UNIVERSIDADE ESTADUAL DO CEARÁ

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO DE CASTRO HONORATO SILVA

OTIMIZAÇÃO DE ROTAS UTILIZANDO ABORDAGENS HEURÍSTICAS EM UM AMBIENTE GEORREFERENCIADO

FORTALE ZA – CE

2013

BRUNO DE CASTRO HONORATO SILVA

Dissertação apresentada ao Programa de Mestrado Acadêmico em Ciência da Computação da Universidade Estadual do Ceará como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Algoritmos em Grafos e Otimização.

Orientador: Prof. Dr. Gerardo Valdísio Rodrigues Viana.

FORTALEZA – CE

2013

BRUNO DE CASTRO HONORATO SILVA

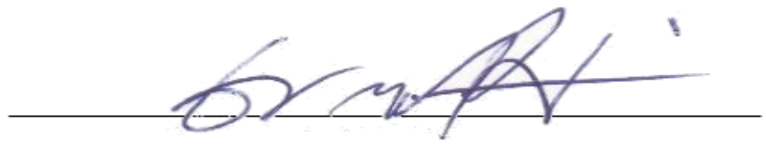
OTIMIZAÇÃO DE ROTAS UTILIZANDO ABORDAGENS HEURÍSTICAS EM UM AMBIENTE GEORREFERENCIADO

Dissertação submetida à Coordenação do curso de Pós-graduação em Ciência da Computação da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Algoritmos em Grafos e Otimização.

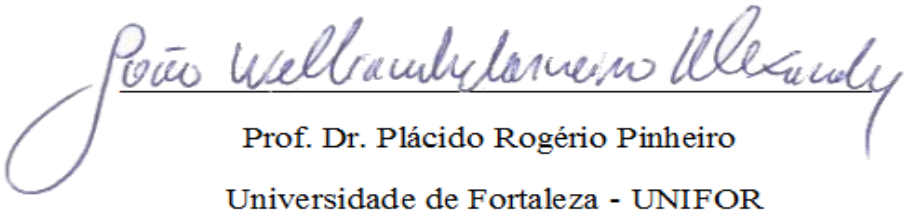
Aprovado em 26/04/2013.

BANCA EXAMINADORA



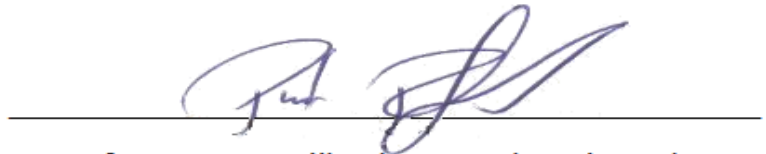
Prof. Dr. Gerardo Valdísio Rodrigues Viana (Orientador)

Universidade Estadual do Ceará – UECE



Prof. Dr. Plácido Rogério Pinheiro

Universidade de Fortaleza - UNIFOR



Prof. Dr. João Welliandre Carneiro Alexandre

Universidade Federal do Ceará - UFC

AGRADECIMENTOS

Em primeiro lugar, aos meus pais, por dedicarem suas vidas na construção e prosperidade desta família e pelo amor concebido aos seus filhos.

Ao meu irmão, parentes e amigos, pelo apoio e compreensão durante toda essa jornada de dedicação e restrição de tempo.

Ao Professor Dr. Gerardo Valdísio Rodrigues Viana pela confiança e orientação deste trabalho.

Ao Professor Dr. José Lassance de Castro Silva pelo apoio computacional nas aplicações dos algoritmos e pelos conselhos.

Aos meus amigos Sergio Rocha e Rony Iglecio pelo apoio teórico nos segmentos de Geodesia e Cartografia.

À todos os colegas, professores e funcionários do Programa de Mestrado Acadêmico em Ciências da Computação/UECE pela convivência harmoniosa durante todo o período do curso.

E por fim, a Deus, por ter me concedido a permissão de chegar até aqui e pelas pessoas colocadas em meu caminho ao longo desses anos.

DEDICATORIA

Dedico este trabalho à minha família, sobretudo aos meus pais que estiveram sempre presentes me apoiando e me incentivando a alcançar meus objetivos.

RESUMO

Mediante ao anseio de organizações empresariais em minimizar os custos com transporte, pesquisas sobre métodos que venham a otimizar o processo de roteirização têm sido realizadas. O termo roteirização pode ser descrito como um processo de sequências de paradas determinadas que um veículo deva percorrer, com o objetivo de atender pontos dispersos geograficamente. Na literatura, o problema relacionado com roteirização mais conhecido é o clássico Problema do Caixeiro Viajante (*PCV*). Neste trabalho, o PCV será abordado por meio de métodos heurísticos propostos a partir de um estudo realizado sobre métodos computacionais aplicados na resolução do problema. Ainda como consequência deste estudo, desenvolveu-se também uma ferramenta computacional que pode dar suporte a vários tipos de empresas que dependem da roteirização para distribuir seus produtos ou serviços com custo mínimo de transporte, através da aplicação prática do PCV clássico podendo atender outros tipos de restrições. Após analisar, desenvolver, implementar e validar os métodos heurísticos propostos resolveu-se ambientá-los à ferramenta computacional sobre uma perspectiva espacial, gerando um Ambiente Georreferenciado de Resolução, composto por um Sistema de Informações Georreferenciadas (*SIG*) robusto, permitindo que instâncias do PCV possam ser criadas e analisadas as suas resoluções com acompanhamento gráfico. Os métodos de resolução proposto para o problema são bastante rápidos e fáceis de serem implementados, adequando-se muito bem a parte prática do problema.

Palavras-chave: Otimização, PCV, Heurísticas, SIG.

ABSTRACT

Through the desire of business organizations to minimize transportation costs, research on methods that will optimize the routing process has been conducted. The routing can be described as a process of certain sequences of stops that a vehicle must traverse in order to meet geographically scattered points. In the literature, the problem related to routing best known is the classic Traveling Salesman Problem (TSP). In this work, the TSP will be addressed through heuristic methods proposed from a study of computational methods applied in solving the problem. As result of this study also was developed a computational tool used in the support for the businesses deliver your products or services with minimum transportation cost, through the practical application of classic TSP with other restrictions. After analyze, develop, implement and validate the proposed heuristic methods was created a Computational Environment on a spatial perspective, called Resolution Georeferenced Environment, composed of a robust Georeferenced Information System (GIS), allowing create TSP instances and analyze their resolutions with accompanying graphic. The proposed heuristics are very fast and low cost computational, adapting itself very well to the practical problem.

Keywords: Optimization, TSP, heuristics, GIS.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG	: Algoritmo Genético
AHC	: Algoritmos Heurísticos Construtivos
AHM	: Algoritmos Heurísticos de Melhoramento
ARG-PCV	: Ambiente de Resolução Georreferenciado do Problema do Caixeiro Viajante
BFS	: *Breadth First Search*
BT	: Busca Tabu
GA	: *Genetic Algorithms*
GRASP	: *Greedy Randomized Adaptive Search Procedures*
HVP	: Heurística do Vizinho mais Próximo
HVP2	: Heurística do Vizinho mais Próximo com 2 Caminhos Paralelos
HVP2B	: Heurística do Vizinho mais Próximo com 2 caminhos paralelos e a inserção de uma cidade por iteração
HVP4	: Heurística do Vizinho mais Próximo com 4 Caminhos Paralelos
HIMB	: Heurística da Inserção mais Barata
HIMBM	: Heurística da Inserção mais Barata Modificada
IBGE	: Instituto Brasileiro de Geografia e Estátistica
ILOS	: Instituto de Logística e Supply Chain
NASA	: *National Aeronautics and Space Administration*
NP	: Não Polinomial
OR	: *Operational Research*
PCV	: Problema do Caixeiro Viajante
PO	: Pesquisa Operacional
POCP:	: Problema de Otimização Combinatória Permutacional
RAND	: *Research And Development*
SA	: *Simulated Annealing*
SAD 69	: Sistema Geodésico Sul-Americano de 1969
SIG	: Sistema de Informações Georreferenciadas

SIRGAS : Sistema de Referência Geocêntrico para as Américas

SGBDE : Sistema Gerencial de Banco de Dados Espacial

SQL : *Structured Query Language*

TSP : *Travelling Salesman Problem*

WGS : *World Geodetic System*

WMS : *Web Map Service*

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

Este Capítulo encontra-se dividido em 4 seções. A primeira seção contém a justificativa e relevância do trabalho. A segunda seção trata da descrição e histórico do PCV. A terceira seção aborda os objetivos do trabalho desenvolvido. Por fim, a quarta seção apresenta o delineamento e a organização deste trabalho.

1.1 Justificativa e relevância do trabalho

No Brasil, um país onde as exportações e importações respondem por 15% e 12%, respectivamente do seu Produto Interno Bruto (PIB), a relevância do sistema logístico é de alta prioridade. O transporte representa o elemento logístico mais custoso e consequentemente, o candidato preponderante para otimização. Segundo o Instituto de Logística e *Supply Chain* (ILOS), os custos com transporte por parte das empresas correspondem a 6,9% do PIB. Este fato, associado ao número considerável de congestionamentos de veículos observados nas malhas viárias das grandes cidades, além do aumento no consumo de combustíveis e do intervalo de tempo para se percorrer um determinado trajeto, tem despertado o interesse, tanto no meio profissional como no acadêmico, por estudos que venham a otimizar o processo de roteirização, afim de minimizar os custos decorrentes deste processo e melhorar o gerenciamento do mesmo.

O uso de técnicas baseadas em Pesquisa Operacional (PO) auxilia o tomador de decisão na geração de vantagem competitiva, tendo em vista a complexidade associada aos problemas de logística urbana, notadamente nas médias e grandes metrópoles brasileiras. A complexidade de um sistema real resulta do fato de que seu comportamento é influenciado por um número muito grande de elementos ou variáveis. Exemplos de aplicações da PO podem ser observados na: determinação do custo mínimo de produção, maximização de lucros, otimização de rotas, resolução de problemas de transporte, designação e sequenciamento de tarefas, entre outros.

Segundo Silva (2010), o ponto chave da Pesquisa Operacional reside na construção de modelos matemáticos a partir dos quais escolhe-se uma técnica adequada para

solucioná-los. A ausência ou incerteza em relação às informações disponíveis sobre cada variável que influencia o sistema real prejudica diretamente a formulação de um plano de trabalho ou de um planejamento de serviços, reduzindo a produtividade e normalmente incorrendo em custos adicionais elevados.

Segundo Andrade (2000), o esforço despendido para a modelagem de um problema leva a uma compreensão mais profunda do próprio problema, identificando melhor seus elementos internos, suas variáveis principais, suas interações com o ambiente externo, as informações necessárias e os resultados possíveis de obter. Essa abordagem quantitativa dos problemas fornece uma estrutura de raciocínio e análise que permite desenvolver a visão sistêmica do processo. O termo roteirização, ou *routing* do idioma inglês, pode ser descrito como um processo de sequências de paradas determinadas que um veículo deva percorrer, com o objetivo de atender pontos dispersos geograficamente (CUNHA, 2000). Na literatura, o problema relacionado com roteirização mais conhecido é o clássico Problema do Caixeiro Viajante, que busca encontrar um roteiro, ou rota, entre "*n*" pontos de passagem, de forma a realizar todo o percurso e minimizar a distância total percorrida.

O PCV pertence a classe dos Problemas de Otimização Combinatória, avaliada por pesquisadores de diferentes áreas do conhecimento científico cuja premissa é atribuir valores a um conjunto de variáveis de decisão, de tal modo que uma função com base nestas variáveis, denominada função objetivo, seja minimizada ou maximizada na presença de um conjunto de restrições (LAWLER, LENSTRA *&* SHMOYS, 1985). A busca por um método computacional de resolução do problema eficiente e eficaz é uma tarefa difícil, tendo em vista que a resolução do problema é bastante complexa, pois o PCV pertence classe dos problemas NP-difíceis (GAREY & JOHNSON, 1979). Cook (1971) e Karp (1972) mostraram que uma grande quantidade de problemas de otimização pode ser reduzida, em tempo polinomial, ao PCV. Com isso, a descoberta de métodos eficientes e eficazes para a resolução do PCV, implica na resolução, também eficiente, de outros problemas complexos importantes da área de otimização.

Neste trabalho, o PCV é abordado por meio de métodos heurísticos a partir de um estudo realizado sobre as técnicas encontradas na literatura. Em seguida, apresenta-se

um conjunto de técnicas de resolução desenvolvidas para o problema, onde foi aplicada e avaliada o desempenho dessas técnicas nas instâncias consolidadas da literatura, mais especificamente as instância da *TSP-Lib*, da *OR-Library*. No segundo momento, do trabalho, realizou-se a ambientação dessas técnicas na resolução de problemas práticos empresariais por meio de uma base espacial georreferenciada, composto pelos métodos heurísticos desenvolvidos propostos e por um padrão de geração de um Sistema de Informação Georreferenciada (*SIG*), eficaz e eficiente na resolução do problema.

Os SIGs em geral são munidos para mostrar dados contextuais em forma de mapas, desenhos, imagens, etc. Geralmente, eles não são programados para realizar tomadas de decisões automatizadas em suas plataformas. No problema em questão, existem *softwares* disponíveis na praça para realizar este objetivo, entretanto estes sistemas são muito caros e de domínio de empresas do exterior, como é o caso do *RoadNet*, cuja propriedade é da empresa americana *Roadnet Technologies Incorporation*. Fundada em 1983, a *Roadnet Technologies Incorporation* foi adquirida pela organização *United Parcel Service* (UPS) em 1986 passando a ser denominada como UPS *Logistics Technologies*. *Thoma Bravo*, uma empresa de capital privado especializada em investir em empresas de medio porte com sede em San Francisco, comprou a divisão da UPS relacionado com roteamento no final de 2010 e restabeleceu a denominação inicial. O *software RoadNet* é o mais popular dos que fazem roteamento e bastante usado aqui no Brasil. O seu uso é obtido através de licenças, cujo preço depende do número de viaturas e clientes a serem tratados no problema.

Nos trabalhos de Bezerra (2012), Gomes (2011) e Campelo Júnior (2010) foram tratadas várias instâncias de problemas práticos do PCV nas empresas Correios-Fortaleza (distribuição de encomendas – SEDEX 10), Thyssenkrupp-Ceará (distribuição de equipes para conserto e manutenção de elevadores) e RB Distribuidora (distribuição de sorvetes Kibon). Em todos estes trabalhos, os autores descrevem a dificuldade de operacionalizar e monitorar as instâncias do PCV e visualizar suas soluções gráficas em um único ambiente computacional integrado. Nossa proposta também visa atender esta demanda. Desta forma, além dos métodos de resolução propostos para o problema, a principal contribuição deste trabalho é diminuir e operacionalizar racionalmente os custos na distribuição dos produtos ou serviços de uma empresa, com base na

roteirização otimizada desta distribuição, através de um ambiente computacional integrado e único.

1.2 O Problema do Caixeiro Viajante

O PCV, ou *Travelling Salesman Problem* (TSP) como ele é conhecido na literatura universal, consiste em um dos mais tradicionais e conhecidos problemas da Pesquisa Operacional. O objetivo é definir a melhor rota (menor distância ou menor custo, por exemplo) de um circuito de pontos ou localidades para que determinado agente percorra todos os pontos uma única vez, sem repetição. Este circuito também é tratado na literatura como ciclo hamiltoniano. A solução do problema fornece a ordem em que as localidades são visitadas e a esta ordenação, pode-se denominar *tour* que conrresponde ao ciclo hamiltoniano de tamanho mínimo (APPLEGATE et al., 2006).

Este problema de sonoridade modesta, derivado de um jogo relacionado ao cálculo de circuitos hamiltonianos inventado em 1800 pelo matemático irlandês W. R. Hamilton e pelo matemático britânico Thomas Kirkman, é de fato um dos problemas mais intensamente investigados em Otimização Combinatória, dado a sua relação com problemas complexos encontrados em diversas áreas cientificas.

Em 1930, o problema passou a ser estudado na academia, mais precisamente em Harvard e Princeton, todavia, foi somente nas décadas de 1950 e 1960 que o problema conquistou a popularidade nos círculos científicos europeus e americanos. Neste período, George Dantzig, Fulkerson Delbert Ray e Selmer Johnson expressaram e resolveram o problema por meio de Programação Linear Inteira, desenvolvendo o método de plano de corte para resolvê-lo de forma ótima.

Karp (1972) com seu artigo histórico, "*Reducibility Among Combinatorial Problems*", mostrou que o Problema do Caminho Hamiltoniano e outros 20 diferentes problemas de combinatória, famosos por sua intratabilidade computacional, pertencem a classe de problemas NP-Completos, fornecendo uma explicação matemática para a dificuldade em resolver de forma eficaz e eficiente o PCV. A classe *NP*, do acrônimo de Tempo Polinomial não Determinístico (*Non-deterministic Polynomial time*), deriva da teoria

fundamentada por Cook (1971) e, como o próprio nome sugere, é composta por problemas com instâncias de médio e grande porte que são intratáveis por algoritmos determinísticos em tempo polinomial, devido ao consumo demasiado de recursos computacionais. Desde então, o desenvolvimento de métodos heurísticos ganhou bastante evidência na resolução dos problemas de otimização, como é o caso do PCV.

Um método heurístico pode ser definido como uma técnica que busca boas soluções, com um custo computacional (tempo) relativamente inferior ao dos métodos determinísticos. Sirenko (2009) classificou métodos heurísticos, mediante a complexidade de suas estruturas, em Heurísticas, Heurísticas Híbridas, Meta-Heurísticas, Meta-Heurísticas Híbridas ou Hiper-Heurísticas. Segundo Sousa (2009), dentre as vantagens a serem consideradas como prerrogativas no uso de métodos heurísticos, destacam-se a capacidade de flexibilização para manipular as variáveis ou características do problema, a proposição de mais de uma solução, possibilitando ao analista verificar qual aquela que tem melhor qualidade para o problema analisado e a geração de soluções satisfatórias sem recorrer ao formalismo matemático aumentando a facilidade de implementação.

