

MARQUES MOREIRA DE SOUSA

**HEURÍSTICAS PARA O PROBLEMA DO CAIXEIRO VIAJANTE
COM SELEÇÃO DE HOTÉIS**

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS – BRASIL
2015

**Ficha catalográfica preparada pela Biblioteca Central da Universidade
Federal de Viçosa - Câmpus Viçosa**

T

S725h
2015

Sousa, Marques Moreira de, 1989-

Heurísticas para o problema do caixeiro viajante com
seleção de hotéis / Marques Moreira de Sousa. – Viçosa, MG,
2015.

xiii, 87f. : il. (algumas color.) ; 29 cm.

Orientador: Luciana Brugiolo Gonçalves.

Dissertação (mestrado) - Universidade Federal de Viçosa.

Referências bibliográficas: f. 83-87.

1. Programação heurística. 2. Algoritmos. 3. Otimização
Combinatória. I. Universidade Federal de Viçosa. Departamento
de Informática. Programa de Pós-graduação em Ciência da
Computação. II. Título.

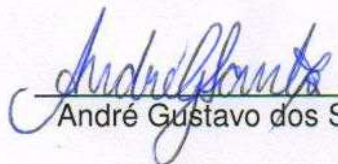
CDD 22. ed. 518

MARQUES MOREIRA DE SOUSA

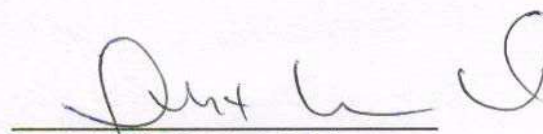
**HEURÍSTICAS PARA O PROBLEMA DO CAIXEIRO VIAJANTE
COM SELEÇÃO DE HOTÉIS**

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

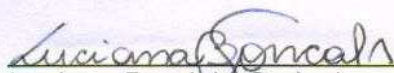
APROVADA: 25 de fevereiro de 2015.



André Gustavo dos Santos



Luiz Satoru Ochi



Luciana Brugiolo Gonçalves
(Orientadora)

À Deus, à minha família e aos meus amigos por me fazerem acreditar que sou capaz.

“Educação é uma descoberta progressiva de nossa própria ignorância”
(Voltaire)

Agradecimentos

Só vemos o quanto as pessoas são importantes, quando não as temos mais ao nosso lado. Assim, o mínimo que podemos fazer é agradecer aqueles que sempre nos ajudaram. Não apenas com sua presença, mas também com pensamentos positivos e orações. Com certeza, de alguma forma, todas as energias são essenciais para que nossas conquistas possam tornar-se reais.

Primeiramente, agradeço a **Deus** por permitir minha existência e por sempre guiar meus passos. Momentos difíceis surgiram e continuarão a surgir, mas acredito que cada um tem seu motivo e representa um aprendizado contínuo ao qual precisamos vivenciar.

Aos meus PAIS **Marieta** e **José**, pessoas as quais não tenho palavras suficientes para agradecer tudo que fizeram e continuam a fazer por mim, sem dúvida fazem parte dos pilares que me sustentam. Vocês me ensinaram a ser humilde, educado, perseverante e a discernir o bem e o mau. Muito obrigado por insistirem pra que eu sempre estudasse e buscasse construir uma carreira e sem dúvida me tornar um ser humano melhor.

A eles, meus IRMÃOS **Carlos** e **Marcos**. Pessoas a quem posso sempre confiar, não importa o momento. Simples palavras também não são suficientes para expressar o quão grato sou por tudo que fizeram, o mínimo que posso fazer é tentar retribuir tudo o que me proporcionaram.

A minha namorada **Mariana**, por ser sempre compreensível e por me aguentar nos dias em que sentia vontade de desistir de tudo. Obrigado, com certeza sem você teria sido muito mais difícil.

As minhas cunhadas **Mônica** e **Liliane** e a minha sobrinha **Julia**, agradeço por sempre me receberem de braços abertos. Obrigado por proporcionarem grandes momentos de alegria.

A meus primos **Raquel** e **Alessandro**, pessoas ímpares as quais pude sempre contar. Considero como irmãos e respeito muito.

A todos meus familiares, em especial aos meus tios, obrigado por todo auxílio

fornecido durante toda a minha vida.

Agradeço imensamente à minha orientadora **Luciana Brugiolo**, por sempre me auxiliar na realização dos trabalhos e acreditar que eu seria capaz de concretizar as tarefas propostas, apesar dos atrasos. Por transmitir os conhecimentos necessário para a execução desta dissertação. Não a vejo apenas como orientadora, mas também como uma grande amiga.

Aos professores do Departamento de Informática (DPI), em especial aos quais tive o prazer de ser aluno. Foram responsáveis por grande parte da minha formação acadêmica e pessoal. Com certeza, devem ser considerados como grandes mestres da sabedoria.

Aos meus professores da graduação que me ensinaram os primeiros passos para que eu fosse capaz de chegar onde estou hoje. Grandes profissionais aos quais tenho grande respeito e admiração.

Aos funcionários do DPI, que de alguma forma contribuíram para minha formação. Em especial ao **Altino**, que sempre se mostrou solícito para resolver qualquer pendência relacionada ao mestrado, além de ser uma pessoa com a qual sempre pude conversar sobre qualquer assunto.

Agradeço aos meus amigos **Gilberto Oliveira** e **Marcos Vinícius** pela amizade, compreensão nos momentos de brincadeira e pela ajuda no desenvolvimento de parte deste trabalho.

Ao meu colega e amigo **Willian Reis**, companheiro de estudos que sempre esteve ao meu lado durante o mestrado e se prontificou a me ajudar na resolução dos problemas que surgiam. Amizade que levo pra sempre.

Ao colega de mestrado **Vinícius Vilar** pelos ensinamentos e por ter se prontificado a ajudar nos momentos em que mais precisei. Sua paciência para explicar o que deveria ser feito e como fazer, foi sem dúvida crucial para o término desta dissertação.

Aos meus colegas de mestrado, que sempre estiveram presentes. Lembro de nossas noites e finais de semana de estudo, do companheirismo e amizade. Foram muitos momentos de descontração, brincadeiras, apreensão e comemoração. Agradeço, por fazerem parte desta caminhada.

Aos colegas do futebol, obrigado por proporcionarem momentos alegres e também uma forma de praticar algum exercício físico, visto que passamos a maior parte do tempo em frente ao computador.

Aos meus colegas de república, que foram como uma família durante o tempo que estive em Viçosa. Muito obrigado por todos os momentos proporcionados.

Agradeço também às pessoas, que de alguma forma participaram direta ou indiretamente desta trajetória em que estive cursando o mestrado. Às vezes, uma pequena contribuição ou até mesmo um incentivo são suficientes para trazer felicidade e vontade de seguir em frente.

Agradeço ao financiamento fornecido pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao qual foi essencial para que pudesse me dedicar exclusivamente a pesquisa e a escrita dessa dissertação.

Agradeço a Universidade Federal de Viçosa e ao Departamento de Informática por fornecerem o suporte necessário para adquirir o título de mestre e o conhecimento para atuar como educador.

Por fim, tenho a dizer que verdadeiros amigos são pra toda a vida e ninguém consegue triunfar sem que existam pessoas ao seu redor, para fornecer o suporte necessário. Obrigado a todos por estarem ao meu lado!

Sumário

Lista de Figuras	ix
Lista de Tabelas	xi
Resumo	xii
Abstract	xiii
1 INTRODUÇÃO	1
1.1 O problema e sua importância	2
1.2 Objetivos	4
1.2.1 Objetivo Geral	4
1.2.2 Objetivos Específicos	4
1.3 Organização da dissertação	4
2 O PROBLEMA DO CAIXEIRO VIAJANTE COM SELEÇÃO DE HOTÉIS	6
2.1 Formulação Matemática	11
3 HEURÍSTICAS	14
3.1 Procedimentos Estruturais	15
3.1.1 Construção da Solução Inicial	15
3.1.2 Estruturas de Vizinhaça	20
3.1.3 Perturbação da Solução	27
3.2 Heurísticas Básicas	28
3.2.1 Busca Tabu	28
3.2.2 Descida de Vizinhaça Variável	31
3.3 Heurísticas Híbridas	33
3.3.1 Algoritmo Memético com Busca Tabu e com RVND	33

3.3.2	Algoritmo Iterated-Greedy com VND	37
4	RESULTADOS	39
4.1	Ambiente Computacional Utilizado	39
4.2	Métrica para Avaliação dos Algoritmos	39
4.3	Grupos de Instâncias Utilizados	40
4.4	Calibração de Parâmetros	41
4.4.1	Calibração dos Parâmetros do AMBT	41
4.4.2	Calibração dos Parâmetros do AMRVND	45
4.4.3	Calibração dos Parâmetros do IGVND	46
4.5	Testes Computacionais	48
5	CONCLUSÕES	80
5.1	Trabalhos Futuros	81
	Referências Bibliográficas	83

Lista de Figuras

1.1	Exemplo de definição das viagens e da rota completa, composta por clientes (C) e hotéis (H).	3
2.1	O jogo do viajante comercial. Adaptado de Applegate et al. [2006b] . . .	7
2.2	Exemplo de uma rota do PCVSH composta por 6 viagens	8
3.1	Exemplo do grafo criado para definição do melhor caminho.	18
3.2	Exemplo de escopo de aplicação de uma estrutura <i>inter-trip</i> e <i>intra-trip</i> .	20
3.3	Exemplo de reconexão dos caminhos pela estrutura 2-OPT.	21
3.4	Exemplo de reconexão dos caminhos pela estrutura 3-OPT.	22
3.5	Movimentos para realocar os 3 primeiros clientes da viagem.	23
3.6	Realocação de um cliente da viagem 1 para viagem 2.	24
3.7	Troca de um cliente entre a viagem 1 e a viagem 2.	25
3.8	Exemplo de possibilidade da troca dos hotéis utilizando a estrutura <i>ChangeHotels</i>	26
3.9	Exemplo de remoção de um hotel intermediário da rota.	26
3.10	Exemplo de remoção de um hotel intermediário e inversão do restante da rota.	27
3.11	Representação da solução em diferentes otimalidades. Adaptado de Talbi [2009].	28
3.12	Representação do funcionamento do Algoritmo Memético.	34
3.13	Representação do cruzamento com dois pontos de quebra.	36
4.1	Gráfico de médias do GAP de todas configurações para Busca Tabu no primeiro teste.	42
4.2	Gráfico de médias do GAP de todas configurações para Busca Tabu no segundo teste.	43
4.3	Gráfico de médias do GAP de todas configurações para o Algoritmo Memético com Busca Tabu no primeiro teste da calibragem.	44

4.4	Gráfico de médias do GAP de todas configurações para o Algoritmo Memético com Busca Tabu no segundo teste da calibragem.	44
4.5	Gráfico de médias do GAP de todas configurações para o Algoritmo Memético com RVND no primeiro teste da calibragem.	45
4.6	Gráfico de médias do GAP de todas configurações para o Algoritmo Memético com RVND no segundo teste da calibragem.	46
4.7	Gráfico de médias do GAP de todas configurações para o Algoritmo IGVND no primeiro teste da calibragem.	47
4.8	Gráfico de médias do GAP de todas configurações para o Algoritmo IGVND no segundo teste da calibragem.	48
4.9	Gráfico de médias do GAP de todas configurações para o Algoritmo IGVND no terceiro teste da calibragem.	49
4.10	Gráfico de probabilidade de alcance do alvo definido para a instância <i>eil51.s5</i>	75
4.11	Gráfico de probabilidade de alcance do alvo definido para a instância <i>r201</i>	76
4.12	Gráfico de probabilidade de alcance do alvo definido para a instância <i>pcb442.s4</i>	77
4.13	Gráfico de convergência da solução para todos as heurísticas comparadas, considerando a instância <i>eil51.s5</i>	77
4.14	Gráfico de convergência da solução para todos as heurísticas comparadas, considerando a instância <i>r201</i>	78
4.15	Gráfico de convergência da solução para todos as heurísticas comparadas, considerando a instância <i>pcb442.s4</i>	79

Lista de Tabelas

4.1	Resumo das características das instâncias utilizadas	41
4.2	Resultados SET_1 e SET_2 modelo exato	50
4.3	Resultados SET_3 e SET_4 modelo exato	50
4.4	Resultados dos Algoritmos Meméticos SET_1	54
4.5	Resultados para AM SET_2 com 10 clientes	55
4.6	Resultados para AM SET_2 com 15 clientes	56
4.7	Resultados para AM SET_2 com 30 clientes	57
4.8	Resultados para AM SET_2 com 40 clientes	58
4.9	Resultados para AM SET_3 com 3 hotéis extras	59
4.10	Resultados para AM SET_3 com 5 hotéis extras	60
4.11	Resultados para AM SET_3 com 10 hotéis extras	61
4.12	Resultados Algoritmos Meméticos para SET_4	62
4.13	Resultados para o Algoritmo <i>Iterated Greedy</i> SET_1	64
4.14	Resultados Algoritmo <i>Iterated Greedy</i> SET_2 com 10 clientes	65
4.15	Resultados Algoritmo <i>Iterated Greedy</i> SET_2 com 15 clientes	66
4.16	Resultados Algoritmo <i>Iterated Greedy</i> SET_2 com 30 clientes	67
4.17	Resultados Algoritmo <i>Iterated Greedy</i> SET_2 com 40 clientes	68
4.18	Resultados Algoritmo <i>Iterated Greedy</i> SET_3 com 3 hotéis extras	70
4.19	Resultados Algoritmo <i>Iterated Greedy</i> SET_3 com 5 hotéis extras	71
4.20	Resultados Algoritmo <i>Iterated Greedy</i> SET_3 com 10 hotéis extras	72
4.21	Resultados Algoritmo <i>Iterated Greedy</i> SET_4	73

Resumo

SOUSA, Marques Moreira de, M.Sc., Universidade Federal de Viçosa, Fevereiro de 2015. **Heurísticas para o Problema do Caixeiro Viajante com Seleção de Hotéis** Orientadora: Luciana Brugiolo Gonçalves.

A otimização de percursos é de grande interesse para empresas que fornecem serviços relacionados com transporte, seja de pessoas ou de mercadorias, visto que podem levar a uma diminuição do tempo e do custo necessário para prestar um serviço e, consequentemente, elevar a lucratividade. Neste trabalho é abordado o Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH), uma variante do clássico Problema do Caixeiro Viajante (PCV). No PCVSH, existe um limite de tempo imposto a uma jornada diária de trabalho. Desta forma, considerando que há um conjunto de clientes que precisam ser atendidos, há casos em que não é possível atender a todos em um mesmo dia. Levando em consideração esta restrição, é necessário escolher hotéis, dentre um conjunto previamente fornecido, para que seja realizada a parada entre duas jornadas diárias consecutivas. O objetivo desta dissertação é apresentar, discutir e tratar o Problema do Caixeiro Viajante com Seleção de Hotéis aplicando heurísticas e comparando os resultados obtidos com aqueles disponíveis na literatura. Foram propostas três heurísticas, sendo duas baseadas em Algoritmo Memético (AM) e outra baseada na metaheurística *Iterated Greedy* (IG), além de um modelo de Programação Linear Inteira alternativo ao existente na literatura.

Abstract

SOUSA, Marques Moreira de, M.Sc., Universidade Federal de Viçosa, February of 2015. **Heuristics to the Travelling Salesperson Problem with Hotel Selection** Adviser: Luciana Brugiolo Gonçalves.

The optimization of routes is of great interest to companies that provide services related to transportation, being of peoples or goods, as they can lead to a decrease in the time and cost required to provide a service, and consequently to a raise in profitability. In this work we deal with the Travelling Salesperson Problem with Hotel Selection (PCVSH), a variant of the classic Travelling Salesperson Problem (TSP). In PCVSH there is a limit of time imposed to a daily journey of work. Thus, considering that there is a set of customers that need to be visit, there are cases in which one cannot visit all in one day. Considering this restriction, one you must choose hotels, among a previously given set, where a break will take place between two working daily journey. The aim of this work is to present, discuss and solve the Travelling Salesperson Problem with Hotel Selection applying heuristics and comparing the results with those available in the literature. Three heuristics, two based on memetic algorithm (AM) and another based on the metaheuristic Iterated Greedy (IG), and an alternative Integer Linear Programming model to the existing on the literature were proposed.

Capítulo 1

INTRODUÇÃO

Com o aumento da competitividade, principalmente devido a globalização, a busca por alternativas que melhor utilizem os recursos disponíveis são demandadas. Para problemas onde o objetivo é encontrar a melhor solução dentre um dado conjunto de possibilidades, diferentes estratégias podem ser aplicadas na busca da melhor solução. A utilização de técnicas que auxiliam na tomada de decisões traz melhorias consideráveis para diversos setores. No setor de transportes, encontrar uma forma de otimizar a utilização de recursos pode representar uma considerável redução no custo final de produtos, como também diminuir a quantidade de gases poluentes lançados no meio ambiente que está diretamente relacionada à redução do consumo de petróleo, lubrificantes e desgaste de componentes, entre outros.

O setor de transportes está diretamente ligado a logística que pode ser implantada nas empresas. Ao otimizar os processos logísticos, uma empresa torna-se mais competitiva frente as demais. Com isso, a globalização das cadeias de suprimentos, o aumento da diversidade dos produtos ofertados e o aumento no nível de exigência dos consumidores obrigam as empresas a suprirem suas necessidades logísticas com rapidez, consistência e flexibilidade, com menor custo possível [Fleury, 2011].

A aplicação de estratégias de otimização em processos logísticos costumeiramente alcança valores de 5% a 20% em economia com frete, diminuição tanto na distância percorrida pela frota quanto em emissão de CO_2 , redução do ciclo de planejamento das rotas de horas para minutos e uma diminuição média da frota necessária que pode chegar a 25% [Neolog, 2014].

No Brasil, o transporte rodoviário é o principal meio para distribuição de alimentos, matérias primas e outros bens essenciais à população. Em recente estudo realizado pela Confederação Nacional do Transporte, constatou-se que cerca de 65% da movimentação de cargas e 90% da movimentação dos passageiros ocorrem pelas

rodovias [CNT, 2013]. Este fato reforça a necessidade de otimizar a utilização deste meio de transporte, melhorando o serviço oferecido.

1.1 O problema e sua importância

Problemas de roteamento são exemplos de problemas do ramo da otimização largamente estudados. O Problema de Roteamento de Veículos (PRV) [Dantzig & Ramser, 1959] busca determinar um grupo de rotas que devem ser percorridas por uma frota de veículos para fornecer serviços a um grupo de clientes. O PRV possui uma grande importância na cadeia de suprimentos de diversas empresas envolvidas com o transporte de bens e pessoas, isto devido a quantidade expressiva de aplicações em situações reais enfrentadas diariamente no ambiente empresarial, como: serviços de entrega de jornais e revistas, coleta de lixo, distribuição de comidas e bebidas, transporte de passageiros, etc.

Dentre os problemas similares ao Problema de Roteamento de Veículos, pode-se citar o Problema do Caixeiro Viajante (PCV) que possui grande usabilidade prática, tendo recebido grande atenção por parte dos pesquisadores da área de otimização [Applegate et al., 2006b]. No PCV, o objetivo é encontrar um ciclo hamiltoniano de custo mínimo.

Neste dissertação, uma nova variante do PCV introduzida na literatura por Vansteenwegen et al. [2012] e chamada de Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH) é tratada. O PCVSH é aplicado em situações onde há um limite de tempo imposto para a jornada de trabalho. Neste caso, podem ocorrer situações em que não é possível atender todos os clientes em apenas uma jornada de trabalho. Torna-se necessário, ao final de uma jornada, procurar um hotel para aguardar o início de uma nova jornada, de onde será possível continuar o atendimento aos clientes no dia seguinte. Assim, ao final de cada jornada, deve ser tomada a decisão de voltar ao ponto de partida ou ficar em um hotel que esteja localizado em sua rota de atendimento aos clientes. Outra característica importante do PCVSH é que um hotel não necessariamente precisa ser utilizado e, caso for, poderá ser utilizado mais de uma vez.

O problema considerado nesta pesquisa ocorre em ambientes empresariais, principalmente no que tange a logísticas de produtos ou serviços. Neste caso, existe um meio de transporte e este recurso deve ser alocado de forma a fornecer um serviço a um grupo de clientes. Todos os clientes devem ser atendidos e o atendimento não necessariamente precisa ocorrer no mesmo dia. Para estas empresas, tratar

problemas similares ao PCVSH, e aplicar na prática, propicia economia média de até 10% dos custos finais com transporte [Toth & Vigo, 2001].

Foram utilizados dois termos importantes para o entendimento e representação de soluções do PCVSH, são eles: Viagem e Rota. Na Figura 1.1 é ilustrado um exemplo de representação de uma solução para o PCVSH. Neste contexto o termo Viagem refere-se a um subconjunto de clientes que foram atendidos em uma jornada de trabalho. A viagem deve, obrigatoriamente, ser iniciada e finalizada em algum dos hotéis disponíveis, levando em consideração que há um limite máximo de tempo a ser utilizado. Já o termo Rota é utilizado para descrever o conjunto de todas as viagens percorridas para atender a todos os clientes. No exemplo ilustrado na figura, a rota é composta por quatro viagens.

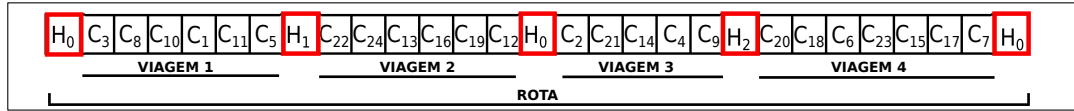


Figura 1.1. Exemplo de definição das viagens e da rota completa, composta por clientes (C) e hotéis (H).

De acordo com Castro et al. [2013], o PCVSH pode ser tratado considerando-se dois objetivos básicos:

- **Minimizar o número de viagens:** definir uma rota que possa ser percorrida pelo caixeiro, considerando o menor número de jornadas de trabalho.
- **Minimizar o tempo total da rota:** está relacionado com a minimização da soma total do tempo gasto pelo caixeiro desde sua saída do hotel de partida até seu retorno, após ter visitado todos os clientes.

Aplicações práticas são vistas em problemas relacionados com coleta e entrega de mercadorias e/ou serviços, vivenciados principalmente por empresas do segmento de transportes (representante de vendas, entregador de mercadorias e encomendas, etc), onde faz-se necessário ter um diferencial para concorrer no mercado cada vez mais competitivo. Entre os diferenciais comerciais apontados por Laudon & Laudon [2007], destaca-se: a redução de custos, cumprimento de prazos e melhoria nos processos decisórios.

Como o clássico Problema do Caixeiro Viajante é reconhecidamente pertencente à classe de problemas NP-difíceis [Garey & Johnson, 1979], o fato do PCVSH ser uma generalização faz com também seja classificado nesta mesma classe de problemas [Castro et al., 2012]. No PCVSH, se o tempo limite de uma jornada de

trabalho for suficiente para visitação de todos os clientes, então o PCVSH se reduz ao PCV.

1.2 Objetivos

Nesta seção serão apresentados os objetivos geral e específicos que serão desenvolvidos ao longo desta dissertação.

1.2.1 Objetivo Geral

O objetivo geral desta dissertação é avaliar o desempenho de diferentes abordagens para o Problema do Caixeiro Viajante com Seleção de Hotéis, verificando a eficácia destas abordagens para os conjuntos de instâncias disponíveis na literatura.

1.2.2 Objetivos Específicos

Nesta pesquisa são considerados três objetivos específicos, detalhados a seguir:

- Propor um modelo de Programação Linear Inteira que utilize uma estratégia de restrição de subciclos diferente do modelo que foi proposto na literatura.
- Propor diferentes abordagens baseadas em métodos heurísticos para tratar o Problema do Caixeiro Viajante com Seleção de Hotéis, verificando qual das abordagens utilizadas consegue ser mais competitiva em relação às abordagens utilizadas na literatura.
- Publicar os resultados encontrados por meio de artigos e da escrita desta dissertação.

1.3 Organização da dissertação

Nesta seção, é realizada uma breve descrição do que é apresentado em cada capítulo desta dissertação.

A dissertação, além do capítulo de introdução possui mais quatro capítulos. O Capítulo 2 apresenta o problema e seu estado da arte. É apresentada uma descrição do Problema do Caixeiro Viajante, no qual se baseia o problema tratado nesta dissertação. Também são apresentados os trabalhos que já foram desenvolvidos acerca do PCVSH, mostrando a evolução e melhoria dos resultados encontrados em cada trabalho. Por fim, é apresentada uma descrição formal do Problema do

Caixeiro Viajante com Seleção de Hotéis e o modelo alternativo de Programação Linear Inteira proposto.

No Capítulo 3 é feita uma descrição das heurísticas utilizadas, detalhando os procedimentos embutidos em cada uma das estratégias. No Capítulo 4 é apresentado o procedimento utilizado para realização da calibração dos parâmetros de cada heurística. São apresentados também os resultados obtidos pelo modelo de Programação Linear Inteira que foi adaptado e os resultados obtidos pelas estratégias heurísticas consideradas nesta dissertação. As estratégias aplicadas são comparadas com as melhores estratégias conhecidas na literatura.

Finalizando, no Capítulo 5 são apresentadas as considerações finais, bem como as principais contribuições obtidas por esta pesquisa, além de sugestões de trabalhos que poderão ser realizados no futuro.

Capítulo 2

O PROBLEMA DO CAIXEIRO VIAJANTE COM SELEÇÃO DE HOTÉIS

Neste capítulo é apresentada uma revisão da literatura acerca do problema alvo desta dissertação, considerando os trabalhos contidos na literatura desde o surgimento do problema.

Apesar de ser um problema largamente estudado por diversos autores nas últimas décadas, a história acerca do Problema do Caixeiro Viajante (PCV) ainda pode ser considerada incompleta, pela falta de registros concretos sobre seu provável criador. Os autores em Applegate et al. [2006b] fizeram um estudo aprofundado e algumas questões foram esclarecidas.

De acordo com Applegate et al. [2006b], a origem do nome do Problema do Caixeiro Viajante ainda é um grande mistério. Esta informação não aparece em nenhum dos documentos disponibilizados. Sabe-se que um dos pesquisadores mais influentes na história do PCV foi Merrill Flood da Universidade de Princeton e a companhia RAND. Uma das primeiras referências contendo o termo aparece em um relatório de 1949 [Applegate et al., 2006b]. Porém, os autores têm ciência de problemas similares desde os anos 1800, quando matemáticos começaram a estudá-los. Um exemplo é o Jogo do Viajante Comercial ilustrado na Figura 2.1.

Com o passar dos anos, desde o surgimento do PCV, novas estratégias para resolver o problema foram surgindo e com isso diversas variantes para o problema também foram criadas. Pode-se citar como trabalhos que abordam o PCV e suas variantes Lin [1965]; Bektas [2006]; Applegate et al. [2006a], sendo que este último pode ser considerado como um marco histórico para o PCV. Isto pois, os autores

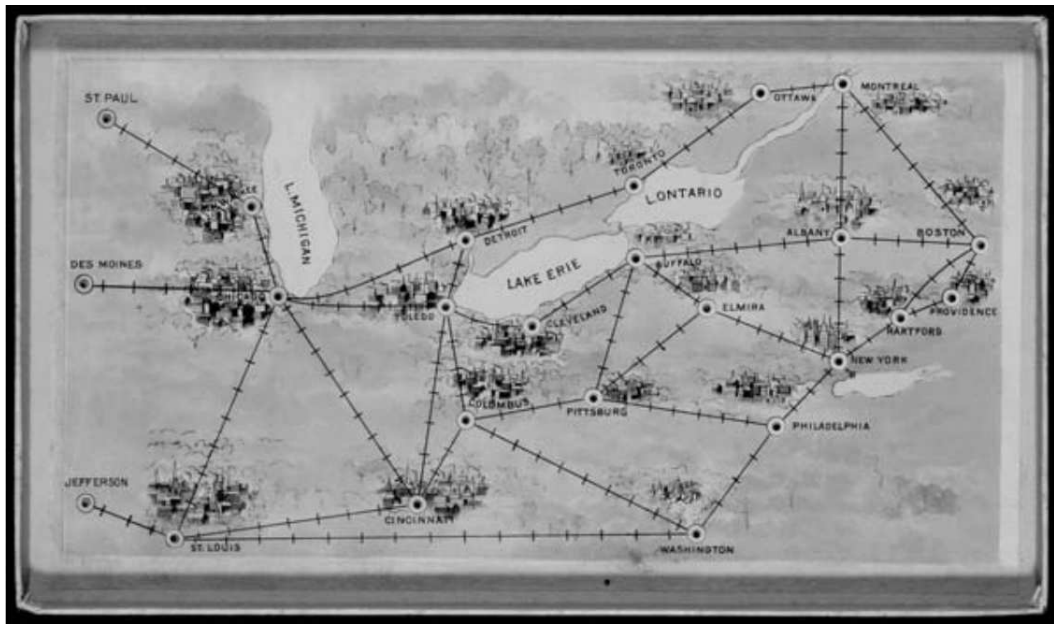


Figura 2.1. O jogo do viajante comercial. Adaptado de Applegate et al. [2006b]

desenvolveram uma biblioteca que combina várias estratégias de Programação Linear e heurísticas capaz de resolver quase todas as instâncias disponibilizadas na literatura, sendo que a maior possui 85900 cidades [Applegate et al., 2006a].