1.3 Objetivos

O objetivo deste trabalho é propor um Ambiente de Resolução Georreferenciado para o PCV, aqui chamado de ARG-PCV, com o intuito de avaliar e verificar o processo de resolução do problema por meio das técnicas criadas e implementadas, assim como ambientar o problema sobre uma perspectiva espacial. O ARG-PCV é composto por um Sistema de Informação Georreferenciado (SIG) robusto suficiente que permite a modelagem de mapas de malhas viárias em grafos, a criação de instâncias georreferenciadas do PCV, a resolução destas instâncias e o acompanhamento gráfico do processo de resolução em tempo real.

O processo de resolução será operado por novos métodos heurísticos, criados e implementados neste trabalho, com base num estudo sobre o Problema do Caixeiro Viajante, onde são descritas as especificações, histórico, métodos clássicos de resolução e relevância prática deste problema. Para mensurar a eficiência desses métodos

heurísticos propostos, foram realizados várias aplicações em instâncias da literatura (TSP-Library), especificamente aquelas abordadas no trabalho de Silva, Soma e Viana (2004), que também são instâncias universais da OR-Library.

1.4 Delineamento e organização do trabalho

O trabalho está dividido em seis capítulos. O Capítulo 1 é a introdução, onde constam a relevância do tema, a contextualização do problema e os objetivos. No Capítulo 2, uma revisão literária científica sobre o PCV é feita, onde constam as características, o histórico, os métodos de resolução, as aplicações do problema, as variações e os problemas semelhantes. O Capítulo 3 destaca diferentes métodos heurísticos aplicados na resolução do PCV bem como as Heurísticas Híbridas propostas neste trabalho. O Capitulo 4 inicia com introduções teóricas sobre a Teoria dos Grafos, Sistemas de Informação Georreferenciado e Bancos de Dados com Extensão Espacial, e termina com a descrição do ARG-PCV proposto. Os experimentos computacionais constam no Capítulo 5, enquanto no Capítulo 6 são apresentadas as conclusões do trabalho e a proposição de trabalhos futuros a serem desenvolvidos.

CAPÍTULO 2 – PROBLEMA DO CAIXEIRO VIAJANTE

Este Capítulo encontra-se dividido em 4 seções. A primeira seção contém a definição juntamente com o histórico do Problema do Caixeiro Viajante. A segunda seção trata da complexidade do PCV. A terceira seção aborda as variações e problemas semelhantes ao PCV no segmento de roteirização. Por fim, a quarta seção apresenta os tipos de métodos mais empregados na literatura para a resolução do problema abordado.

2.1 Definição do PCV

Um Problema de Otimização Combinatória Permutacional (*POCP*) pode ser definido por um terno (S, g, n), onde S é o conjunto de todas as soluções viáveis (soluções que satisfazem as restrições do problema e $| S | = n!$), g é a função objetivo que associa a cada solução $s \in S$ um número real e n é uma instância do problema. O objetivo é encontrar a solução $s \in$ S que minimize a função objetivo g. Podemos representar s como uma permutação de n elementos distintos, ou seja, $s=< a_1\ a_2\ ...\ a_n >$. N(s) é chamada *vizinhança de s*, contendo todas as soluções que podem ser alcançadas a partir de s por um simples movimento. Aqui, o significado de um *movimento* é aquele de um operador que transforma uma solução para uma outra com menor diferença simétrica estendida.

No PCV tem-se um conjunto com n cidades $c_1, c_2, ..., c_n$ e todas as distâncias (ou custo) entre elas, onde $d_{i,j}$ representará a distância (ou custo) para "*ir*" da cidade c_i até a cidade c_j. O objetivo do problema consiste em determinar uma rota, para o caixeiro, que percorra todas as n cidades somente uma vez com a menor distância percorrida. O PCV pode ser modelado como um POCP, $P = (S, g, n)$, de acordo com a definição dada anteriormente, da seguinte forma:

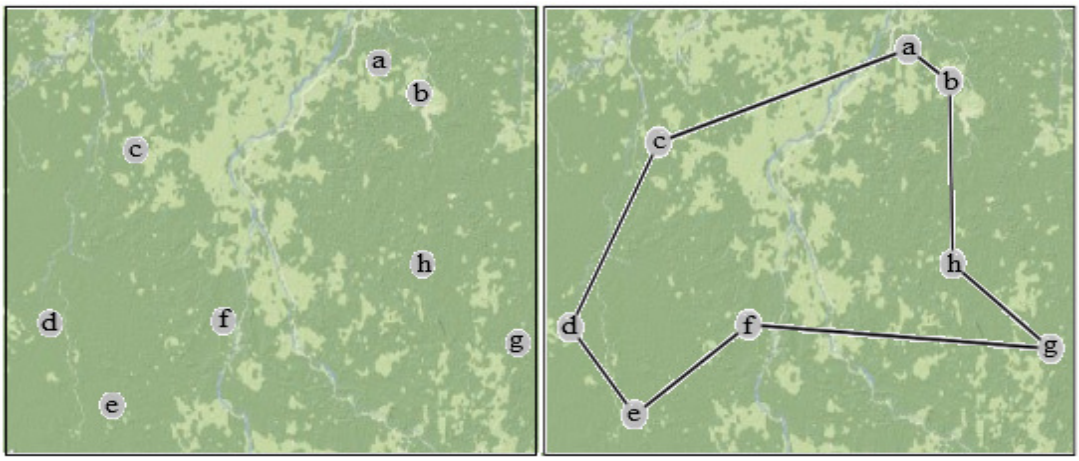
a) Um elemento $s=< a_1\ a_2\ ...\ a_n >$ do conjunto de soluções viáveis S é representado por uma permutação das n cidades, com a ordem de s determinando a seqüência (rota) na qual as cidades serão visitadas;

b) A função *g* que calcula a distância total percorrida numa sequência *s* é dada por

$$g(s) = (\sum_{i=1}^{n-1} d_{c_i c_{i+1}}) + d_{c_n c_1}.$$

Outra formulação do PCV ocorre na da Teoria dos Grafos, onde são definidos dois conjuntos, um composto por elementos chamados de vértices e outro composto por uma coleção de relações binária de pares entre eles, denominados de arestas. No caso, seja *G = (N, A, D)* um grafo onde *N* é o conjunto de vértices do grafo que representam as localidades a serem visitadas, *A* é o conjunto de arestas que representam as possibilidades de composição de uma rota e *D* é uma matriz completa composta por pesos não negativos associados aos arcos, que podem representar, por exemplo, distância ou tempo gasto para ir de um ponto a outro. O PCV consiste em determinar o circuito com o menor custo, passando por cada vértice uma única vez. A este circuito, denomina-se ciclo ou circuito hamiltoniano. Um exemplo de circuito hamiltoniano em um grafo pode ser visto na Figura 2.1. Se a matriz *D* é simétrica, então o PCV é classificado como Simétrico e há *(n-1)!/2* circuitos hamiltonianos possíveis distintos. Caso contrário, o PCV é dito Assimétrico e o número de circuitos hamiltonianos possíveis passa a ser *(n-1)!*.

Figura 2.1 – Exemplo de ciclo hamiltoniano com *n*=7.



A essência do Problema do Caixeiro Viajante é evidente dentro de muitas aplicações práticas. Em todos estes casos, o custo ou a distância entre duas localidades quaisquer é conhecido, sendo o objetivo fundamental, determinar uma sequência na qual os locais

especificados aparecem ordenados de acordo com o custo total incorrido para percorrê-los, passando por cada localidade uma única vez. Na prática, o PCV pode ser aplicado na definição de planos de rotas (HOFFMAN & PADBERG, 2012), na confecção de placas de circuito impresso (VITTES, 1999), na análise da estrutura de cristais (BLAND & SHALLCROSS, 1987), no suporte à rede de abastecimento de graneis líquidos, no sequenciamento de tarefas, na fabricação de chips (KORTE, 1989), no mapeamento de genoma (AVNER et al., 2001), no sequenciamento de DNA (GONNET KOROTENSKY & BENNER, 2000), dentre outras. Neste trabalho, O PCV é abordado no contexto de roteirização, atividade que tem por fim buscar os melhores trajetos que um veículo deve fazer através de uma malha (BALLOU, 2001).

O PCV é tido por muitos pesquisadores como sendo uma viagem pelo mundo, onde pretende-se encontrar um itinerário contínuo pelas arestas do dodecaedro, saindo de uma determinada cidade, que passasse só uma vez por cada cidade. Com um fio de lã marcava-se este itinerário, que dava uma volta no eixo de cada cidade por onde passava e se atava em Dublin. A Figura 2.2 mostra uma ilustração deste fato, proposto por Hamilton.

Figura 2.2 – Exemplar do jogo proposto por Hamilton.

A visibilidade acadêmica do PCV veio por volta das décadas de 1920 e 1930, quando o problema passou a ser estudado em Viena, mais precisamente por Karl Menger e posteriormente, em Princeton, por Hassler Whitney e Merril Flood (MAREDIA, 2010). Foi na decada de 1940, que Merrill e Flood, definiram o nome do problema como *Traveling Salesman Problem* (Problema do Caixeiro Viajante) e o divulgou para a *RAND Corporation* (*Research And Developmen*), que se interessou pelo problema devido ao estimulo financeiro oferecido pela Força Aérea Norte Americana em perspectiva da obtenção de soluções ótimas para problemas de transporte. A *RAND*, fundada em 1948, é uma organização privada, sem fins lucrativos, que ao longo dos anos dedicou-se a resolver problemas interdisciplinares e quantitativos, traduzindo conceitos teóricos de Ciências Aplicadas e Pesquisa Operacional em aplicações úteis a outras áreas como educação, saúde, sociologia, defesa, justiça e economia aplicada.

Na década de 1950, o PCV ficou popular, dado o prestígio da *RAND Corporation* e sua relação com outros problemas de otimização estudados naquele período. Foi também neste período que Dantzig, Fulkerson e Johnson propuseram o primeiro método para resolver de forma otimizada uma instância de 49 cidades do PCV, o que para os padrões da época, era uma instância grande. Aproveitando o sucesso sobre as inúmeras aplicações do algoritmo SIMPLEX (projetado por Dantzig em 1947), os três pesquisadores formularam o problema por meio de Programação Linear Inteira através da seguinte forma:

$$\min\ Z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}.x_{ij} \qquad (1)$$

Sujeito às restrições:

$$x_{ii} = 0, \quad i = 1, 2, 3, \ldots, n \qquad (2)$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, 2, 3, \ldots, n \qquad (3)$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n \qquad (4)$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \qquad (5)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leqslant |S| - 1, |S \subset V : S \neq \emptyset \qquad (6)$$

A equação (1) representa a função objetivo do problema e consiste em minimizar o custo para que o caixeiro viajante percorra todos os *n* pontos distintos uma única vez e retornar ao ponto de partida inicial. As restrições do problema são dadas pelas equações (2) até (5). A equação (2) restringe que não é possível ir de uma cidade para ela mesma, enquanto que a equação (3) representa uma variável binária x_{ij} de confirmação de deslocamento, tal que se $x_{ij}=1$, o caixeiro vai diretamente da cidade *j* à cidade *i* na rota que está sendo apresentada. Em caso contrário, $x_{ij}=0$. As equações (4) e (5) determinam que o caixeiro deve passar uma única vez em cada ponto, seja na chegada, ou na partida, devendo seguir para o próximo ponto até que todo o percurso seja realizado e retorne ao ponto inicial. A Equação 6, evita que ocorra ciclagem menor que *n* na rota, ou seja, que sejam produzidas rotas somente de ciclos hamiltonianos.

No inicio da década de 1970, Held e Karp (1971) abordaram com sucesso o problema valendo-se de programação dinâmica e relaxação lagrangeana, otimizando uma instância com 64 cidades. Em 1977, Gröetschel (1977), utilizando métodos de programação matemática, encontrou para uma instância de 120 cidades, o percurso ótimo. Em 1987, os pesquisadores Padberg e Rinaldi (1987), atuando como funcionários da *Tektronics Incorporated,* mostraram que o problema também pode ser usado em diversas áreas e não apenas para cidades, quando realizaram a otimização de 2392 pontos de um *layout* com um algoritmo *Branch & Cut*.

Dado os avanços tecnológicos que se sucederam a partir da década de 1990, nos anos de 1994, 1998 e 2001, nos laboratórios da AT&T Bell, os pesquisadores Applegate, Bixby, Chvátal e Cook, usando métodos heurísticos e plano de corte, otimizaram instâncias contendo 7397, 13509 e 15112 cidades respectivamente. Em 2004, o mesmo grupo de pesquisadores, juntamente com Helsgaun, após 14 meses de testes, propuseram a solução ótima para uma instância com 24978 cidades. Em 2006, Applegate *et. al.* (2006) alcançaram o último recorde que se tem conhecimento ao otimizarem uma instância com 85600 pontos através do Concorde TSP Solver.

2.2 Complexidade do PCV

O PCV, assim como outros problemas de otimização, possuem complexidade computacional de tal forma que todos os esforços programáveis conhecidos para resolver tais problemas crescem exponencialmente com o tamanho do problema. Em 1972, Richard M. Karp mostrou que o problema pertence a classe de problemas *NP-Completos*, fornecendo assim uma explicação matemática para a complexidade em resolver de forma otimazada o PCV.

A complexidade de um problema computacional é dada pelo consumo de recursos computacionais (tempo, memória, etc.) de algoritmos para apresentar uma solução aceitável, dado um critério específico do problema. A teoria da complexidade computacional, fundamentada por Cook (1971), estuda a complexidade de problemas, classificando-os em duas classes genéricas conhecidas como *P (Polynomial time)* e *NP (Non-Deterministic Polynomial time)*. A classe de complexidade *P* remete ao conjunto de todos os problemas que aceitam o tempo polinomial no pior caso, em outras palavras, a função de complexidade destes problemas é dado pela função $O(p(n))$, onde $p(n)$ é um polinômio. A classe *NP* é composta por problemas onde as instâncias de médio e grande porte são intratáveis pelos algoritmos determinísticos conhecidos em tempo polinomial, devido ao consumo demasiado de recursos computacionais. A função de complexidade dos problemas da classe *NP* é $O(c^n)$, com c>1, sendo a função de complexidade do PCV da ordem de $O(n!)$. Pode-se afirmar que $P \subset NP$, pois um algoritmo polinomial determinístico usado para resolver problema da classe *P*, é também aplicável para verificar a viabilidade de uma solução para um determinado problema da classe *NP*.

Tabela 2.1 – Explosão combinatória.

n	n!	Tempo
5	120	0,00012 segundos
10	3628800	3,62880 segundos
12	479001600	8 minutos
15	1307674368000	15 dias
20	2,43E+018	77.147 anos

50	3.041409320171337804361 2E+0064	∞
100	9.332621544394415268169 9E+0157	∞
500	1.220136825991110068791 2E+1134	∞
1000	4.023872600770937735436 2E+2567	∞

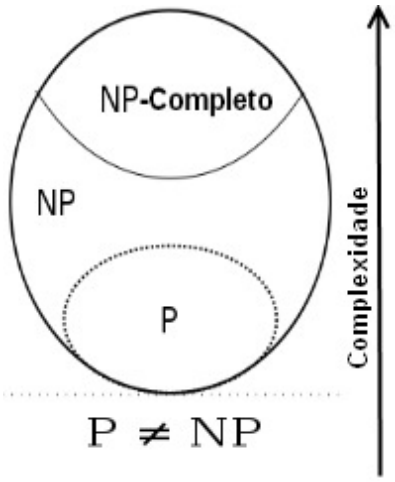
Sendo a função de complexidade do PCV exponencial, torna-se inviável a análise de todas as soluções possíveis mesmo para problemas de pequena complexidade, ou seja, para valores de $n<10$, conforme apresenta a Tabela 2.1, que mostra a explosão combinatorial de problemas desta natureza. O tempo estimado está calculado com base numa máquina hipotética da ordem de 1 microssegundo, para a execução das instruções de controle do programa, de acesso aos dados, cálculo das distâncias, comparações, chamadas a subprogramas, etc. (VIANA, 1998).

Garey e Johnson (1979) classificaram os problemas segundo sua complexidade em indecidíveis, tratáveis e intratáveis. Um problema é tido como indecidível se nenhum algoritmo pode ser dado para resolvê-lo. Já os problemas intratáveis são problemas decidíveis, porém difíceis para os quais possivelmente não existem algoritmos que os resolvam em tempo polinomial. Por fim, os problemas classificados como tratáveis, são aqueles que possuem algoritmos aptos a resolvê-los em tempo polinomial. Na literatura, problemas tratáveis e intratáveis são classificados ainda como sendo de decisão, que consistem na verificação da veracidade ou não de uma determinada questão para o problema, de localização, que buscam uma estrutura que satisfaça requisitos especificados por uma questão do problema, e de otimização, que visam obter a melhor solução possível, dentre as soluções viáveis para o problema. O PCV pertence a classe de problemas de otimização, porém pode ser formulado como um problema de decisão da seguinte forma: dadas n cidades e a distância c_{ij} entre duas cidades i e j, e um inteiro não negativo k, verificar se há um roteiro para o caixeiro cujo custo seja menor que k. Como problema de localização, o PCV segue o seguinte escopo: dadas n cidades e distância c_{ij} entre duas cidades i e j e um inteiro não negativo k, localizar, caso exista, uma sequência de vértices $u, \ldots, v$ com custo inferior ou igual a k.

Vários trabalhos direcionaram seus estudos na busca por um método polinomial para resolver o PCV, pois Cook (1971) e Karp (1972) mostraram que uma grande quantidade

de problemas intratáveis pode ser reduzido, em tempo polinomial, ao problema do caixeiro. Porém, o histórico literário mostra que grandes instâncias deste problema requerem sempre um tempo exponencial. Uma importante noção neste contexto é do subconjunto de *NP* a qual pertence o PCV, denominado *NP-Completo*. Os problemas mais difíceis da classe *NP* são atribuídos a classe *NP-Completo*, isto deve-se ao fato de que nenhum algoritmo de tempo polinomial foi descoberto para qualquer problema desta classe. Se descoberto algum algoritmo de tempo polinomial para qualquer problema *NP–Completo*, então todos os problemas desta classe poderão ser resolvidos em tempo polinomial. Um diagrama sobre as classes de complexidade *P, NP* e *NP-Completo* é apresentado na Figura 2.3.

Figura 2.3 – Diagrama euleriano representando as classes *P*, *NP* e *NP-Completo*.



2.3 Variações e Problemas Semelhantes

As variações clássicas do PCV encontradas na literatura foram propostas para contemplar problemas cuja formulação clássica não poderia representar de maneira satisfatória. A seguir, são apresentadas algumas destas variações:

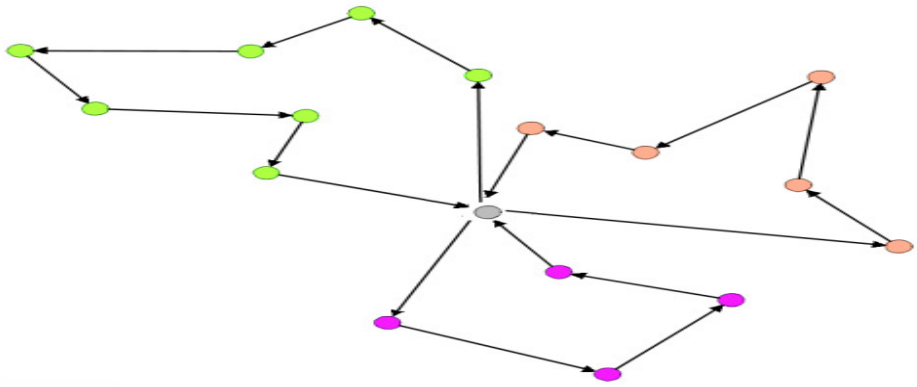
PCV com *Backhauls*: Nesta variação, o conjunto de localidades a serem visitadas é particionado em dois subconjuntos, assim, deve-se visitar todas as localidades do primeiro subconjunto e só depois, visitar as localidades do segundo subconjunto. Um exemplo prático dessa variação pode ser observado no problema de distribuição de

peças, onde um mesmo veículo deve primeiramente descarregar todas as peças para só depois coletar outras. Süral et. al. (2010) realizaram um estudo sobre este problema.

PCV Estocástico: Esta variação foi formulada para contemplar problemas em que as localidades que se deseja visitar juntamente com os custos para visitá-las podem variar conforme os resultados obtidos durante o processo de resolução. Toriello (2012) abstraiu a aplicação prática do PCV Estocástico no roteamento dinâmico do veículos, no qual o custo de uma decisão é conhecida apenas probabilisticamente de antemão, mas é revelado dinamicamente antes de a decisão ser executada.

PCV Múltiplo: Como o próprio nome sugere, nesta variação, múltiplos caixeiros são designados para que todos os pontos do problema sejam visitados. Logo, roteiros com o custo mínimo são configurados para que cada caixeiro viajante atenda ao menos a um ponto da rede. A Figura 2.4 ilustra um esquema básico de resolução do PCV Múltiplo. Bektas (2006) realizou uma revisão literária deste problema, enfatizando a aplicação prática desse problema juntamente com técnicas propostas para sua resolução.

Figura 2.4 – Esquema básico de resolução do PCV Múltiplo.



No contexto de roteirização, há problemas semelhantes ao PCV que sugiram da necessidade de se refletir, por meio da inclusão de restrições e especificações, a realidade das decisões que precisam ser tomadas diariamente por organizações ou entidades do segmento de transporte. As dificuldades de se modelar um problema de roteirização advém da grande quantidade de parâmetros que podem influenciar esse tipo de problema, logo, uma classificação adequada permite que seja implementada uma

estratégia de resolução precisa. Bodin *et al.* (1983) classificaram os problemas de roteamento em *roteirização pura* e *programação de veículos.* Problemas de *roteirização pura* visam construir rotas viáveis com o menor custo possível considerando apenas aspectos espaciais. Já em problemas de *programação de veículos,* constam tanto os aspectos espaciais quanto os temporais, como restrições de horários preestabelecidos para cada atividade a ser executada. Bodin e Golden (1981) relataram que são os problemas de *programação de veículos* que normalmente ocorrem na prática.

Ronen (1988) baseou-se no ambiente operacional e nos objetivos dos problemas para distingui-los em três classes. A primeira classe é formada por problemas relacionados com o transporte de passageiros, a segunda é composta por problemas referentes a prestação e programação de serviços, como roteirização de coleta de lixo ou entrega postal, e a terceira classe remete á problemas de transporte de cargas. Desrochers, Lenstra e Savelsbergh (1990) desenvolveram um esquema que suporta o desenvolvimento de modelos e sistemas de transporte, servindo também para classificar diversos problemas de roteirização e programação de veículos. A ideia é fornecer diretrizes a uma representação teórica do problema real, com base nos tipos de veículos, endereços, objetivos e características do problema, servindo como base para o desenvolvimento de modelo e sistemas, e facilitando escolha do algoritmo mais apropriado. Abaixo, são descritos alguns problemas presentes em Bodin *et al.* (1983), Ronen (1988), Desrochers, Lenstra e Savelsbergh (1990) e outros clássicos encontrados na literatura:

Problema do Carteiro Chinês (*Chinese Postman Problem* – CPP): Neste problema, o objetivo é determinar uma rota com custo mínimo que permita ao carteiro chinês passar por todos os arcos de uma rede sem repeti-los. Detofeno e Steiner (2010) abstrairam este problema na otimização de rotas para a coleta de resíduos sólidos urbanos.

Problema de Roteirização de Veículos (*Vehicle Routing Problem* – VRP): O objetivo deste problema é traçar caminhos mínimos para que uma frota de veículos, inicialmente em um ou mais depósitos, possa atender uma demanda de consumidores dispersos em um espaço geográfico, em um determinado horário, com diferentes demandas pelos

produtos a serem distribuídos. A diferença deste problema para o problema de múltiplos caixeiros está no acréscimo da restrição de capacidade de veículos. Há também outras restrições aplicadas a este problema, como janela de tempo para o atendimento de clientes e tempo máximo de viagem dos veículos. O VRP foi proposto em 1959 por Dantzig e Ramser, através de um estudo realizado sobre o problema de distribuição de gasolina por uma frota de veículos. Um estudo recente sobre este problema pode ser encontrado na publicação de Drexl (2012).

Problema de Dimensionamento e Roteirização de uma Frota de Veículos (*Fleet Size and Vehicles Routing Problem* – FSVRP): Trata-se de uma variação do problema de roteamento de veículos, onde deve-se primeiramente especificar a quantidade de veículos necessários (frota ilimitada) para o roteamento, e depois, determinar o roteiro mínimo de cada veículo, para que no final, as demandas de todos os clientes sejam atendidas com os custos fixos de veículos e os custos variáveis de roteirização sendo mínimos. Neste problema, se os veículos da frota possuem a mesma capacidade e custos fixos, pode-se dizer que a frota é homogênea. Caso contrário, a frota é denominada heterogênea. Subramanian et. al. (2011) abordaram este problema por meio de Meta-Heurísticas Híbridas.

Problema de Coleta e Entrega (*Pickup and Delivery Problems* – PDP): Este problema é tido como uma variação do VRP, onde há uma restrição de precedência no atendimento da demanda dos clientes, pois as cargas são transportadas pelos veículos do depósito aos clientes e entre os clientes. Estratégias de resolução do PDP podem ser vistas em Berbeglia, Cordeau e Laporte (2010).

2.4 Métodos de Resolução

Na literatura, destacam-se na resolução de problemas complexos como o PCV, métodos exatos ou heurísticos. Métodos exatos visam obter sempre o resultado ótimo, com isso, se faz necessário um maior esforço computacional para que o resultado seja alcançado em um tempo aceitável. Como exemplos de métodos exatos, pode-se citar Programação Linear, Programação Dinâmica, os algoritmos *Branch* & *Bound* (LAND & DOIG, 1960) e *Branch* & *Cut* (CROWDER, JOHNSON & PADBERG, 1983).

A Programação Dinâmica é uma metodologia de construção de algoritmos voltados para a resolução de problemas de otimização, cujo processo de obtenção de soluções particiona o problema em subproblemas, trabalhando do fim para o princípio (SZWARCCFITER, 1984). Proposta por Bellman em 1954, esta metodologia visa reduzir a dificuldade de resolução decompondo um problema numa sequência de problemas interrelacionados mais simples. Em Programação Linear, um modelo matemático é formulado para o problema através de uma função objetivo sujeita á restrições e solucionado pelo algoritmo SIMPLEX. Este método foi proposto por Dantzig em 1947 e até hoje é aplicado em problemas de otimização, seja por meio de técnicas computacionais ou *softwares* comerciais como o *LINGO* da empresa *Lindo Systems*.

O Algoritmo *Branch & Bound* opera a partir da enumeração inteligente dos pontos candidatos à solução ótima de um problema, realizando sucessivas partições do espaço de busca e cortando a árvore de pesquisa através da consideração de limites inferiores e superiores calculados ao longo da enumeração. Os limites inferiores são configurados a partir de métodos de relaxação que removem uma ou mais restrições do problema em questão. Na grande parte das aplicações deste algoritmo, os limites superiores são obtidos por heurísticas que produzem boas soluções em curtos intervalos de tempo. O esquema de enumeração divide o problema em vários sub-problemas menos complexos até que o limite inferior seja igual ao superior ou o limite inferior seja maior que a melhor solução corrente. Merz (2000) citou que esta abordagem produz uma ramificação (*branching*) na qual cada nó corresponde a um problema e os nós descendentes, os sub-problemas.

O algoritmo *Branch & Cut* baseia-se na combinação de um algoritmo *Branch & Bound* com técnicas de planos de corte visando melhorar a etapa de relaxamento e a exploração do politopo definido pelas soluções viáveis do problema considerado. Este algoritmo é indicado para problemas que são impossíveis de serem tratados eficientemente apenas por ferramentas de programação linear devido ao seu grande número de restrições, e funciona a partir do seguinte conceito: se uma solução ótima associada à relaxação linear é inviável, um novo problema de separação deve ser "resolvido" buscando a identificação de uma ou mais restrições violadas pela relaxação corrente (*cutting*

procedure). O novo problema obtido (com restrições adicionais) é novamente resolvido via programação linear e o processo é repetido até que novas classes de desigualdades violadas não sejam mais encontradas (MERZ, 2000). Exemplos de algoritmos *Branch & Bound* e *Branch & Cut* aplicados ao PCV podem ainda ser encontrados em Crowder e Padberg (1980), Balas e Toth (1985), Jfinger, Reinelt e Thienel (1994), Applegate et al. (1998, 2006 e 2009) e Erdogan, Cordeau e Laporte (2010).

Dumitrescu e Stützle (2003) atentaram para vantagens e desvantagens no uso de abordagens exatas na resolução de problemas de otimização. Dentre as vantagens, destacam que soluções ótimas podem ser obtidas caso o algoritmo tenha sucesso na sua execução. Como desvantagem, citam o consumo demasiado de recursos computacionais durante a resolução de instâncias de grande porte. Hoffman e Padberg (1985) e Wolsey (1998) elaboraram um estudo sobre a aplicação destes métodos em problemas de otimização. Em Laporte (1992), podem ser encontradas outras abordagens exatas aplicadas ao PCV.

Na prática, quando não se necessita achar uma solução ótima para o problema, dado que o tempo necessário para a mesma possa ser demasiadamente longo, utilizam-se então métodos heurísticos, que conseguem encontrar boas soluções ou, em alguns casos, soluções ótimas, com um consumo inferior de recursos computacionais ao das abordagens exatas. No próximo capítulo, é elaborado um estudo sobre métodos heurísticos de resolução para o problema.

CAPÍTULO 3 – MÉTODOS HEURÍSTICOS

Neste capítulo é apresentado um estudo sobre alguns dos métodos heurísticos aplicados ao PCV e a descrição das novas Heurísticas Híbridas criadas para serem aplicadas especificamente no PCV. As seções estão dispostas segundo a classificação proposta por Sirenko (2009) mencionada no primeiro capítulo deste trabalho. Na primeira seção, consta um estudo sobre Heurísticas aplicadas ao PCV. A segunda seção aborda o conceito de Meta-Heurística e alguns destes métodos aplicados ao PCV. Nas Seções 3.3 e 3.4 são descritas outras formas de resolver o problema, através de Meta-Heurísticas Híbridas e Hiper-Heurísticas. Por fim, na Seção 3.5 é apresentada uma das principais contribuição deste trabalho para a resolução do problema, onde seis novas formas de resolver o PCV foram criadas, implementadas e analisadas, através de heurísticas híbridas, sendo descritos os pseudocódigos e exemplos ilustrativos associados a execução de cada um desses métodos.

3.1 Heurística

Heurísticas, nome derivado da palavra grega *heuriskein* que significa descobrir (REEVES, 1995) refere-se a uma técnica que visa encontrar uma solução, não necessariamente a melhor, de forma rápida valendo-se de conhecimento sobre a estrutura e instância do problema. Consoante a dificuldade ou mesmo impossibilidade de obtenção de soluções exatas para problemas combinatórios de larga escala, foi proposto o desenvolvimento de algoritmos aproximativos ou métodos heurísticos. Heurísticas são concebidas para um tipo especifico de problema, contemplando apenas as particularidades e restrições do escopo do mesmo. Sendo assim, uma heurística dificilmente irá produzir boas soluções para problemas com escopo divergente daquele para o qual foi projetada.

Viana (1998, p. 91) conceituou heurística como qualquer método ou técnica criada ou desenvolvida, para solução de determinado tipo de problema. Trata-se de um método aproximativo desenvolvido especificamente para resolver um problema em tempo

polinomial. Reeves (1995, p.6) descreveu heurística como uma técnica que procura boas soluções com um tempo computacional razoável sem garantir nem viabilidade nem otimalidade, e ainda em muitos casos, sem definir quão próxima uma solução viável encontra-se da solução ótima de determinado problema específico. Rich e Knight (1993) citaram que para a resolução eficiente de muitos problemas difíceis, geralmente é necessário comprometer as exigências de mobilidade e sistematicidade e construir uma estrutura de controle que não garanta encontrar a melhor resposta, mas que quase sempre encontre uma resposta muito boa. Na literatura, encontram-se dois tipos de heurísticas sendo aplicadas ao PCV, são elas: Heurísticas Construtivas (*HC*) e Heurísticas de Melhoramento (*HM*).

3.1.1 Heurística Construtiva

Uma Heurística Construtiva têm por característica a construção de soluções por meio de um processo iterativo que inicia com uma solução vazia e adiciona um novo elemento a cada iteração até a obtenção de uma solução viável. A escolha de cada elemento a ser inserido em cada passo varia de acordo com a função de avaliação especificada pelo problema abordado. Schepke *et. al.* (2004) citou que heurísticas construtivas utilizam técnicas de adição na construção do problema de forma gradativa e contínua.

Heurísticas pertencentes a esta classe, normalmente, tendem a ordenar os elementos candidatos utilizando uma função gulosa, que calcula o benefício da inserção de cada elemento, e somente o "melhor" elemento é inserido a cada iteração. Se aplicado ao PCV, uma HC tentará encontrar uma "boa" rota, considerando a cada interação somente o próximo passo, ou seja, o critério de escolha é basicamente local (CAMPELLO & MACULAN, 1994). Algoritmos construtivos não possuem mecanismos de *backtracking*, logo, após inserida uma cidade, não é possível retirá-la da rota.

Na literatura, este tipo de Heurística é normalmente utilizada para gerar soluções iniciais para outros métodos heurísticos. Como exemplo de HC pode-se citar: Heurística do Vizinho mais Próximo, Heurística da Inserção Mais Barata e Heurística de Clark e Wright.

3.1.1.1 Heurística do Vizinho mais Próximo

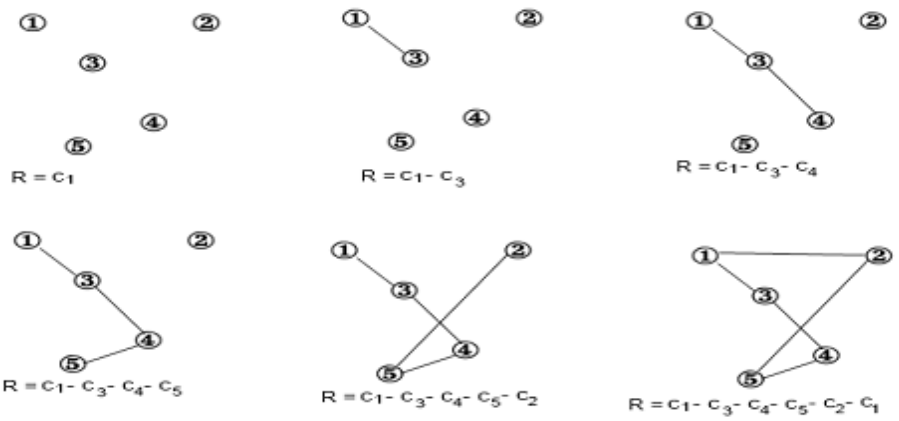
Neste método heurístico, define-se uma cidade inicial da rota e segue-se inserindo na rota corrente, a cidade que apresentar o menor custo para se percorrer em relação à última cidade adicionada. Este processo iterativo de adição repete-se até que todos os elementos sejam percorridos e a rota seja concluída. Uma vez que este método constrói a solução passo a passo, segundo um conjunto de critérios pre-estabelecidos, pode ser denominado como método guloso (g*reedy)*. Na literatura, estudos relacionados a aplicação desta Heurística ao PCV podem ser encontrados em Menger (1932), Gavett (1965), Bellmore e Nemhauser (1968), Solomon (1987) e Kurz (2011). A Tabela 3.1 serve como instância do PCV para n=5, contendo as distâncias entre as cidades.

Tabela 3.1 - Dados utilizados para exemplificar a aplicação das heurísticas construtivas.