O problema abordado neste trabalho, denominado Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH) é considerado uma generalização/variante do clássico PCV. Relembrando, no PCVSH o caixeiro deve partir de um determinado local, atender a todos os clientes, utilizar algum dos hotéis disponíveis caso necessário e retornar ao local de partida. Considerando que há casos em que não é possível realizar o atendimento de todos os clientes na mesma jornada de trabalho devido ao limite de tempo imposto, o caixeiro necessita interromper a visitação dos clientes ao final de uma jornada de trabalho e escolher um hotel para efetuar uma parada. Posteriormente, novas jornadas devem ser estabelecidas de acordo com a necessidade, até que seja completado o atendimento da clientela. A característica de variante do PCV é comprovada, pois se for definido um limite de tempo da jornada de trabalho grande o suficiente para que o caixeiro necessite de apenas uma jornada para atender a todos os clientes, então o PCVSH se resumirá no PCV.

Além de herdar as características do PCV, o PCVSH também compartilha características com o Problema de Roteamento de Veículos (PRV). Por exemplo, no que concerne ao fato que não há uma rota simples que parte de um depósito e volta para o mesmo, a rota no PCVSH é constituída pelo conjunto de viagens que

iniciam e terminam em um dado hotel, sendo que o hotel inicial da primeira viagem e o hotel final da última viagem são pré-definidos. Assim, caso a rota do PCVSH tenha mais de uma viagem, o hotel inicial e final de uma viagem assemelham-se ao depósito do PRV, mas não necessariamente precisam ser idênticos como ocorre no PRV [Vansteenwegen et al., 2012].

Na Figura 2.2 é ilustrada uma possível solução para uma instância do PCVSH contendo 18 clientes e 8 hotéis. Os clientes são representados pelos círculos, os hotéis intermediários representados pelos quadrados e o hotel inicial/final é definido pelo pentágono. Neste caso, o caixeiro viajante realizou 6 viagens para atender a todos os clientes. A rota é definida a partir do hotel h_0 , sendo a sequência de viagens $(V_1, V_2, V_3, V_4, V_5, V_6)$, onde $V_1 = (H_0, C_1, C_3, C_6, C_4, H_2)$, $V_2 = (H_2, C_8, C_7, C_{10}, H_5)$, $V_3 = (H_5, C_{11}, C_{13}, H_5)$, $V_4 = (H_5, C_9, C_{12}, C_{19}, C_{15}, H_3)$, $V_5 = (H_3, C_{17}, C_{14}, C_2, H_7)$ e $V_6 = (H_7, C_{18}, C_5, C_{16}, H_0)$. Os hotéis H_1, H_4 e H_6 não foram visitados durante o percurso, realçando a característica que o caixeiro não é obrigado a visitar todos os hotéis disponíveis.

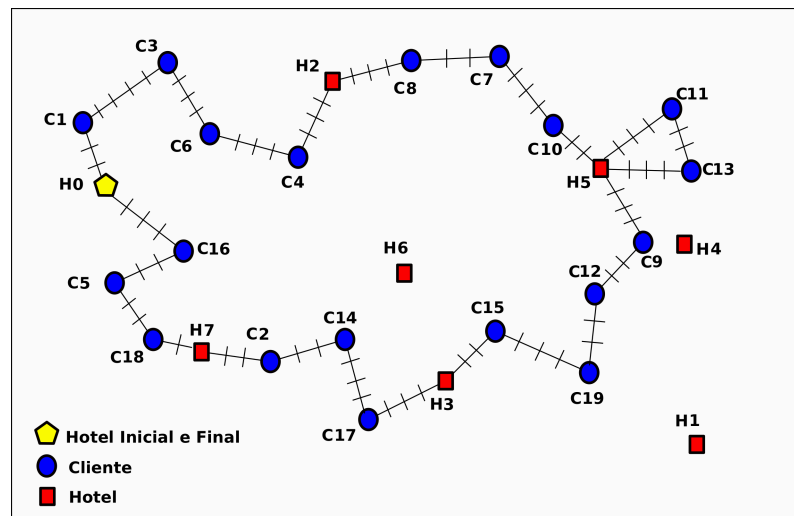


Figura 2.2. Exemplo de uma rota do PCVSH composta por 6 viagens

Os autores em Vansteenwegen et al. [2012] foram os responsáveis por introduzir o PCVSH na literatura. Neste primeiro trabalho foi apresentada uma formulação matemática, destacando a diferença entre o PCVSH e outros problemas de otimização contidos na literatura. Além do modelo, foi apresentada uma heurística que utiliza dois procedimentos para construção da solução inicial e um procedimento de melhoria composto por diferentes operadores de busca local. Por fim, outra contribuição relevante foi o desenvolvimento de um conjunto de instâncias, com diferentes tamanhos e níveis de dificuldade. Neste primeiro trabalho, os autores compararam

os resultados obtidos pela heurística, com o resultado obtido pelo modelo. Para instâncias em que o ótimo foi encontrado pelo modelo, comprovou-se também a eficácia da abordagem heurística proposta.

Dando continuidade à pesquisa sobre o PCVSH, em Castro et al. [2012] o problema, pela primeira vez, foi tratado com uma metaheurística. A metaheurística aplicada foi uma combinação de uma heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) [Feo & Resende, 2003] com uma heurística *Variable Neighbourhood Descent* (VND) [Hansen & Mladenović, 2001]. Para o conjunto de instâncias contido na literatura, a heurística provou ser capaz de encontrar, quase na totalidade, os melhores resultados conhecidos até o momento e ainda conseguiu encontrar melhores resultados para um subconjunto das instâncias.

Partindo para um tipo de estratégia ainda não testada para tratar o problema, os autores de Castro et al. [2013] utilizaram uma metaheurística populacional. Nesta abordagem, denominada Algoritmo Memético (AM) [Moscato et al., 2004], as operações básicas, como cruzamento e mutação dos indivíduos da população são realizadas alterando-se apenas os hotéis. Para a definição das melhores posições para cada cliente, foi utilizada uma heurística baseada na Busca Tabu (BT) [Glover, 1989, 1990]. Os autores utilizaram estruturas de vizinhança e heurísticas de geração da solução inicial conhecidas na literatura e que obtiveram bons resultados em trabalhos anteriores [Vansteenwegen et al., 2012; Castro et al., 2012]. Os resultados alcançados foram melhores que os obtidos nos trabalhos anteriores que trataram o PCVSH.

Com o objetivo de encontrar soluções para as instâncias conhecidas do PCVSH em um tempo computacional pequeno, os autores em Castro et al. [2014] desenvolveram uma heurística capaz de alcançar, em tempos computacionais bem reduzidos, boas soluções para todas as instâncias conhecidas. Os autores propuseram uma nova formulação matemática para o PCVSH, que diferentemente da formulação apresentada em Castro et al. [2013], é baseada na utilização de viagens, ao invés de arestas. A criação de uma solução inicial pela heurística proposta utiliza um método *order-first split-second*. Em um primeiro momento, se define uma solução viável para o PCV e, posteriormente, particiona esta solução tornando-a uma solução para o PCVSH. Este método de geração da solução inicial é combinado com uma heurística *Iterated Local Search* (ILS) [Lourenço et al., 2003], que teve sua estrutura típica de busca local substituída pela metaheurística VND [Hansen et al., 2008, 2010]. Os autores também criaram duas estruturas de perturbação da solução, a primeira considerando mudanças apenas nos clientes e a segunda modificando apenas os hotéis intermediários da solução. As estruturas de vizinhança utilizadas no VND foram as mesmas utilizadas em trabalhos anteriores [Vansteenwegen et al., 2012; Castro

et al., 2012, 2013]. A proposta encontrou resultados competitivos quanto a qualidade da solução em tempo computacional inferior aos consumidos pelos demais métodos conhecidos até o momento.

O PCVSH tem recebido grande atenção por parte de alguns pesquisadores. Com isso, novas extensões/variantes do problema vem sendo apresentadas, o que contribui ainda mais para a exploração de abordagens ainda não utilizadas e que podem ajudar a resolver problemas similares no campo da otimização.

Como variante do PCVSH, pode-se citar o problema abordado no trabalho de Baltz et al. [2014] que introduz e estuda o Problema do Caixeiro Viajante com Múltiplas Janelas de Tempo e Seleção de Hotéis (PCV-MJTSH). O PCV-MJTSH consiste em determinar uma rota para o caixeiro viajante que visita vários clientes em diferentes localidades, obedecendo as janelas de tempo. Considera ainda que o caixeiro tem a opção de parar em um hotel, caso não consiga atender a todos os clientes em uma única jornada de trabalho. O objetivo nesta variante é minimizar os custos da rota, que envolvem o salário do caixeiro, as despesas com hotéis, despesas de viagem e as taxas de penalização para os clientes que não forem atendidos. Os autores apresentaram um modelo de programação linear inteira mista e uma abordagem heurística que combina estratégias de inserção mais barata, 2-OPT e reinício randomizado. Para pequenas instâncias o modelo encontrou as soluções ótimas em tempo razoável. A heurística por sua vez, obtém as soluções encontradas pelo modelo matemático nas instâncias pequenas e provou ser rápida, eficiente e eficaz para grandes instâncias. O PCV-MJTSH é uma variante que começou a ser explorada recentemente e tem potencial para que, a partir dela, sejam realizados trabalhos de grande qualidade.

Outro problema que possui grande similaridade com o PCVSH é o Problema do Caixeiro Viajante com Grupamentos (PCVG) [Chisman, 1975]. O PCVG possui como objetivo encontrar um ciclo hamiltoniano de custo mínimo que passe por todos os vértices do grafo associado e visita os vértices de cada grupo de forma contígua. Esta variante do PCV apresenta grande similaridade uma vez que pode-se assimilar a construção das viagens no PCVSH com a definição dos grupos no PCVG. A escolha da ligação entre dois grupos quaisquer é equivalente a escolha do melhor hotel entre duas viagens no PCVSH. Diversos autores realizaram trabalhos acerca do PCVG desde seu surgimento, sendo os trabalhos mais recentes realizados por Mestria et al. [2013]. Nesse trabalho, o autor utiliza metaheurísticas como GRASP e ILS, além de, produzir um algoritmo exato para tratar tal problema. Ambas as abordagens propostas apresentaram bons resultados quando comparadas às existentes na literatura.

2.1 Formulação Matemática

Para definição formal do Problema do Caixeiro Viajante com Seleção Hotéis (PCVSH), considere um grafo $G = (V, A)$ onde $V = H \cup C$, sendo H o conjunto não vazio que representa os hotéis e C o conjunto que representa os clientes que devem ser visitados. No conjunto $A = \{(i, j) | i, j \in V, i \neq j\}$, cada aresta (i, j) representa a ligação entre os clientes/hotéis i e j . Cada cliente $i \in C$ requer um tempo de serviço ou tempo de visita τ_i (com $\tau_i = 0$, para todo $i \in H$). O tempo c_{ij} necessário para viajar da localidade i para j é conhecido para todos os pares de localidades. O mesmo hotel ($i = 0, i \in H$) deve iniciar e finalizar a rota, podendo ser utilizado como hotel intermediário entre viagens. Os outros hotéis contidos em H , podem ser utilizados quando forem necessários para constituir a rota. Como um hotel H pode ser visitado várias vezes na mesma rota, a solução do PCVSH pode ou não ser representada por um ciclo simples [Vansteenwegen et al., 2012].

Definidas as características básicas do PCVSH, outras restrições complementares são aplicadas, como a restrição de que cada viagem deve iniciar e finalizar em um dos hotéis disponíveis, o tempo total percorrido em uma viagem não pode exceder o tempo limite L e por fim, uma viagem deve iniciar em um hotel onde a viagem prévia terminou. Estas restrições foram definidas de acordo com as características específicas do problema [Castro et al., 2013].

Os autores em Castro et al. [2013] apresentaram um modelo de Programação Linear Inteira (PLI) que modifica a formulação apresentada em Vansteenwegen et al. [2012]. Esta nova formulação altera a anterior principalmente considerando a função objetivo que neste caso, considera a minimização do número de viagens e o tempo total percorrido. Tratando-se de um problema de minimização, com o intuito de priorizar rotas que contenham um número menor de viagens, uma constante M suficientemente grande foi utilizada na função objetivo. Assim, ao multiplicar o número de viagens por esta constante, rotas com menor número de viagens serão priorizadas.

Dado x_{ij}^d uma variável binária que recebe o valor 1 se, na d -ésima viagem, uma visita a um cliente ou hotel i é seguida pela visita a um cliente ou hotel j ou o valor 0, caso contrário. A variável binária y^d receberá o valor 1 se na viagem d no mínimo um cliente ou hotel é visitado ou 0, caso contrário. Assim, y^d receberá o valor zero se nenhuma viagem for necessária no dia d . As variáveis y^d e x_{ij}^d são usadas na função objetivo, respectivamente, para minimizar o número de viagens e o tempo total percorrido. Por fim, a constante D representa o número máximo de viagens contidas na solução e foi definida pela solução encontrada na heurística

proposta em Castro et al. [2013].

$$\min \quad M \sum_{d=1}^D y^d + \sum_{d=1}^D \left(\sum_{(i,j) \in A} c_{ij} x_{ij}^d \right) \quad (2.1)$$

$$s.t. \quad \sum_{d=1}^D \sum_{i \in V} x_{ij}^d = 1, \forall j \in C \quad (2.2)$$

$$\sum_{i \in V} x_{ij}^d = \sum_{i \in V} x_{ji}^d, \forall j \in C, \forall d = 1, \dots, D \quad (2.3)$$

$$\sum_{h \in H} \sum_{j \in V \setminus \{h\}} x_{hj}^d = y^d, \forall d = 1, \dots, D \quad (2.4)$$

$$\sum_{h \in H} \sum_{i \in V \setminus \{h\}} x_{ih}^d = y^d, \forall d = 1, \dots, D \quad (2.5)$$

$$\sum_{(i,j) \in A} (c_{ij} + \tau_j) x_{ij}^d \leq L, \forall d = 1, \dots, D \quad (2.6)$$

$$\sum_{j \in V \setminus \{0\}} x_{0j}^1 = 1 \quad (2.7)$$

$$\sum_{i \in V \setminus \{0\}} x_{i0}^d \geq y^d - y^{d+1}, \forall d = 1, \dots, D - 1 \quad (2.8)$$

$$\sum_{i \in V} x_{ih}^d + y^d \geq \sum_{i \in V} x_{hi}^{d+1} + y^{d+1}, \forall h \in H, \forall d = 1, \dots, D - 1 \quad (2.9)$$

$$\sum_{i \in V} x_{ih}^d - \sum_{i \in V} x_{hi}^{d+1} \leq 1 - y^{d+1}, \forall h \in H, \forall d = 1, \dots, D - 1 \quad (2.10)$$

$$x_{ij}^d \leq y^d, \forall (i, j) \in A, \forall d = 1, \dots, D \quad (2.11)$$

$$y^d \geq y^{d+1}, \forall d = 1, \dots, D - 1 \quad (2.12)$$

$$\sum_{i \in \kappa} \sum_{j \in \kappa \setminus \{i\}} x_{ij}^d \leq |\kappa| - 1, \kappa \subset C, 2 \leq |\kappa| \leq |C| - 1, \forall d = 1, \dots, D \quad (2.13)$$

$$x_{ij}^d \in \{0, 1\}, \forall (i, j) \in A, \forall d = 1, \dots, D \quad (2.14)$$

$$y^d \in \{0, 1\}, \forall d = 1, \dots, D \quad (2.15)$$

Considerando o modelo matemático, a função objetivo (2.1) visa minimizar o número de viagens e o tempo total percorrido. As restrições (2.2) garantem que cada cliente será visitado exatamente uma vez. As restrições (2.3) garantem que haja conectividade entre cada viagem contida na rota, as restrições (2.4) e (2.5) garantem que cada viagem inicia e termina em um dos H hotéis disponíveis. As restrições (2.6) impõem um limite ao tempo total de uma viagem (inclui tempo gasto para percorrer o caminho e tempos de visita). As restrições (2.7) e (2.8) definem que a rota deve iniciar e terminar no hotel 0. As restrições (2.9) e (2.10) indicam que se uma viagem termina em um dado hotel, então a próxima viagem deve, obrigatoriamente, iniciar neste hotel. As restrições (2.11) definem que uma

viagem está sendo utilizada só se acontece no mínimo a visita a um cliente ou a um hotel naquela jornada de trabalho. Restrições (2.12) garantem que as viagens serão realizadas em dias consecutivos, iniciando-se no primeiro dia. As restrições (2.13) são aplicadas as viagens e garantem que não serão formados subciclos desconexos da origem. Por fim, as restrições (2.14) e (2.15) indicam o domínio das variáveis.

Neste trabalho foi avaliada uma alteração no modelo de Programação Linear Inteira (PLI) proposto no trabalho de Castro et al. [2013], com o intuito de fornecer uma formulação alternativa que utilize conceitos distintos da que foi baseada. Diferentemente da formulação em que foi baseada, nesta proposta é utilizado um modelo baseado em fluxo, onde o fluxo de entrada nos vértices intermediários deve ser sempre menor que o fluxo de saída desses vértices. Para o hotel inicial e final, foram criadas restrições específicas para tratar o fluxo. Este novo modelo retira do modelo original [Castro et al., 2013] as restrições (2.13) e acrescenta as restrições (2.16) à (2.19).

Uma nova variável inteira f_{ij} foi criada e receberá o valor do fluxo que passará pela aresta que liga os vértices i e j . Caso uma aresta seja utilizada, o fluxo desta aresta que liga o vértice i ao j será sempre menor do que o fluxo que liga o vértice j a outro vértice k (ainda não visitado).

$$f_{0j} = x_{0j}^1, \forall j \in V \setminus \{0\} \quad (2.16)$$

$$\sum_{j \in V} f_{ij} = \sum_{j \in V} f_{ji} + \sum_{d=1}^D \sum_{j \in V} x_{ji}^d, \forall i \in V \setminus \{0\} \quad (2.17)$$

$$f_{ij} \leq (|C| + |D|) \sum_{d=1}^D x_{ij}^d, \forall (i, j) \in V \quad (2.18)$$

$$f_{ij} \geq 0, (i, j) \in A | i \neq j \quad (2.19)$$

As restrições (2.16) definem que o fluxo na aresta que sai do hotel 0 para qualquer outro vértice na primeira viagem será igual a 1, garantindo assim, que saia apenas uma aresta do hotel inicial na primeira jornada de trabalho. As restrições (2.17) garantem que o fluxo de entrada em um vértice i será sempre uma unidade menor que o fluxo de saída. A restrição (2.18) garante que se uma determinada aresta entre i e j não for utilizada, então o valor do fluxo naquela aresta será 0. As restrições (2.19) indicam o domínio das variáveis.

Capítulo 3

HEURÍSTICAS

Neste capítulo são apresentadas as diferentes estratégias heurísticas que foram aplicadas ao Problema do Caixeiro Viajante com Seleção de Hotéis e os procedimentos necessários para suas composições. Para cada estratégia, primeiramente são apresentados seus componentes essenciais e posteriormente, os algoritmos contendo todos os procedimentos. O conteúdo deste capítulo foi parcialmente publicado em Sousa & Gonçalves [2014].

Heurísticas são métodos aproximativos utilizados dentro do contexto dos métodos de otimização. As heurísticas possuem como vantagem a possibilidade de geração de soluções classificadas como de alta qualidade, em um tempo computacional viável para aplicações práticas [Talbi, 2009]. O uso de heurísticas para tratar problemas em otimização vem sendo há muitos anos uma alternativa aos métodos exatos. Relembrando, métodos exatos são aqueles que garantem encontrar o ótimo global, mas podem demandar uma grande quantidade de recursos computacionais e tempo para isto.

Entre as técnicas heurísticas, tem-se as heurísticas específicas e as metaheurísticas. Enquanto as heurísticas específicas são projetadas e adaptadas com intuito de resolver um dado problema de otimização, as metaheurísticas são estratégias de propósito geral que podem ser aplicadas na resolução de grande parte dos problemas de otimização [Talbi, 2009]. Metaheurísticas ainda podem ser definidas como um processo de geração iterativo que guia heurísticas específicas, combinando de forma inteligente diferentes conceitos para exploração do espaço de busca, escapando de ótimos locais de baixa qualidade [Osman & Laporte, 1996].

A aplicação de heurísticas e metaheurísticas para tratamento de problemas da área de otimização vem crescendo devido a sua capacidade de gerar soluções com qualidade aceitável para problemas que possuem instâncias consideravelmente

grandes e difíceis de serem tratadas.

3.1 Procedimentos Estruturais

Nesta seção são apresentadas as descrições sobre construção de uma solução inicial, estruturas de vizinhança utilizadas e estratégia de perturbação da solução.

3.1.1 Construção da Solução Inicial

Para que seja possível aplicar estratégias de melhoria de uma solução, com o objetivo de otimizá-la, é necessário primeiramente construir uma solução de partida. A forma como a solução inicial do problema é determinada pode impactar diretamente na qualidade e no tempo computacional necessário para chegar à solução final. Logo, definir um bom método para geração da solução inicial é essencial para obter bons resultados.

Heurísticas de construção aplicadas ao PCV podem ser classificadas como algoritmos que geram um circuito viável a partir de um conjunto inicial de vértices, e esse conjunto é modificado de tempos em tempos. Durante este processo, ocorre a busca por soluções que são consideradas boas, que necessitam de um tempo computacional razoável, porém, sem ser capaz de garantir otimalidade ou até, em vários casos, de estabelecer quão perto uma dada solução viável está da solução considerada ótima para tal problema [Benevides, 2011].

Existem diversas heurísticas de construção da solução inicial que são aplicáveis a problemas de roteamento e que podem ser extensíveis ao PCVSH. Utilizar heurísticas que são aplicadas ao clássico Problema do Caixeiro Viajante, adaptadas para o PCVSH, podem levar a bons resultados se aplicadas ao PCVSH. Como exemplo, temos a heurística que será apresentada na seção a seguir.

A seguir, são apresentadas as duas heurísticas utilizadas para a construção de soluções para PCVSH.

3.1.1.1 Heurística de Lin-Kernighan com Dijkstra

A primeira estratégia de geração da solução inicial usada nesta dissertação foi proposta no trabalho de Castro et al. [2013], sendo uma composição de diferentes abordagens. Esta estratégia é considerada como pertencente ao grupo de técnicas para geração de soluções (*order-first split-second*) que primeiramente ordenam os clientes da solução e posteriormente realizam a divisão introduzindo hotéis conforme

necessário [Prins et al., 2014]. A primeira fase utiliza a heurística de Lin-Kernighan (LKH) [Lin & Kernighan, 1973] e a segunda fase o Algoritmo de Dijkstra [Dijkstra, 1959]. As operações primordiais que devem ser realizadas nesta estratégia são apresentadas no Algoritmo 1 e descritas com maiores detalhes a seguir.

Algoritmo 1: Algoritmo de geração da solução inicial (LKH + Dijkstra)

```

1 início
2   Gerar a rota com a heurística de Lin-Kernighan;
3   Definir vértices do grafo auxiliar  $G'$ ;
4   Criar grafo auxiliar  $G'$ ;
5   Aplicar Algoritmo de Dijkstra e obter o caminho mínimo;
6   Inserir os hotéis de acordo com a ordem definida pelo Algoritmo de
   Dijkstra;
7 fim

```

A primeira parte desta estratégia é determinada pela criação de uma solução para o Problema do Caixeiro Viajante. Esta solução inicia e termina no hotel (H_0) visitando todos os clientes, sem considerar a restrição do limite de tempo das viagens. Para construir a rota é utilizada a heurística de Lin-Kernighan da forma como foi implementada em Applegate et al. [2006a].

A heurística de Lin-Kernighan é considerada como uma das melhores estratégias conhecidas para gerar soluções ótimas ou próximas às ótimas quando se trata do PCV. No entanto, sua eficiência é proporcional ao trabalho necessário para realizar sua implementação, havendo várias maneiras diferentes de fazê-la. Logo, detalhes de sua implementação tem uma influência significativa no seu desempenho [Helsgaun, 2000].

A Heurística de Lin-Kernighan é um procedimento iterativo adaptativo que tem como base a heurística k -opt [Golden & Stewart, 1985] (ver Subseção 3.1.2.1 e Subseção 3.1.2.2). A cada iteração, de acordo com a avaliação da qualidade da solução corrente, o valor de k é redefinido. Caso a solução seja melhorada pelo processo de aplicação da heurística k -opt, a solução é redefinida e os novos subconjuntos de k arestas a serem testadas são também redefinidos. Este processo se repete até que todos os subconjuntos de k arestas sejam analisados e a aplicação deles na solução corrente não resulte em uma melhoria na solução.

Considerando a construção da rota s realizada pela heurística LKH, esta rota na maioria das vezes será inviável para o PCVSH, pois extrapola a restrição de tempo definido para uma viagem. Uma rota é considerada viável, se todas as viagens contidas nesta rota respeitam o limite L de tempo imposto a uma viagem. Para

tornar a rota gerada viável ao PCVSH, um procedimento definido como divisão da rota deve ser empregado. A divisão da rota gerada pela heurística LKH em uma rota viável é realizada por meio de um procedimento inspirado no processo de divisão apresentado em Prins [2004], que baseia-se no Algoritmo de Dijkstra. O método de divisão busca de forma otimizada particionar a rota por meio da inserção de hotéis intermediários, garantindo a viabilidade de cada viagem gerada.

Considerando os clientes ordenados de acordo com as regras estabelecidas pela heurística LKH, no procedimento de divisão [Castro et al., 2013] é construído um grafo auxiliar G' contendo $mn + 1$ vértices, com $m = |H|$ e $n = |C|$. O primeiro nó do grafo representa o hotel inicial e é definido com o índice 1. O restante dos vértices do grafo representam uma combinação de hotéis e clientes. Para representar a visita a um hotel p que foi precedido pela visita ao i -ésimo cliente s_i , é utilizada a notação i^p . O nó final do grafo é definido como $V^{(mn+1)-(m-1)}$, e determina o hotel final da rota, depois que a visitação a todos os clientes foi realizada. Uma aresta (i^p, j^q) é adicionada ao grafo se uma viagem partindo do hotel p , visitando os clientes de s_{i+1} até s_j e chegando ao hotel q , for viável. Por fim, o peso da aresta (i^p, j^q) é caracterizado pela soma dos tempos necessários para o caixeiro viajar de i^p para j^q , passando pelos clientes de s_{i+1} até s_j , somados os tempos de visita de cada cliente. Para cada aresta inserida no grafo, foi acrescido ao seu peso o valor de uma constante (γ) que deve ser um valor relativamente grande, definido de acordo com as instâncias a serem testadas e que contribuirá no momento de minimização do número de viagens. O peso destas arestas é expresso pela Equação 3.1.

O resultado retornado pela equação é composto pelo tempo necessário para percorrer o trajeto entre o hotel p e o cliente s_{i+1} , somando-se o tempo de viagem entre os clientes s_v e s_{v+1} , mais o tempo de visita associado a cada cliente s_v . Neste caso, a variável v assume os valores de $(i + 1)$ à $(j - 1)$. Por fim, é acrescentado o tempo de visita do cliente s_j , o tempo necessário para o caixeiro se deslocar do cliente s_j até o hotel q e o valor da constante γ . A soma total representa o peso associado a aresta (i^p, j^q) .

$$c_{ps_{i+1}} + \sum_{v=i+1}^{j-1} (c_{s_v s_{v+1}} + \tau_{s_v}) + \tau_{s_j} + c_{s_j q} + \gamma \quad (3.1)$$

Uma observação importante que deve ser levada em consideração é que há a possibilidade de serem criadas viagens que não possuem nenhum cliente, neste caso o hotel p deve ser obrigatoriamente diferente do hotel q . Essas viagens sem clientes

podem ser úteis para fazer ligações entre áreas que não possuem clientes próximos aos hotéis.

No exemplo da Figura 3.1, tem-se as variáveis $m = 5$ e $n = 2$. Cada vértice do grafo corresponde a um respectivo hotel que foi precedido pelo cliente i , da forma em que aparece na figura. Cada aresta contida no grafo representa a saída de um dado hotel, visita a uma sequência de clientes e a parada em outro hotel. Somente os caminhos que não excedem o limite L receberam arestas no grafo. Ao sair do vértice V_1 que representa H_0 , não é possível chegar diretamente ao vértice de destino V_{10} , sendo necessária a realização de uma parada no vértice V_7 que representa o hotel H_1 , precedido pelo cliente 1. Caso houver mais de um caminho entre a origem e o destino, o melhor caminho deve ser escolhido, conforme descrito a seguir. Para este exemplo, o caminho no grafo que gera o menor tempo total é (V^1, V^7, V^{10}) , com V^1 a $V^7 = (H_0, C_2, C_4, C_1, H_1)$ e V^7 a $V^{10} = (H_1, C_5, C_3, H_0)$.