Matriz D	c_1	c_2	c_3	c_4	c_5
c_1	0	8	4	9	9
c_2	8	0	6	7	10
c_3	4	6	0	5	6
c_4	9	7	5	0	4
c_5	9	10	6	4	0

Aplicando a Heurística do Vizinho mais Próximo aos elementos da Tabela 3.1 obtém-se então a rota R de ordem (c_1-c_3-c_4-c_5-c_2-c_1) cujo custo total é 31. A Figura 3.1 ilustra a rota formada com os seguintes passos:

Figura 3.1 – Rota da Heurística do Vizinho Mais Próximo, peso igual a 31, iniciando por c_1.



Passo 1: Seja $R = (c_1)$, adiciona-se c_3 a R, tendo em vista que c_3 é a cidade com menor custo na vizinhança de c_1. $R = (c_1\text{-}c_3)$;

Passo 2: Adiciona-se c_4 a *R*, pois ela é o melhor vizinho de c_3. $R = (c_1\text{-}c_3\text{-}c_4)$;

Passo 3: Adiciona-se c_5 a *R*, pois ela é o melhor vizinho de c_4. $R = (c_1\text{-}c_3\text{-}c_4\text{-}c_5)$;

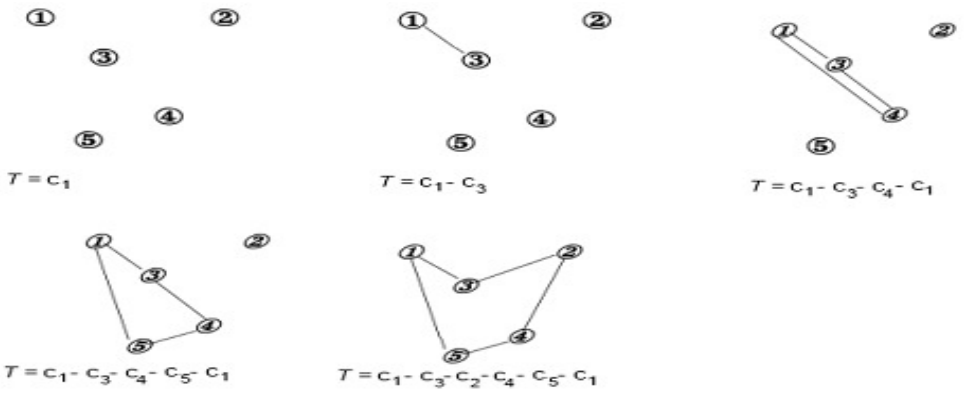
Passo 4: Adiciona-se c_2 a *R*, pois ela é o melhor vizinho de c_5. $R = (c_1\text{-}c_3\text{-}c_4\text{-}c_5\text{-}c_2)$.

Passo 5: Conclui-se a rota conectando c_2 a c_1, tendo em vista que todas as cidades já foram visitadas uma única vez e $R = (c_1\text{-}c_3\text{-}c_4\text{-}c_5\text{-}c_2\text{-}c_1)$.

3.1.1.2 Heurística da Inserção Mais Barata

Proposta por Karg e Thompson (1964) e referenciada como Heurística da Inserção Mais Barata (*HIMB*) por Solomon (1987), este método heurístico procura definir um roteiro inicial com somente três cidades, sendo as demais cidades ainda não incluídas no roteiro, selecionadas para inserção no roteiro parcial, aquelas que proporcionam menor acréscimo de distância total percorrida. Esse procedimento é repetido sucessivamente, com a análise da inserção entre cada par de cidades do roteiro parcial, até que todas as cidades sejam inseridas. O mecanismo de escolha da cidade a ser avaliada para inserção pode ser implementado para proceder de forma sequencial, aleatória ("Inserção Aleatória") ou baseado na cidade que está mais distante do roteiro parcial ("Inserção Mais Distante"). Em Rei (1994), são descritas variantes desta heurística.

Figura 3.2 – Rota construída pela HIMB com peso igual a 30.



A Figura 3.2, dada anteriormente, mostra uma aplicação de HIMB aplicada no problema dado na Tabela 3.1, onde uma cidade inicial c_i é escolhida arbitrariamente e logo depois, o método procura pela cidade c_j mais próximo de c_i, formando uma sub-rota $T=(c_i\text{-}c_j\text{-}c_i)$. Em seguida, procura-se pela cidade c_k que proporcione o menor custo de inserção $D_{ik} + D_{kj} - D_{ij}$ entre algum par de cidades c_i e c_j já incluídas na sub-rota T e insere-se c_k entre este respectivo par de cidades que compõem T. Este processo de inserção é iterado até que todas as cidades do problema sejam inseridas em T.

3.1.1.3 Heurística de Clark e Wright

A Heurística de Clark e Wright, também conhecida como Heurística de *Savings* (economias), é um método econômico-iterativo construtivo de busca de soluções para problemas de roteirização. Proposta em 1964, este método heurístico inicia com um processo iterativo que procura percorrer todas as cidades, duas a duas, de forma a calcular as economias deste deslocamento e considerando ainda, o custo de retornar à cidade inicial. Durante este processo iterativo, a função gulosa de inserção escolhe sempre a maior economia dentre as possíveis.

Schepke *et. al.* (2004) destacaram que a noção de economia pode ser definida como o custo da combinação, ou união, de duas sub-rotas existentes. Se esta heurística for aplicada a um problema de roteirização, matematicamente, as economias E_{ij} podem ser definidas como:

$E_{ij} = D_{ik} + D_{kj} - D_{ij}$, onde E_{ij} representa a economia de seguir a rota $c_i\text{-}c_k\text{-}c_j$ em vez de $c_i\text{-}c_j$.

Assim, as cidades mais próximas são descobertas e, em seguida, inicia-se outro processo iterativo que busca o melhor *saving*, considerando as duas cidades mais próximas uma da outra, roteando-as à cidade base.

3.1.2 Heurística de Melhoria de Roteiros

Heurísticas de Melhoria de Rotas, denominadas de *k-Opt*, também chamadas de técnicas de Busca Local ou de Refinamento, constituem uma família de Heurísticas utilizadas

para melhorar uma solução viável, proposta por outro método heurístico. Estes métodos genéricos podem assistir o processo de obtenção de soluções de vários problemas de otimização sem a necessidade de um profundo entendimento dos mesmos e funcionam com o conceito de vizinhança, dado a seguir.

Seja S o espaço de soluções viáveis de um problema de otimização e f uma função objetivo de minimização deste problema, a vizinhança de s_0 (solução inicial) remete ao conjunto $N(s_0) \subset S$ que reúne um numero finito de soluções candidatas s. Cada solução s é denominada como *vizinho* de s_0 e é obtida a partir de um *movimento*. Por *movimento,* entende-se como a operação que transforma uma solução s em outra, s'. O processo de execução de um algoritmo que funciona sobre este conceito de vizinhança cessa após um determinado número de *movimentos* realizados. Com isso, é obtida uma solução correspondente a um ótimo local, porém não há garantias de que esta solução seja um ótimo global. A Figura 3.3 abaixo apresenta a estrutura básica deste tipo de algoritmo.

Figura 3.3 – Estrutura básica de algoritmo de Busca Local.

```
Selecione uma solução inicial s0 ∈ S
Repita
        Selecione s ∈ N(s0) tal que f(s) < f(s0)
        s0 ← s
Até que f(s) > f(s0) para todo s ∈ N(s0)
```

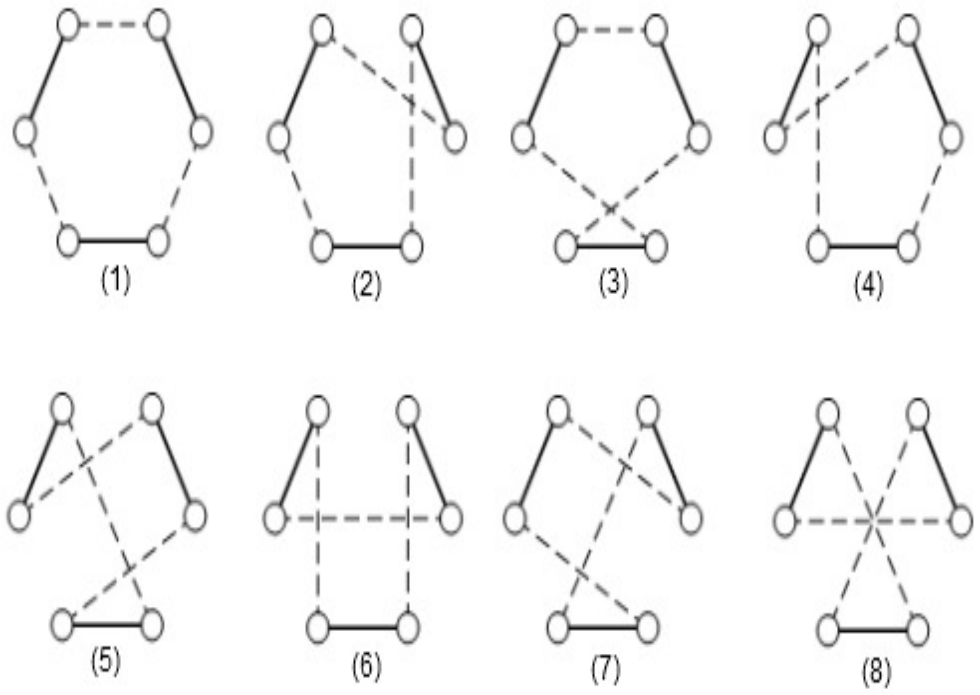
Na literatura, *k-Opt* destaca-se pela simplicidade e praticidade. Este método baseia-se na ideia da troca de conexões entre cidades para otimizar o percurso, sendo k, equivalente a um número de reconexões a serem designados para cada parte da rota, as vezes tratada como sendo uma permutação de n. A primeira abordagem remete a 1958 no artigo de Croes, que tratou da troca de 2 conexões não consecutivas. Posteriormente, em 1965, Lin abordou esse método para 3 reconexões. Caldas e Santos (2008) destacaram que melhor será a solução obtida se maior for o valor de r, todavia, maior também será o tempo computacional empregado.

Abordagens comuns do método *k-Opt* são 2-*Opt* e 3-*Opt*. Schepke (2004) citou que o método de melhoramento 2-*Opt* intra-rotas consiste na possível troca de duas conexões, não consecutivos, de uma rota e refazer as conexões. Refeitas as conexões, tem-se uma nova rota cujo custo é avaliado e se for menor em relação ao custo da rota original, a

nova rota é mantida. Normalmente, utiliza-se como critério de parada, um número máximo de iterações, mesmo sem que haja melhora na solução. Existe também a heurística 2-*Opt* inter-rotas, que consiste na criação de duas novas rotas a partir das possíveis trocas de dois arcos destas duas rotas diferentes. Assim como a abordagem anterior, após a reconexão das cidades, as novas rotas somente são mantidas se o custo delas forem inferiores ao custo das rotas originais.

Geralmente, na heurística *k-Opt*, *k* conexões são eliminadas em um *movimento* e o número de reconexões a serem testadas é dado por $(k-1)!2^{(k-1)}$. Sendo $k = 3$, as 8 combinações de ligações entre as cidades são ilustradas na Figura 3.4.

Figura 3.4 – Reconexões realizadas pelo algoritmo 3-*Opt*.



3.2 Meta-Heurística

Os métodos heurísticos propostos para resolver problemas de otimização combinatória normalmente são específicos e dedicados a um determinado problema. No final da década de 80 observou-se a evolução deste paradigma, dado um crescente interesse em

técnicas flexíveis para contemplar a resolução de diversos tipos de problemas de otimização, e surgiu então o conceito de Meta-Heurística.

Meta-Heurísticas podem fornecer boas soluções para diferentes tipos de problemas, assim sendo consideradas heurísticas de uso e aplicação geral. A característica primordial de uma Meta-Heurística é sua capacidade de diversificar o campo de soluções evitando paradas em ótimos locais. Paradas, prematuras, em ótimos locais no processo iterativo de busca de soluções são evitadas pelas Meta-Heurísticas por meio de mecanismos de *fuga*, como combinação de escolhas aleatórias e conhecimento histórico dos resultados anteriores adquiridos. Estes mecanismos atuam direcionando a busca para outras vizinhanças dentro do vasto espaço de pesquisa.

Viana (1998) citou que como as Meta-Heurísticas utilizam funções de probabilidade, não há garantia de obtenção de um mesmo resultado para uma determinada instância de um problema, o que não ocorre com as heurísticas. Sousa (2009) atentou para o fato de que a grande diferença entre as Meta-Heurísticas está no algoritmo de pesquisa da vizinhança, onde o processo de busca precisa ser eficiente para que se possa encontrar uma solução de boa qualidade ou até mesmo a solução ótima. Para Hillier e Lieberman (2010), Meta-Heurísticas podem ser descritas como métodos genéricos de resolução capazes de fornecerem, tanto diretrizes estratégicas, quanto estruturas gerais de desenvolvimento para um método heurístico específico que se ajuste a um tipo específico de problema.

Hillier e Lieberman (2010) citaram também que sempre que um algoritmo garantir otimalidade para um problema prático, deve-se usar então esse algoritmo. Blum e Roli (2003) subdividiram Meta-Heurísticas em: métodos que produzem uma solução única por iteração, como *Simulated Annealing* (*SA*), *Tabu Search* (*TS*), *Greedy Randomized Adaptive Search Procedure* (*GRASP*), *Iterated Local Search* (*ILS*), *Variable Neighborhood Search* (VNS); métodos que produzem uma população de soluções por iteração, como *Evolutionary Algorithms* (EA), Algoritmos Genéticos (*AG*), *Scatter Search* (SS) e *Adaptive Memory Procedures* (AMP); e métodos que, a cada iteração, obtém soluções através de mecanismos de aprendizagem, como Redes Neurais (RN),

Colonia de Formigas (CF), *Particle Swarm Optimization* (PSO). A seguir, destacam-se algumas destas Meta-Heurísticas de aplicação comum no PCV.

3.2.1 Simulated Annealing

A Meta-Heurística *Simulated Annealing* (SA) é um algoritmo introduzido por Kirkpatrick, Gelatt e Vecchi (1983) em otimização combinatória, e mais tarde, aperfeiçoada por Cerny (1985), inspirado nos processos de têmpera de sólidos desenvolvidos em 1953 por Metropolis. Estes pesquisadores, observaram que o procedimento usado para a mudança de estado físico do material poderia ser também aplicado na resolução de um problema de otimização, tendo em vista similaridades entre suas estruturas. Assim, eles propuseram um algoritmo capaz de simular um processo semelhante ao *annealing*, para encontrar a configuração ótima de um problema complexo. O termo *annealing* refere-se a um processo de tratamento térmico que começa pela liquidificação de um cristal, à alta temperatura, seguido pela lenta e gradativa diminuição de sua temperatura, até que o ponto de solidificação seja alcançado, quando o sistema atinge um estado de energia mínima (IGNACIO, FERREIRA FILHO e GALVÃO, 2000).

Por analogia, a temperatura do processo físico se torna um parâmetro de controle a ser determinado para conseguir os resultados desejados e a energia livre do material é tida como a função objetivo do problema. Brião (2008) citou que a implementação deste método heurístico consiste em um laço externo que controla a temperatura e um laço interno que realiza três passos: perturbação, avaliação e decisão. A partir de uma solução corrente, uma perturbação gera uma nova solução, vizinha à anterior. A nova solução é então avaliada e tem-se um ΔC = *custo final – custo inicial* indicando a qualidade da nova solução em relação à anterior. Se a solução proposta é melhor que a anterior, então esta é a nova solução corrente. Se for pior, aqui está o ponto importante do SA: a solução proposta é aceita com probabilidade $e^{-\Delta_{C/Kb.T}}$ onde ΔC é a variação do custo da solução, Kb é a constante de *Boltzman* e T é a temperatura do sistema. Ainda segundo Brião (2008), o sistema deve evoluir de acordo com a distribuição de Boltzman.

Schneider (2002) apresentou bons resultados da metaheuristica SA ao aplicá-la no famoso problema das 127 fábricas de cerveja da cidade de Augsburg, na Alemanha. Este problema é representado pela instância BIER127, que pode ser encontrada na TSPLIB. Outros estudos relacionados à aplicação da Meta-Heurística SA ao PCV podem ser encontrados em Viana (1998) e Chen e Zhang (2006). A Figura 3.5 ilustra um pseudocódigo desta Meta-Heurística.