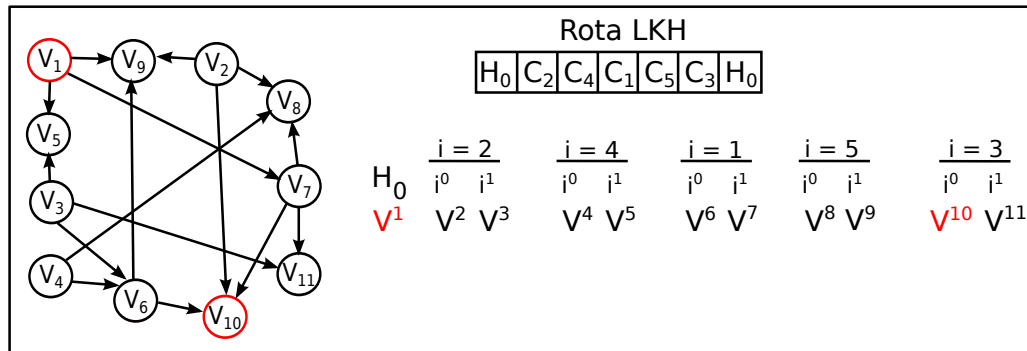


Figura 3.1. Exemplo do grafo criado para definição do melhor caminho.

Após a construção do grafo auxiliar da forma apresentada acima, o conhecido Algoritmo de Dijkstra [Dijkstra, 1959] é utilizado para encontrar o caminho mínimo entre os hotéis inicial e final. Como foi inserido no peso de cada aresta a constante γ , o caminho definido pelo Algoritmo de Dijkstra prioriza a utilização do menor número de viagens necessárias para atender a todos os clientes. Desta forma, é realizado simultaneamente a minimização do número de arestas utilizadas (correspondem ao número de viagens) e o tempo total viajado.

3.1.1.2 Heurística de Inserção Mais Barata Randomizada

A segunda estratégia utilizada para construir uma solução inicial para o PCVSH é conhecida como Heurística de Inserção Mais Barata (HIMBR). A HIMBR é bastante difundida para problemas que envolvem a geração de um ciclo [Benevides, 2011; Silva et al., 2013; Mine et al., 2010]. Devido ao fator de aleatoriedade

incluído no processo, diferentemente da heurística de Lin-Kernighan que constrói uma solução determinística, a solução criada nesta heurística é classificada como não determinística.

O Algoritmo 2 descreve o processo de funcionamento da HIMBR. Basicamente, consiste na geração de uma rota para o PCV que contém o hotel H_0 no início e no fim. Para deixar a estratégia não determinística, os dois primeiros clientes serão sempre escolhidos aleatoriamente. Para o restante dos clientes, a cada iteração, um cliente é sorteado e sua inserção é testada em cada uma das posições possíveis. O cliente é então inserido na posição em que representa o menor acréscimo ao tempo total da rota. Ao final, todos os clientes foram inseridos na rota que deve ser particionada em viagens viáveis, de forma a não extrapolar o limite de tempo L .

Algoritmo 2: Algoritmo de geração da solução inicial HIMBR.

```

1 início
2    $lista\_clientes \leftarrow$  todos os clientes da instância;
3    $rota \leftarrow$  aleatoriamente dois clientes da  $lista\_clientes$ ;
4   remover clientes sorteados da  $lista\_clientes$ ;
5   enquanto ( $lista\_clientes$ )  $\neq \emptyset$  faça
6      $cli \leftarrow$  aleatoriamente um cliente da  $lista\_clientes$ ;
7     para (cada posição  $i$  entre dois elementos da rota) faça
8       se inserção de  $cli$  é mais barata então
9          $posicao \leftarrow i$ ;
10      fim
11    fim
12    inserir  $cli$  na rota de acordo com a variável  $posicao$ ;
13    remover  $cli$  da  $lista\_clientes$ ;
14  fim
15  retornar  $rota$ ;
16 fim

```

Para que a rota possa ser particionada utilizando o Algoritmo de Dijkstra, esta deve ser de boa qualidade, ou seja, deve prover ao menos um caminho possível entre a origem e o destino, levando em consideração a visitação à todos os clientes. Ao utilizar a HIMBR este requisito pode não ser alcançado. Para que não haja casos de insucesso na divisão, a estratégia de melhoramento da solução por meio da estrutura de vizinhança 3-OPT (ver Subseção 3.1.2.2) é utilizada.

Após finalizar os procedimentos descritos acima, a solução inicial do PCVSH define um limite superior para o tempo total da solução final. Assim, espera-se que ao aplicar estratégias como estruturas de vizinhança, seja possível otimizar o

posicionamento dos clientes e hotéis na solução, melhorando significativamente a qualidade da solução final.

3.1.2 Estruturas de Vizinhança

Nesta subseção serão apresentadas as diferentes estruturas de vizinhança que foram utilizadas no desenvolvimento desta dissertação. As estruturas de vizinhança são estratégias utilizadas para realização de movimentos de melhora nas soluções obtidas em cada heurística construtiva. Estas estruturas podem ser classificadas de duas formas distintas. A primeira forma de classificação é definida como estruturas *intra-trips*, que abrangem as estruturas de vizinhança que operam somente sobre os clientes contidos dentro das viagens, ou seja, sobre o subconjunto de clientes compreendidos entre dois hotéis consecutivos. Já, as estruturas *inter-trips* realizam operações considerando o conjunto de todas as viagens que compõem a rota do caixeiro. A Figura 3.2 ilustra os diferentes escopos que uma estrutura de vizinhança pode abranger.

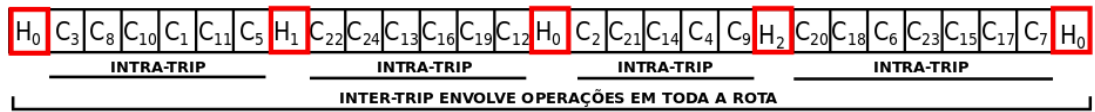


Figura 3.2. Exemplo de escopo de aplicação de uma estrutura *inter-trip* e *intra-trip*

A seguir, as estruturas de vizinhança serão detalhadas conforme seu funcionamento e classificadas de acordo com sua utilização, conforme apresentado acima.

As estruturas que serão detalhadas nesta subseção compreendem 2-OPT, 3-OPT, *Relocate*, *Exchange*, *Jointrips*, *Changehotels* e perturbação (*Swap*) envolvendo clientes. Todas estas estruturas foram utilizadas no desenvolvimento desta dissertação devido sua recorrente utilização para tratar o Problema do Caixeiro Viajante com Seleção de Hotéis [Vansteenwegen et al., 2012; Castro et al., 2012, 2013, 2014].

3.1.2.1 2-OPT

A estrutura de vizinhança 2-OPT foi originalmente proposta por Croes [1958] com o objetivo de tratar o problema do caixeiro viajante. Apesar de ser simples, possui capacidade de otimizar uma solução do PCVSH e de diversos outros problemas na área de otimização.

A ideia de funcionamento desta estrutura é bastante simples e consiste na realização da remoção de duas arestas não consecutivas de uma solução, reconectando-as

de maneira distinta. A Figura 3.3 ilustra como é realizada a operação. Ao retirar as arestas entre os elementos (K_i, K_{i+1}) e (L_j, L_{j+1}) , devem ser geradas duas novas arestas que ligam (K_i, L_j) e (K_{i+1}, L_{j+1}) .

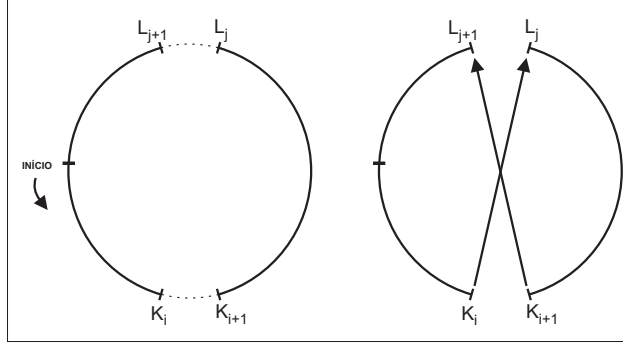


Figura 3.3. Exemplo de reconexão dos caminhos pela estrutura 2-OPT.

Se a nova forma de conexão dos vértices for melhor que a anterior, de maneira que o tempo total da rota seja diminuído, então, a configuração do movimento testado é armazenada. Prossiga testando as configurações remanescentes até que sejam testadas todas as possibilidades. Desta forma, é realizado um teste completo que vai comparar cada combinação válida de movimentos. Ao final dos testes, caso haja uma configuração que otimiza a ordem de visita dos elementos, o melhor movimento sobre a rota original para produzir a rota melhorada é aplicado.

A estrutura 2-OPT pode ser aplicada tanto em viagens (*intra-trip*), quanto em uma rota (*inter-trip*).

3.1.2.2 3-OPT

A estrutura 3-OPT [Lin, 1965] foi desenvolvida com o intuito de tratar o PCV. Para instâncias com pequeno número de elementos, a estrutura 3-OPT é capaz de obter soluções próximas a ótima e até mesmo a solução ótima em alguns casos. À medida que é aumentado o número de elementos na instância, a estrutura necessita de tempo computacional proporcional ao crescimento da instância [Garey & Johnson, 1979].

A Figura 3.4 ilustra as diferentes formas de reconectar a rota (b, c, d, e) após a remoção de três arestas. Tomando como exemplo a rota (a) apresentada na figura, foram escolhidos três arestas para remoção. As arestas são representadas entre os clientes (K_i, K_{i+1}) , (L_j, L_{j+1}) e (M_p, M_{p+1}) . No caso da reconexão (b), seriam definidas novas arestas entre os clientes (K_i, L_{j+1}) , (M_p, L_j) e (K_{i+1}, M_{p+1}) . Para todas as combinações possíveis de remoção das três arestas, serão testadas

as quatro configurações de movimentos para reconexão. Após realizados todos os testes, apenas o movimento que melhor otimizar a rota (melhor aprimorante), ou seja, o que mais diminuir o tempo total da rota será aplicado. Caso não haja nenhum movimento que represente melhora, a rota será mantida na forma original.

Semelhante à 2-OPT, distingue-se apenas na quantidade de arestas que são removidas, e consequentemente, na quantidade de possíveis combinações de reconexão.

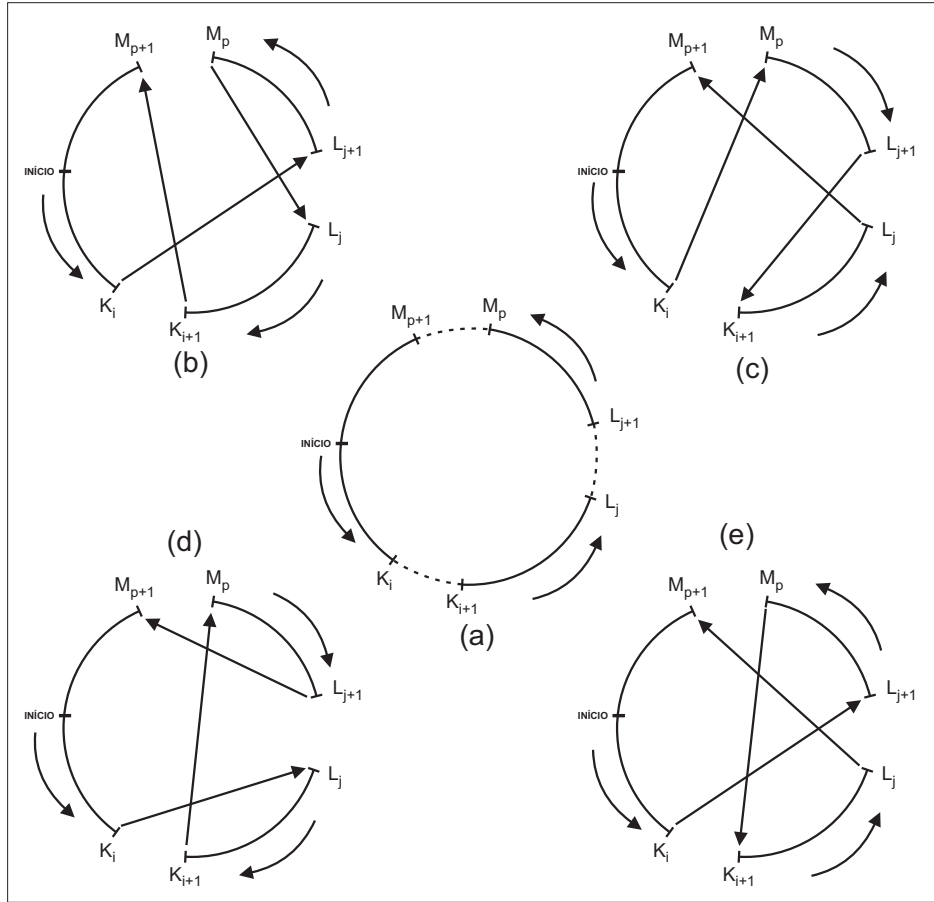


Figura 3.4. Exemplo de reconexão dos caminhos pela estrutura 3-OPT.

Devido a sua simplicidade de implementação e tempo computacional baixo, a estrutura foi utilizada tanto no escopo *inter-trip*, como também no *intra-trip*.

3.1.2.3 Or-OPT

A estrutura Or-OPT [Or, 1976] possui como característica o uso de movimentos que realocam uma cadeia de k clientes dentro de uma viagem. Para cada viagem, a estrutura deve ser aplicada seguindo a sequência expressa no conjunto $\{3, 2, 1\}$ e

após testadas todas as possibilidades, apenas o movimento que contribuir mais para a melhora da solução será aplicado, ou seja, o melhor aprimorante. A Figura 3.5 ilustra as possibilidades de realocar os três ($k = 3$) primeiros clientes internamente à uma viagem, por meio do uso da estrutura Or-OPT.

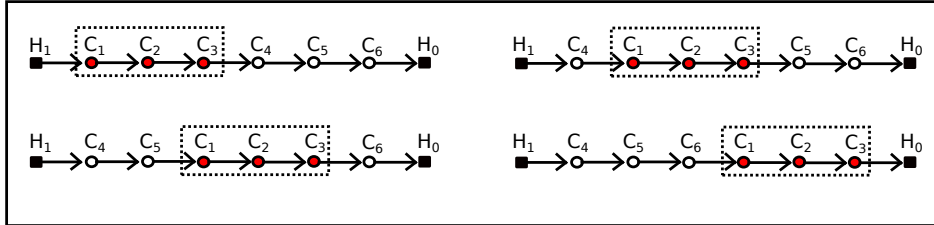


Figura 3.5. Movimentos para realocar os 3 primeiros clientes da viagem.

Nesta dissertação, o valor da variável K para a estrutura Or-OPT foi definido pelo conjunto $\{3, 2, 1\}$. Como esta estrutura foi aplicada apenas no âmbito de uma viagem, logo é classificada como uma estrutura *intra-trip*. As demais estruturas de vizinhança que serão apresentadas a seguir são todas voltadas para a aplicação na rota, logo, classificadas como *inter-trip*.

3.1.2.4 Relocate

Continuando com estratégias que buscam a realocação dos clientes para outras posições na solução, temos a estrutura *Relocate* [Laporte et al., 2000]. A estrutura *Relocate* apresenta grande semelhança com relação a estrutura Or-OPT. A única diferença entre as duas é o fato do *Relocate* operar no âmbito da rota e o Or-OPT apenas no escopo da viagem. Assim, o *Relocate* realiza a realocação de uma determinada cadeia de clientes de tamanho k , onde $k \in \{3, 2, 1\}$. O procedimento inicia-se com a tentativa de realocar $k = 3$ clientes de uma viagem para todas as outras viagens, em todas as posições possíveis. O procedimento encerra-se após serem testadas todas as combinações de realocação de k clientes entre todas as viagens, seguindo a ordem de $k = 3$, $k = 2$ e $k = 1$. Somente o movimento de realocação que gerar a maior melhora no resultado da solução será aplicado. Caso não haja nenhum movimento que caracterize melhora na solução, a solução de entrada do procedimento permanecerá inalterada após a aplicação da estrutura. A Figura 3.6 ilustra o processo de aplicação da estrutura utilizando um exemplo de solução para o PCVSH. Neste exemplo o cliente C_3 foi removido da primeira viagem e adicionado logo após o hotel H_1 , na segunda viagem.

Por atuar realizando movimentos que envolvem viagens distintas, esta estrutura é classificada como uma estrutura *inter-trip*. Uma observação importante sobre

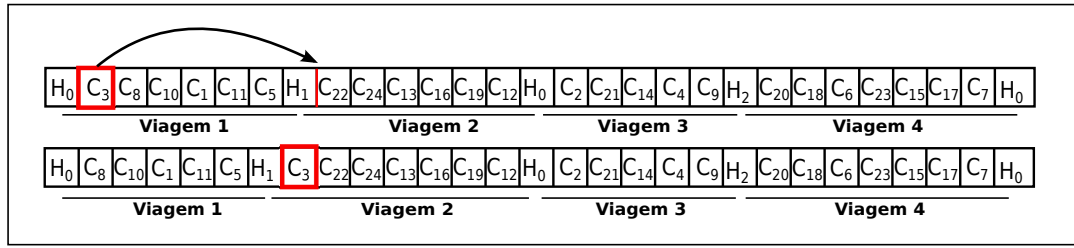


Figura 3.6. Realocação de um cliente da viagem 1 para viagem 2.

a estrutura é que o valor de k pode ser alterado para um número maior, porém quanto maior o valor de k , maiores são as chances dos movimentos gerarem soluções inviáveis.

3.1.2.5 Exchange

O simples movimento de realocação de um conjunto de clientes de uma viagem para outra oferece melhoras significativas na solução explorada, logo estruturas que realizam movimentos similares devem ser consideradas. Outra forma de realocar grupos de clientes entre duas viagens distintas, é definida nesta dissertação como *Exchange* [Laporte et al., 2000]. Esta estrutura de vizinhança consiste na realização de trocas considerando uma cadeia de clientes de tamanho k , onde $k \in \{3, 2, 1\}$. Os valores de k são aplicados sempre em ordem decrescente. Para cada possível cadeia de clientes de tamanho k contida nas viagens que compõem a rota, é realizada uma simulação da troca das cadeias entre as viagens. Para que seja concretizada a troca das cadeias de clientes, apenas o movimento que representar a maior diminuição do tempo total da rota será aplicado.

Uma consideração importante sobre esta estrutura é que se a rota contiver apenas viagens viáveis (obedecendo o valor limite L), movimentos que tornem a rota inviável não serão aceitos. Porém, caso a rota tenha alguma viagem inviável, será aceito um movimento que mantenha a inviabilidade, desde que o tempo total da rota seja diminuído. A estrutura *Exchange*, bem como as anteriores foram aplicadas nos trabalhos de Castro et al. [2013, 2014] e obtiveram bons resultados para o PCVSH.

A Figura 3.7 ilustra o processo de troca de uma cadeia de clientes ($k = 1$) entre duas viagens. Considerando as soluções definidas pelos vetores, onde os hotéis são representados por H_i com $i \in \{0, 1, 2\}$ e os clientes representados por C_j com $j \in \{1, \dots, 24\}$, uma possibilidade de movimento seria trocar o cliente C_3 que está na primeira viagem, pelo cliente C_{22} que está na segunda viagem. Caso este movimento seja o melhor encontrado após todos os testes, a solução é modificada da forma como

é exposta no segundo vetor da Figura 3.7.

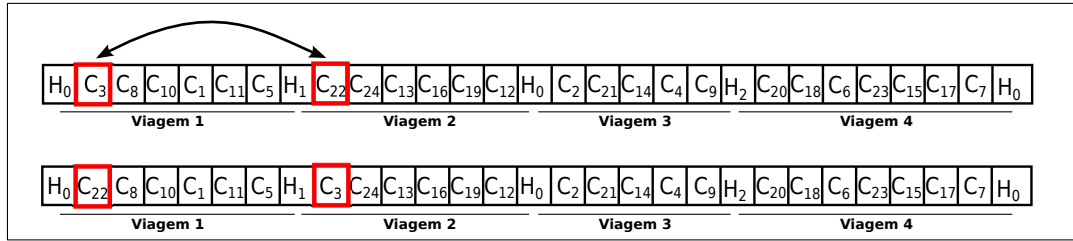


Figura 3.7. Troca de um cliente entre a viagem 1 e a viagem 2.

Esta estrutura de vizinhança faz parte do conjunto de estruturas que envolvem troca de informações entre duas viagens distintas, caracterizando-a assim como uma estrutura *inter-trip*. A seguir, são apresentadas duas estruturas que diferentemente das abordadas até aqui, realizam modificações no conjunto de hotéis da solução.

3.1.2.6 ChangeHotels

A estrutura de vizinhança *ChangeHotels*, diferentemente das demais estruturas apresentadas anteriormente, não é classificada como *inter-trip* ou *intra-trip*. Esta estrutura é tida como um operador de seleção de hotéis. Isto pois, trabalha exclusivamente realizando a alteração dos hotéis intermediários da rota.

A estrutura *ChangeHotels* realiza basicamente a troca de cada um dos hotéis intermediários por outro hotel do conjunto de hotéis disponíveis, caso a troca represente uma melhora na solução. Considerando que esteja disponível o seguinte conjunto de hotéis $H\{H_0, H_1, H_2, H_3\}$. Caso haja, em alguma posição intermediária da rota, o hotel H_1 , logo deve ser simulada a troca deste hotel por cada um dos outros que compõem o conjunto H . Um hotel somente será considerado apto para a troca se a aplicação desta troca não tornar a rota inviável e, ao mesmo tempo diminuir o tempo total da mesma. Quando a rota inicial é inviável, um hotel será considerado apto se diminuir o tempo total da rota.

Um exemplo gráfico é ilustrado na Figura 3.8, onde os hotéis intermediários H_2 e H_3 podem ser substituídos por outro hotel do conjunto, com o intuito de melhorar a solução. Como o hotel inicial e final são sempre definidos como H_0 , estes não são submetidos ao processo de busca.

3.1.2.7 Jointrips

A segunda estrutura utilizada para modificações dos hotéis de uma rota é a estrutura *JoinTrips*. Nesta estrutura, o objetivo é retirar um hotel intermediário

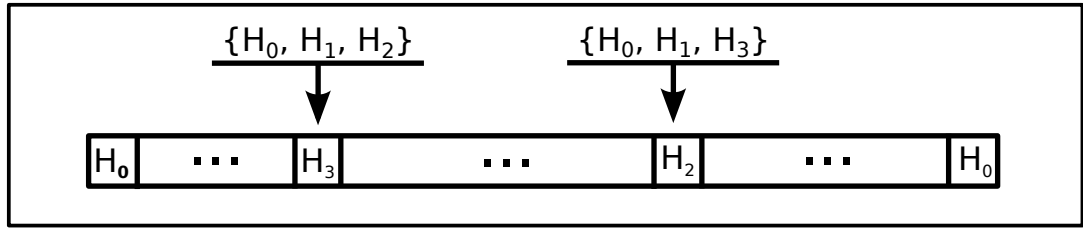


Figura 3.8. Exemplo de possibilidade da troca dos hotéis utilizando a estrutura *ChangeHotels*.

que realiza a conexão entre duas viagens e, conseqüentemente, diminuir o número de viagens da rota. Considerando um conjunto $V = \{V_1, V_2, V_3, V_4, V_5, V_6\}$ de viagens, a remoção de um hotel pode ser realizada de duas formas. A primeira tenta unir duas viagens consecutivas (Figura 3.9) e testa a remoção dos hotéis intermediários. Caso a remoção de um hotel intermediário não torne a nova viagem inviável, então o hotel poderá ser removido com sucesso.

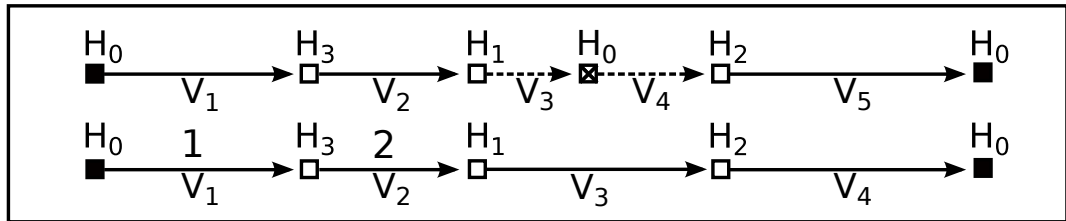


Figura 3.9. Exemplo de remoção de um hotel intermediário da rota.

A segunda forma para realizar a remoção de hotéis é ilustrada na Figura 3.10, onde a rota é composta por seis viagens, enumeradas em ordem crescente. Neste exemplo, como um dos hotéis intermediários é o mesmo hotel definido como origem da rota, então é possível retirar este hotel intermediário e fazer uma inversão na visita dos demais clientes e hotéis da rota verificando se a viabilidade é mantida. Esta segunda forma somente pode ser aplicada quando existem hotéis que se repetem no decorrer da rota. A remoção do hotel somente poderá ser concretizada se a operação não tornar a nova viagem, fruto da junção da viagem que precede e sucede o hotel, inviável. No exemplo da Figura 3.10, a viagem V_3^* consiste na junção da viagem V_3 e V_6 , sendo esta última percorrida no sentido inverso ao original. As outras viagens V_4^* e V_5^* representam basicamente a inversão no sentido de visita dos clientes e hotéis, quando comparadas com as viagens originais (V_4 e V_5).

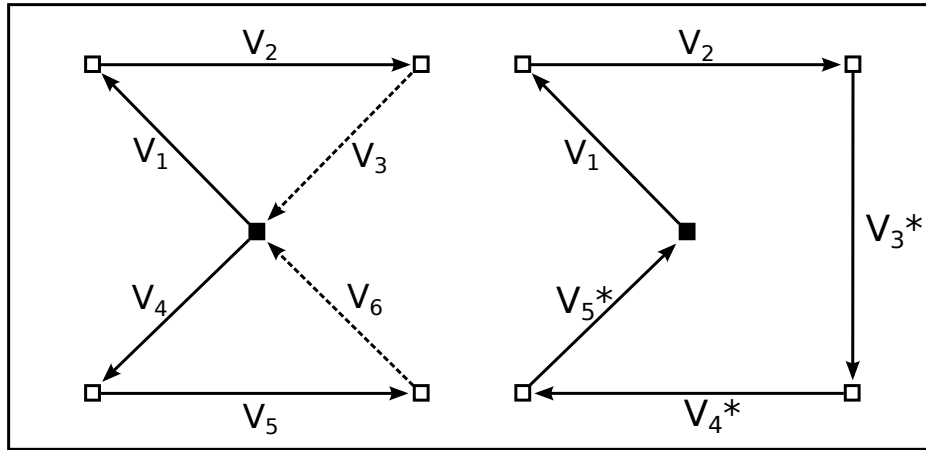


Figura 3.10. Exemplo de remoção de um hotel intermediário e inversão do restante da rota.

3.1.3 Perturbação da Solução

Ao aplicar estruturas de vizinhança a uma solução para determinado problema na área de otimização, objetiva-se obter uma solução que seja melhor que a atual. Porém, há casos em que, mesmo aplicando diferentes estruturas de vizinhança não é possível melhorar a solução. Quando isto ocorre, diz-se que a solução encontra-se num ótimo local. Encontrar uma maneira para explorar novos espaços de busca, pode levar ao encontro do ótimo global, ou seja, a melhor solução possível para o problema em questão.

A Figura 3.11 ilustra o espaço de busca e a representação de diferentes soluções por meio da curva plotada. São observadas três soluções que podem ser classificadas como ótimos locais e a melhor destas definida como ótimo global. O objetivo principal de se aplicar métodos de perturbação da solução é fazer com que o algoritmo escape de ótimos locais e, explorando uma nova região do espaço de busca consiga chegar ao ótimo global. A aplicação da perturbação precisa levar em consideração que o percentual de perturbação deve ser equilibrado, de tal forma que, se for muito pequeno não será suficiente para retirar a solução de um ótimo local e se for muito grande, a solução sofrerá tanta modificação que será caracterizada como um reinício aleatório.

3.1.3.1 Perturbação da Solução Alterando os Clientes

Nesta dissertação, foi utilizado um método de perturbação simples e que apresentou-se eficiente para o PCVSH [Castro et al., 2014]. Este método consiste na perturbação da solução considerando a mudança apenas dos clientes. A pertur-

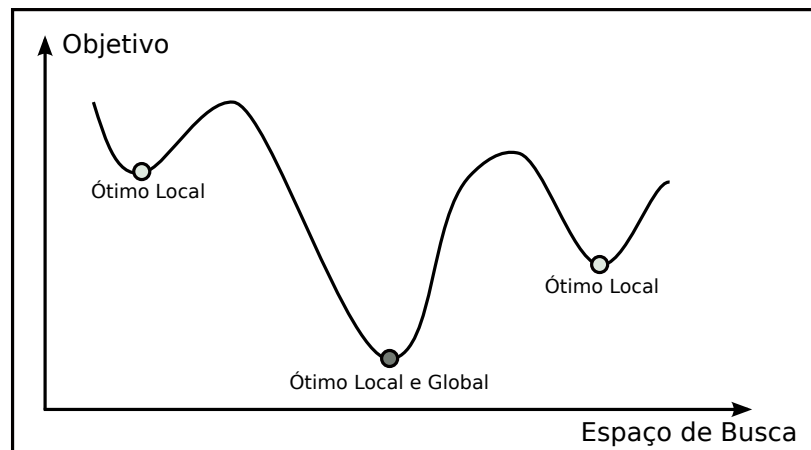


Figura 3.11. Representação da solução em diferentes otimalidades. Adaptado de Talbi [2009].

bação consiste na alteração da posição dos clientes na rota. O primeiro passo para sua aplicação é a definição de um parâmetro teta (θ) que indica quantos clientes da solução devem ter sua posição alterada na rota. O próximo passo é escolher de forma aleatória quais clientes terão suas posições alteradas e quais serão suas novas posições.

3.2 Heurísticas Básicas

Nesta seção são apresentadas as duas heurísticas básicas que foram utilizadas no desenvolvimento desta dissertação. As heurísticas utilizadas foram baseadas nos trabalhos de Castro et al. [2013] e Castro et al. [2014].

3.2.1 Busca Tabu

A metaheurística Busca Tabu (do inglês *Tabu Search* - BT), proposta por Glover [1989, 1990], é uma metaheurística que utiliza estruturas de vizinhança de forma adaptativa para explorar um espaço de soluções. Uma característica particular da BT é a utilização de memória para armazenar informações sobre o processo de busca, evitando retornar a soluções já visitadas [Talbi, 2009]. Formalmente, a estratégia parte de uma solução inicial s_0 e, a cada iteração, a vizinhança N_s da solução corrente s é explorada. A solução s' pertencente à vizinhança com o melhor valor da função objetivo torna-se a nova solução corrente, mesmo que s' seja pior que s .

Sem um mecanismo que evite retornar à mesma sequência de soluções já geradas em iterações passadas, o algoritmo pode percorrer soluções ciclicamente. Para evitar que este tipo de problema ocorra, existe uma lista Λ denominada tabu. A lista tabu armazena os últimos movimentos realizados que são considerados proibidos. Esta lista funciona como uma fila de tamanho fixo, de forma que, quando um novo movimento é adicionado o movimento mais antigo é retirado. Apesar da lista tabu fornecer um meio para evitar ciclagem, ela também pode proibir movimentos que caracterizariam melhora na solução. Para isto, existe a função de aspiração que é um mecanismo que retira, sob certas circunstâncias, um movimento da lista tabu.

A BT utilizada nesta dissertação foi proposta no trabalho de Castro et al. [2013] e escolhida por apresentar bons resultados no tratamento de problemas de roteamento de veículos [Archetti et al., 2006; Gendreau et al., 2006].

Neste trabalho, com o objetivo de aceitar soluções que diminuam a inviabilidade da solução inicial, foi utilizada uma função que atua penalizando de forma ponderada as soluções que excedem o limite L de tempo total de uma viagem. Esta função é definida pela Equação 3.2 e pode ser subdividida em duas parcelas, sendo que a primeira representa o tempo total necessário para percorrer a rota, desconsiderando-se os tempos de visita aos clientes. A segunda parte representa a soma do tempo que cada viagem excede o limite de tempo L para cada uma das viagens. O objetivo é minimizar o valor de $F(S)$.

$$F(S) = \sum_{d=1}^D \sum_{(i,j) \in A} c_{ij} x_{ij}^d + \varphi \sum_{d=1}^D \left[\sum_{(i,j) \in A} (c_{ij} + \tau_j) x_{ij}^d - L \right]^+ \quad (3.2)$$

Na Equação 3.2, a variável binária x_{ij}^d assume o valor 1 se o caminho de um cliente/hotel para outro é realizado, e 0 caso contrário. A constante τ_j representa o tempo gasto para se visitar um cliente j , este custo é contabilizado no tempo total da viagem. A penalização é aplicada se em alguma das viagens d do total D , for excedido o limite de tempo L ($[z]^+ = \max(z, 0)$). Desta forma, quanto maior for a inviabilidade, maior vai ser a penalização e pior será considerada a solução. No Algoritmo 3 é apresentada a estrutura da Busca Tabu utilizada.

Na implementação da BT, a variável S' representa a melhor solução viável encontrada e a variável S^* representa a solução que contém o melhor valor da função ponderada $F(S)$. As estruturas de vizinhança utilizadas são o *relocate*, *exchange*, *jointrips* e 3-OPT (ver Seção 3.1.2). A lista definida por Γ representa a lista de movimentos candidatos que podem vir a ser executados em uma dada iteração e

Algoritmo 3: Busca Tabu

Entrada: $(S, Imax, \theta^-, \theta^+, U_p, U_c)$

```

1 início
2   inicializar:  $time = 0, \Lambda = \emptyset, S^* = S;$ 
3   se  $S$  é viável então
4      $S' = S;$ 
5   fim
6   enquanto critério de parada não encontrado faça
7      $time++;$ 
8     resetar lista de candidatos  $\Gamma = \emptyset;$ 
9     para cada possível movimento  $\mu \in N(S)$  faça
10      adicionar  $\mu$  à  $\Gamma$  de acordo condições tabu e aspiração;
11    fim
12    selecionar o melhor movimento  $\mu_{best} = \operatorname{argmin}_{\mu \in \Gamma} F(S \oplus \mu);$ 
13    atualizar solução  $S = S \oplus \mu_{best};$ 
14    atualizar lista tabu  $\Lambda;$ 
15    se  $time \bmod U_c = 0$  então
16       $3 - OPT(S);$ 
17    fim
18    jointrips( $S$ );
19    atualizar  $S^*$  e  $S'$  se aplicável;
20    se  $time \bmod U_p = 0$  então
21      atualizar o fator de penalização  $\varphi;$ 
22    fim
23  fim
24 fim

```

a lista Λ contém os movimentos considerados tabu, ou seja, que não podem ser executados em uma dada iteração a menos que satisfaçam o critério de aspiração. Os dois grupos citados anteriormente, contém todos os possíveis movimentos a serem explorados pelo *relocate* e *exchange* em uma dada iteração.

Para o perfeito funcionamento do Algoritmo que implementa a BT, cinco parâmetros são necessários: $Imax$ (número máximo de iterações sem melhora), θ^- (número mínimo de iterações que um movimento é considerado tabu), θ^+ (número máximo de iterações que um movimento é considerado tabu), U_c (periodicidade de aplicação da otimização da rota) e U_p (periodicidade de atualização do fator de penalização). A BT finaliza sua execução se durante as últimas $Imax$ iterações nenhuma solução melhor que S^* ou S' for encontrada. A cada iteração do algoritmo a lista tabu (Λ) deve ser atualizada de acordo com os movimentos realizados. Um movimento é considerado tabu durante θ^{med} iterações, onde este valor é um inteiro definido aleatoriamente no intervalo $[\theta^-, \theta^+]$. Desta forma, o tamanho da lista tabu

é alterado a cada chamada da Busca Tabu.

Se um movimento é considerado tabu, então ele fica proibido de ser executado durante θ^{med} iterações. Porém, há uma exceção que pode desfazer a proibição de um movimento, ou seja, o critério de aspiração. O critério de aspiração aqui definido diz que, se um movimento proibido levar a uma solução que é melhor que S' ou a uma solução que é melhor que S^* de acordo com a Equação 3.2, então o movimento é adicionado a lista de candidatos Γ e automaticamente excluído da lista Λ .

O algoritmo realiza periodicamente duas ações adicionais para otimizar a rota e forçar a busca tabu a aceitar apenas soluções viáveis ou que diminuam a inviabilidade. A periodicidade é definida pelos parâmetros U_p e U_c . Como pode ser observado na linha 15 do Algoritmo 3, o primeiro define que, a cada U_p iterações, a solução deve ser otimizada com a estrutura 3-OPT. Na sequência, o operador *join-trips* realiza uma tentativa de remoção de algum dos hotéis internos da rota (linha 18). O segundo parâmetro que considera um número de iterações é o U_c e, tem como funcionalidade a atualização do fator de penalização φ de acordo com o trabalho de Gendreau et al. [1994]. Basicamente, a cada U_c iterações é analisado se nas últimas U_c iterações todas as soluções foram viáveis. Se sim, o fator de penalização é reduzido por um fator de 2, caso contrário, é incrementado a um fator de 2 (linha 21).

3.2.2 Descida de Vizinhança Variável

A heurística conhecida como Descida de Vizinhança Variável (do inglês *Variable Neighborhood Descent* - VND) foi proposta originalmente por Mladenović & Hansen [1997] e consiste em um método de refinamento que explora o espaço de soluções realizando trocas entre diferentes estruturas de vizinhança $N = \{N_1, N_2, \dots, N_r\}$. Apenas soluções de melhora da solução corrente são aceitas e a cada melhora, o procedimento é reiniciado aplicando-se a primeira estrutura de vizinhança. Sua ideia é baseada na premissa que um ótimo local utilizando uma estrutura de vizinhança não é necessariamente o mesmo, quando utilizada outra estrutura de vizinhança [Castro et al., 2014].

A heurística utilizada nesta dissertação foi proposta no trabalho de Castro et al. [2014]. Como em alguns casos uma solução pode ser inviável para o Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH), ao utilizar tal solução o VND deve ser capaz não somente de aceitar soluções viáveis a partir da solução inicial, mas também de melhorar uma solução inviável com o intuito de torná-la viável. Para isto, a Equação 3.3 viabiliza o aceite de soluções que diminuem a inviabilidade

de uma solução. Dada uma solução de entrada y , a heurística VND tenta minimizar a seguinte função $F(y)$.

$$F(y) = \sum_{d=1}^D (M + tempo_d) + \varphi \sum_{d=1}^D \max(tempo_d - L, 0) \quad (3.3)$$

A Equação 3.3 apresenta duas parcelas bem definidas. A primeira é responsável por somar os tempos necessários para percorrer cada viagem, acrescentando-se para cada viagem o valor de uma constante M com valor suficientemente grande para priorizar soluções que contenham um número menor de viagens. A segunda realiza a soma dos tempos que excedem o limite L para cada viagem. Esta soma é multiplicada pela constante φ , fixada com um valor maior que a melhor solução conhecida na literatura. Desta forma, a equação busca a minimização do número de viagens, bem como a inviabilidade.

O VND proposto para o Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH) considera quatro estruturas de vizinhanças N_k , $k = 1, \dots, 4$, que são aplicadas na seguinte ordem: *relocate*, *exchange*, *changehotels* e *jointrips*. A estrutura do algoritmo VND é apresentada no Algoritmo 4 e consiste basicamente em um laço de repetição que é executado enquanto pelo menos uma das estruturas de vizinhança estiverem atuando positivamente sobre a solução fornecida.

Algoritmo 4: Algoritmo VND

Entrada: (y)

```

1 início
2    $k \leftarrow 1$ ;
3   enquanto ( $k < 4$ ) faça
4      $y' \leftarrow \text{BUSCA}(N_k, y)$ ;
5     se ( $y'$  melhor que  $y$ ) então
6        $y \leftarrow y'$ ;
7        $k \leftarrow 1$ ;
8     fim
9     senão
10       $k \leftarrow k + 1$ ;
11    fim
12  fim
13  Retornar  $y$ ;
14 fim
```

É importante observar que internamente o VND realiza a chamada para o Algoritmo 5 definido como BUSCA, é neste algoritmo que estão as chamadas para

as estruturas de vizinhança. Na linha 4 do Algoritmo 5 é analisada a k -ésima vizinhança da solução y , considerando a função descrita na Equação 3.3. Toda vez que uma solução melhor é encontrada, a nova solução é otimizada pelos operadores intra-trip 2-OPT e OR-OPT. A busca é repetida até que nenhuma solução melhor seja encontrada.

Algoritmo 5: Algoritmo BUSCA

Entrada: (N_k, y)

```

1 início
2    $y' \leftarrow y$ ;
3   repita
4      $y'' \leftarrow \operatorname{argmin}_{S \in N_k(y')} F(S)$ ;
5     se  $(y'' \text{ melhor que } y')$  então
6        $y' \leftarrow \text{otimizar } y'' \text{ com 2-OPT/OR-OPT}$ ;
7     fim
8   até que não seja encontrada mais melhora;
9   Retornar  $y'$ ;
10 fim
```

Nesta dissertação, também foi utilizada uma variante da heurística VND que é denominada RVND (do inglês *Random Variable Neighborhood Descent*). Na variante RNVD, a ordem de execução das estruturas de vizinhança a serem aplicadas à solução é definida de forma aleatória a cada iteração do algoritmo. Esta estratégia vem apresentando bons resultados para problemas de roteamento [Subramanian et al., 2010; Penna et al., 2013].

3.3 Heurísticas Híbridas

Nesta seção são apresentadas as heurísticas híbridas que foram consideradas nesta dissertação. As Heurísticas Híbridas são caracterizadas pela combinação de estratégias distintas de otimização, com o intuito de alcançar melhores resultados para problemas como o PCVSH. Assim, a seguir são apresentadas abordagens que combinam Algoritmo Memético e Algoritmo *Iterated Greedy* com as heurísticas apresentadas na Seção 3.2, Busca Tabu e Descida de Vizinhança Variável (ou RVND).

3.3.1 Algoritmo Memético com Busca Tabu e com RVND

Um Algoritmo Memético (AM) é um algoritmo evolucionário que, a cada geração, busca melhorar sua população por meio de cruzamentos e aplicações de

estruturas de vizinhança, convergindo a população para melhores soluções. O AM utilizado nesta dissertação é baseado no trabalho de Amorim et al. [2012] e seu pseudocódigo é apresentado no Algoritmo 6. Este algoritmo trabalha com uma população de cromossomos (soluções) e realiza a cada geração (iteração do algoritmo) cruzamentos entre os membros da população, modificações nos membros por meio de perturbação e melhoras realizadas pelo procedimento de busca local que utiliza as estruturas de vizinhança definidas na Subseção 3.1.2.

Neste trabalho, a população (p) é dividida em classes da seguinte maneira: Classe A (elite) composta pelos melhores indivíduos e possui tamanho correspondente a $(p * 0, 2)$; Classe C composta por $(p * 0, 15)$ representando os piores membros da população, ou seja, aqueles que possuem o pior valor da função objetivo e por fim, a Classe B com os membros que não pertencem à elite e nem a Classe C, seu tamanho é igual a $(p * 0, 65)$.

Para definição da população inicial P , o primeiro cromossomo é definido pelo procedimento construtivo LKH + Algoritmo de Dijkstra (ver Subseção 3.1.1.1) (linha 2). Os demais cromossomos $(p - 1)$ são definidos pelo procedimento construtivo HIMBR (ver Subseção 3.1.1.2)(linha 3). Após o preenchimento da população P ser finalizado, 20% da população será otimizada pela aplicação da estrutura de vizinhança 2-OPT. A população é então ordenada pelo valor da função objetivo considerando uma ordem crescente, para que sejam definidas as classes (A, B e C) descritas anteriormente.

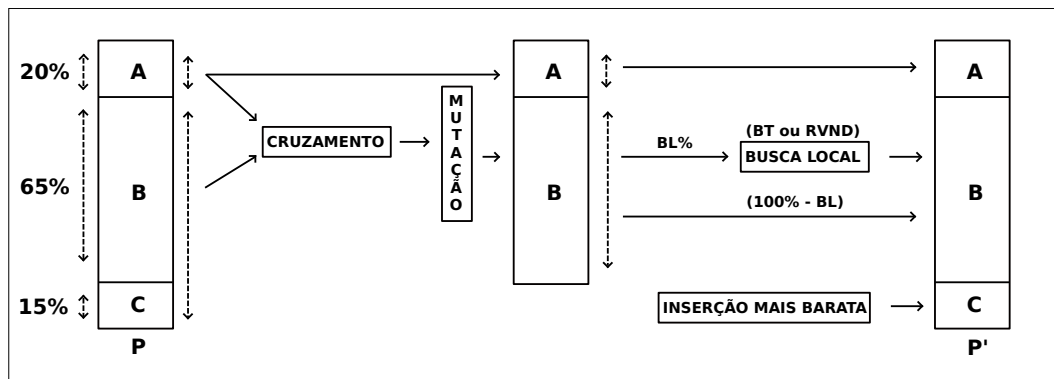


Figura 3.12. Representação do funcionamento do Algoritmo Memético.

O próximo passo é melhorar a população inicial, este processo é repetido por $maxIsm$ iterações sem melhora (linha 6). Como ilustrado na Figura 3.12, com intuito de privilegiar os indivíduos que compõe a elite de P , ou seja, aqueles pertencentes a Classe A, a nova população P' que será analisada na próxima geração receberá a elite de P (linha 7). Para realização do cruzamento da população são

Algoritmo 6: Algoritmo Memético

Entrada: $(p, maxIsm, mut, bl, \theta)$

```

1  início
2  |  $P \leftarrow \text{construtor\_LKH}(1);$ 
3  |  $P \leftarrow P + \text{construtor\_HIMBR}(p - 1);$ 
4  |  $2\text{-OPT}((p * 0,2), P);$ 
5  |  $ism \leftarrow 0;$ 
6  |  $\text{ordenar}(P);$ 
7  | enquanto  $(ism < maxIsm)$  faça
8  |   |  $P' \leftarrow \text{elite}(P);$ 
9  |   | para  $((0,65 * p) / 2)$  faça
10 |   |   |  $\text{pai1} \leftarrow \text{rand}(\text{elite}(P));$ 
11 |   |   |  $\text{pai2} \leftarrow \text{rand}(P \setminus \text{elite}(P));$ 
12 |   |   |  $P' \leftarrow P' + \text{cruzamento}(\text{pai1}, \text{pai2});$ 
13 |   | fim
14 |   |  $\text{pmut} \leftarrow \text{mínimo}((mut * ism), 0,2);$ 
15 |   | para  $i = 0$  até  $i < (p * \text{pmut})$  faça
16 |   |   |  $\text{mutação}(\text{rand}(P' \setminus \text{melhorIndivíduo}(P')), \theta);$ 
17 |   | fim
18 |   | para  $i = 0$  até  $i < (p * bl)$  faça
19 |   |   |  $\text{buscaLocal}(\text{rand}(P' \setminus \text{elite}(P')));$ 
20 |   | fim
21 |   | enquanto  $(|P'| < p)$  faça
22 |   |   |  $P' \leftarrow P' + \text{construtor\_HIMBR}(1);$ 
23 |   | fim
24 |   | se  $(\text{melhorIndivíduo}(P') = \text{melhorIndivíduo}(P))$  então
25 |   |   |  $ism \leftarrow ism + 1;$ 
26 |   | fim
27 |   | senão
28 |   |   |  $ism \leftarrow 0;$ 
29 |   | fim
30 |   |  $P \leftarrow P';$ 
31 | fim
32 |  $\text{Retornar}(\text{melhorIndivíduo}(P));$ 
33 fim

```

escolhidos aleatoriamente dois cromossomos (pais) de P : um cromossomo que faz parte da elite (mantém as características das melhores soluções) e outro cromossomo que faz parte do restante da população, sem considerar a elite. Escolhidos, os cromossomos serão cruzados utilizando um procedimento de cruzamento com dois pontos de quebra (Figura 3.13)(linhas 8 à 12). Para este procedimento são escolhidos aleatoriamente dois pontos na rota e o conteúdo que estiver entre estes dois pontos será trocado entre os dois cromossomos pais. Caso o conteúdo que estiver entre os dois pontos seja diferente, ou seja, apresentar clientes distintos, é verificado se o cliente já existe no cromossomo de destino. Caso afirmativo, a posição em que o cliente seria inserido permanece vazia até o término da tentativa de inserção de todo o conteúdo originado do cromossomo pai. Assim, é verificado qual ou quais cliente(s) estão faltando no cromossomo filho gerado e, realiza-se a inserção destes na ordem em que aparecem no cromossomo pai de origem. Os dois cromossomos filhos gerados pelo cruzamento serão adicionados a nova população P' . O cruzamento é responsável por preencher $(p * 0,65)$ da população P' .

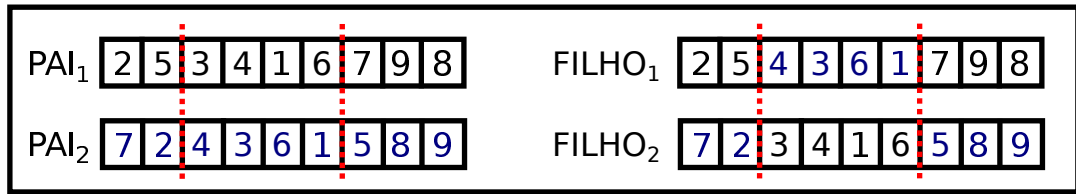


Figura 3.13. Representação do cruzamento com dois pontos de quebra.

As soluções resultantes do processo de refinamento são submetidas ao procedimento de mutação (ver Subseção 3.1.3.1). Para definir a quantidade de cromossomos de P' que serão afetados, o parâmetro *mut* é utilizado como pode ser observar nas linhas 13 à 16 do Algoritmo 6. A perturbação será então aplicada a no máximo 20% da população P' , lembrando que o melhor indivíduo não poderá ser afetado. Seguindo adiante, o procedimento de busca local é aplicado a $(p * bl)$ cromossomos da população P' desconsiderando-se a elite (linhas 17 à 19). Nesta etapa, foram definidas duas formas de realização da busca: utilizando a Busca Tabu (ver Subseção 3.2.1) ou utilizando RVND (ver Subseção 3.2.2). Com isto, foram criados dois algoritmos distintos que mantêm apenas a estrutura do AM em comum.

O restante da população de P' é então preenchida (linhas 20 à 22) utilizando-se o procedimento de construção de soluções HIMBR (ver Subseção 3.1.1.2). Antes que uma nova iteração seja iniciada, a população P é atualizada com os cromossomos de P' (linha 29). Novas populações são obtidas até que o critério de parada seja

alcançado, ou seja, até que sejam executadas *ism* iterações sem melhora no melhor indivíduo da população.

Ao final da execução do algoritmo AM, a solução retornada será o melhor indivíduo da população P . Na próxima seção será apresentado o algoritmo *Iterated-Greedy* com VND, que diferente do AM não faz uso de uma população de soluções.

3.3.2 Algoritmo Iterated-Greedy com VND

O algoritmo *Iterated-Greedy* (IG) é uma metodologia recente quando comparada a outras como AM, VND e BT. Tem alcançado bons resultados para problemas de roteamento [Duarte et al., 2012]. Este algoritmo é baseado em um princípio simples e efetivo: gerar soluções iterativas por meio de uma heurística gulosa aplicando duas fases (destruição da solução e posteriormente reconstrução). O processo de destruição considerado aqui retira uma porcentagem dos hotéis intermediários da solução e a reconstrução reinsere hotéis da melhor forma possível. O Algoritmo 7 descreve os processos básicos do IG e também os processos que foram adicionados para conferir maior robustez na resolução do PCVSH especificamente. A adaptação do IG criada nesta dissertação faz uso do procedimento VND (ver Subseção 3.2.2) para realização da busca local, sendo nomeado IGVND.

O algoritmo desenvolvido inicia seu procedimento criando uma solução inicial por meio da heurística LKH + Algoritmo de Dijkstra (ver Subseção 3.1.1.1) (linha 2). A quantidade de hotéis a serem retirados pelo processo de destruição é então definida de acordo com o parâmetro *dest*. Diferente do critério de parada utilizado no Algoritmo Memético, no IGVND o critério é definido pelo parâmetro *Imax* e determina o número de iterações que o algoritmo será executado. Seu critério de parada foi definido de forma similar ao critério utilizado no trabalho de Castro et al. [2014].

Internamente ao laço de repetição do Algoritmo 7, é aplicado o procedimento de perturbação da solução (ver Subseção 3.1.3.1) que realoca uma porcentagem (θ) de clientes na solução (linha 7). Caso a solução possua um número de hotéis suficientes para viabilizar a destruição, é sorteado um dos hotéis intermediários de onde deve iniciar o processo de destruição (linha 10). Ao retirar os hotéis uma nova viagem (S') contendo apenas os hotéis que eram vizinhos aos que foram removidos e os respectivos clientes que estão entre estes hotéis é criada. Para otimizar essa viagem, é aplicada a busca local 2-OPT (linha 11). Antes de realizar a reconstrução da rota, é realizada a divisão da viagem S' utilizando apenas o procedimento de divisão de rota baseado no Algoritmo de Dijkstra (ver Subseção 3.1.1.1) (linha 12).

Algoritmo 7: Algoritmo IGVND

Entrada: ($dest$, $Imax$, θ)

```

1 início
2    $S0 \leftarrow \text{construtor\_LKH}(1)$ ;
3    $S \leftarrow S0$ ;
4    $hoteis\_retirar \leftarrow hoteis\_rota * dest$ ;
5    $i \leftarrow 0$ ;
6   enquanto ( $i < Imax$ ) faça
7     Pertubar( $S$ );
8     se ( $hoteis\_rota \geq (hoteis\_retirar + 2)$ ) então
9        $pos \leftarrow \text{rand}() \% (hoteis\_rota - hoteis\_retirar - 1) + 1$ ;
10       $S' \leftarrow \text{Destruir}(S, S_{pos}, S_{pos+hoteis\_retirar})$ ;
11      2-OPT( $S'$ );
12      Dijkstra( $S'$ );
13      Reconstruir ( $S$ ,  $S'_0$ ,  $S'_n$ );
14    fim
15    VND( $S$ );
16    se ( $S$  melhor  $S0$ ) então
17       $S0 \leftarrow S$ ;
18    fim
19     $i++$ ;
20  fim
21  Retornar  $S0$ ;
22 fim

```

Terminado o processo de divisão, S' não será mais uma viagem e sim uma subrota que contém clientes e hotéis que foram retirados de S pelo processo de desconstrução.

O processo de reconstrução (linha 13) realiza então, a reinserção dos hotéis e realocação dos clientes seguindo a disposição dos mesmos como apresentados em S' (subrota que foi retirada) em S . A heurística VND é então aplicada à solução S e o critério de aceitação da solução é testado (linhas 15 à 18). Se a solução retornada pelo VND for melhor que a melhor solução corrente armazenada ($S0$), então a melhor solução corrente é atualizada. Ao final da execução do IGVND, será retornada a melhor solução encontrada pelo procedimento.

Capítulo 4

RESULTADOS

Nesta seção, são apresentadas informações sobre o ambiente computacional utilizado, a métrica de avaliação dos resultados, os grupos de instâncias, a calibração dos algoritmos e os resultados obtidos pelas diferentes estratégias abordadas nesta dissertação.

4.1 Ambiente Computacional Utilizado

Para realização dos testes computacionais, todos os algoritmos foram codificados na linguagem C++, compilados com gcc 4.3.4 e executados em uma máquina com processador Intel(R) Xeon(R) X5650 @ 2.66GHz e 24 GB de RAM DDR3 1333 MHz. Para análise estatística dos parâmetros de cada abordagem proposta, foi utilizado o software Statgraphics Centurion XVII (versão trial).

4.2 Métrica para Avaliação dos Algoritmos

Para verificar a qualidade de uma solução heurística foi computado o GAP em relação as melhores soluções conhecidas. Para avaliação dos resultados obtidos neste trabalho foi utilizada a diferença percentual entre a melhor solução encontrada pelo algoritmo a ser comparado e o melhor resultado contido na literatura (Equação 4.1).

$$GAP = 100 * \frac{f_{algoritmo} - f_{melhor}}{f_{melhor}} \quad (4.1)$$

Na equação, $f_{\text{algoritmo}}$ é a melhor solução encontrada pelo algoritmo que está sendo avaliado (valor da função objetivo) e f_{melhor} é a melhor solução obtida pelos algoritmos da literatura.

4.3 Grupos de Instâncias Utilizados

Os grupos de instâncias utilizados para realizar os testes computacionais dos algoritmos desenvolvidos nesta dissertação foram idealizados por Vansteenwegen et al. [2012] e são divididos em quatro grupos que levam em consideração características como o tamanho da instância e a dificuldade para encontrar as melhores soluções para tais instâncias.

O primeiro grupo (definido por SET_1) é baseado em seis instâncias do Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) [Solomon, 1987] e em outras dez instâncias do Problema de Roteamento de Veículos com Janela de Tempo e múltiplos Depósitos (PRVJTMD) [Cordeau et al., 2001]. As primeiras instâncias possuem uma quantidade de 100 clientes e as demais possuem uma quantia definida entre 48 e 288 clientes. Para estas instâncias, cinco hotéis foram adicionados em pontos que já existem algum cliente. Desta forma, atendendo a restrição do problema em que todos os clientes devem ser visitados, ao visitar algum destes hotéis há a possibilidade de definição de uma rota onde a visita aos hotéis não represente um aumento no tempo total da rota.