Figura 3.5 – Pseudocódigo da Meta-Heurística Simulated Anealing.

```
PROCEDIMENTO SIMULATED_ANNEALING()
INPUT: temperaturaInicial, temperaturaFinal: REAL;
OUTPUT: solução: ESTADO

solução = soluçãoAleatória
temperatura = temperaturaInicial;

ENQUANTO (temperatura > temperaturaFinal) FAÇA
        REPITA
                novaSolução = Perturbação(solução);
                ΔC = Avaliação(solução, novaSolução);
                SE (Decisão(ΔC, temperatura) == ACEITA) ENTÃO
                        solução = novaSolução;
                FIM_SE
        ATÉ (sistema em equilíbrio nessa temperatura);
        temperatura = funçãoDeResfriamento(temperatura);
FIM_ENQUANTO
FIM_PROCEDIMENTO

FUNÇÃO Perturbação(solução:ESTADO): ESTADO
RETORNA estado válido, vizinho a solução;
FIM_FUNÇÃO

FUNÇÃO Avaliação(solução:ESTADO, novaSolução:ESTADO): REAL
RETORNA Custo(novaSolução) - Custo(solução);
FIM_FUNÇÃO

FUNÇÃO Decisão(ΔC:REAL, temperatura:REAL): BOOLEAN
Aceita_condição:BOOLEAN
        SE (ΔC < 0) OU (e^(-ΔC/KbT) < rand())
        ENTÃO aceita_condição = ACEITA;
        SENÃO aceita_condição = REJEITA;
RETORNA aceita_condição;
```

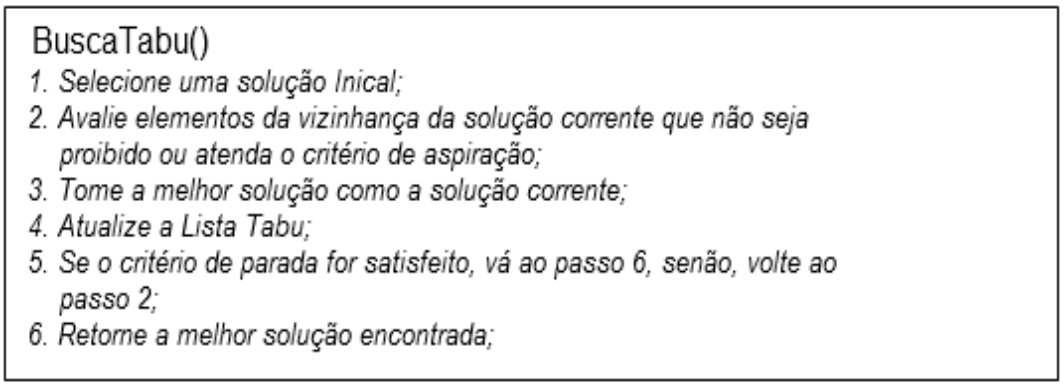
3.2.2 Busca Tabu

A Meta-Heurística Busca Tabu (BT), do inglês *Tabu Search*, foi criada por Fred Glover no final da década de 1980 para tratar problemas de Programação Inteira. Este algoritmo

consiste basicamente em guiar outros algoritmos heurísticos para gerar soluções melhores que as que seriam geradas por estes algoritmos, começando com uma solução inicial e progredindo iterativamente de uma solução para outra até que um critério de parada seja satisfeito. Cada solução pertence a uma vizinhança dentro do espaço de busca e para cada vizinhança, busca-se por uma solução de melhor qualidade. Ao processo de passagem da solução corrente para a outra solução com melhor configuração da vizinhança corrente ou a melhor dentre as visitadas, denomina-se *movimento*. Guimarães (2005) citou que as vizinhanças consideradas no processo de busca desta Meta-Heurística podem ter suas estruturas expandidas dinamicamente caso não seja encontrada uma boa solução dentro da vizinhança corrente.

Visando evitar ciclagem e retornos as configurações já visitadas, utiliza-se uma estrutura de dados denominada *lista tabu*. Esta lista guarda os atributos das configurações visitadas anteriormente. Para impedir o retorno a configurações já visitadas, estes atributos recebem o status de *proibido*. Em casos em que uma solução candidata possuir atributos tidos como *proibidos*, porém for melhor que a solução corrente, uma função do algoritmo, denominada *critério de aspiração*, é acionada. Esta função irá eliminar o processo de proibição caso a solução candidata satisfaça a um determinado *critério de aspiração*. Estudos sobre a aplicação desta Meta-Heurística aplicada ao PCV podem ser encontrados em Viana (1998) e Gomes (2011). Um pseudocódigo simples da Meta-Heurística BT é mostrado na Figura 3.6.

Figura 3.6 – Algoritmo da Metahurística Busca Tabu.



3.2.3 Algoritmo Genético

Apresentada em 1975 pelo pesquisador John H. Holland, esta Meta-Heurística apoia-se nos mecanismos de seleção e adaptação natural das espécies propostos por Charles Darwin para tratar problemas de otimização. O nome Algoritmo Genético deve-se a abstração de sistemas artificiais com propriedades similares aos sistemas naturais realizada pelo processo de busca deste algoritmo, que utiliza em cada iteração operadores genéticos e critérios de seleção, para avaliar e substituir uma população de indivíduos por outra, com valores de aptidão em média melhores.

De forma análoga, um algoritmo genético inicia produzindo uma geração de indivíduos (soluções candidatas) aleatoriamente. A cada geração, a adaptação dos indivíduos é avaliada, onde alguns são selecionados para serem recombinados ou mutados para formar uma nova geração. A nova geração então é utilizada como entrada para a próxima iteração do algoritmo. Uma das características mais importantes dos algoritmos genéticos é a manipulação de diversas soluções ao mesmo tempo. Com esta característica a Meta-Heurística tem a capacidade de explorar com mais eficiência o espaço de busca, melhorando a qualidade das soluções (GUIMARÃES, 2009). A Figura 3.7 apresenta um pseudocódigo para implementação de um AG.

Figura 3.7 – Pseudocódigo de um Algoritmo Genético.

```
ALGORITMOGENETICO( )
1.  t = 0;
2.  criar população P(t);
3.  for cada indivíduo i de P(t)
4.     Avaliar aptidão indivíduos(i)
5.  end for
6.  while condição de para não satisfeita
7.     t = t+1;
8.     Selecionar população P(t) de P(t-1);
9.     Aplicar operadores de cruzamento sobre P(t);
10.    Aplicar operadores de mutação sobre P(t);
11.    Avaliar P(t);
12. end while
```

Viana (1998) destacou que esta Meta-Heurística usa técnicas de randomização, regras probabilísticas (não determinísticas) e trabalha unicamente com o valor da função objetivo. Bons resultados foram obtidos com a aplicação do AG no PCV: Silva, Soma e Viana (2004) e Whitley, Hains e Howe (2010).

3.2.4 GRASP

GRASP (*Greedy Randomized Adaptive Search Procedures*) foi proposta por Thomas A. Feo e Mauricio G. C. Resende (1989) para tratar problemas de otimização complexos e de grande porte. Trata-se de uma técnica iterativa em que uma solução aproximada para o problema é gerada a cada iteração por meio de transições aleatórias. A primeira solução de cada iteração é dada por um algoritmo construtivo aleatório e as demais soluções são obtidas aplicando-se à solução inicial um algoritmo de busca local. Gonçalves (2009) citou que a partir de uma solução *s* obtida na fase de construção, a fase de busca local, procura por uma melhor solução s_0 em uma vizinhança de *s*, *N(s)*. Enquanto houver melhoria na solução, o processo de busca local continua até encontrar um ótimo local, isto é, encontrar uma solução *s'* melhor que qualquer *s* em *N(s')*. Ao final de todas as iterações, a solução resultante é a melhor solução gerada. A Figura 3.8 apresenta um pseudocódigo para o GRASP.

Figura 3.8 – Pseudocódigo para a Meta-Heurística GRASP.

```
GRASP()
1.  enquanto um critério de parada não for satisfeito
2.      s ← fase_de_construcao()
3.      s ← busca_local(s)
4.      atualizar_melhor_solucao(s*)
5.  fim_enquanto
```

É importante destacar que a fase de busca local tenta melhorar as soluções geradas no decorrer da execução, pois não há garantias de que as soluções geradas na fase de construção não direcionem a busca para ótimos locais. A eficiência da fase construtiva pode ser melhorada com uma implementação minuciosa e o uso de estruturas de dados

personalizadas. Com uma fase construtiva eficiente, provavelmente, a fase de busca local também será (GONÇALVES, 2009). Estudos relacionados à aplicação da Meta-Heurística GRASP ao PCV podem ser encontrados em Viana (1998) e Marinakis, Migdalas e Pardalos (2008).

3.2.5 Colônia de Formigas

A Meta-Heurística Colônia de Formigas simula o comportamento de uma colônia de formigas (agentes inteligentes) que cooperam entre si por meio de mecanismos de comunicação simples visando a resolução de um determinado problema de otimização. Criada em 1992 por Marco Dorigo, esta Meta-Heurística só se tornou conhecida em 1996 na publicação do trabalho Dorigo, Maniezzo e Colorni (1996). Nela, os autores, visando demonstrar a eficiência do método, designaram uma população de agentes com orientações diversas e guiados por uma força gananciosa (feromônio) para resolver o Problema do Caixeiro Viajante. Foi demonstrado que os agentes optavam pelo caminho que possuísse a maior quantidade de feromônio e quando um caminho era escolhido, a quantidade de feromônio neste era reforçada. Desta forma os agentes interagiam a fim de indicar a melhor rota possível para o problema.

O efeito sinérgico gerado pelas interações dos agentes é uma característica marcante deste método, pois a qualidade da solução obtida tende a aumentar com uma maior cooperação dos agentes para resolver um mesmo problema. Carbajal, Corne e Reid (2010) aplicaram esta Meta-Heurística no *Desafio de Mona Lisa.* O *Desafio de Mona Lisa* foi proposto Robert Bosch em 2009 e trata-se de uma instância do PCV com n=100.000, cujo processo de resolução da mesma tende a reconstruir o quadro *Mona Lisa* de Leonardo da Vinci. Esta instância foi construída a partir de técnicas de Computação Gráfica desenvolvidas por Bosch e Herman (2004). Outros estudos sobre a aplicação desta Meta-Heurística aplicada ao PCV podem ser encontrados em Kötzing *et. al.* (2010) e López-Ibáñez e Blum (2010). A Figura 3.9 ilustra o pseudocódigo desta Meta-Heurística aplicada ao PCV.

Figura 3.9 – Pseudocódigo para a Meta-Heurística Colônia de Formigas.

```
ColôniaDeFormigas()
1   repita
2       Aleatoriamente posicione m formigas em n nós
3       para nó = 1 até n faça
4           para formiga = 1 até m faça
5               Escolha, probabilísticamente, o próximo nó para se mover
6           fim para
9       fim para
7       Calcule a distância percorrida de cada formiga
8       Aplique o rastro feromônio de cada formiga
10      Aplique a evaporação do feromônio
11  até Condição-de-saída
```

3.3 Meta-Heurística Híbrida

Trata-se de um método heurístico construído a partir da junção de duas ou mais Meta-Heurísticas em um mesmo procedimento. A intenção teórica desta proposta está em tentar evitar com maior facilidade os ótimos locais, refletindo assim, em uma melhora nos resultados obtidos na prática com os métodos heurísticos combinados. Viana (1998) combinou as Meta-Heurísticas Busca Tabu, GRASP, Algoritmo Genético e técnicas de processamento paralelo na resolução do PCV. O termo paralelismo refere-se ao processamento simultâneo existente na execução de programas, sub-rotinas e instruções (VIANA, 1998). Outros pesquisadores que estudam Meta-Heurísticas Híbridas e Paralelismo são Cotta, Talbi e Alba (2005).

Mestria (2011) aplicou Meta-Heurísticas Híbridas em instâncias do PCV com até 2.000 cidades. As abordagens desenvolvidas por Mestria foram baseadas nas Meta-Heurísticas GRASP, *Iterated Local Search*, Vizinhança Variável e em módulos de Reconexão de Caminhos (*RC*), originalmente proposto por Glover (1996) como uma estratégia de intensifícação que explora trajetórias conectando soluções elite obtidas pela Busca Tabu e *Scatter Search* (MESTRIA, 2011). Outros estudos sobre a aplicação de Meta-Heurísticas Híbridas podem ser encontrados em Baraglia, Hidalgo e Perego (2001), Tsai *et. al.* (2004) e Nguyen *et. al.* (2007).

3.4 Hiper-Heurística

Por Hiper-Heurística entende-se como uma técnica de otimização capaz de controlar um conjunto de outras heurísticas na resolução de problemas de otimização. Este método visa equilibrar vantagens e desvantagens de um conjunto de heurísticas específicas para um tipo de problema onde, em cada ponto de decisão, uma heurística é devidamente selecionada para ser aplicada. A diferença entre Hiper-Heurísticas e os demais métodos está no fato de que as Hiper-Heurísticas operam em um espaço de busca composto por Heurísticas menos complexas, denominadas Heurísticas de baixo nível, e os demais métodos heurísticos tendem a buscar dentro de um espaço composto por soluções candidatas. Similar ao conceito de Meta-Heurística, uma Hiper-Heurística pode ser aplicada facilmente a tipos diferentes de problemas devido a sua a natureza flexível e robusta.

Hiper-Heurísticas ganharam destaque na resolução de problemas de otimização no começo da década de 2000 por se mostrarem como uma técnica genérica e robusta, aplicável em diferentes problemas de otimização. Anterior aos anos 2000, segundo Chakhlevitch e Cowling (2008), o uso de Hiper-Heurística devia-se à necessidade em superar dificuldades adversas encontradas na codificação e manutenção da viabilidade de soluções. Na literatura, alguns pesquisadores utilizaram o PCV para validarem a eficiência de suas Hiper-Heurísticas, dentre os quais pode-se mencionar Chen, Kendall e Berghe (2007) e Kendall e Li (2012).

3.5 Heurística Híbrida

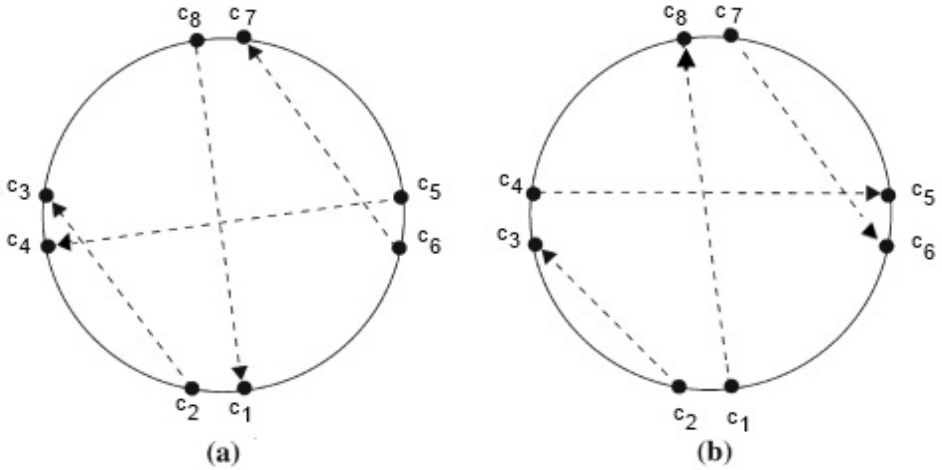
Uma Heurística Híbrida concebida para a resolução de problemas de otimização é uma abordagem que incorpora características de mais de uma heurística. Também são conhecidas por Heurísticas Compostas. Na teoria, a premissa para se construir um método híbrido é a preservação das boas características e eliminação das más características dos métodos heurísticos originais. Porém, nem sempre isso ocorre e o que se observa na prática, é que os métodos tendem a se complementar. De qualquer forma, os resultados obtidos acabam sendo melhores que os dos métodos isolados. A

vantagem deste tipo de abordagem sobre as Meta-Heurísticas Híbridas se dá na obtenção de boas soluções com um consumo inferior de recursos computacionais.

Neste trabalho, foram implementadas Heurísticas Híbridas capazes de apresentar soluções satisfatórias para o PCV com um consumo de tempo baixo. Estas abordagens híbridas são aplicáveis tanto no PCV Simétrico quanto no Assimétrico e possuem uma fase de construção de roteiros e uma fase de melhoria de roteiros. A fase de construção de cada uma destas abordagens híbridas é realizada por uma variação da Heurística do Vizinho mais Próximo (*HVP*) ou da Inserção Mais Barata(HIMB).

Já a fase de melhoria de roteiros de todas as heurísticas híbridas desenvolvidas, é feita por uma Heurística 4-*Opt*. Esta heurística, também conhecida como, método *double-bridge*, foi proposta por Martin, Otto e Felten (1991) para o PCV como um algoritmo de melhoria de roteiros cujo *movimento* consiste na remoção de quatro arcos de uma determinada rota e na inserção de outras quatro de tal maneira a gerar uma nova rota.

Figura 3.10 – Exemplo de um *movimento 4-Opt* viável(a) e um inviável(b).



Helsgaun (2009) citou que a avaliação da viabilidade de um *movimento k-Opt* (Figura 3.10) é uma questão frequente desta heurística. A Figura 3.10a mostra um movimento que resulta no ciclo (c_2-c_3-c_8-c_1-c_6-c_7-c_5-c_4-c_2), tornando assim, este movimento viável. Por outro lado, o movimento de *4-Opt* na Figura 3.10b produz dois ciclos, (c_2-c_3-c_2) e (c_4-c_5-c_7-c_6-c_1-c_8-c_4), não gerando uma rota viável. Uma maneira de avaliar se um

movimento é viável, seria por meio de um procedimento capaz de percorrer a sequência de cidades dispostos na rota produzida verificando se não há mais de um ciclo entre as cidades. Helsgaun (2009) propõe outra maneira de verificar a viabilidade de um *movimento*.

Segundo o autor, se for possível compor um caminho que percorra em sequência apenas as cidades incluídas no *movimento* considerando ainda as 4 novas ligações criadas, então o *movimento* é dado como viável. Por exemplo, na Figura 3.10a, o *movimento* é viável, por que, partindo da cidade c_6, é possível percorrer as demais cidades considerando as ligações resultantes do *movimento* operado. Dessa forma, as 8 cidades são percorridas na seguinte sequência: c_6, c_7, c_5, c_4, c_2, c_3, c_8, c_1.

3.5.1 Heurística Híbrida do Vizinho mais Próximo

Nesta Heurística, desenvolvida por nos, o processo de construção da rota seleciona uma cidade inicial e segue adicionando cidades à rota em formação. A cidade *i* mais próxima de um dos extremos da rota, ainda não visitada, é colocada na rota a cada iteração. Para diversificar a fase de construção, o processo de escolha da cidade inicial é repetido para todas as cidades que constituem o itinerário almejado. Desta forma é constituída uma rota iniciando de cada cidade do problema, sendo a melhor, mantida para a fase de melhoria de roteiros. É fácil ver que esta heurística construtiva possui complexidade $O(n^2)$, onde um pseudocódigo é apresentado na Figura 3.11.