O segundo grupo de instâncias (definido por SET_2) foi desenvolvido com intuito de ser um grupo fácil, de tal modo que um modelo matemático seja capaz de encontrar a maioria de suas soluções ótimas em um tempo razoável. Foram selecionadas 13 das 16 instâncias do SET_1 e para cada uma das 13 instâncias, foi considerado uma quantia de 10, 15, 30 e 40 clientes totalizando uma quantia de 52 instâncias. Neste grupo, apenas dois hotéis são disponibilizados em cada instância.

Para viabilizar o teste de heurísticas, estes dois últimos grupos são compostos por instâncias mais complexas. O terceiro grupo (definido por SET_3) é composto por instâncias derivadas do clássico Problema do Caixeiro Viajante e possuem de 51 a 1002 clientes, além de possuírem 3, 5 ou 10 hotéis extras que podem ser utilizados no percurso do caixeiro. Como a definição diz, estes hotéis extras não incluem o hotel de partida (H_0). Para este grupo as melhores soluções são conhecidas e representam a solução do PCV.

Por fim, o último grupo (SET_4) foi desenvolvido a partir de 15 instâncias do SET_3 que contam com 10 hotéis extras. Para dificultar ainda mais, os hotéis foram

definidos em posições estratégicas, não permitindo que seja afirmado que a solução ótima seja a mesma solução do PCV. O tempo limite L de uma viagem para cada uma das instâncias é detalhado em maiores detalhes no trabalho de Vansteenwegen et al. [2012]. Na Tabela 4.1 é possível observar as principais características dos grupos de instâncias de forma resumida.

Tabela 4.1. Resumo das características das instâncias utilizadas

Grupo	Nível de Dificuldade	Nº de Clientes	Nº Hotéis	Total Instâncias
SET_1	Médio	48 a 288	5	16
SET_2	Fácil	10 a 40	2	52
SET_3	Difícil	51 a 1002	4 a 11	48
SET_4	Difícil	51 a 1002	11	15

4.4 Calibração de Parâmetros

Com o intuito de determinar a melhor configuração dos valores para os parâmetros de cada algoritmo, um experimento computacional foi realizado com base na metodologia Desenho de Experimentos (DOE) [Montgomery, 2006] onde cada fator é um parâmetro controlado. O desenho fatorial completo é usado para todos os fatores de cada algoritmo, assim todas as possibilidades de combinações possíveis foram analisadas.

Os experimentos foram realizados utilizando uma amostra de 15 instâncias escolhidas aleatoriamente do grupo de instâncias definidas na literatura [Vansteenwegen et al., 2012]. Para cada instância os algoritmos foram executados trinta vezes utilizando cada configuração de parâmetros, e o valor da função objetivo foi utilizado para calcular o GAP em cada uma das trinta vezes. As melhores configurações para os parâmetros foram determinadas por meio do Teste de Kruskal-Wallis e pelo Teste de Comparações Múltiplas.

4.4.1 Calibração dos Parâmetros do AMBT

Para o algoritmo AMBT foram analisados 10 parâmetros, sendo eles 5 parâmetros utilizados na Busca Tabu e 5 parâmetros utilizados no Algoritmo Memético. Para realizar o fatorial completo com os 10 parâmetros seria inviável, pois a quantia de combinações diferentes seria muito grande. Para evitar o número exponencial de combinações, foi realizada primeiramente a calibração da busca tabu e posteriormente a calibração do Algoritmo Memético com Busca Tabu utilizando os parâmetros da Busca Tabu fixados conforme a calibração já realizada.

Os seguintes conjuntos de valores foram testados para a Busca Tabu: $\text{Imax} \in \{50; 100; 130\}$ (representa o número máximo de iterações da Busca Tabu), $\theta^- \in \{7; 10; 13\}$ (representa o número mínimo de iterações que um movimento é considerado tabu), $\theta^+ \in \{17; 20; 23\}$ (representa o número máximo de iterações que um movimento é considerado tabu), $U_p \in \{10; 15; 20\}$ (representa a quantidade de iterações necessárias para testar se a penalização deve ser atualizada) e $U_c \in \{2; 5; 10\}$ (representa a quantidade de iterações necessárias para realizar uma otimização da rota com a estrutura 3 – OPT). Note que ao todo foram testadas 243 configurações de parâmetros. Foi obtido a partir do teste de Kruskal-Wallis um *valor – P* = 1,0. Como é necessário obter um *valor – P* menor que 0,05 para afirmar que há uma diferença estatisticamente significativa de uma configuração de parâmetros em relação a outra, considerando um nível de confiança de 95%, foi constatado que diversas configurações conseguiam alcançar os melhores resultados para o grupo de instâncias selecionado para a calibração (Figura 4.1).

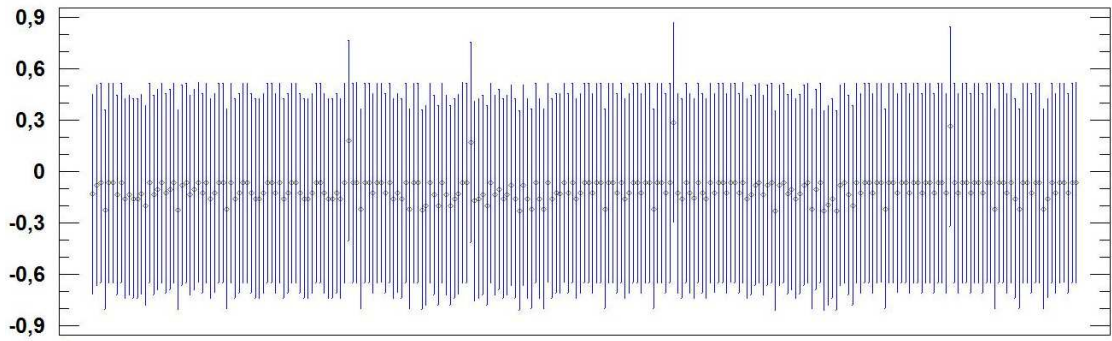


Figura 4.1. Gráfico de médias do GAP de todas configurações para Busca Tabu no primeiro teste.

Como várias configurações compartilham a mesma média mínima do GAP, foi escolhida a que demandou o menor tempo computacional médio. A configuração selecionada foi a 106, com os parâmetros: $\text{Imax} = 100$, $\theta^- = 7$, $\theta^+ = 23$, $U_p = 20$ e $U_c = 2$. Como a maioria dos parâmetros apresentam-se em algum extremo dos conjuntos testados, foram definidos novos intervalos para estes parâmetros, sendo eles $\theta^- \in \{3; 5; 7\}$, $\theta^+ \in \{13; 15; 17\}$ e $U_p \in \{15; 20; 25\}$. Os parâmetros U_p e Imax foram mantidos no mesmo intervalo. Após realizar os testes computacionais com os novos intervalos para os parâmetros e aplicar o teste de Kruskal-Wallis e de Comparações Múltiplas, constatou-se novamente um *valor – P* = 1,0 o que indica que não houve diferença estatisticamente significativa entre as configurações testadas. Para determinar qual configuração utilizar, analisou-se os valores médios do GAP para todas as configurações (Figura 4.2).

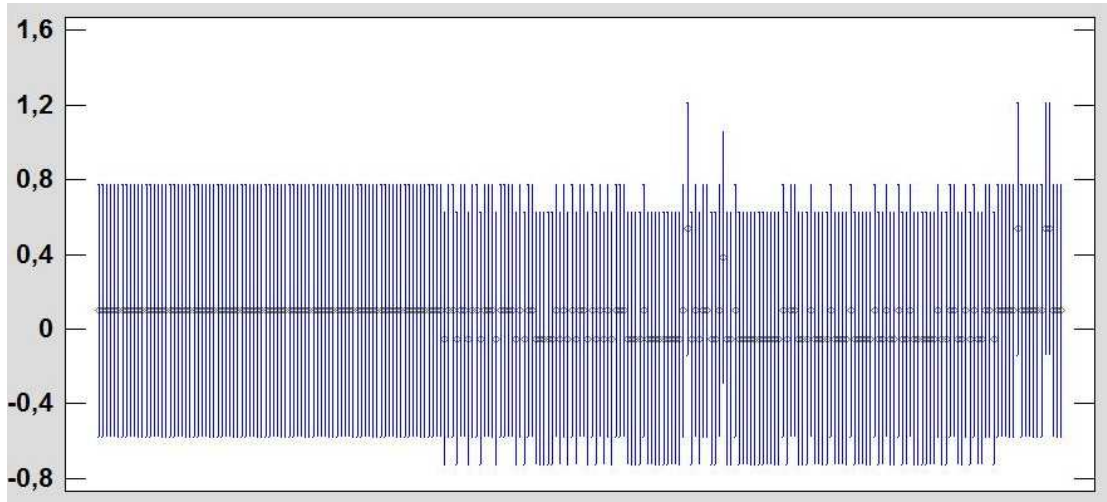


Figura 4.2. Gráfico de médias do GAP de todas configurações para Busca Tabu no segundo teste.

Devido à grande quantidade de configurações apresentarem a mesma média mínima para o GAP, foi selecionada a que demandou menor tempo computacional e que contém os valores centrais dos intervalos de parâmetros. A configuração escolhida foi a 121 e os parâmetros foram: $Imax = 100$, $\theta^- = 5$, $\theta^+ = 15$, $U_p = 20$ e $U_c = 2$.

Com os parâmetros da Busca Tabu definidos, os parâmetros do Algoritmo Memético puderam ser também analisados. Os parâmetros a serem calibrados são: $p \in \{5; 10; 20\}$ (tamanho da população), $maxIsim \in \{10; 15; 20\}$ (número máximo de iterações sem melhora), $mut \in \{0,1; 0,2; 0,3\}$ (percentual de mutação), $bl \in \{0,2; 0,3; 0,5\}$ (percentual de aplicação de busca local) e $\theta \in \{0,07; 0,15; 0,2\}$ (percentual de perturbação dos clientes). Após os testes computacionais e estatísticos, obteve-se um $valor - P = 1,0$, ou seja, não há diferença significativa entre as diferentes configurações utilizando o teste de Kruskal-Wallis. Após a realização do Teste de Comparações Múltiplas (Figura 4.3), foram analisadas as três configurações com os melhores GAPs (neste caso, foram iguais) e foi selecionada a configuração que demandou menor tempo computacional. A configuração escolhida foi a 219 com os valores: $p = 20$, $maxIsim = 20$, $mut = 0,1$, $bl = 0,2$ e $\theta = 0,2$. Como a configuração escolhida apresenta valores nos extremos dos intervalos testados, novos intervalos foram atribuídos aos parâmetros: $p \in \{20; 30; 40\}$, $maxIsim \in \{20; 30; 40\}$, $mut \in \{0,1; 0,2; 0,3\}$, $bl \in \{0,2; 0,5; 0,7\}$ e $\theta \in \{0,07; 0,15; 0,2\}$.

Novamente, para cada uma das instâncias do conjunto escolhido para realização da calibragem o algoritmo foi executado 30 vezes e os resultados foram analisados

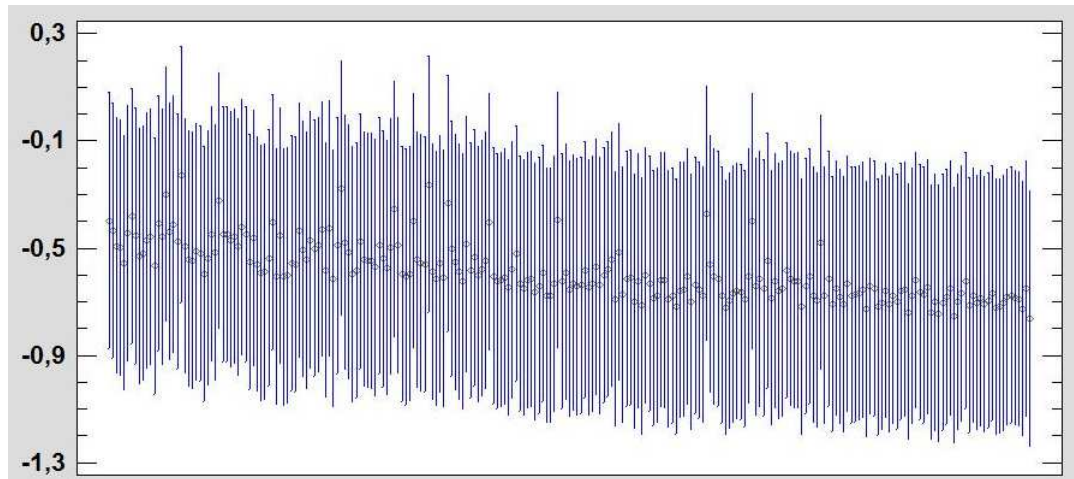


Figura 4.3. Gráfico de médias do GAP de todas configurações para o Algoritmo Memético com Busca Tabu no primeiro teste da calibragem.

aplicando-se o teste de Kruskal Wallis e de Comparações Múltiplas. Concluiu-se que nenhuma das configurações apresentava-se estatisticamente diferente das demais. Foi gerado o gráfico de médias do GAP (Figura 4.4) e como houve uma grande quantidade de configurações com a mesma média mínima idêntica, optou-se por aquela que melhor distribuiu-se pelo intervalo dos níveis de cada parâmetro e ao mesmo tempo demandou um dos menores tempos computacionais médios para a execução da calibragem. A configuração escolhida foi a 137, com os seguintes valores para cada parâmetro: $p = 30$, $maxIsm = 40$, $mut = 0,1$, $bl = 0,2$ e $\theta = 0,15$.

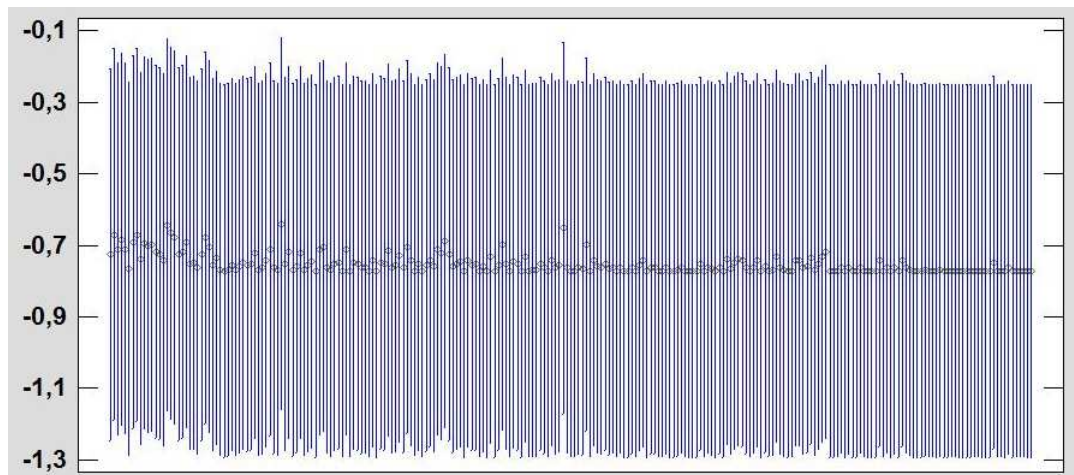


Figura 4.4. Gráfico de médias do GAP de todas configurações para o Algoritmo Memético com Busca Tabu no segundo teste da calibragem.

4.4.2 Calibração dos Parâmetros do AMRVND

Para o algoritmo AMRVND foram analisados 5 parâmetros que fazem parte do Algoritmo Memético e para a heurística RVND não foi necessário calibrar nenhum parâmetro. Os parâmetros analisados e os respectivos intervalos testados foram os seguintes: $p \in \{20; 30; 50\}$ (tamanho da população), $maxIsM \in \{20; 30; 50\}$ (número máximo de iterações sem melhora), $mut \in \{0; 0,1; 0,2\}$ (percentual de mutação), $bl \in \{0,2; 0,3; 0,5\}$ (percentual de aplicação de busca local) e $\theta \in \{0,2; 0,3; 0,5\}$ (percentual de perturbação dos clientes).

Após a realização dos testes computacionais, foram aplicados aos dados obtidos o teste de Kruskal-Wallis e o Teste de Comparações Múltiplas. Ambos os testes determinaram que não existe diferença estatisticamente significativa entre as 243 configurações possíveis. O gráfico de médias (Figura 4.5) ilustra a grande similaridade entre a média do GAP para as diversas configurações e demonstra que o algoritmo apresenta convergência mesmo para os dados que estão no limite inferior dos intervalos testados. A configuração escolhida foi a que apresentou a menor média do GAP e o menor tempo médio. Neste caso, foi utilizada a configuração 16, definindo os parâmetros com os seguintes valores: $p = 20$, $maxIsM = 20$, $mut = 0,1$, $bl = 0,5$ e $\theta = 0,2$.

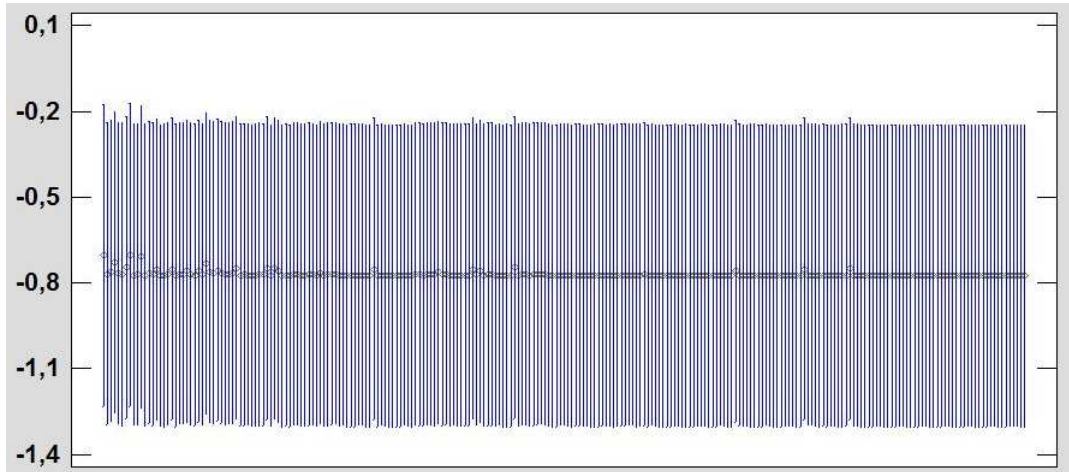


Figura 4.5. Gráfico de médias do GAP de todas as configurações para o Algoritmo Memético com RVND no primeiro teste da calibragem.

Como os parâmetros definidos pelo primeiro teste utilizam valores definidos no extremo inferior do intervalo de valores testados, novos intervalos foram definidos: $p \in \{5; 10; 20\}$, $maxIsM \in \{10; 15; 20\}$, $mut \in \{0,1; 0,2; 0,3\}$, $bl \in \{0,2; 0,3; 0,5\}$ e $\theta \in \{0,07; 0,15; 0,2\}$ (percentual de perturbação dos clientes). O algoritmo foi então executado novamente para as instâncias definidas para a calibragem do algoritmo e

o teste de Kruskal-Wallis resultou em um *valor* - $P = 1,0$. O gráfico de médias do GAP ilustra que quanto mais no extremo os parâmetros forem definidos, mais rápido o algoritmo converge.

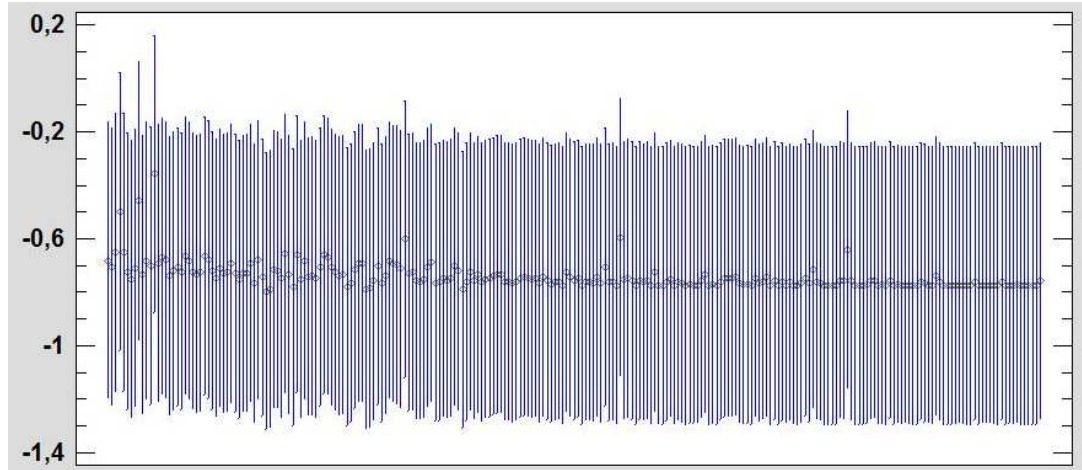


Figura 4.6. Gráfico de médias do GAP de todas configurações para o Algoritmo Memético com RVND no segundo teste da calibragem.

Posteriormente, após realizar o Teste de Comparações Múltiplas e constatar que não existe uma diferença estatisticamente significativa entre as diferentes configurações de parâmetros, optou-se por aquele que está com o valor no centro do intervalo para maioria dos parâmetros. A configuração escolhida foi a 42, contendo os seguintes valores para os parâmetros: $p = 5$, $maxIsn = 15$, $mut = 0,2$, $bl = 0,3$ e $\theta = 0,2$. O parâmetro P que foi definido no extremo inferior do intervalo foi mantido sem realizar uma nova variação no intervalo, pois o parâmetro já se encontra pequeno e caso fosse diminuído perderia a característica populacional do algoritmo. Já o parâmetro θ não foi variado para um intervalo que considera valores maiores, pois caso fosse alterado, deixaria o algoritmo com uma característica de reinício aleatório.

4.4.3 Calibração dos Parâmetros do IGVND

Para calibração do algoritmo IGVND foram analisados três parâmetros. Para cada um dos parâmetros foi avaliado um conjunto contendo três níveis que foram definidos de acordo com experimentos empíricos realizados. Os parâmetros e seus respectivos conjuntos são: $I_{max} \in \{10; 20; 30\}$ (número máximo de iterações), $dest \in \{0,4; 0,5; 0,6\}$ (porcentagem de destruição dos hotéis) e $\theta \in \{0,03; 0,05; 0,07\}$ (percentual de perturbação dos clientes). Foram realizados os testes computacionais executando-se cada uma das 27 configurações diferentes por trinta vezes e os

resultados foram analisados estatisticamente por meio do Teste de Kruskal-Wallis e pelo Teste de Comparações Múltiplas. Ambos os testes determinaram que não existe nenhuma configuração que se difere de forma estatisticamente significativa de uma outra configuração dentre as analisadas. Um exemplo gráfico da plotagem das médias do GAP é ilustrado na Figura 4.7.

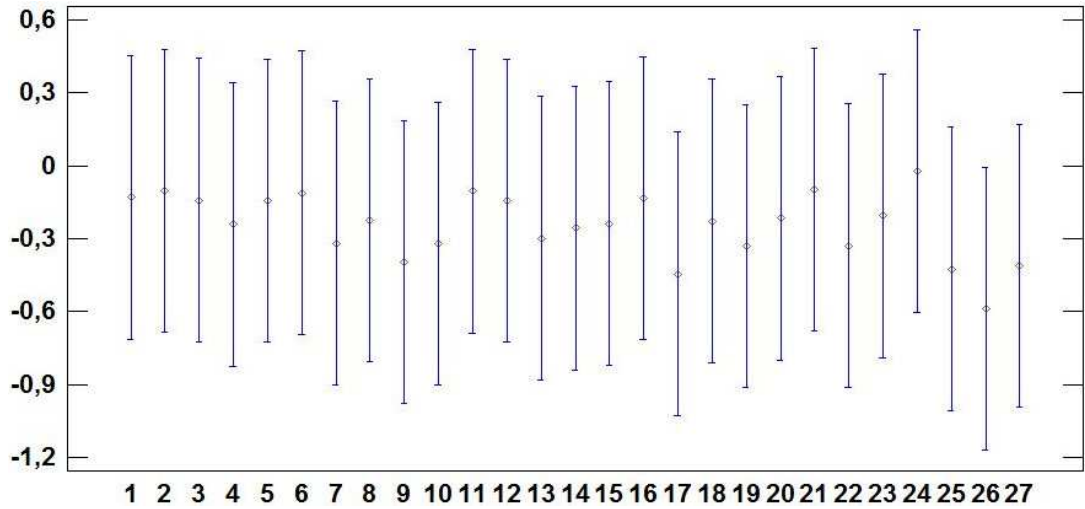


Figura 4.7. Gráfico de médias do GAP de todas configurações para o Algoritmo IGVND no primeiro teste da calibragem.

Considerando que a melhor média do GAP foi obtida pela configuração 26, representada pelos valores $I_{max} = 30$, $dest = 0,6$ e $\theta = 0,07$, onde quase todos os valores dos parâmetros foram definidos no extremo superior dos níveis dos conjuntos testados, novos níveis foram definidos com intuito de melhorar ainda mais o GAP médio. Os novos níveis a serem analisados são definidos pelos seguintes conjuntos: $I_{max} \in \{20; 30; 40\}$, $dest \in \{0,6; 0,7; 0,8\}$ e $\theta \in \{0,05; 0,07; 0,09\}$. Mais uma vez, após realizar os testes computacionais e estatísticos (Kruskal-Wallis e Comparações Múltiplas) não houve uma configuração que apresentou-se com uma diferença considerada estatisticamente diferente para as demais. Porém, foi observado que a média do GAP foi diminuída (Figura 4.8) em comparação com a média do primeiro teste. Desta forma, foi escolhida a configuração 21 com os valores de parâmetros $I_{max} = 40$, $dest = 0,6$ e $\theta = 0,09$.

Como o parâmetro $dest$ apresentou-se com o valor central do nível, logo, foi fixado seu valor e novos níveis para os demais parâmetros foram definidos. Os novos níveis são: $I_{max} \in \{40; 50; 60\}$, $dest \in \{0,5; 0,6; 0,7\}$ e $\theta \in \{0,09; 0,11; 0,13\}$. Todos os testes computacionais foram refeitos para as instâncias definidas para calibragem dos algoritmos e os testes estatísticos foram conduzidos novamente

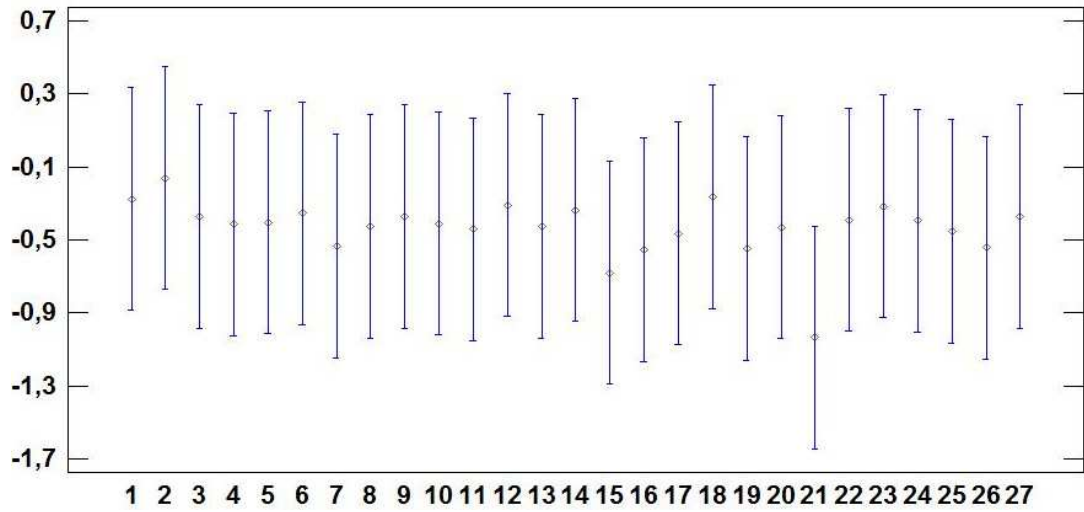


Figura 4.8. Gráfico de médias do GAP de todas configurações para o Algoritmo IGVND no segundo teste da calibragem.

com o intuito de testar se a variação nos níveis dos parâmetros seria suficiente para alcançar melhores resultados. Os testes estatísticos determinaram que não existe uma diferença estatisticamente significativa entre as configurações analisadas. Após analisar o gráfico de médias da segunda (Figura 4.8) e da terceira (Figura 4.9) amostra de configurações, verificou-se que os limites inferior e superior da média pertencente à segunda amostra são menores que os apresentados na terceira. Além do mais, na terceira amostra o tempo médio gasto estava aumentando muito pelo alto número de iterações do algoritmo e a perturbação dos clientes tendia a tornar a solução com características de reinício aleatório.

Com isso, a configuração escolhida para executar o algoritmo IGVND foi a 21, com os seguintes parâmetros: $Imax = 40$, $dest = 0,6$ e $\theta = 0,09$. Definidas as configurações de parâmetros para os três algoritmos propostos, os resultados de suas execuções são apresentados na seção apresentada a seguir.

4.5 Testes Computacionais

Nesta seção, são feitas as comparações dos resultados obtidos pelas abordagens propostas com os melhores resultados contidos na literatura. Para cada uma das instâncias os algoritmos foram executados 30 vezes, considerando os parâmetros definidos na Seção 4.4 e os resultados obtidos pela função objetivo foram utilizados para calcular o GAP. O tempo de execução dos algoritmos foi contabilizado considerando o tempo de CPU. O tempo máximo definido para a execução do modelo

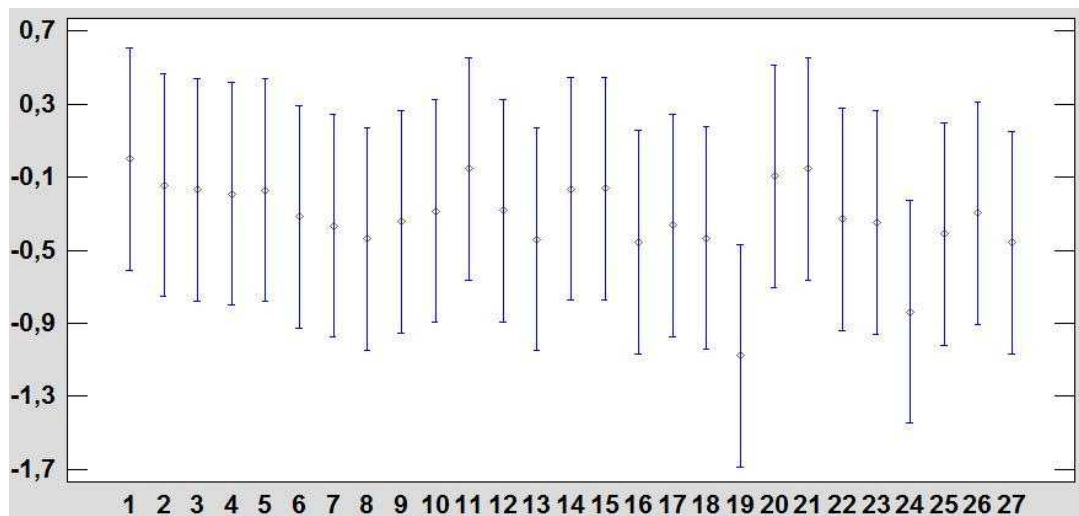


Figura 4.9. Gráfico de médias do GAP de todas configurações para o Algoritmo IGVND no terceiro teste da calibragem.

matemático foi fixado em 24 horas.

Nas tabelas a seguir, os resultados em **negrito** identificam os melhores resultados encontrados para o PCVSH e os símbolos mais (+) e menos (-) representam o aumento ou diminuição no número de viagens que foi utilizado para percorrer a rota. A Melhor Solução Conhecida (MSC) é definida com base nos trabalhos de Castro et al. [2013, 2014].

Para o grupo de instâncias SET_2, foram conduzidos os testes com o modelo matemático proposto neste trabalho, rotulado nas tabelas a seguir como Modelo Fluxo. Os resultados obtidos são apresentados na Tabela 4.2 e Tabela 4.3 e comparados com os resultados obtidos pelo modelo matemático proposto no trabalho de Castro et al. [2013]. A primeira linha das tabelas a seguir contém a descrição do grupo de instâncias utilizado e a quantidade de clientes que existe em cada instância. A segunda linha das tabelas contém a descrição da abordagem utilizada. A primeira coluna das tabelas contém o nome das instâncias. A segunda, quarta, oitava e nona coluna contém o número de viagens para cada instância. A terceira, quinta, oitava e décima coluna contém os tempos totais das rotas para cada instância. Por fim, a sexta e décima primeira coluna rotuladas com a sigla GAP representam a diferença percentual entre a melhor solução conhecida e a solução encontrada pelo modelo matemático proposto.

Os resultados obtidos pelo modelo matemático na Tabela 4.2 demonstram que para o conjunto de instâncias que contém 10 ou 15 clientes, o modelo matemático proposto é eficaz para encontrar a solução ótima. Para o conjunto de instâncias

que contém 30 e 40 clientes (Tabela 4.3), o modelo proposto consegue encontrar a solução ótima para a maioria das instâncias. Para a instância c101 o modelo alcançou resultados para o GAP menores ou iguais a 0,15% e para duas instâncias (r101 e rc101) o modelo não conseguiu encontrar uma solução durante o tempo máximo definido.

Tabela 4.2. Resultados SET_1 e SET_2 modelo exato

	SET_2 com 10 clientes					SET_2 com 15 clientes				
	MSC		Modelo Fluxo			MSC		Modelo Fluxo		
instância	V	T. Total	V	T. Total	GAP	V	T. Total	V	T. Total	GAP
c101	1	955,1	1	955,1	0,00	2	1452,2	2	1452,2	0,00
r101	2	272,8	2	272,8	0,00	2	379,8	2	379,8	0,00
rc101	1	237,5	1	237,5	0,00	2	303,2	2	303,2	0,00
pr01	1	426,6	1	426,6	0,00	1	590,4	1	590,4	0,00
pr02	1	661,9	1	661,9	0,00	1	745,6	1	745,6	0,00
pr03	1	553,3	1	553,3	0,00	1	632,9	1	632,9	0,00
pr04	1	476,4	1	476,4	0,00	1	683,4	1	683,4	0,00
pr05	1	528,9	1	528,9	0,00	1	621,2	1	621,2	0,00
pr06	1	597,4	1	597,4	0,00	1	685,2	1	685,2	0,00
pr07	1	670,2	1	670,2	0,00	1	795,3	1	795,3	0,00
pr08	1	573,4	1	573,4	0,00	1	707,2	1	707,2	0,00
pr09	1	645,5	1	645,5	0,00	1	771,7	1	771,7	0,00
pr10	1	461,5	1	461,5	0,00	1	611,9	1	611,9	0,00
Média					0,00					0,00

Tabela 4.3. Resultados SET_3 e SET_4 modelo exato

	SET_2 com 30 clientes					SET_2 com 40 clientes				
	MSC		Modelo Fluxo			MSC		Modelo Fluxo		
instância	V	T. Total	V	T. Total	GAP	V	T. Total	V	T. Total	GAP
c101	3	2829,4	3	2863,6	1,21	4	3817,5	4	3874,4	1,49
r101	3	655,2	-	-	-	4	842,9	-	-	-
rc101	3	610,0	-	-	-	3	652,1	-	-	-
pr01	1	964,8	1	964,8	0,00	2	1160,5	2	1160,5	0,00
pr02	2	1078,3	2	1078,3	0,00	2	1336,9	2	1336,9	0,00
pr03	1	952,5	1	952,5	0,00	2	1303,4	2	1303,4	0,00
pr04	2	1091,6	2	1091,6	0,00	2	1259,5	2	1259,5	0,00
pr05	1	924,7	1	924,7	0,00	2	1200,7	2	1200,7	0,00
pr06	2	1063,2	2	1063,2	0,00	2	1242,9	2	1242,9	0,00
pr07	2	1130,4	2	1130,4	0,00	2	1407,0	2	1407,0	0,00
pr08	2	1006,2	2	1006,2	0,00	2	1222,2	2	1222,2	0,00
pr09	2	1091,4	2	1091,4	0,00	2	1284,2	2	1284,4	0,02
pr10	1	918,9	1	918,9	0,00	2	1200,4	2	1200,4	0,00
Média					0,11					0,14

O trabalho da literatura que apresenta a maioria dos melhores resultados para o PCVSH é o Algoritmo Memético com Busca Tabu (AMBTlit) desenvolvido por Castro et al. [2013]. Analisando este trabalho, foram observadas algumas caracte-

rísticas no Algoritmo Memético que poderiam ser aprimoradas. Com isso, foi desenvolvida uma nova abordagem de Algoritmo Memético (AM) com características distintas do AM da literatura, porém combinado com a mesma Busca Tabu presente em Castro et al. [2013]. Posteriormente, uma nova abordagem de AM também foi desenvolvida onde utilizou-se o RVND no lugar da Busca Tabu, objetivando-se com isso reduzir o tempo de execução, já que a Busca Tabu consumia a maior parte do tempo de processamento.

A comparação entre os resultados das três diferentes abordagens de AMs: Algoritmo Memético com Busca Tabu da literatura, Algoritmo Memético com Busca Tabu proposto e Algoritmo Memético com Descida de Vizinhaça Variável proposto é apresentada da Tabela 4.4 à Tabela 4.12. Nestas tabelas, o rótulo traz como informações o nome do grupo a que as instâncias pertencem. As siglas que identificam os diferentes algoritmos, são: Melhor Solução Conhecida (MSC), Algoritmo Memético com Busca Tabu da literatura (AMBTLit), Algoritmo Memético com Busca Tabu proposto (AMBT) e Algoritmo Memético com RVND proposto (AMRVND). Ambos os algoritmos AMBT e AMRVND foram implementados de acordo com a descrição apresentada na Seção 3.3.1. A primeira coluna das tabelas contém o nome das instâncias. A segunda, quarta, oitava e décima segunda coluna contém a quantia de viagens que foram necessárias para percorrer as rotas. Os símbolos de mais (+) e menos (-) que aparecem junto ao número de viagens representam que houve um aumento ou diminuição no número de viagens. A terceira, quinta, nona e décima terceira colunas representam o tempo total necessário para percorrer as rotas. A sexta, décima e décima quarta colunas contém o tempo de CPU gasto para execução de cada algoritmo. A sétima, décima primeira e décima quinta colunas representam o percentual de diferença entre as soluções encontradas pelos algoritmos e a melhor solução conhecida da literatura, intituladas GAP. Os dados que são apresentados em **negrito** correspondem aos melhores resultados encontrados para cada instância.

Os resultados encontrados pelos Algoritmos Meméticos propostos para o SET_1 são apresentados na Tabela 4.4 e demonstram que os algoritmos são eficazes obtendo valores médios para o GAP menores que 0,12 % (AMBT) e 0,14 % (AMRVND). Ainda observando os resultados obtidos, o algoritmo AMBT consegue encontrar para quatro instâncias as melhores soluções conhecidas e para outras sete instâncias melhora os resultados contidos na literatura. O AMBT diminui o número de viagens gastas para uma instância e aumenta para outra. O algoritmo AMRVND consegue encontrar quatro novas melhores soluções, mantendo os números de viagens iguais aos da melhor solução contida na literatura até então. Para outras duas instâncias o AMRVND necessita de uma viagem extra para percorrer a rota. Ou-

tra observação importante a se considerar é o tempo médio gasto pelas diferentes abordagens. Ao comparar o tempo médio do AMBTLit contido na literatura com a abordagem AMBT proposta, observou-se que o algoritmo proposto levou em média 15 vezes mais tempo para realizar a execução de todas as instâncias. Quando compara-se o tempo do algoritmo AMRVND com o AMBTLit, este primeiro necessita de uma quantidade menor de tempo, chegando a ser 5 vezes mais rápido que o AMBTLit.

Para o segundo grupo de instâncias (SET_2), ambos os algoritmos propostos encontram a maioria das melhores soluções conhecidas, chegando no pior caso a apresentar um GAP médio de 2,81%. No quesito tempo computacional, o maior valor médio de tempo gasto fica em torno de 3,2 segundos para as instâncias que possuem 40 clientes. Vale a pena ressaltar, que para a instância rc101.k30 ambas abordagens propostas diminuíram o tempo total da rota, porém apresentaram um aumento no número de viagens necessárias. Para este conjunto de instâncias o AMRVND necessitou de uma viagem a mais para três instâncias e o AMBT para duas instâncias. Esses resultados são apresentados nas Tabela 4.5, Tabela 4.6, Tabela 4.7 e Tabela 4.8.

Os resultados observados para o SET_3 com a inclusão de 3 hotéis extras em cada instância são apresentados na Tabela 4.9. Estes resultados demonstram que ambos os algoritmos propostos são qualitativamente similares. Para o AMBT o GAP médio foi de 0,10% e para o AMRVND foi de 0,01%. Novamente, conforme resultado obtido na Tabela 4.4 o tempo médio do AMBT foi superior ao AMBTLit e o AMRVND obteve tempo médio inferior aos demais. Para a instância a280.s3 o AMBT conseguiu alcançar um resultado melhor que a melhor solução conhecida na literatura, diminuindo uma viagem no total necessário para percorrer a rota. Ainda foram observadas algumas instâncias em que as três abordagens de Algoritmo Memético comparadas aumentam o número de viagens necessárias. Entre os três algoritmos analisados, o que obteve o tempo médio de execução menor, foi o AMRVND.

Ainda no grupo de instâncias denominado SET_3, a Tabela 4.10 traz os resultados para instâncias com 5 hotéis extras. Mais uma vez o AMBT mostrou-se robusto para quase todas as instâncias testadas, obtendo para a instância a280.s5 mais de 1 % de redução no tempo total da rota. Em contrapartida a sua qualidade, destaca-se que seu tempo médio é várias vezes maior que o AMBTLit e também maior quando comparado ao tempo do AMRVND. Tanto o AMBT, quanto o AMRVND alcançaram quase todas as melhores soluções conhecidas ficando com o GAP médio próximo a zero. Em três casos, um em cada abordagem, o número de

viagens necessárias para percorrer a rota foi incrementado.

O último teste realizado com o SET_3 compreende as instâncias compostas por 10 hotéis extras. Neste conjunto de instâncias, novamente os algoritmos propostos obtiveram resultados compatíveis com os melhores conhecidos (Tabela 4.11). O AMRVND consegue encontrar três soluções com tempo total da rota menor que a melhor solução conhecida, finalizando as execuções das instâncias com um GAP médio de -0,11 %. Em contrapartida, para as três instâncias que foi notada a melhora, o número de viagens foi aumentado. O AMBT foi prejudicado por seu grande tempo computacional, o que ocorreu principalmente pelo alto custo computacional necessário às três últimas instâncias que apresentam uma grande quantidade de clientes. Foram observadas duas melhoras quanto ao tempo total da rota, mas para estas instâncias foram utilizadas mais viagens que na melhor solução conhecida na literatura.

As instâncias que apresentam maior dificuldade para serem resolvidas estão agrupadas no grupo SET_4 e os resultados obtidos são apresentados na Tabela 4.12. O AMBT encontrou três soluções com tempo total da rota inferior ao melhor conhecido, sendo que em duas delas o número de viagens gasto foi mantido igual a melhor solução conhecida. Para três instâncias, o número de viagens necessárias foi acrescido pelo AMBT. De forma similar, no AMRVND foram encontradas quatro soluções com tempo total da rota reduzido quando comparado ao melhor resultado conhecido na literatura. Para duas destas soluções o número de viagens é mantido igual ao melhor conhecido e nas outras duas é acrescida uma viagem. Somente para este grupo o AMRVND demandou um maior tempo computacional que o AMBTLit, isto ocorreu devido à complexidade da maior instância do conjunto. O AMBT se comportou similarmente ao ocorrido nos grupos de instâncias anteriores, com resultados próximos aos melhores, porém demandando um alto tempo computacional.

Tabela 4.4. Resultados dos Algoritmos Meméticos SET_1

SET_1														
	MSC		AMBTLit				AMBT				AMRVND			
instância	V	Tamanho	V	Tamanho	T(s)	GAP(%)	V	Tamanho	T(s)	GAP(%)	V	Tamanho	T(s)	GAP(%)
c101	9	9595,6	9	9595,6	24,1	0,00	8 ⁻	9680,4	210,7	0,88	9	9582,7	1,4	-0,13
r101	8	1704,6	8	1704,6	24,0	0,00	8	1722,2	75,9	1,03	9 ⁺	1718,3	4,7	0,80
rc101	8	1674,1	8	1674,1	29,5	0,00	8	1673,9	101,2	-0,01	8	1677,4	3,2	0,20
c201	3	9560,0	3	9560,0	16,2	0,00	3	9560,4	279,5	0,00	3	9562,0	4,6	0,02
r201	2	1643,4	2	1643,4	11,6	0,00	2	1640,9	169,7	-0,15	2	1642,6	6,3	-0,05
rc201	2	1642,7	2	1642,7	12,4	0,00	2	1642,7	112,4	0,00	2	1642,7	5,1	0,00
pr01	2	1412,2	2	1412,2	2,8	0,00	2	1412,2	9,3	0,00	2	1412,2	0,2	0,00
pr02	3	2543,3	3	2543,3	18,1	0,00	3	2547,0	56,1	0,15	3	2549,2	2,9	0,23
pr03	4	3415,1	4	3415,1	48,4	0,00	4	3406,2	370,7	-0,26	4	3421,6	10,7	0,19
pr04	5	4217,4	5	4217,4	165,8	0,00	5	4213,8	770,1	-0,09	5	4218,6	14,7	0,03
pr05	5	4958,7	5	4958,7	331,8	0,00	6 ⁺	4957,3	3772,6	-0,03	6 ⁺	4966,9	46,4	0,17
pr06	7	5963,1	7	5963,1	327,7	0,00	7	5971,5	5065,7	0,14	7	5992,8	115,6	0,50
pr07	3	2070,3	3	2070,3	13,1	0,00	3	2070,3	17,3	0,00	3	2070,3	0,6	0,00
pr08	4	3372,0	4	3372,0	64,9	0,00	4	3372,5	435,6	0,01	4	3366,8	12,1	-0,15
pr09	5	4420,3	5	4420,3	228,0	0,00	5	4417,8	2995,3	-0,06	5	4440,4	19,6	0,45
pr10	7	5940,5	7	5940,5	409,0	0,00	7	5951,0	12359,2	0,18	7	5934,3	88,3	-0,10
Média					108,0	0,00			1675,1	0,11			21,0	0,13

Tabela 4.5. Resultados para AM SET_2 com 10 clientes

SET_2 com 10 clientes														
	MSC		AMBTLit				AMBT				AMRVND			
instância	V	T. Total	V	T. Total	T(s)	GAP(%)	V	T. Total	T(s)	GAP(%)	V	T. Total	T(s)	GAP(%)
c101.k10	1	955,1	1	955,1	0,0	0,00	1	955,1	0,1	0,00	1	955,4	0,0	0,03
r101.k10	2	272,8	2	272,8	0,0	0,00	2	272,8	0,1	0,00	2	272,8	0,0	0,00
rc101.k10	1	237,5	1	237,5	0,0	0,00	1	237,5	0,1	0,00	1	237,5	0,0	0,00
pr01.k10	1	426,6	1	426,6	0,0	0,00	1	426,6	0,1	0,00	1	426,6	0,0	0,00
pr02.k10	1	661,9	1	661,9	0,0	0,00	1	661,9	0,1	0,00	1	661,9	0,0	0,00
pr03.k10	1	553,3	1	553,3	0,0	0,00	1	553,3	0,1	0,00	1	553,3	0,0	0,00
pr04.k10	1	476,4	1	476,4	0,0	0,00	1	476,4	0,1	0,00	1	476,4	0,0	0,00
pr05.k10	1	528,9	1	528,9	0,0	0,00	1	528,9	0,1	0,00	1	528,9	0,0	0,00
pr06.k10	1	597,4	1	597,4	0,0	0,00	1	597,4	0,1	0,00	1	597,4	0,0	0,00
pr07.k10	1	670,2	1	670,2	0,0	0,00	1	670,2	0,1	0,00	1	670,2	0,0	0,00
pr08.k10	1	573,4	1	573,4	0,0	0,00	1	573,4	0,1	0,00	1	573,4	0,0	0,00
pr09.k10	1	645,5	1	645,5	0,0	0,00	1	645,5	0,1	0,00	1	645,5	0,0	0,00
pr10.k10	1	461,5	1	461,5	0,0	0,00	1	461,5	0,1	0,00	1	461,5	0,0	0,00
Média					0,0	0,00			0,1	0,00			0,0	0,00

Tabela 4.6. Resultados para AM SET_2 com 15 clientes

SET_2 com 15 clientes														
	MSC		AMBTLit				AMBT				AMRVND			
instância	V	T. Total	V	T. Total	T(s)	GAP(%)	V	T. Total	T(s)	GAP(%)	V	T. Total	T(s)	GAP(%)
c101.k15	2	1452,2	2	1452,2	0,0	0,00	2	1452,2	0,2	0,00	2	1452,2	0,0	0,00
r101.k15	2	379,8	2	379,8	0,0	0,00	2	379,8	0,2	0,00	2	379,8	0,0	0,00
rc101.k15	2	303,2	2	303,2	0,0	0,00	2	303,2	0,2	0,00	2	303,2	0,0	0,00
pr01.k15	1	590,4	1	590,4	0,0	0,00	1	590,4	0,2	0,00	1	590,4	0,0	0,00
pr02.k15	1	745,6	1	745,6	0,0	0,00	1	745,6	0,2	0,00	1	745,6	0,0	0,00
pr03.k15	1	632,9	1	632,9	0,0	0,00	1	632,9	0,2	0,00	1	632,9	0,0	0,00
pr04.k15	1	683,4	1	683,4	0,0	0,00	1	683,4	0,2	0,00	1	683,4	0,0	0,00
pr05.k15	1	621,2	1	621,2	0,0	0,00	1	621,2	0,2	0,00	1	621,2	0,0	0,00
pr06.k15	1	685,2	1	685,2	0,0	0,00	1	685,2	0,2	0,00	1	685,2	0,0	0,00
pr07.k15	1	795,3	1	795,3	0,0	0,00	1	795,3	0,2	0,00	1	795,3	0,0	0,00
pr08.k15	1	707,2	1	707,2	0,0	0,00	1	707,2	0,2	0,00	1	707,2	0,0	0,00
pr09.k15	1	771,7	1	771,7	0,0	0,00	1	771,7	0,2	0,00	1	771,7	0,0	0,00
pr10.k15	1	611,9	1	611,9	0,0	0,00	1	611,9	0,2	0,00	1	611,9	0,0	0,00
Média					0,0	0,00			0,2	0,00			0,0	0,00

Tabela 4.7. Resultados para AM SET_2 com 30 clientes

SET_2 com 30 clientes														
	MSC		AMBTLit				AMBT				AMRVND			
instância	V	Tamanho	V	Tamanho	T(s)	GAP(%)	V	Tamanho	T(s)	GAP(%)	V	Tamanho	T(s)	GAP(%)
c101.k30	3	2829,4	3	2863,2	0,0	1,19	3	2863,6	3,6	1,21	3	2864,3	0,0	1,23
r101.k30	3	655,2	3	655,2	0,0	0,00	3	656,9	1,6	0,26	3	673,4	0,0	2,78
rc101.k30	3	610,0	3	705,5	0,1	15,66	4	683,8	1,8	12,10	4	683,8	0,0	12,10
pr01.k30	1	964,8	1	964,8	0,0	0,00	1	964,8	1,8	0,00	1	964,8	0,0	0,00
pr02.k30	2	1078,3	2	1078,3	0,0	0,00	2	1086,9	1,9	0,80	2	1080,6	0,0	0,21
pr03.k30	1	952,5	1	952,5	0,0	0,00	1	952,5	1,7	0,00	1	952,5	0,0	0,00
pr04.k30	2	1091,6	2	1091,6	0,0	0,00	2	1091,6	1,0	0,00	2	1091,6	0,0	0,00
pr05.k30	1	924,7	1	924,7	0,0	0,00	1	924,7	1,0	0,00	1	924,7	0,0	0,00
pr06.k30	2	1063,2	2	1063,2	0,0	0,00	2	1063,2	1,0	0,00	2	1063,2	0,0	0,00
pr07.k30	2	1130,4	2	1130,4	0,0	0,00	2	1130,4	1,0	0,00	2	1130,4	0,0	0,00
pr08.k30	2	1006,2	2	1006,2	0,0	0,00	2	1006,2	1,0	0,00	2	1007,6	0,0	0,14
pr09.k30	2	1091,4	2	1091,4	0,0	0,00	2	1091,4	1,2	0,00	2	1091,4	0,0	0,00
pr10.k30	1	918,9	1	918,9	0,0	0,00	1	918,9	1,0	0,00	1	918,9	0,0	0,00
Média					0,0	1,30			1,5	1,10			0,0	1,27

Tabela 4.8. Resultados para AM SET_2 com 40 clientes

SET_2 com 40 clientes														
	MSC		AMBTLit				AMBT				AMRVND			
instância	V	Tamanho	V	Tamanho	T(s)	GAP(%)	V	Tamanho	T(s)	GAP(%)	V	Tamanho	T(s)	GAP(%)
c101.k40	4	3817,4	4	3866,1	2,6	1,28	4	3870,1	3,0	1,38	4	3866,7	0,1	1,29
r101.k40	4	842,9	4	862,8	2,0	2,36	4	862,8	6,1	2,36	4	880,0	0,1	4,40
rc101.k40	3	652,1	3	850,3	2,2	30,39	4	850,3	6,8	30,39	4	850,3	0,0	30,39
pr01.k40	2	1160,5	2	1160,5	0,0	0,00	2	1160,5	2,3	0,00	2	1160,5	0,0	0,00
pr02.k40	2	1336,9	2	1336,9	0,0	0,00	2	1340,1	3,2	0,24	2	1341,2	0,0	0,32
pr03.k40	2	1303,4	2	1303,4	0,0	0,00	2	1303,4	2,8	0,00	2	1303,4	0,0	0,00
pr04.k40	2	1259,5	2	1259,5	0,0	0,00	2	1259,5	2,4	0,00	2	1259,5	0,0	0,00
pr05.k40	2	1200,7	2	1200,7	0,0	0,00	2	1200,7	2,6	0,00	2	1200,7	0,1	0,00
pr06.k40	2	1242,9	2	1242,9	0,1	0,00	2	1242,9	2,6	0,00	2	1242,9	0,0	0,00
pr07.k40	2	1407,0	2	1407,0	0,0	0,00	2	1407,2	2,6	0,01	2	1407,8	0,0	0,06
pr08.k40	2	1222,2	2	1222,2	0,0	0,00	2	1222,2	2,1	0,00	2	1222,2	0,0	0,00
pr09.k40	2	1284,2	2	1284,2	0,0	0,00	2	1284,4	2,7	0,02	2	1284,4	0,0	0,02
pr10.k40	2	1200,4	2	1200,4	0,0	0,00	2	1200,4	2,3	0,00	2	1200,4	0,0	0,00
Média					0,5	2,62			3,2	2,65			0,0	2,81

Tabela 4.9. Resultados para AM SET_3 com 3 hotéis extras

SET_3 com 3 hotéis extras														
	MSC		AMBTLit				AMBT				AMRVND			
instância	V	T. Total	V	T. Total	T(s)	GAP(%)	V	T. Total	T(s)	GAP(%)	V	T. Total	T(s)	GAP(%)
eil51.s3	4	426	4	426	3,8	0,00	4	426	4,6	0,00	4	426	0,1	0,00
berlin52.s3	4	7542	4	7542	3,2	0,00	4	7542	4,8	0,00	4	7542	0,1	0,00
st70.s3	4	675	4	675	7,0	0,00	4	675	16,1	0,00	4	675	0,2	0,00
eil76.s3	4	538	4	538	27,7	0,00	4	538	23,7	0,00	4	538	0,3	0,00
pr76.s3	4	108159	4	108159	14,6	0,00	4	108159	23,1	0,00	4	108159	0,2	0,00
kroA100.s3	4	21282	4	21282	14,2	0,00	4	21282	46,6	0,00	4	21282	0,5	0,00
kroC100.s3	4	20749	4	20749	15,1	0,00	4	20749	44,7	0,00	4	20749	0,6	0,00
kroD100.s3	4	21294	4	21294	15,9	0,00	4	21294	44,1	0,00	4	21294	0,6	0,00
rd100.s3	4	7910	4	7910	15,7	0,00	4	7910	46,5	0,00	4	7910	0,6	0,00
eil101.s3	4	629	4	629	16,9	0,00	4	629	30,6	0,00	4	629	0,5	0,00
lin105.s3	4	14379	4	14379	16,4	0,00	4	14379	51,4	0,00	4	14379	0,9	0,00
ch150.s3	4	6528	4	6528	35,7	0,00	4	6528	141,0	0,00	4	6528	1,9	0,00
tsp225.s3	4	3916	4	3916	93,4	0,00	4	3916	536,8	0,00	4	3916	7,3	0,00
a280.s3	5	2591	5	2591	228,4	0,00	4 ⁻	2583	1534,6	-0,31	5	2594	28,6	0,12
pcb442.s3	4	50778	4	50778	672,1	0,00	4	50778	6352,7	0,00	5 ⁺	50825	137,9	0,09
pr1002.s3	4	259045	5 ⁺	259045	3172,8	0,00	5 ⁺	264143	273211,1	1,97	4	259045	3040,8	0,00
Média					272,1	0,00			17632,0	0,10			201,3	0,01