Figura 3.11 – Pseudocódigo da HVP proposta.

```
HHVP( )
    1.  n ← Número de cidades do itinerário;
    2.  eX = null;        // Cidade localizada na extremidade X da Rota
    3.  eY = null;        // Cidade localizada na extremidade Y da Rota
    4.  vx = null;        // vizinho de eX
    5.  vY = null;        // vizinho de eY
    6.  count = null;  // Contador de cidades incluídas na Rota
    7.  for i=1, n do
    8.      eX ← i     // i é a cidade inicial;
    9.      eY ← Selecionar cidade mais próxima de eX;
    10.     Adicionar eX e eY à Rota nesta ordem;
    11.     count := 2;
```

```
12.  while count < n do
13.          v_X ← Selecionar cidade ainda não visitada mais próxima de e_X;
14.          v_Y ← Selecionar cidade ainda não visitada mais próxima de e_Y;
15.          if (distância de v_X para e_X ) < (distância de v_Y para e_Y ) then;
16.                  Adicona v_X ao início da Rota;
17.                  e_X := v_X;
18.          else
19.                  Adicona v_Y ao fim da Rota;
20.                  e_Y := v_Y;
21.          end if
22.          count ++;
23.  end while
24.  Conectar e_x e e_y para concluir a Rota;
25.  Comparar Rota recém-formada com a Rota formada na iteração anterior;
26.  Manter melhor Rota;
27. end for
```

A Tabela 3.2 foi elaborada para abstrair uma instância do PCV, com *n*=8, e exemplificar a aplicação da HHVP e das próximas Heurísticas Construtivas, também desenvolvidas por nos, a serem apresentadas.

Tabela 3.2 - Dados utilizados para exemplificar a aplicação das heurísticas construtivas.

Cidade	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	8	4	9	9	7	6	5
2	8	0	6	7	1	15	3	3
3	4	6	0	5	6	5	3	5
4	9	7	5	0	4	6	4	9
5	9	10	6	4	0	3	8	15
6	7	15	5	6	3	0	10	13
7	6	3	3	4	8	10	0	6
8	5	3	5	9	15	13	6	0

Aplicando HHVP aos dados da Tabela 3.2, obtêm-se, como pode ser visto na Figura 3.12, dada a seguir, a rota *R* de ordem (c_8-c_4-c_5-c_6-c_1-c_3-c_7-c_2-c_8) cujo custo total é 37. Os passos para construção de *R* são descritos abaixo, tomando a cidade inicial sendo c_1:

Passo 1: Adiciona-se c_3, como elemento seguinte, já que é a cidade com menor custo na vizinhança de c_1, sendo c_3 a segunda extremidade da rota em formação. $R = (c_1\text{-}c_3)$;

Passo 2: Adiciona-se c_7, por ser esta, a cidade mais próxima de um dos dois extremos e representar o menor acréscimo a distância total de *R*. A cidade c_7 substitui c_3 como extremo. $R = (c_1\text{-}c_3\text{-}c_7)$;

Passo 3: Adiciona-se c_2, já que é esta, a cidade que representar o menor acréscimo a distância total de *R*. Neste passo, c_2 substitui c_7 como extremo de *R*. $R = (c_1\text{-}c_3\text{-}c_7\text{-}c_2)$;

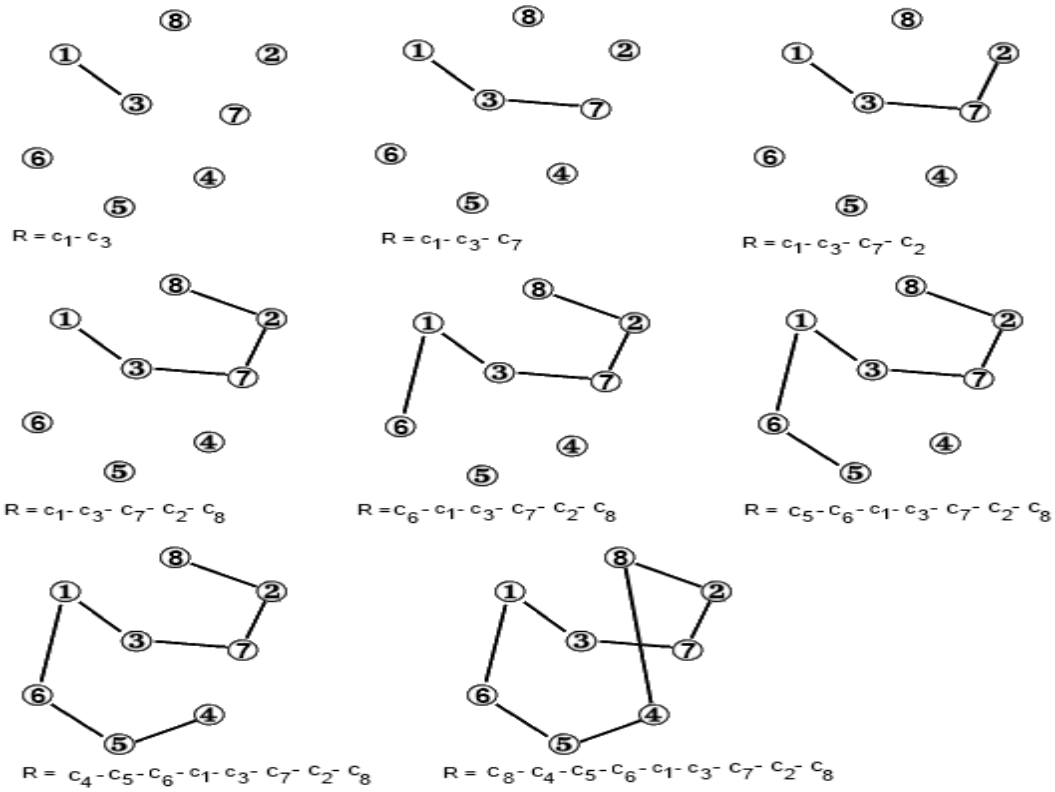
Passo 4: Adiciona-se c_8, já que é esta, a cidade que representar o menor acréscimo a distância total de *R*. Neste passo, c_8 substitui c_2 como extremo de *R*. $R = (c_1\text{-}c_3\text{-}c_7\text{-}c_2\text{-}c_8)$;

Passo 5: A cidade que representa o menor acréscimo a distância total de *R* desta vez é c_6, substituindo c_1 como extremo. $R = (c_6\text{-}c_1\text{-}c_3\text{-}c_7\text{-}c_2\text{-}c_8)$;

Passo 6: Neste passo, c_5 substitui c_6 como extremidade da rota, por ser esta, a cidade a cidade mais próxima de uma das extremidades. $R = (c_5\text{-}c_6\text{-}c_1\text{-}c_3\text{-}c_7\text{-}c_2\text{-}c_8)$;

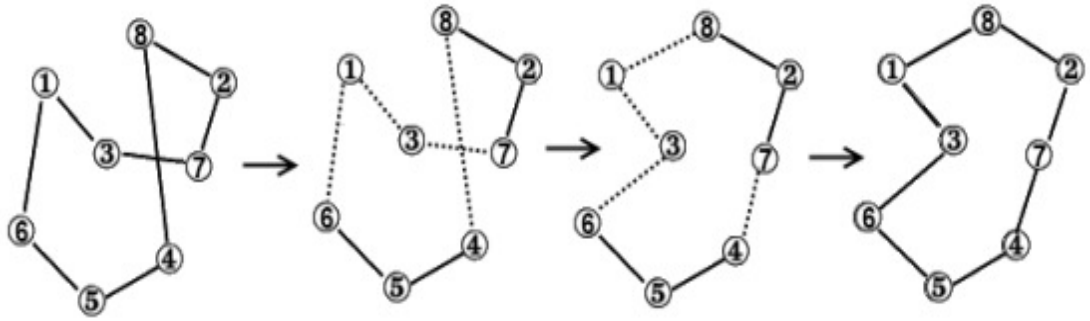
Passo 7: Por fim, c_4, a última cidade ainda não visitada, é adicionada a R e completa-se a rota conectando c_4 a c_8. $R = (c_8\text{-}c_4\text{-}c_5\text{-}c_6\text{-}c_1\text{-}c_3\text{-}c_7\text{-}c_2\text{-}c_8)$.

Figura 3.12 – Rota proposta pela HVP.



Após a fase de construção foi aplicada a fase de melhoria de roteiros por meio da heurística *4-Opt*. Assim como em 3-*Opt*, o número de reconexões a serem testadas entre os 4 pares de cidades é igual a 24 combinações de conexões entre os respectivos pares de cidades. A Figura 3.13 mostra uma reconexão que produz uma rota R= (c_1-c_3-c_6-c_5-c_4-c_7-c_2-c_8-c_1) de custo 31, inferior ao custo 37 da rota inicial.

Figura 3.13 – Aplicação de *4-Opt*.



3.5.2 Heurística Híbrida do Vizinho mais Próximo com 2 Caminhos Paralelos

A fase construtiva desta nova Heurística Híbrida, desenvolvida por nos e denominada de Heurística do Vizinho mais Próximo com 2 Caminhos Paralelos (*HVP2*), é realizada por uma variação da HHVP, dada na seção anterior, cujo processo de construção de rotas é feito a partir da conexão de 2 caminhos configurados paralelamente. O processo de seleção de cidades a serem inseridas em um dos dois caminhos em formação utiliza o mesmo critério de avaliação da HHVP, onde é selecionada a cidade mais próxima de um dos extremos dos dois caminhos paralelos. O pseudocódigo da HVP2 proposta é apresentado na Figura 3.14.

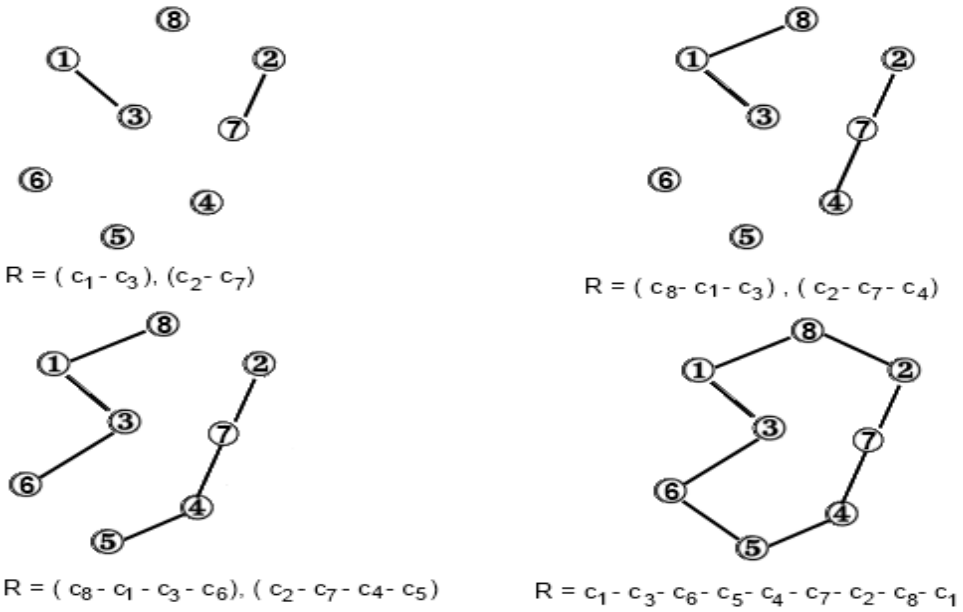
Figura 3.14 – Pseudocódigo da HVP2 proposta.

```
HVP2( )
    1. n ← Número de cidades do itinerário;
    2. C1 = null;      // Caminho 1 em formação;
    3. eX1 = null;     // Cidade localizada na extremidade X de C1
    4. eY1 = null;     // Cidade localizada na extremidade Y de C1
    5. vX1 = null;     // vizinho de eX1
    6. vY1 = null;     // vizinho de eY1
```

```
7.  C2 = null;     // Caminho 2 em formação;
8.  eX2 = null;    // Cidade localizada na extremidade X de C2
9.  eY2 = null;    // Cidade localizada na extremidade Y de C2
10. vX2 = null;    // vizinho de eX2
11. vY2 = null;    // vizinho de eY2
12. count = null;     // Contador de cidades visitadas
13. for i=1, (n-1) do
14.   for j=(i+1), n do
15.         eX1 ← i;    //cidade inicial de C1;
16.         eY1 ← Selecionar cidade mais próxima de eX1;
17.         Adicionar eX1 e eY1 a C1;
18.         eX2 ← j;   // cidade inicial de C2;
19.         eY2 ← Selecionar cidade mais próxima de eX2;
20.         Adicionar eX2 e eY2 a C2;
21.         count := 4;
22.         while count < n do
23.               vX1 ← Selecionar cidade ainda não visitada mais próxima de eX1;
24.               vY1 ← Selecionar cidade ainda não visitada mais próxima de eY1;
25.               if (distância de vX1 para eX1) < (distância de vY1 para eY1) then;
26.                     Adiciona vX1 a C1;
27.                     eX1 := vX1;
28.               else
29.                     Adiciona vY1 a C1;
30.                     eY1 := vY1;
31.               end – if
32.               count ++;
33.               vX2 ← Selecionar cidade ainda não visitada mais próxima de eX2;
34.               vY2 ← Selecionar cidade ainda não visitada mais próxima de eY2;
35.               if (distância de vX2 para eX2) < (distância de vY2 para eY2) then;
36.                     Adiciona vX2 a C2;
37.                     eX2 := vX2;
38.               else
39.                     Adiciona vY2 a C2;
40.                     eY2 := vY2;
41.               end – if
42.               count ++;
43.         end – while
44.         Conectar C1 a C2 e formando uma Rota;
45.         Comparar Rota recém-formada com a Rota formada na iteração anterior;
46.         Manter melhor Rota;
47. end - for
```

Aplicando HVP2 no problema da Tabela 3.2, e considerando c_1 e c_2 como as cidades iniciais dos caminhos 1 (C_1) e 2 (C_2), obtêm-se, como pode ser visto na Figura 3.15, dada a seguir, uma rota sendo (c_1-c_3-c_6-c_5-c_4-c_7-c_2-c_8-c_1) com custo igual a 31. Abaixo, são descritos os passos operados na composição desta rota *R*:

Figura 3.15 – Rota proposta pela HVP2.



Passo 1: Adiciona-se em C_1, a cidade c_3, como elemento seguinte a c_1, já que c_3 é a cidade com menor custo na vizinhança. Em C_2, adiciona-se c_7 seguindo o mesmo procedimento de C_1. $R = (c_1\text{-}c_3)\ (c_2\text{-}c_7)$;

Passo 2: Adiciona-se c_8 a C_1, substituindo-se c_1 como um dos extremos. Em C_2, adiciona-se c_4, substituindo-se c_7 como um dos extremos. $R = (c_8\text{-}c_1\text{-}c_3)\ (c_2\text{-}c_7\text{-}c_4)$;

Passo 3: Adiciona-se c_6 a C_1, substituindo-se c_3 como um dos extremos. Em C_2, adiciona-se c_5, substituindo-se c_4 como um dos extremos. $R = (c_8\text{-}c_1\text{-}c_3\text{-}c_6)\ (c_2\text{-}c_7\text{-}c_4\text{-}c_5)$;

Passo 4: Conectam-se as extremidades de C1 e C2 avaliando a opção que proporcione o menor acréscimo de distância total. Assim $R = (c_1\text{-}c_3\text{-}c_6\text{-}c_5\text{-}c_4\text{-}c_7\text{-}c_2\text{-}c_8\text{-}c_1)$.

A fase de melhoria de roteiros por meio do algoritmo *4-Opt* também se repetiu aqui para a HVP2. Implementou-se uma modificação no critério de seleção para uma cidade ser inserida em um dos dois caminhos paralelos, dando origem a uma nova metodologia de tentar solucionar o problema específico, denominada de Heurística do Vizinho mais Próximo com 2 caminhos paralelos e a inserção de uma cidade por iteração (*HVP2B*). Este critério insere uma cidade por iteração, sendo selecionada aquela cidade que estiver mais próxima de um dos dois caminhos paralelos. Antes eram inseridas duas cidades em

cada iteração, uma para cada caminho paralelo. Neste novo procedimento isto não mais ocorre.

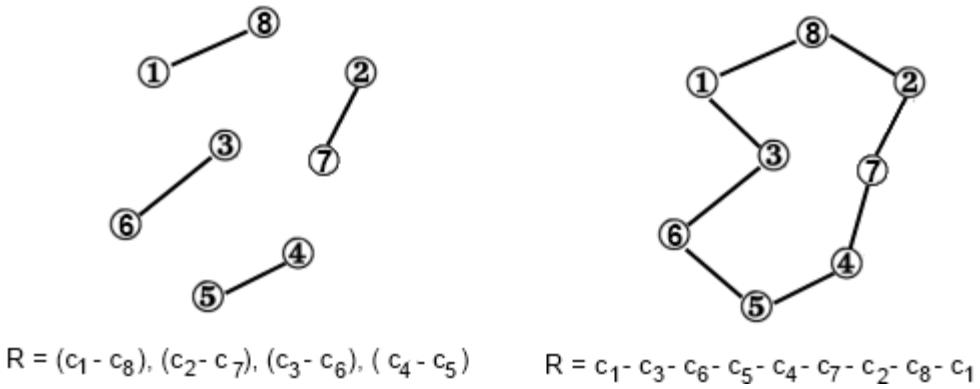
3.5.3 Heurística Híbrida do Vizinho mais Próximo com 4 Caminhos Paralelos

A construção desta nova heurística, também desenvolvida por nos, levou em consideração a metodologia empregada para HVP2. A diferença deste método para HVP2 é que aqui são utilizados 4 caminhos paralelos. Denominada de HVP4, essa heurística construtiva possui complexidade $O(n^6)$. Aplicando-a aos dados da Tabela 3.2 e considerando como cidades iniciaisl c_1, c_2, c_3 e c_4 dos caminhos 1 (C_1), 2 (C_2), 3 (C_3), e 4 (C_4), respectivamente, tem-se então como ordem encontrada (c_1-c_3-c_6-c_5-c_4-c_7-c_2-c_8-c_1). Este procedimento irá formar mais de uma rota viável para o problema, devido as várias formas de unir estes caminhos na geração de uma rota. Os passos do processo de formação da rota são descritos a seguir e a Figura 3.16 ilustra o processo de formação desta:

Passo 1: Adiciona-se em C_1, a cidade c_3, em C_2, adiciona-se c_7, em C_3, adiciona-se c_6 e em C_4, adiciona-se c_5, por estas serem as cidades com menor custo na vizinhança das cidades iniciais de cada caminho. R = (c_1-c_3), (c_2-c_7), (c_3-c_6), (c_4-c_5);

Passo 2: Uma vez que as 8 cidades já foram incluídas em um dos caminhos, conectam-se as extremidades de C_1, C_2, C_3 e C_4 considerando a opção que proporcione o menor custo de distância total produzida. Assim R = (c_1-c_3-c_6-c_5-c_4-c_7-c_2-c_8-c_1).