Tabela 4.10. Resultados para AM SET_3 com 5 hotéis extras

SET_3 com 5 hotéis extras														
	MSC		AMBTLit				AMBT				AMRVND			
instância	V	T. Total	V	T. Total	T(s)	GAP(%)	V	T. Total	T(s)	GAP(%)	V	T. Total	T(s)	GAP(%)
eil51.s5	6	426	6	426	3,5	0,00	6	426	6,3	0,00	6	426	0,1	0,00
berlin52.s5	6	7542	6	7542	3,6	0,00	6	7542	5,3	0,00	6	7542	0,1	0,00
st70.s5	6	675	6	675	7,1	0,00	6	675	12,2	0,00	6	675	0,3	0,00
eil76.s5	6	538	6	538	9,3	0,00	6	538	14,6	0,00	6	538	0,3	0,00
pr76.s5	6	108159	6	108159	8,1	0,00	6	108159	14,2	0,00	6	108159	0,3	0,00
kroA100.s5	6	21282	6	21282	14,5	0,00	6	21282	32,3	0,00	6	21282	0,6	0,00
kroC100.s5	6	20749	6	20749	13,8	0,00	6	20749	31,2	0,00	6	20749	0,6	0,00
kroD100.s5	6	21294	6	21294	14,7	0,00	6	21294	31,2	0,00	6	21294	0,6	0,00
rd100.s5	6	7910	6	7910	13,5	0,00	6	7910	31,1	0,00	6	7910	0,7	0,00
eil101.s5	6	629	6	629	16,3	0,00	6	634	32,1	0,79	6	629	0,7	0,00
lin105.s5	6	14379	6	14379	15,4	0,00	6	14379	35,0	0,00	6	14379	0,8	0,00
ch150.s5	6	6528	6	6528	40,2	0,00	6	6528	107,3	0,00	6	6528	2,1	0,00
tsp225.s5	6	3916	6	3916	86,8	0,00	6	3916	414,1	0,00	6	3916	6,7	0,00
a280.s5	7	2646	7	2646	193,1	0,00	7	2605	831,8	-1,55	7	2651	41,0	0,19
pcb442.s5	6	50778	6	50778	483,2	0,00	6	50778	5859,7	0,00	7 ⁺	50832	185,2	0,11
pr1002.s5	6	259045	7 ⁺	259774	3882,4	0,28	7 ⁺	259365	126003,3	0,12	6	259045	2688,7	0,00
Média					300,3	0,02			8341,4	-0,04			183,1	0,02

Tabela 4.11. Resultados para AM SET_3 com 10 hotéis extras

SET_3 com 10 hotéis extras														
	MSC		AMBTLit				AMBT				AMRVND			
instância	V	Tamanho	V	Tamanho	T(s)	GAP(%)	V	Tamanho	T(s)	GAP(%)	V	Tamanho	T(s)	GAP(%)
eil51.s10	10	426	10	426	3,3	0,00	10	426	6,0	0,00	10	426	0,2	0,00
berlin52.s10	8	7864	8	7864	3,9	0,00	8	7951	20,3	1,11	9 ⁺	7542	0,3	-4,09
st70.s10	10	675	10	675	6,6	0,00	10	675	13,1	0,00	10	675	0,4	0,00
eil76.s10	11	538	11	538	9,0	0,00	11	538	17,9	0,00	11	564	2,4	4,83
pr76.s10	11	108159	11	108159	7,9	0,00	11	108159	17,7	0,00	11	108159	0,6	0,00
kroA100.s10	11	21282	11	21282	14,1	0,00	11	21282	36,0	0,00	11	21282	1,0	0,00
kroC100.s10	11	20749	11	20749	13,8	0,00	11	20749	39,9	0,00	11	20749	1,0	0,00
kroD100.s10	11	21294	11	21294	14,1	0,00	11	21294	41,2	0,00	11	21294	1,0	0,00
rd100.s10	10	7910	10	7910	14,3	0,00	10	7910	47,1	0,00	10	7910	1,0	0,00
eil101.s10	11	629	11	629	15,2	0,00	11	629	42,1	0,00	11	629	1,2	0,00
lin105.s10	10	14379	10	14379	15,3	0,00	10	14379	46,7	0,00	10	14379	1,2	0,00
ch150.s10	11	6528	11	6528	35,8	0,00	11	6528	122,0	0,00	11	6528	2,7	0,00
tsp225.s10	11	3916	11	3916	79,9	0,00	11	3916	422,0	0,00	11	3916	9,6	0,00
a280.s10	11	2613	11	2613	134,1	0,00	12 ⁺	2612	933,9	-0,04	12 ⁺	2590	19,1	-0,88
pcb442.s10	11	51774	11	51774	511,4	0,00	12 ⁺	50919	5006,2	-1,65	12 ⁺	50919	181,6	-1,65
pr1002.s10	11	259045	11	259045	3224,1	0,00	12 ⁺	267049	159397,4	3,09	11	259045	5386,3	0,00
Média					256,4	0,0			10388,1	0,16			350,6	-0,11

Tabela 4.12. Resultados Algoritmos Meméticos para SET_4

SET_4														
	MSC		AMBTLit				AMBT				AMRVND			
instância	V	Tamanho	V	Tamanho	T(s)	GAP(%)	V	Tamanho	T(s)	GAP(%)	V	Tamanho	T(s)	GAP(%)
eil51.s4	6	429	6	429	3,8	0,00	6	429	12,1	0,00	6	438	0,4	2,10
berlin52.s4	7	8642	7	8642	4,2	0,00	7	8642	13,9	0,00	7	8651	0,3	0,10
st70.s4	6	723	6	723	8,5	0,00	7 ⁺	723	53,6	0,00	7 ⁺	719	1,7	-0,55
eil76.s4	6	539	6	548	12,4	1,67	6	539	79,0	0,00	6	539	1,3	0,00
pr76.s4	6	118061	6	118061	8,2	0,00	7 ⁺	116046	122,4	-1,71	7 ⁺	116021	1,5	-1,73
kroA100.s4	6	22343	6	22343	19,2	0,00	6	22138	163,7	-0,92	6	22074	2,8	-1,20
kroC100.s4	6	20933	6	20933	12,8	0,00	6	20933	52,4	0,00	6	20933	2,9	0,00
kroD100.s4	6	21464	6	21664	17,3	0,93	6	21464	58,6	0,00	6	21554	4,5	0,42
rd100.s4	6	8244	6	8244	24,2	0,00	6	8255	66,2	0,13	6	8401	2,2	1,90
eil101.s4	6	634	6	634	22,8	0,00	6	634	112,2	0,00	6	637	3,6	0,47
ch150.s4	6	6647	6	6647	54,7	0,00	6	6562	688,9	-1,28	6	6625	5,4	-0,33
tsp225.s4	6	4502	6	4571	118,7	1,53	7 ⁺	4859	3311,1	7,93	7 ⁺	4690	55,3	4,18
a280.s4	6	2646	6	2646	158,7	0,00	6	2690	4445,4	1,66	6	2731	52,2	3,21
pcb442.s4	6	54339	6	54339	872,5	0,00	6	58683	39336,2	7,99	6	55742	910,3	2,58
pr1002.s4	7	290110	7	292690	3423,4	0,89	7	297147	271148,0	2,43	7	299967	16865,3	3,40
média					317,4	0,33			21310,9	1,08			1193,9	0,97

Após o desenvolvimento das abordagens que utilizam Algoritmo Memético, um novo trabalho surgiu na literatura [Castro et al., 2014]. Nesse trabalho, o algoritmo que combina um *Iterated Local Search* com procedimentos de perturbação (PLS) apresentado, demanda um tempo computacional inferior à todos os trabalhos contidos na literatura até o momento. Além do seu tempo computacional reduzido, suas soluções apresentam-se competitivas e o seu GAP máximo não ultrapassou 3% para nenhuma das instâncias. Com o intuito de comparar o PLS com um algoritmo que demandasse um tempo computacional similar, foi desenvolvido o algoritmo IGVND.

Para as tabelas a seguir, a primeira e segunda linha de cada tabela identificam o grupo de instâncias e as respectivas abordagens heurísticas que foram utilizadas. A primeira coluna representa o nome de cada instância. A segunda, quarta e oitava coluna definem a quantidade de viagens necessárias para percorrer a rota. A terceira, quinta e nona coluna contém o tempo total necessário para percorrer a rota. A sexta e décima coluna correspondem ao tempo computacional que cada algoritmo demandou para cada instância. Por fim, a sétima e décima primeira coluna indicam a diferença percentual entre a solução encontrada pela abordagem em relação a Melhor Solução Conhecida (MSC).

Para o SET_1 apresentado na Tabela 4.13, o algoritmo IGVND proposto apresentou-se competitivo em relação ao PLS. A média do GAP observada para o PLS foi de 0,34%, enquanto a do IGVND ficou em 0,76%. Além da pequena diferença entre os GAPs médios, o IGVND ainda conseguiu melhorar a solução para a instância *pr04*. Quanto a diferença de tempo computacional, apesar do algoritmo proposto demandar mais tempo para o processamento das instâncias, esta diferença pode ser considerada viável para ser utilizada em problemas que precisam ser solucionados em tempo real. Para a instância *pr05* tanto o PLS quanto o IGVND necessitaram de uma viagem a mais para percorrer a rota.

Os resultados obtidos na Tabela 4.14, Tabela 4.15, Tabela 4.16 e Tabela 4.17 estão relacionados ao grupo SET_2. Os resultados demonstram que o IGVND é capaz de encontrar a maioria das melhores soluções conhecidas instantaneamente. Ao comparar o PLS com IGVND os algoritmos apresentam um GAP médio de no máximo 2,98% e 2,90% respectivamente. Quanto ao número de viagens o IGVND em apenas duas instâncias aumenta a quantidade necessária de viagens em uma unidade, já o PLS aumentou o número de viagens para duas instâncias, acrescentando para uma delas duas viagens extras.

Tabela 4.13. Resultados para o Algoritmo *Iterated Greedy* SET_1

SET_1										
	MSC		PLS				IG			
instância	Viagens	T. Total	Viagens	T. Total	Tempo(s)	GAP(%)	Viagens	T. Total	Tempo(s)	GAP(%)
c101	9	9595,6	9	9596,9	0,1	0,01	9	9595,6	0,5	0,00
r101	8	1704,6	8	1717,4	0,2	0,75	8	1761,9	0,6	3,36
rc101	8	1674,1	8	1674,3	0,2	0,01	8	1694,8	0,5	1,24
c201	3	9560,0	3	9563,1	0,1	0,03	3	9563,1	0,6	0,03
r201	2	1643,4	2	1648,1	0,1	0,29	2	1648,7	0,3	0,32
rc201	2	1642,7	2	1644,3	0,2	0,10	2	1644,7	0,3	0,12
pr01	2	1412,2	2	1412,2	0,0	0,00	2	1412,2	0,0	0,00
pr02	3	2543,3	3	2551,3	0,2	0,31	3	2544,8	0,6	0,06
pr03	4	3415,1	4	3421,1	0,3	0,18	4	3441,8	2,2	0,78
pr04	5	4217,4	5	4217,4	0,6	0,00	5	4214,6	3,1	-0,07
pr05	5	4958,7	6 ⁺	4974,7	1,1	0,32	6 ⁺	5023,1	10,4	1,30
pr06	7	5963,1	7	6032,0	1,6	1,16	7	6022,2	11,0	0,99
pr07	3	2070,3	3	2070,3	0,0	0,00	3	2070,3	0,3	0,00
pr08	4	3372,0	4	3399,9	0,4	0,83	4	3388,5	1,9	0,49
pr09	5	4420,3	5	4445,7	1,1	0,57	5	4505,9	5,4	1,94
pr10	7	5940,5	7	5991,5	2,4	0,86	7	6032,5	12,1	1,55
Média					0,5	0,34			3,1	0,76

Tabela 4.14. Resultados Algoritmo *Iterated Greedy* SET_2 com 10 clientes

SET_2 com 10 clientes										
	MSC		PLS				IG			
instância	Viagens	T. Total	Viagens	T. Total	Tempo(s)	GAP(%)	Viagens	T. Total	Tempo(s)	GAP(%)
c101.k10	1	955,1	1	955,1	0,0	0,00	1	955,4	0,0	0,03
r101.k10	2	272,8	2	272,8	0,0	0,00	2	272,8	0,0	0,00
rc101.k10	1	237,5	1	237,5	0,0	0,00	1	237,7	0,0	0,08
pr01.k10	1	426,6	1	426,6	0,0	0,00	1	426,6	0,0	0,00
pr02.k10	1	661,9	1	661,9	0,0	0,00	1	661,9	0,0	0,00
pr03.k10	1	553,3	1	553,3	0,0	0,00	1	553,3	0,0	0,00
pr04.k10	1	476,4	1	476,4	0,0	0,00	1	476,4	0,0	0,00
pr05.k10	1	528,9	1	528,9	0,0	0,00	1	528,9	0,0	0,00
pr06.k10	1	597,4	1	597,4	0,0	0,00	1	597,4	0,0	0,00
pr07.k10	1	670,2	1	670,2	0,0	0,00	1	670,2	0,0	0,00
pr08.k10	1	573,4	1	573,4	0,0	0,00	1	573,4	0,0	0,00
pr09.k10	1	645,5	1	645,5	0,0	0,00	1	645,5	0,0	0,00
pr10.k10	1	461,5	1	461,5	0,0	0,00	1	461,5	0,0	0,00
Média					0,0	0,00			0,0	0,01

Tabela 4.15. Resultados Algoritmo *Iterated Greedy* SET_2 com 15 clientes

SET_2 com 15 clientes										
	MSC		PLS				IG			
instância	Viagens	T. Total	Viagens	T. Total	Tempo(s)	GAP(%)	Viagens	T. Total	Tempo(s)	GAP(%)
c101.k15	2	1452,2	2	1452,2	0,0	0,00	2	1452,2	0,0	0,00
r101.k15	2	379,8	2	379,8	0,0	0,00	2	391,5	0,0	3,08
rc101.k15	2	303,2	2	303,2	0,0	0,00	2	303,2	0,0	0,00
pr01.k15	1	590,4	1	590,4	0,0	0,00	1	590,4	0,0	0,00
pr02.k15	1	745,6	1	745,6	0,0	0,00	1	745,6	0,0	0,00
pr03.k15	1	632,9	1	632,9	0,0	0,00	1	632,9	0,0	0,00
pr04.k15	1	683,4	1	683,4	0,0	0,00	1	683,4	0,0	0,00
pr05.k15	1	621,2	1	621,2	0,0	0,00	1	621,4	0,0	0,03
pr06.k15	1	685,2	1	685,2	0,0	0,00	1	685,2	0,0	0,00
pr07.k15	1	795,3	1	795,3	0,0	0,00	1	795,3	0,0	0,00
pr08.k15	1	707,2	1	707,2	0,0	0,00	1	707,2	0,0	0,00
pr09.k15	1	771,7	1	771,7	0,0	0,00	1	772,9	0,0	0,16
pr10.k15	1	611,9	1	611,9	0,0	0,00	1	611,9	0,0	0,00
Média					0,0	0,0			0,0	0,25

Tabela 4.16. Resultados Algoritmo *Iterated Greedy* SET_2 com 30 clientes

SET_2 com 30 clientes										
	MSC		PLS				IG			
instância	Viagens	Tamanho	Viagens	Tamanho	Tempo(s)	GAP	Viagens	Tamanho	Tempo(s)	GAP
c101.k30	3	2829,4	3	2863,6	0,0	1,21	3	2864,3	0,0	1,23
r101.k30	3	655,2	3	655,2	0,0	0,00	3	657,6	0,0	0,37
rc101.k30	3	610,0	4 ⁺	683,8	0,0	12,10	4 ⁺	684,4	0,0	12,20
pr01.k30	1	964,8	1	964,8	0,0	0,00	1	964,8	0,0	0,00
pr02.k30	2	1078,3	2	1078,3	0,0	0,00	2	1078,3	0,0	0,00
pr03.k30	1	952,5	1	952,5	0,0	0,00	1	952,5	0,0	0,00
pr04.k30	2	1091,6	2	1091,6	0,0	0,00	2	1091,6	0,0	0,00
pr05.k30	1	924,7	1	924,7	0,0	0,00	1	924,7	0,0	0,00
pr06.k30	2	1063,2	2	1063,2	0,0	0,00	2	1063,2	0,0	0,00
pr07.k30	2	1130,4	2	1130,4	0,0	0,00	2	1130,4	0,0	0,00
pr08.k30	2	1006,2	2	1006,2	0,0	0,00	2	1006,2	0,0	0,00
pr09.k30	2	1091,4	2	1091,4	0,0	0,00	2	1091,4	0,0	0,00
pr10.k30	1	918,9	1	918,9	0,0	0,00	1	918,9	0,0	0,00
Média					0,0	1,02			0,0	1,06

Tabela 4.17. Resultados Algoritmo *Iterated Greedy* SET_2 com 40 clientes

SET_2 com 40 clientes										
	MSC		PLS				IG			
instância	Viagens	Tamanho	Viagens	Tamanho	Tempo(s)	GAP	Viagens	Tamanho	Tempo(s)	GAP
c101.k40	4	3817,4	4	3866,1	0,0	1,28	4	3866,4	0,0	1,28
r101.k40	4	842,9	4	873,5	0,0	3,63	4	882,4	0,0	4,69
rc101.k40	3	652,1	5 ⁺⁺	870,8	0,0	33,54	4 ⁺	850,9	0,0	30,49
pr01.k40	2	1160,5	2	1160,5	0,0	0,00	2	1160,5	0,0	0,00
pr02.k40	2	1336,9	2	1336,9	0,0	0,00	2	1342,3	0,0	0,40
pr03.k40	2	1303,4	2	1303,4	0,0	0,00	2	1303,4	0,0	0,00
pr04.k40	2	1259,5	2	1259,5	0,0	0,00	2	1259,5	0,0	0,00
pr05.k40	2	1200,7	2	1200,7	0,0	0,00	2	1201,1	0,0	0,03
pr06.k40	2	1242,9	2	1242,9	0,0	0,00	2	1242,9	0,0	0,00
pr07.k40	2	1407,0	2	1410,3	0,0	0,23	2	1417,8	0,0	0,77
pr08.k40	2	1222,2	2	1222,2	0,0	0,00	2	1222,2	0,0	0,00
pr09.k40	2	1284,2	2	1284,4	0,0	0,02	2	1284,7	0,0	0,04
pr10.k40	2	1200,4	2	1200,4	0,0	0,00	2	1200,4	0,0	0,00
Média					0,0	2,98			0,0	2,90

Os resultados apresentados na Tabela 4.18 foram obtidos utilizando as instâncias definidas no grupo SET_3 com 3 hotéis extras. Observou-se que o algoritmo proposto apresenta um GAP médio menor que do PLS e ainda consegue melhorar uma das soluções diminuindo o número de viagens e também o tempo total da rota. Foi aumentado o número de viagens em duas instâncias do PLS e uma do IGVND. Quanto ao tempo computacional, o algoritmo proposto é rápido para a maioria das instâncias, demandando um tempo maior apenas para as duas que apresentam uma quantidade maior de clientes.

Para o grupo SET_3 com 5 hotéis extras os resultados são apresentados na Tabela 4.19. Para este conjunto de instâncias o algoritmo proposto alcança bons resultados quando comparados aos melhores conhecidos, sendo que para a instância *a280.s5* conseguiu melhorar mais de 2% a melhor solução conhecida, mantendo o mesmo número de viagens. Ao comparar o GAP médio entre o IGVND e o PLS, observa-se que as soluções obtidas pelo algoritmo proposto são em média 0,51% melhores que as encontradas pelo algoritmo PLS. Em relação a quantidade de viagens necessárias para realizar cada rota, o IGVND não necessitou de nenhuma viagem extra, enquanto o PLS teve acréscimo para uma das instâncias.

O último conjunto de instâncias que pertence ao grupo SET_3 possui como característica o acréscimo de 10 hotéis extras e seus resultados são apresentados na Tabela 4.20. Os resultados encontrados demonstram que o algoritmo proposto consegue soluções similares ao PLS para todas as instâncias. Ao analisar o GAP médio é possível constatar que o IGVND consegue soluções melhores que o PLS a uma porcentagem média de 0,27%. Quando comparado o número de viagens necessárias para percorrer as rotas, para três instâncias o algoritmo proposto obteve um número maior de viagens que o apresentado pela melhor solução conhecida. Para o PLS, em quatro instâncias foi aumentado o número de viagens necessárias. Novamente, o tempo computacional médio do IGVND elevou-se pelo grande tempo demandando nas duas últimas instâncias.

Para o grupo SET_4, considerado de difícil resolução os resultados são apresentados na Tabela 4.21. Neste grupo, o PLS encontrou resultados melhores que o algoritmo proposto, obtendo uma porcentagem de melhora de mais de 4%. Quanto a quantidade de viagens o IGVND e o PLS conseguem na maioria das instâncias encontrar soluções que demandam a mesma quantia de viagens que a melhor solução conhecida. Apenas em três instâncias foi observado aumento no número de viagens. Em relação ao tempo computacional, a maioria das instâncias são executadas quase instantaneamente, apenas as duas últimas demandam um tempo computacional maior.

Tabela 4.18. Resultados Algoritmo *Iterated Greedy* SET_3 com 3 hotéis extras

SET_3 com 3 hotéis extras										
	MSC		PLS				IG			
instância	Viagens	T. Total	Viagens	T. Total	Tempo(s)	GAP(%)	Viagens	T. Total	Tempo(s)	GAP(%)
eil51.s3	4	426	4	426	0,0	0,00	4	426	0,1	0,00
berlin52.s3	4	7542	4	7542	0,0	0,00	4	7542	0,1	0,00
st70.s3	4	675	4	675	0,0	0,00	4	675	0,2	0,00
eil76.s3	4	538	5 ⁺	556	0,0	3,35	4	538	0,3	0,00
pr76.s3	4	108159	4	108159	0,1	0,00	4	108159	0,5	0,00
kroA100.s3	4	21282	4	21282	0,1	0,00	4	21282	0,6	0,00
kroC100.s3	4	20749	4	20749	0,0	0,00	4	20749	0,6	0,00
kroD100.s3	4	21294	4	21294	0,1	0,00	4	21294	0,7	0,00
rd100.s3	4	7910	4	7910	0,1	0,00	4	7910	0,5	0,00
eil101.s3	4	629	4	629	0,1	0,00	4	629	0,4	0,00
lin105.s3	4	14379	4	14379	0,1	0,00	4	14379	0,8	0,00
ch150.s3	4	6528	4	6528	0,2	0,00	4	6528	1,6	0,00
tsp225.s3	4	3916	4	3916	0,5	0,00	4	3916	5,1	0,00
a280.s3	5	2591	5	2615	0,8	0,93	4 ⁻	2579	13,8	-0,46
pcb442.s3	4	50778	5 ⁺	51144	3,7	0,72	5 ⁺	51137	48,4	0,71
pr1002.s3	4	259045	4	259045	34,5	0,00	4	259045	1178,3	0,00
Média					2,5	0,31			78,2	0,02

Tabela 4.19. Resultados Algoritmo *Iterated Greedy* SET_3 com 5 hotéis extras

SET_3 com 5 hotéis extras										
	MSC		PLS				IG			
instância	Viagens	Tamanho	Viagens	Tamanho	Tempo(s)	GAP(%)	Viagens	Tamanho	Tempo(s)	GAP(%)
eil51.s5	6	426	6	426	0,0	0,00	6	426	0,1	0,00
berlin52.s5	6	7542	6	7542	0,0	0,00	6	7542	0,1	0,00
st70.s5	6	675	6	675	0,0	0,00	6	675	0,2	0,00
eil76.s5	6	538	6	566	0,1	5,20	6	538	0,2	0,00
pr76.s5	6	108159	6	108159	0,1	0,00	6	108159	0,4	0,00
kroA100.s5	6	21282	6	21282	0,1	0,00	6	21282	0,6	0,00
kroC100.s5	6	20749	6	20749	0,1	0,00	6	20749	0,7	0,00
kroD100.s5	6	21294	6	21294	0,1	0,00	6	21294	0,7	0,00
rd100.s5	6	7910	6	7910	0,1	0,00	6	7910	0,5	0,00
eil101.s5	6	629	6	629	0,1	0,00	6	629	0,5	0,00
lin105.s5	6	14379	6	14379	0,1	0,00	6	14379	0,6	0,00
ch150.s5	6	6528	6	6528	0,2	0,00	6	6528	1,6	0,00
tsp225.s5	6	3916	6	3916	0,4	0,00	6	3916	5,4	0,00
a280.s5	7	2646	7	2652	0,7	0,23	7	2590	9,9	-2,12
pcb442.s5	6	50778	7 ⁺	51087	3,1	0,61	6	50778	47,4	0,00
pr1002.s5	6	259045	6	259045	20,5	0,00	6	259045	1020,3	0,00
Média					1,6	0,38			68,1	-0,13

Tabela 4.20. Resultados Algoritmo *Iterated Greedy* SET_3 com 10 hotéis extras

SET_3 com 10 hotéis extras										
	MSC		PLS				IG			
instância	Viagens	T. Total	Viagens	T. Total	Tempo(s)	GAP(%)	Viagens	T. Total	Tempo(s)	GAP(%)
eil51.s10	10	426	10	426	0,0	0,00	10	426	0,1	0,00
berlin52.s10	8	7864	9 ⁺	7542	0,0	-4,09	9 ⁺	7542	0,1	-4,09
st70.s10	10	675	10	675	0,0	0,00	10	675	0,2	0,00
eil76.s10	11	538	12 ⁺	567	0,1	5,39	11	538	0,3	0,00
pr76.s10	11	108159	11	108159	0,1	0,00	11	108159	0,3	0,00
kroA100.s10	11	21282	11	21282	0,1	0,00	11	21282	1,0	0,00
kroC100.s10	11	20749	11	20749	0,1	0,00	11	20749	0,7	0,00
kroD100.s10	11	21294	11	21294	0,1	0,00	11	21294	0,6	0,00
rd100.s10	10	7910	10	7910	0,1	0,00	10	7910	0,6	0,00
eil101.s10	11	629	11	629	0,1	0,00	11	629	0,6	0,00
lin105.s10	10	14379	10	14379	0,1	0,00	10	14379	0,6	0,00
ch150.s10	11	6528	11	6528	0,2	0,00	11	6528	1,8	0,00
tsp225.s10	11	3916	11	3916	0,4	0,00	11	3916	3,9	0,00
a280.s10	11	2613	12 ⁺	2596	0,7	-0,65	12 ⁺	2611	9,5	-0,08
pcb442.s10	11	51774	12 ⁺	50919	2,6	-1,65	12 ⁺	51195	61,0	-1,12
pr1002.s10	11	259045	11	259045	11,9	0,00	11	259173	982,1	0,05
Média					1,0	-0,06			66,5	-0,33

Tabela 4.21. Resultados Algoritmo *Iterated Greedy* SET_4

SET_4										
	MSC		PLS				IG			
instância	Viagens	T. Total	Viagens	T. Total	Tempo(s)	GAP(%)	Viagens	T. Total	Tempo(s)	GAP(%)
eil51.s4	6	429	6	436	0,0	1,63	6	433	0,1	0,93
berlin52.s4	7	8642	7	8642	0,0	0,00	7	9058	0,1	4,81
st70.s4	6	723	6	731	0,0	1,11	6	737	0,4	1,94
eil76.s4	6	539	6	539	0,0	0,00	6	568	0,6	5,38
pr76.s4	6	118061	7 ⁺	118719	0,1	0,56	7 ⁺	121421	1,1	2,85
kroA100.s4	6	22343	7 ⁺	22044	0,1	-1,34	7 ⁺	22116	0,9	-1,02
kroC100.s4	6	20933	6	21116	0,1	0,87	6	23088	1,4	10,29
kroD100.s4	6	21464	6	21464	0,1	0,00	6	23138	0,8	7,80
rd100.s4	6	8244	7 ⁺	8245	0,1	0,01	6	8379	1,4	1,64
eil101.s4	6	634	6	652	0,1	2,84	6	661	1,0	4,26
ch150.s4	6	6647	6	6728	0,2	1,22	6	6738	2,1	1,37
tsp225.s4	6	4502	6	4502	0,9	0,00	7 ⁺	4867	8,2	8,11
a280.s4	6	2646	6	2658	0,9	0,45	6	2857	12,9	7,97
pcb442.s4	6	54339	6	55134	5,7	1,46	6	59468	72,1	9,44
pr1002.s4	7	290110	7	290110	29,5	0,00	7	323102	2007,8	11,37
média					2,5	0,59			140,7	5,14

Com o objetivo de avaliar probabilisticamente os resultados obtidos por cada heurística, uma análise de desempenho empírico foi realizada. Esta análise é importante quando as heurísticas possuem componentes aleatórios que podem fazer com que sejam geradas soluções distintas a cada execução de uma dada instância. Este formato de análise foi proposto por Aiex et al. [2002] e consiste em capturar os tempos de processamento que cada heurística requer para atingir um valor definido como alvo na função objetivo. Assim, no momento em que a heurística encontra uma solução que seja menor ou igual (para problemas de minimização) ao valor alvo fixado, o tempo é registrado e a execução da heurística interrompida. Caso o valor alvo não seja encontrado, a heurística finalizará sua execução após uma quantidade de tempo definida muito acima do tempo computacional observado normalmente para execução de tais instâncias.

Na realização dos testes, cada heurística foi executada 100 vezes para cada instância avaliada. Após realizadas as 100 execuções, os tempos foram apanhados e dispostos em ordem crescente numa lista Q . A cada tempo computacional t obtido, foi associada a probabilidade $p_i = (i - \frac{1}{2})/100$, onde i define a ordem que t aparece na lista ordenada Q . Em seguida, foi realizada a disposição das informações no gráfico tomando-se cada ponto (t_i, p_i) . Os resultados do AMBTLit e PLS foram encontrados pela reimplementação dos algoritmos da literatura e os demais resultados são provenientes das abordagens propostas.

Para plotagem dos gráficos foi utilizado o *time-to-target plots* que mostram no eixo das ordenadas a probabilidade de uma determinada heurística encontrar uma solução tão boa quanto a solução alvo, considerando o tempo computacional necessário. Esta técnica de análise e comparação é utilizada em vários trabalhos de otimização combinatória, como por exemplo em Aiex et al. [2002]; Gonçalves [2005]; Resende & Ribeiro [2014]. O tempo computacional necessário é definido no eixo das abscissas. Para conduzir os experimentos a seguir, foram consideradas três instâncias distintas, escolhidas dentre as existentes e classificadas de acordo com sua dificuldade. Os diferentes níveis de dificuldade das instâncias utilizadas são definidos por fácil, médio e difícil. A instância considerada de fácil tratamento é a *eil51.s5* pertencente ao grupo SET_3 com 5 hotéis extras. A segunda instância *r201* classificada como média, pertence ao grupo SET_1. E por fim, a instância *pcb442.s4* pertencente ao grupo SET_4 foi utilizada como uma instância de difícil tratamento. O valor considerado alvo foi definido para todas as instâncias com base no melhor tempo total necessário para percorrer a rota do PCVSH contido na literatura.

Para a primeira instância o valor alvo foi encontrado por todas as heurísticas

tratadas nesta dissertação pelo menos uma vez. Os resultados obtidos pela execução da instância *eil51.s5* são apresentados na Figura 4.10. Para esta instância o valor alvo definido foi a melhor solução conhecida na literatura.

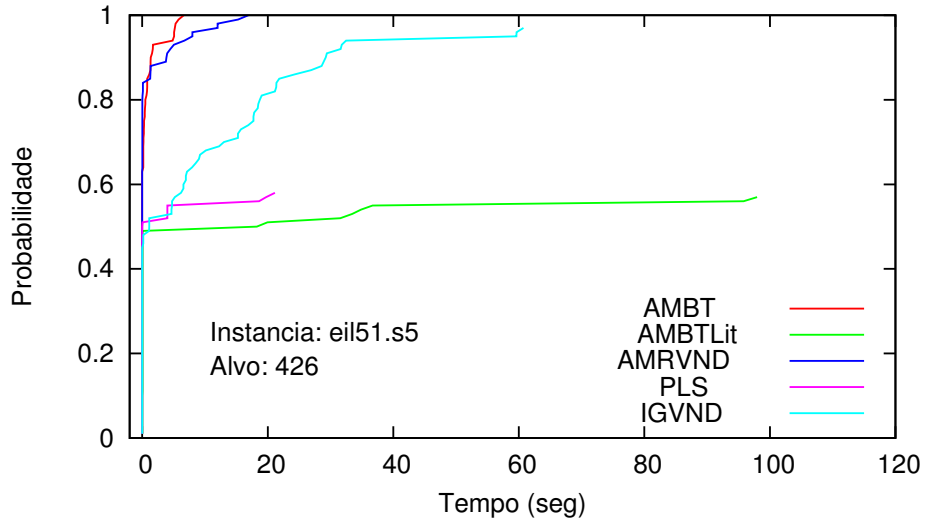


Figura 4.10. Gráfico de probabilidade de alcance do alvo definido para a instância *eil51.s5*.

Analisando o gráfico da Figura 4.10 é possível concluir que as heurísticas que possuem a maior probabilidade de encontrar a solução alvo são AMBT, AMRVND e IGVND. A heurística AMBT obteve os melhores resultados, encontrando a solução alvo com uma probabilidade de mais de 80% e tempo computacional quase instantâneo. Quanto a heurística AMBTLit, sua probabilidade de encontrar a solução alvo foi de aproximadamente 50%. De forma similar, a heurística PLS possui probabilidade de pouco mais de 50% no encontro à solução alvo. A segunda bateria de testes para a plotagem dos gráficos de probabilidade foi realizada utilizando-se a instância *r201* do grupo SET_1 e os resultados obtidos são apresentados na Figura 4.11.

O alvo para a instância *r201* foi definido utilizando uma porcentagem de aproximadamente 5% a mais em relação a melhor solução conhecida na literatura. A heurística IGVND teve para todas suas execuções um tempo computacional próximo a zero, sendo sua curva de probabilidade plotada quase sobre o eixo das ordenadas ($x = 0$). Neste teste, todas heurísticas foram capazes de encontrar o valor alvo definido para todas as execuções realizadas. A heurística PLS consegue encontrar o alvo quase instantaneamente para mais de 70% das execuções, porém necessita de uma grande porção de tempo computacional para atingir o alvo no restante das

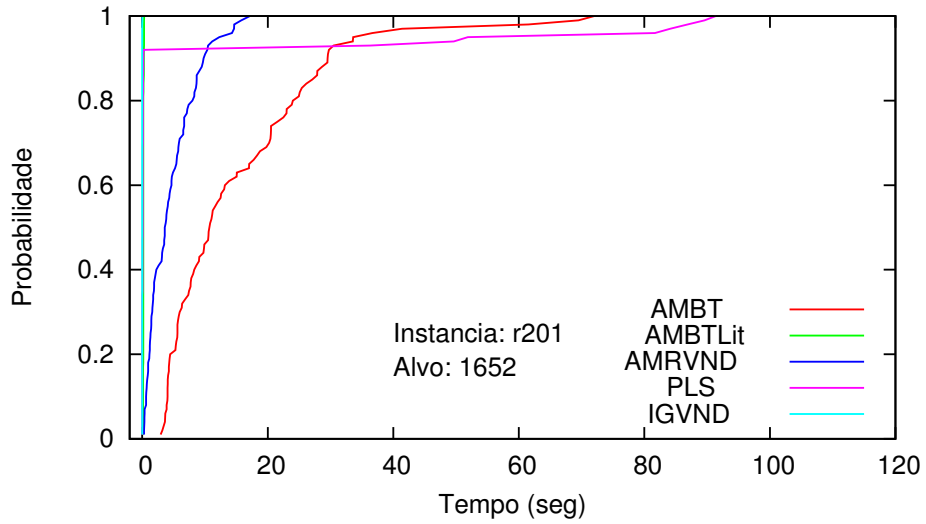


Figura 4.11. Gráfico de probabilidade de alcance do alvo definido para a instância *r201*.

execuções. Ambas heurísticas populacionais propostas apresentam uma curva de probabilidade similar, mas a AMRVND demanda menor tempo computacional.

Finalizando, temos na Figura 4.12 a ilustração dos resultados obtidos pela execução da instância *pcb442.s4*. O valor alvo foi definido acrescentando-se ao tempo total da rota uma quantia de aproximadamente 10% em relação ao melhor valor conhecido na literatura. Para esta instância considerada de difícil tratamento, somente as heurísticas AMRVND e IGVND conseguiram alcançar o alvo para todas as execuções. A heurística PLS consegue alcançar o valor alvo com uma probabilidade de 40% e o AMBTLit de 10%. Apesar da grande quantidade de tempo disponibilizada para a execução das heurísticas, o AMBT conseguiu encontrar o valor alvo em apenas 2%, sendo que o alvo somente foi alcançado próximo ao término do tempo estabelecido.

Foi realizado também um experimento com intuito de analisar a convergência das heurísticas propostas e das heurísticas da literatura, que foram reimplementadas. Neste experimento foi registrado o valor da função objetivo obtido por cada heurística durante sua execução, considerando as melhoras que foram feitas a cada iteração.

Cada uma das abordagens heurísticas foi executada 10 vezes, sendo o critério de parada definido de forma igualitária entre as abordagens. Os respectivos tempos máximos de execução foram assim definidos: 2 minutos para as instâncias classificadas como fácil e média e 2 horas para a instância classificada como difícil. A execução que levou ao melhor resultado, entre as 10 execuções, foi utilizada para

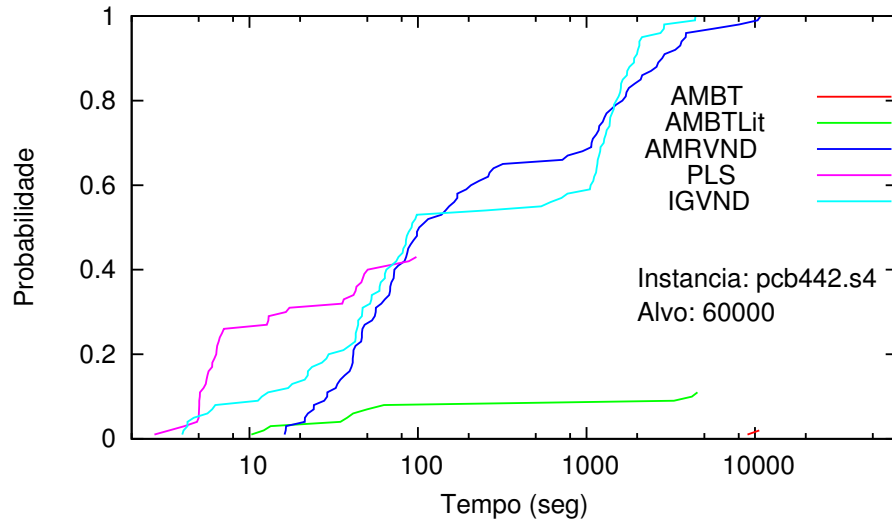


Figura 4.12. Gráfico de probabilidade de alcance do alvo definido para a instância *pcb442.s4*.

análise e plotagem dos gráficos.

A Figura 4.13, Figura 4.14 e Figura 4.15 ilustram para cada uma das abordagens comparadas, a curva de convergência da solução. O valor da função objetivo é definido no eixo das ordenadas e o tempo computacional é definido no eixo das abscissas. Foi adicionada uma linha pontilhada que representa o melhor valor da função objetivo conhecida, sendo que o tempo computacional neste caso é desconsiderado.

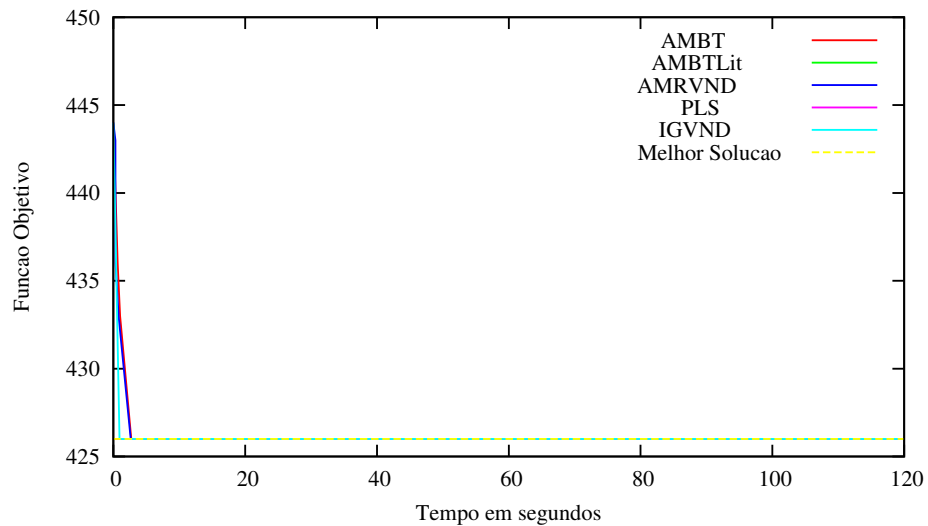


Figura 4.13. Gráfico de convergência da solução para todas as heurísticas comparadas, considerando a instância *eil51.s5*.

Na Figura 4.13 é ilustrado o gráfico de convergência da solução para a instân-

cia de fácil tratamento. É possível observar que todas as heurísticas convergem para a melhor solução conhecida necessitando de pouco tempo computacional. Os resultados apresentados na Figura 4.14 foram obtidos a partir da instância *r201* e pelo fato de ser uma instância considerada de dificuldade média, nem todas as heurísticas foram capazes de convergir para a melhor solução conhecida. Um comportamento interessante foi identificado neste teste, pois as duas heurísticas propostas AMBT e AMRVND conseguiram melhorar a qualidade da solução. Esta melhora também foi observada nos testes computacionais, tendo os resultados apresentados na Tabela 4.4. A Heurística que convergiu mais rápido para uma melhor solução foi o AMRVND, sendo cerca de 10 segundos mais rápida que o AMBT no quesito convergência.

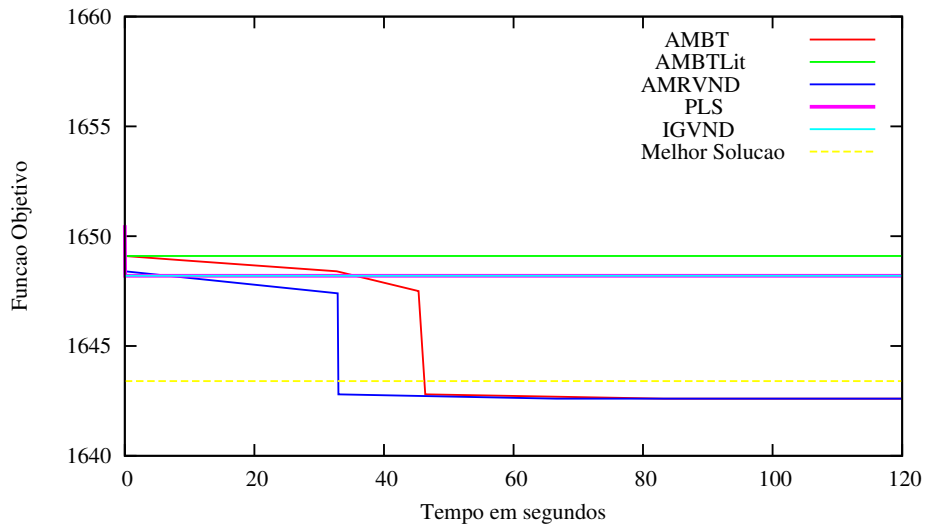


Figura 4.14. Gráfico de convergência da solução para todas as heurísticas comparadas, considerando a instância *r201*.

O último teste com o objetivo de analisar a convergência da solução em relação ao tempo computacional demandado, foi realizado com a instância *pcb442.s4* que possui grande dificuldade para ser tratada. Na Figura 4.15 são ilustradas as curvas de convergências das heurísticas, sendo possível observar que o AMRVND foi a heurística que conseguiu aproximar-se mais da melhor solução conhecida. A maior parte de sua melhora foi obtida antes do término de 25% do tempo computacional disponível. Para esta instância, as heurísticas que apresentaram pior desempenho foram a AMBTLit e AMBT. Os resultados obtidos neste teste ilustram de forma gráfica os resultados apresentados na Tabela 4.12 e Tabela 4.21.

Com base em todos os testes computacionais realizados, as conclusões acerca do trabalho desenvolvido nesta dissertação são apresentadas no capítulo a seguir.

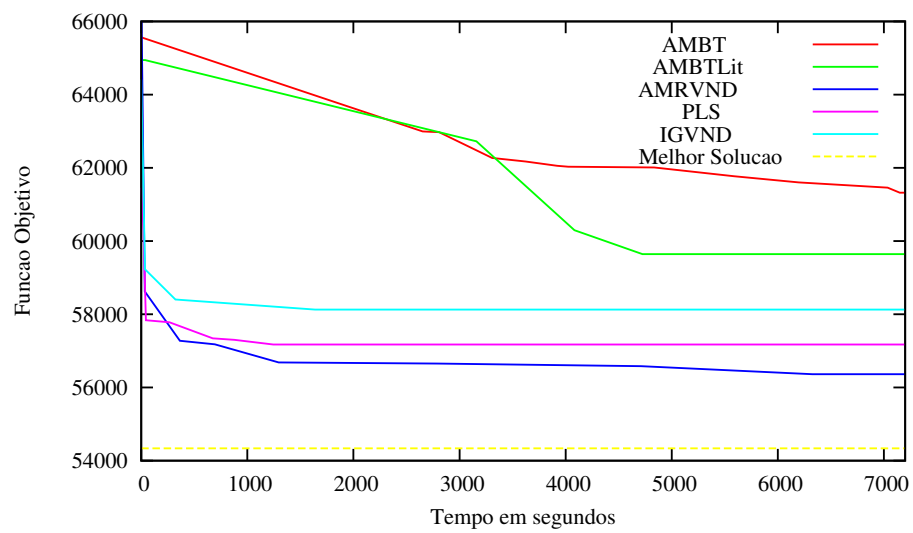


Figura 4.15. Gráfico de convergência da solução para todas as heurísticas comparadas, considerando a instância *pcb442.s4*.

Capítulo 5

CONCLUSÕES

Neste trabalho foi apresentado um estudo de uma variante do Problema do Caixeiro Viajante (PCV) aqui denotado como Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH).

O objetivo do PCVSH é atender todos os seus clientes levando em consideração que há um tempo limite para sua jornada de trabalho. Caso não seja possível atender a todos os clientes no mesmo dia, o caixeiro deve escolher um hotel para descansar e então, no dia seguinte, continuar seu trabalho a partir deste hotel, até que retorne ao seu local de partida.

Com o intuito de criar um modelo matemático alternativo, foi proposta uma formulação matemática que utiliza o conceito de fluxo para evitar a formação de subciclos. Este modelo foi aplicado ao grupo de instâncias SET_2 e encontrou a solução ótima para 46 das 52 instâncias deste grupo, resultado similar ao observado para o modelo proposto por Castro et al. [2013].

Para tratar o PCVSH, abordagens heurísticas foram propostas utilizando-se conceitos clássicos de construção da solução inicial amplamente utilizados em heurísticas para tratamento de problemas de roteamento. Em relação as estruturas de vizinhança, todas foram em algum momento utilizadas nos trabalhos acerca do PCVSH contidos na literatura.

Na avaliação das abordagens propostas, observou-se que o método de construção da solução baseado na heurística de Lin-Kernighan oferece uma solução inicial com qualidade não muito distante das melhores soluções conhecidas. Nota-se ainda que o uso da heurística Busca Tabu faz com que o tempo computacional seja demasiadamente alto para as configurações de parâmetros definidas. A combinação do Algoritmo Memético com RVND obteve bons resultados, obtendo um custo/benefício interessante em relação a abordagem que utiliza Busca Tabu. A abordagem base-

ada no Algoritmo *Iterated Greedy* com Descida de Vizinhaça Variável mostrou ser eficaz para a maioria dos grupos de instâncias conhecidas, chegando a obter resultados similares aos melhores conhecidos na literatura tanto em qualidade da solução, quanto no tempo necessário para sua execução.

A análise probabilística empírica mostrou que para a instância com nível de dificuldade considerado fácil, as abordagens que utilizam Algoritmo Memético propostas neste trabalho se destacaram encontrando o alvo definido para todas as execuções, demandando uma menor quantia de tempo em relação as demais abordagens. Para as instâncias que apresentam dificuldade média e difícil, houve uma inversão na qualidade das abordagens, sendo o IGVND e o AMRVND probabilisticamente melhores que as demais abordagens comparadas.

Os resultados apresentados nos gráficos de convergência da solução demonstram que a abordagem que converge para a solução no menor tempo computacional é o AMRVND. Considerando a instância com nível de dificuldade considerado fácil, todas as abordagens comparadas convergem rapidamente para a melhor solução conhecida. Em relação a instância que apresenta dificuldade média, a abordagem que apresenta-se mais competitiva em relação ao AMRVND é a AMBT. Para a instância de alta dificuldade, o PLS consegue obter a solução mais próxima ao AMRVND, seguida pela abordagem IGVND.

Dados estes resultados, conclui-se que as abordagens propostas apresentam-se competitivas em relação às contidas na literatura, obtendo para algumas instâncias, soluções melhores que as conhecidas até então.

Os resultados encontrados neste trabalho foram parcialmente publicados em Sousa & Gonçalves [2014].

5.1 Trabalhos Futuros

Como trabalhos futuros para o Problema do Caixeiro Viajante com Seleção de Hotéis, pode-se incluir a realização de um aperfeiçoamento das heurísticas propostas, como também o uso de outras metaheurísticas. Uma estratégia interessante seria a utilização do método de geração de colunas para obter uma heurística hibridizada, combinando um modelo matemático com diferentes heurísticas. Este tipo de estratégia tem tido bons resultados para problemas de roteamento [Subramanian et al., 2013a,b; Taş et al., 2014].

Outra vertente que pode ser explorada é a aplicação das heurísticas definidas nesta dissertação para problemas similares. Como exemplos de problemas, temos:

Problema do Caixeiro Viajante Múltiplo com Seleção de Hotéis, Problema do Caixeiro Viajante com Múltiplas Janelas de Tempo e Seleção de Hotéis [Baltz et al., 2014], Problema de Orientação com Seleção de Hotéis [Divsalar et al., 2013] e o Problema de Roteamento de Veículos com Seleção de Hotéis como proposta de problema ainda não explorado.

Referências Bibliográficas

- Aiex, R. M.; Resende, M. G. C. & Ribeiro, C. C. (2002). Probability distribution of solution time in grasp: An experimental investigation. *Journal of Heuristics*, 8(3):343 – 373.
- Amorim, L. E.; Gonçalves, L. B. & de Magalhaes, S. V. G. (2012). Um algoritmo memético para a solução do problema de mínima latência. In *Simpósio Brasileiro de Pesquisa Operacional*, pp. 2247 – 2258.
- Applegate, D.; Bixby, R. E.; Chvatal, V. & Cook, W. J. (2006a). Concorde tsp solver. Disponível em <http://www.tsp.gatech.edu/concorde>, acessado 19-Outubro-2014.
- Applegate, D. L.; Bixby, R. E.; Chvatal, V. & Cook, W. J. (2006b). *The traveling salesman problem: a computational study*. Princeton University Press.
- Archetti, C.; Speranza, M. G. & Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40(1):64 – 73.
- Baltz, A.; Ouali, M. E.; Jäger, G.; Sauerland, V. & Srivastav, A. (2014). Exact and heuristic algorithms for the travelling salesman problem with multiple time windows and hotel selection. *Journal of the Operational Research Society*.
- Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209 – 219.
- Benevides, P. F. (2011). Aplicação de heurísticas e metaheurísticas para o problema do caixeiro viajante em um problema real de roteirização de veículos. Dissertação de mestrado, Universidade Federal do Paraná, UFPR.
- Castro, M.; Sörensen, K.; Vansteenwegen, P. & Goos, P. (2014). A fast metaheuristic for the travelling salesperson problem with hotel selection. *4OR*, pp. 1 – 20.

- Castro, M.; Sörensen, K.; Vansteenwegen, P. & Goos, P. (2012). A simple grasp+vnd for the travelling salesperson problem with hotel selection. Working papers, University of Antwerp, Faculty of Applied Economics.
- Castro, M.; Sörensen, K.; Vansteenwegen, P. & Goos, P. (2013). A memetic algorithm for the travelling salesperson problem with hotel selection. *Computers & Operations Research*, 40(7):1716 – 1728.
- Chisman, J. A. (1975). The clustered traveling salesman problem. *Computers & Operations Research*, 2(2):115 – 119.
- CNT (2013). Pesquisa confederação nacional do transporte de rodovias. Disponível em <http://pesquisarodovias.cnt.org.br/Paginas/index.aspx>, acessado 08-Outubro-2014.
- Cordeau, J.-F.; Laporte, G. & Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 52(8):928 – 936.
- Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791 – 812.
- Dantzig, G. B. & Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80 – 91.
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische matematik*, 1(1):269–271.
- Divsalar, A.; Vansteenwegen, P. & Cattrysse, D. (2013). A variable neighborhood search method for the orienteering problem with hotel selection. *International Journal of Production Economics*, 145(1):150 – 160.
- Duarte, A.; Martí, R.; Álvarez, A. & Ángel-Bello, F. (2012). Metaheuristics for the linear ordering problem with cumulative costs. *European Journal of Operational Research*, 216(2):270 – 277.
- Feo, T. A. & Resende, M. G. C. (2003). *Greedy randomized adaptive search procedures*. Handbook of metaheuristics.
- Fleury, P. (2011). Logística no brasil: situação atual e transição para uma economia verde. Disponível em <http://www.fbds.org.br/IMG/pdf/doc-7.pdf>, acessado 08-Novembro-2014.

- Garey, M. R. & Johnson, D. S. (1979). Computers and intractability: A guide to the theory of np-completeness. *Freeman & Company*, 31.
- Gendreau, M.; Hertz, A. & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276 – 1290.
- Gendreau, M.; Iori, M.; Laporte, G. & Martello, S. (2006). A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3):342 – 350.
- Glover, F. (1989). Tabu search-part i. *ORSA Journal on computing*, 1(3):190 – 206.
- Glover, F. (1990). Tabu search-part ii. *ORSA Journal on computing*, 2(1):4 – 32.
- Golden, B. L. & Stewart, W. R. (1985). Empirical analysis of heuristics in the traveling salesman problem. *John Wiley & Sons*, pp. 207 – 249.
- Gonçalves, L. B. (2005). Heurísticas grasp para um problema de roteamento periódico de veículos. Dissertação de Mestrado, Universidade Federal Fluminense.
- Hansen, P. & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449 – 467.
- Hansen, P.; Mladenović, N.; Brimberg, J. & Pérez, J. A. M. (2008). Variable neighbourhood search: methods and applications. *4OR*, 6(4):319 – 360.
- Hansen, P.; Mladenović, N.; Brimberg, J. & Pérez, J. A. M. (2010). *Variable neighbourhood search*. Gendreau M, Potvin J-Y (eds) Handbook of metaheuristics. Springer, Berlin, pp 61 - 86.
- Helsgaun, K. (2000). An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130.
- Laporte, G.; Gendreau, M.; Potvin, J.-Y. & Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, 7(4-5):285 – 300.
- Laudon, K. & Laudon, J. P. (2007). *Sistemas de Informações Gerenciais*. Pearson Education.
- Lin, S. (1965). Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245 – 2269.

- Lin, S. & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498 – 516.
- Lourenço, H. R.; Martin, O. C. & Stützle, T. (2003). *Handbook of Metaheuristics: Iterated local search*. Springer.
- Mestria, M.; Ochi, L. S. & de Lima Martins, S. (2013). Grasp with path-relinking for the symmetric euclidean clustered traveling salesman problem. *Computers & Operations Research*, 40:3218 – 3229.
- Mine, M. T.; de Souza Alves Silva, M.; Ochi, L. S.; Souza, M. J. F. & da Silva, T. C. B. (2010). O problema de roteamento de veículos com coleta e entrega simultânea: uma abordagem via iterated local search e genius. *Transportes*, 18(3):60–71.
- Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100.
- Montgomery, D. C. (2006). *Design and Analysis of Experiments*. John Wiley & Sons.
- Moscato, P.; Cotta, C. & Mendes, A. (2004). Memetic algorithms. In *New optimization techniques in engineering*, pp. 53 – 85. Springer.
- Neolog (2014). Resultados do diagnóstico logístico. Disponível em <http://www.neolog.com.br/neolog/avaliacao.html>, acessado 08-Novembro-2014.
- Or, I. (1976). *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. Xerox University Microfilms.
- Osman, I. H. & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63(5):511–623.
- Penna, P. H. V.; Subramanian, A. & Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2):201 – 232.
- Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985 – 2002.
- Prins, C.; Lacomme, P. & Prodhon, C. (2014). Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies*, 40:179 – 200.

- Resende, M. G. C. & Ribeiro, C. C. (2014). Grasp: greedy randomized adaptive search procedures. In *Search methodologies*, pp. 287 – 312. Springer.
- Silva, G. A. N.; Silva, F. A.; Russi, D. T. A.; Pazoti, M. A. & Siscoutto, R. A. (2013). Algoritmos heurísticos construtivos aplicados ao problema do caixeiro viajante para a definição de rotas otimizadas. 5(2):30 – 46.
- Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254 – 265.
- Sousa, M. M. & Gonçalves, L. B. (2014). Comparação de abordagens heurísticas baseadas em algoritmo memético para o problema do caixeiro viajante com seleção de hotéis. XLVI Simpósio Brasileiro de Pesquisa Operacional, Salvador - BA.
- Subramanian, A.; de A Drummond, L. M.; Bentes, C.; Ochi, L. S. & Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899 – 1911.
- Subramanian, A.; Uchoa, E. & Ochi, L. S. (2013a). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519 – 2531.
- Subramanian, A.; Uchoa, E.; Pessoa, A. A. & Ochi, L. S. (2013b). Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. *Optimization Letters*, 7(7):1569 – 1581.
- Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*. John Wiley & Sons.
- Taş, D.; Gendreau, M.; Dellaert, N.; van Woensel, T. & de Kok, A. (2014). Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. *European Journal of Operational Research*, 236(3):789 – 799.
- Toth, P. & Vigo, D. (2001). *The vehicle routing problem*. Siam.
- Vansteenwegen, P.; Souffriau, W. & Sörensen, K. (2012). The travelling salesperson problem with hotel selection. *Journal of the Operational Research Society*, 63(2):207 – 217.