Figura 3.16 – Rota proposto pela HVP4.



Da mesma forma, como nos métodos heurísticos apresentados nas duas seções anteriores, é aplicada a fase de melhoria de roteiros por meio da heurística *4-Opt* afim de melhorar a solução obtida na fase construtiva.

3.5.4 Heurística Híbrida da Inserção Mais Barata Modificada

A estratégia proposta na Heurística Híbrida da Inserção Mais Barata Modificada (HIMBM) é aplicar o método *4-Opt* no processo de construção de rotas da Heurística da Inserção Mais Barata (HIMB). Desta forma, quando uma sub-rota (ciclo com menos de *n* cidades) é formada pelo princípio da HIMB com 8 ou mais cidades, aplica-se o método 4-*Opt* para cada nova cidade inserida nesta sub-rota. Este processo termina quando tiver sido inserido na sub-rota todas as n cidades. Em HIMBM, o processo de escolha da cidade inicial é selecionado para cada uma das *n* cidades, gerando *n* soluções distintas e tomando-se a melhor delas como a solução do problema. Uma vez que a HIMB e *4-Opt* possuem complexidade $O(n^2)$ e $O(n^4)$, respectivamente, a complexidade de HIMBM passa a ser limitada superiormente por $O(n^6)$. A Figura 3.17 apresenta o pseudocódigo da HIMBM.

Figura 3.17 – Pseudocódigo da HIMBM.

```
HIMBM( )
n ← Número de cidades do itinerário;
D ← Matriz de distâncias das n cidades;
c0 = null;     // Cidade inicial
ck = null;     // Cidade a ser inserida na sub-rota
count = null; // Contador de cidades incluídas na sub-rota
x = 0;        // Posição m que será inserido ck na sub-rota
for y =1, n do
   c0 ← y ;        // y é a cidade inicial
   ck ← Selecionar cidade mais próxima de c0;
   Adicionar c0 e ck a sub-rota;
```

```
   count = 2;
   for k = 1, n do    // Avalia cada uma das cidades que ainda não estão na sub-rota
      if (Cidade k não tiver na sub-rota) then
            (Dik + Dkj – Dij))= Min { Dwk + Dkv – Dwv; para todos as arestas (w,v) da sub-rota};
            x ← Retorna posição de inserção de k na sub-rota entre i  e j;
            Adicionar cidade k a sub-rota na posição x;
               count ++;
            if (count >= 8) then  OPT4(sub-rota) end if
      end if
   end for
   Comparar a rota atual com as rotas anteriores e guardar a melhor delas;
end for
```

Como pode ser visto no pseudocódigo apresentado acima, a cidade inicial é escolhida com base na variável *y*, em seguida procura-se pelo vizinho mais próximo da cidade inicial. Depois, as cidades restantes são avaliadas e inseridas com base na iteração da estrutura de repetição implementada na linha 12. Desta forma, ao aplicar HIMBM nos dados da Tabela 3.2 e considerando como cidade inicial c_1, obtêm-se então a rota (c_1-c_8-c_2-c_7-c_4-c_5-c_6-c_3-c_1) cujo custo total é 31. A Figura 3.18 ilustra o processo de formação desta rota e os passos deste processo são descritos a seguir:

Passo 1: Adiciona-se c_3, já que é a cidade com menor custo na vizinhança de c_1. R = (c_1-c_3-c_1);

Passo 2: Adiciona-se c_2 entre c_1 e c_3, já que é esta a primeira cidade ainda não visitada disposta na sequência. R = (c_1-c_2-c_3-c_1);

Passo 3: Seguindo a sequência, adiciona-se c_4 entre c_2 e c_3, já que é esta a posição em que a inserção de c_4 proporciona o menor acréscimo de distância a rota corrente. R = (c_1-c_2-c_4-c_3-c_1);

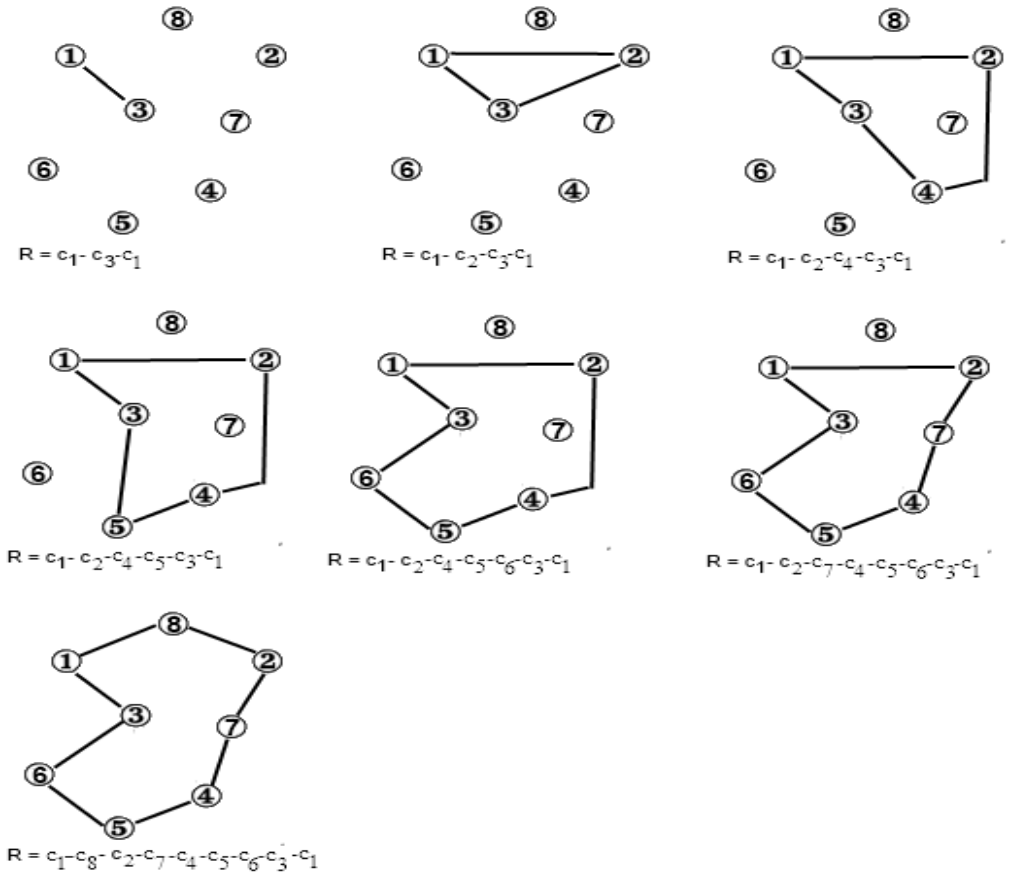
Passo 4: Seguindo a sequência, adiciona-se c_5 entre c_4 e c_3, já que é esta a posição em que a inserção de c_5 proporciona o menor acréscimo de distância a rota corrente. R = (c_1-c_2-c_4-c_5-c_3-c_1);

Passo 5: A próxima cidade da sequência é c_6 e a posição em que a inserção de c_6 proporciona o menor acréscimo de distância a rota corrente é entre c_5 e c_3. $R = (c_1\text{-}c_2\text{-}c_4\text{-}c_5\text{-}c_6\text{-}c_3\text{-}c_1)$;

Passo 6: A penúltima cidade da sequência é c_7 e a posição em que a inserção de c_7 proporciona o menor acréscimo de distância a rota corrente é entre c_2 e c_4. $R = (c_1\text{-}c_2\text{-}c_7\text{-}c_4\text{-}c_5\text{-}c_6\text{-}c_3\text{-}c_1)$;

Passo 7: A última cidade da sequência é c_8 e a posição em que a inserção de c_8 acrescenta menos distância total da rota corrente é entre c_1 e c_2. $R = (c_1\text{-}c_8\text{-}c_2\text{-}c_7\text{-}c_4\text{-}c_5\text{-}c_6\text{-}c_3\text{-}c_1)$.

Figura 3.18 – Rota proposta pela HIMBM.



Após a realização de vários testes, com as instâncias da literatura, adotou-se um novo critério aplicado nas heurísticas *HVP4* e *4-Opt*, denominado de *Salto*, com o intuito de evitar um gasto excessivo de tempo sem melhoria na busca pela solução ótima. Este

critério diminui o número de avaliações com base nos valores das variáveis *i*, *j*, *k* e *l* dentro dos comandos **for** nos algoritmos destas duas heurísticas.

Assim, estes dois algoritmos ganharam uma versão diferente da anterior com cinco alterações: Uma linha contendo a variável inteira *salto*=*n*/50, no início de cada algoritmo; e no final de cada **for** *i*, *j*, *k* e *l* as atribuições *i*=*i*+*salto, j*=*j*+*salto*, *k*=*k*+*salto* e *l*=*l*+*salto*, respectivamente. Logo, quando executadas *HVP4* e *4-Opt*, com este procedimento, haverá uma grande diminuição de tempo para valores de *n* entre médio e grande porte, basta ver que *i*, *j*, *k* e *l* não vão mais variar de uma unidade e sim de *salto*+1 valores. Exemplificando, se *n*=200 o salto vai ser de 5 (200/50 + 1) unidades para os valores de *i*, *j*, *k* e *l*, ou seja, estas quatro variáveis vão saltar 5 unidades tendo a seguinte variação *i*=1, 6, 11, 16, … , *j*=2, 7, 12, …, etc.

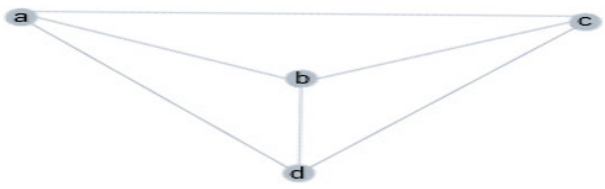
CAPÍTULO 4 – AMBIENTE DE RESOLUÇÃO GEORREFERENCIADO

Este capítulo contempla a fundamentação teórica utilizada como base para o desenvolvimento do Ambiente de Resolução Georreferenciado para o PCV (ARG-PCV) proposto bem como a descrição das características e funcionalidades deste Ambiente. Na primeira seção, é feita uma introdução teórica sobre a Teoria dos Grafos, visto que a modelagem baseia-se numa estrutura de Grafos, que possibilita sua associação com o mundo real. Na segunda seção, os conceitos de Geodesia e Sistemas de Informação Georreferenciados são abordados. A terceira seção contém uma descrição sobre o módulo espacial do Sistema Gerencial de Banco de Dados escolhido para suportar o SIG proposto. Por fim, na quarta seção, o ARG-PCV é apresentado, sendo ele a principal contribuição deste trabalho.

4.1 Teoria dos Grafos

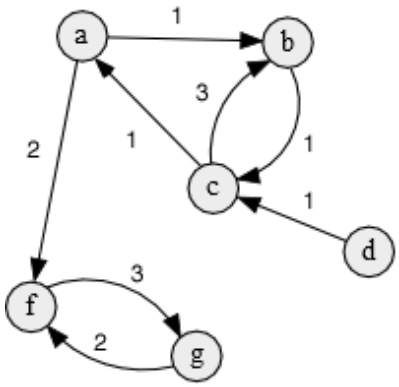
A Teoria dos Grafos é um ramo da matemática que remete ao estudo de objetos combinatórios denominados Grafos. Grafos são representados por pontos dispostos em posições arbitrárias denominados de nós, ou vértices, conectados por curvas chamadas de arestas. Deste modo, um Grafo com n nós e m arestas pode ser representado por $G = (V, E)$, onde $V=\{1, 2,, 3, \ldots, n\}$, $E=\{e_1, e_2, \ldots, e_m\}$ e cada $e_k=(i,j)$, com $1\leq k\leq m$ e $1\leq i, j\leq n$. Se um grafo G admite que seu conjunto V de vértices seja particionado em dois subconjuntos U e W tais que cada aresta de G possua um extremo em U e outro em W, então G é dito bipartido. Se G poder ser representado no plano de tal forma que suas arestas não se cruzem, então este G é classificado como *planar* (Figura 4.1).

Figura 4.1 – Exemplo de grafo planar.



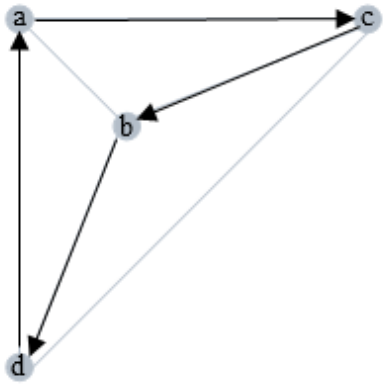
Quando as arestas de um grafo têm direção associada ou sentido, elas são chamadas de arcos e o respectivo grafo é chamado de grafo *orientado* ou *dígrafo*. Quando há pesos associados às arestas ou aos vértices do grafo, classifica-se este grafo como *valorado* ou *ponderado*. A Figura 4.2 apresenta um grafo *orientado* e *valorado*. Quaisquer nós *v* e *w*, são ditos adjacentes se uma aresta *e* liga-os diretamente, neste caso, a aresta *e* é classificada como incidente dos vértices *v* e *w*. O número de vértices adjacentes a um vértice é chamado de *grau do vértice*, dessa forma, o grau de um vértice *v* em um grafo *G* é o número de arestas incidindo em *v* e se todos os vértices possuírem o mesmo grau, o grafo é classificado como *regular*.

Figura 4.2 – Grafo *orientado* e *valorado*.



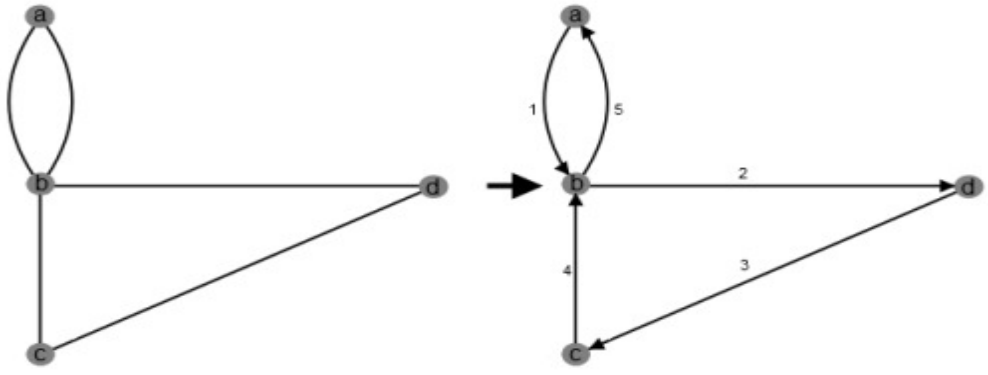
Uma sequência alternada de vértices e arestas que começa no vértice v_i e termina no vértice v_j é denominada *passeio* de v_i até v_j. Se todos os vértices que compõe um passeio, forem distintos, denomina-se esse *passeio* como *caminho*. Um *caminho* é classificado como sendo um *ciclo* se o primeiro e o último vértice da sequência são iguais. De forma efémera, um *ciclo* pode ser expressado como um *caminho* v_i,..., v_j, sendo $v_i = v_j$. Grafos que não possuem *ciclos* são ditos *acíclicos*. Se para quaisquer que sejam os vértices v_i e v_j distintos em um grafo, existe sempre um passeio que os une, então este grafo é classificado como *conexo*. Um *caminho* que permite passar por cada vértice do grafo uma única vez é denominado *caminho hamiltoniano*, assim, um *ciclo*, cuja sequência de vértices v_i,..., v_j seja um *caminho hamiltoniano*, será chamado de *ciclo hamiltoniano*. A Figura 4.3 ilustra com destaque um *ciclo hamiltoniano* formado no grafo apresentado na Figura 4.1.

Figura 4.3 – Ilustração de um c*iclo hamiltoniano* em um grafo.



Um *passeio* em um grafo que admite repetição de vértices, mas possua todas as arestas distintas, é denominado *trajeto* ou *trilha*. Qualquer *trajeto* ou *ciclo*, que contenha cada aresta do grafo exatamente uma vez, é denominado *trajeto euleriano*. Informalmente, pode-se dizer que um grafo *G* é *Euleriano* se for possível desenhar uma representação gráfica desse grafo *G* sem tirar o lápis do papel e voltar ao ponto de partida, sem passar mais de uma vez por nenhuma aresta (LUCCHESI, 1979). Se um grafo *G* possuir um *ciclo hamiltoniano* ou *euleriano*, então, *G* será classificado respectivamente como um grafo *Hamiltoniano* ou *Euleriano*. Na Figura 4.4 é apresentada a ordem da composição de um *ciclo euleriano* em um grafo conexo.

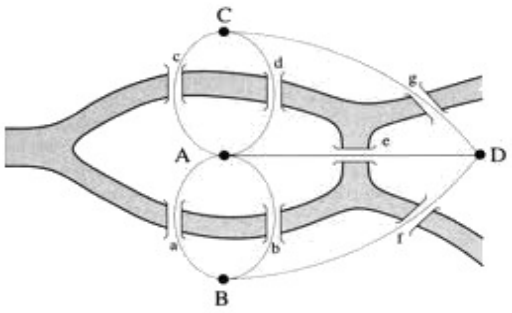
Figura 4.4 – Ilustração de um c*iclo euleriano* em um grafo.



O surgimento da Teoria dos Grafos aconteceu no ano de 1736, quando o matemático Leonhard Euler estudou o problema das pontes de Konigsberg, uma cidade da antiga

Prússia do século XVIII, atual Kaliningrado (Rússia), onde há duas ilhas que, juntas com a parte continental, eram ligadas por 7 pontes. O objetivo do problema era encontrar uma maneira de se atravessar todas as sete pontes sem repeti-las. Euler abordou o problema, transformando os caminhos das pontes em retas e suas interseções em pontos, o que para muitos (Figura 4.5), resultou no primeiro grafo da história. Baseado no fato de que em cada ponto deveria haver um número par de caminhos, os quais representariam a chegada e a saída, Euler concluiu que só seria possível atravessar o caminho inteiro passando uma única vez em cada ponte se houvesse no máximo dois pontos de onde sairia um número ímpar de caminhos. Sendo assim, os dois pontos com caminhos ímpares, seriam referentes ao incio e fim do percurso, pois estes não necessitavam de um caminho de chegada ou saída.

Figura 4.5 – Esquema proposto por Euler para tratar o problema das pontes de Konigsberg.



Boaventura Netto (2003) mencionou que o interesse na utilização da teoria dos grafos, como instrumento de modelagem para diversos problemas, se deu a partir de 1847 através das pesquisas de Kirchhoff, cidadão de Konigsberg, sobre circuitos elétricos utilizando modelos de Árvore, um tipo de grafo *bipartido*, *conexo* e *planar*. Em 1852, Morgan relacionou a teoria dos grafos com a formulação do Problema das 4 Cores, cuja premissa é definir a quantidade mínima de cores para colorir um mapa de tal forma que países de fronteira comum possuam cores diferentes. Em 1859, Hamilton estudou os problemas dos caminhos mínimos e, em 1869, Jordan buscou estabelecer certo formalismo matemático à Teoria das Árvores de Kirchhoff. Com a aplicabilidade da teoria dos grafos em problemas de otimização e o interesse crescente que se deu pela

Pesquisa Operacional nos anos pós 2ª guerra mundial, muitos estudos foram feitos acerca de algoritmos em grafos.

Dois algoritmos foram considerados fundamentais durante muito tempo em grafos são eles: o *Depth First Search* (DFS) e *Breadth First Search* (BFS). Ambos serviram de base para o desenvolvimento de novos algoritmos em grafos encontrados na literatura (CORMEN et al., 2001). O algoritmo DFS, também denominado como método de Busca em Profundidade, tem por objetivo explorar os nós de um grafo. Para isso, este algoritmo começa num nó raiz (selecionando algum nó como sendo a raiz, no caso de um grafo) e explora cada uma de suas conexões (arestas) tanto quanto possível, antes de retroceder (*backtracking*). No algoritmo BFS, também denominado como método de Busca em Largura, a busca inicia em um vértice raiz e explora todos os vértices vizinhos. Para cada um desses vértices mais próximos, são explorados os seus vértices vizinhos inexplorados e assim por diante, até que se encontre o alvo da busca. O funcionamento deste algoritmo produz uma *árvore em extensão* onde o nível em que se encontra um vértice representa exatamente a sua distância (ou número de arestas) até o vertice inicial (CORMEN et al., 2001).

Na literatura há vários algoritmos eficientes na Teoria dos Grafos baseados nos métodos DFS e BFS: Algoritmo de Dijkstra, Algoritmo de Bellman-Ford, Algoritmo de Prim e o Algoritmo Ford-Fullkerson. Boaventura Netto (2003) citou que os primeiros trabalhos brasileiros sobre a Teoria dos Grafos no Brasil foram apresentados no I Simpósio Brasileiro de Pesquisa Operacional, em 1968. No âmbito do Geoprocessamento, mais precisamente, no desenvolvimento de Sistemas de Informações Georreferenciadas, a Teoria dos Grafos é usada para a representação da malha viária de uma região por meio de um grafo, viabilizando assim, a inserção de técnicas computacionais nessas estruturas com o objetivo de unir a teoria à prática, e vice-versa.

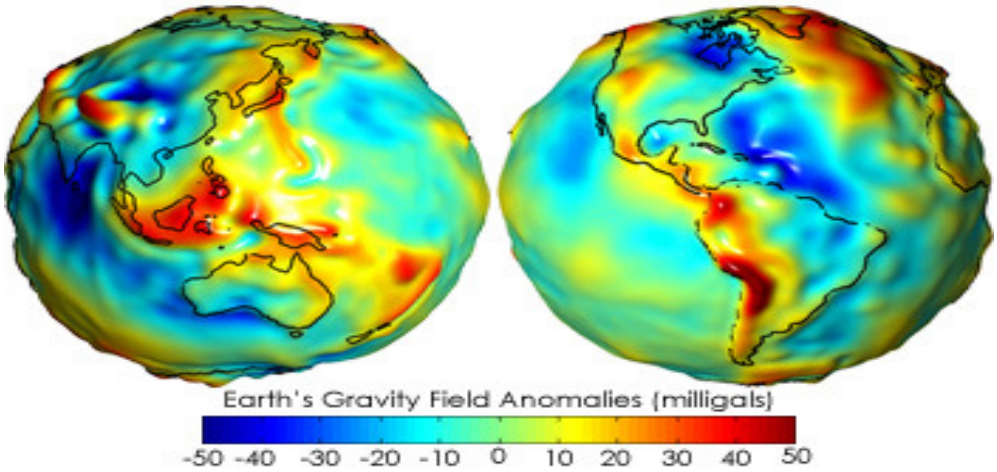
4.2 Sistemas de Informações Georreferenciadas

Um Sistema de Informações Georreferenciadas (SIG) de modo mais amplo, é uma ferramenta que permite manipular dados georreferenciados e alfanuméricos para, a partir de análises espaciais, apoiar a tomada de decisão espacial, como a definição do

melhor roteiro de entregas a ser seguido, por exemplo (FARKUH NETO & LIMA, 2005). Georreferenciar significa atribuir coordenadas geográficas a um evento geográfico ou entidade de forma a posicioná-lo sobre a superfície terrestre com base em um sistema de coordenadas geográficas. A partir do georreferenciamento, torna-se viável a integração dos diferentes objetos geográficos de um SIG, porém, antes de georreferenciar algo, é preciso definir um modelo representativo da superfície física do planeta Terra. Surge então o conceito de Geodésia, que segundo Gemael (1994), é a ciência que tem por objetivo determinar a forma e as dimensões da Terra e os parâmetros definidores do campo gravífico.

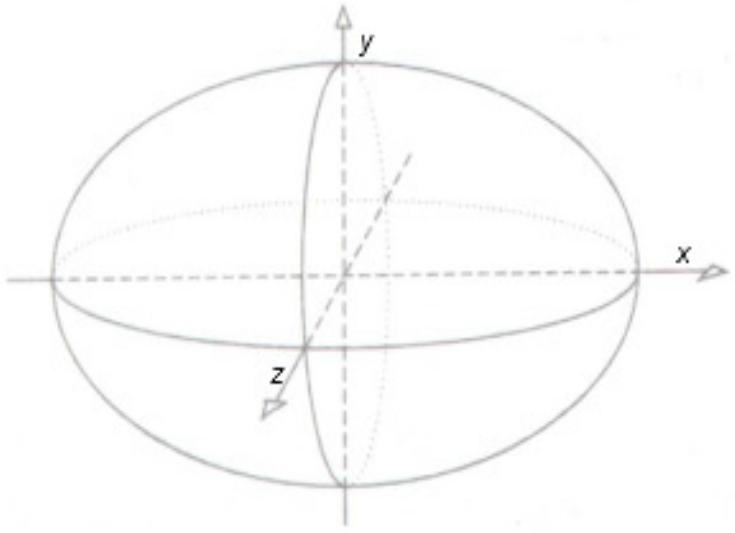
Existem diferentes tipos de figuras usados para representar fisicamente e matematicamente a superfície terrestre. Em Geodésia, a *geoide* é a figura que melhor representa fisicamente o planeta Terra e pode ser entendido como uma superfície equipotencial do campo de gravidade do planeta Terra que coincide com o nível médio não perturbado dos mares (IBGE, 2012). A superfície *geoidal* pode ser abstraída como uma superfície prolongada através dos continentes, com um formato ondulatório levemente irregular que acompanha as variações da estrutura de distribuição de massa do planeta Terra. O termo *geoide* foi concebido em 1873, por J. F. Listing, e até hoje, é tido como a verdadeira figura representativa do planeta Terra ou qualquer outro corpo planetário, onde o seu equacionamento matemático é complexo, dificultando às vezes seu uso prático como referência geométrica. A Figura 4.6 representa o geoide do planeta Terra segundo a NASA (*National Aeronautics and Space Administration*).

Figura 4.6 – Representação da Terra por um *geoide* segundo a NASA.



Do ponto de vista geométrico, o *elipsoide* se destaca como a melhor figura para representar o planeta Terra. Como pode ser visto na Figura 4.7, o *elipsoide* é um sólido que resulta da rotação de uma *elipse* em torno de um dos seus eixos. Em casos efêmeros, devido à facilidade de representação, utiliza-se da *esfera* como uma aproximação do *elipsoide*. A relativa simplicidade que possuí o modelo *elipsoidal* em relação ao *geoidal*, torna-o a superfície de referência utilizada em todos os cálculos básicos que assistem a elaboração de uma representação cartográfica.

Figura 4.7 – Ilustração de um *elipsoide* e seus respectivos três eixos.



Segundo Rosa (2004), medições e levantamentos feitos na superfície terrestre (*geoide*) são matematicamente solucionados no *elipsoide*. Os sistemas *geodésicos* buscam uma melhor correlação entre o *geoide* e o *elipsoide*, elegendo um *elipsoide de revolução* como aquele que melhor se ajusta ao *geoide local* e estabelece uma origem para as coordenadas *geodésicas* referenciadas neste *elipsoide*, através de um *datum planimétrico* e um *datum altimetrico*. *Datum* (plural *data*) é um modelo matemático teórico da superfície terrestre. Para que a superfície de uma região possa ser modelada em um *datum*, é necessário definir um *elipsoide* e posicioná-lo em relação ao planeta Terra, considerando a preservação do paralelismo entre o eixo de rotação da Terra e o do *elipsoide*. Definida estas e outras restrições matemáticas mais complexas, é estabelecido o *datum planimétrico*. O *datum altimetrico* é concebido para se trabalhar

com dados que contenham informações de elevação, utilizando parra isso, o nível médio dos mares.

Os sistemas geodésicos de cada país são baseados em um *datum planimétrico* e um *datum altimétrico* que melhor representam sua superfície terrestre. Em consequência disso, existem inúmeros *data* para diferentes regiões do globo terrestre. Por exemplo, se utilizar do *datum* planimétrico adotado no Brasil para representar o território da Rússia implicaria em muitas distorções deste espaço geométrico. Outro fator que contribui para diversidade de *data* geodésicos é o desenvolvimento de técnicas e instrumentos *geodésicos*. Com isso, a medida que a Geodesia evolui, surgem versões de *data* oficiais melhores acurados para uma determinada região. O Brasil utilizou até o final da década de 1970 como *datum planimétrico* oficial o Córrego Alegre, e posteriormente, adotou o *datum* SAD 69 (Sistema Geodésico Sul-Americano de 1969) como oficial. Atualmente, o sistema *geodésico* brasileiro está em fase de transição do SAD 69 para o SIRGAS (Sistema de Referência Geocêntrico para as Américas), definido como novo *datum* planimétrico oficial. Outro *datum* bastante utilizado não só no Brasil, mas no mundo, é o WGS 84 (*World Geodetic System*). Estabelecido pelo Departamento de Defesa Americano, desde 1960, com o objetivo de fornecer posicionamento e navegação em qualquer parte do globo, este *datum* é utilizado em trabalhos que necessitam de sistema geodésico a nível global e como sistema geodésico padrão de GPS (*Global Positioning Systems*). O *datum altimétrico* oficial adotado no Brasil é o *Imbituba Santa Catarina*.

A definição do sistema de paralelos e meridianos sobre a superfície elipsoidal do *datum* é baseada em um sistema de coordenadas *geodésicas*, onde, dado um ponto sobre a superfície do *elipsoide* de referência de um certo *datum planimétrico*, a latitude *geodésica* é o ângulo entre a normal ao *elipsoide*, no ponto, e o plano do equador. A longitude *geodésica* é o ângulo entre o meridiano que passa no ponto e o meridiano origem (*Greenwich*, por convenção). Os sistemas de coordenadas planas (ou Projeções Cartográficas) concebidos para viabilizar a transferência de uma imagem da superfície curva do planeta para um plano da carta, são desenvolvidos por meio de um mapeamento, normalmente guiado por imposições de equivalência, de pontos sobre a superfície curva do elipsoide de referência para pontos sobre uma superfície plana. Este mapeamento traduz-se por uma relação entre coordenadas planas e coordenadas

geodésicas. Uma vez que todo SIG precisa especificar um sistema de coordenadas planas, torna-se evidente a definição correta de um *datum planimétrico* na exatidão *geodésica* do conteúdo da base de dados do SIG (D'ALGE, 1999).

O conceito de SIG foi formulado a cerca de 40 anos por um grupo de geógrafos e pesquisadores de outras áreas do conhecimento científico. Na Inglaterra e nos Estados Unidos, no final da década de 1950 e durante a década de 1960, visando a produção de um inventário de recursos naturais e a redução de custos envolvidos na produção e manutenção de planos cartográficos, se iniciou o desenvolvimento de tecnologias para automatizar parte do processamento de dados com componentes espaciais. Todavia, a interface destas tecnologias eram complexas e a capacidade de armazenamento e a velocidade de processamento eram baixas. Na década de 1970, com o desenvolvimento de tecnologias mais acessíveis no segmento, surgiu a expressão *Geographic Information System* (GIS). Na década de 1980, com o advento da computação, o desenvolvimento de tecnologias relacionadas ao SIG se acelerou. A década de 1990 marca a saída definitiva do meio acadêmico para o mercado por parte do geoprocessamento. Desde então, instituições governamentais e empresas passaram a investir no uso de tecnologias de coleta da informação espacial ou as denominadas geotecnologias, resultando no aparecimento de aplicativos mais simples, com funcionalidades básicas de consulta a mapas e as bases alfanuméricas. Esta tendência deu origem a ferramentas como *Wikimapia, Google Maps* e *Earth*, dentre outras.

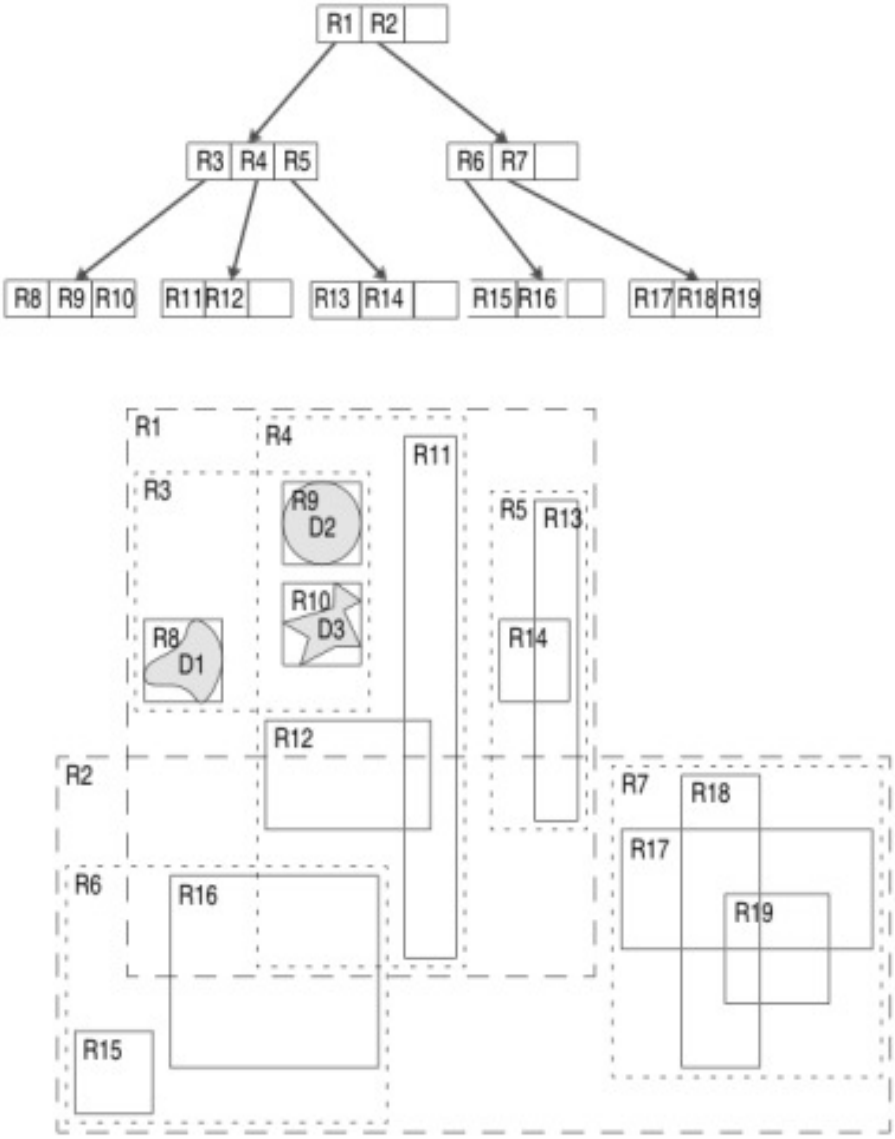
Como exemplos de geotecnologias, cujo desenvolvimento foi preponderante para a evolução do SIG, pode-se citar: o *Sensoriamento Remoto*, que trata das imagens de alta resolução geradas por satélites e a confecção de ortoimagens; o *GPS - Sistemas de Posicionamento Global* que permitem determinar o correto posicionamento de objetos sobre a superfície terrestre; a *Aerofotogrametria* que permite a transformação de fotografias aéreas verticais em mapas digitais; e os *Sistemas Gerenciais de Banco de Dados Espaciais* que permitem armazenar dados sobre informações espaciais. Com o advento destas novas tecnologias convencionou-se o termo SIG passou a ser denominado de *Sistemas de Informações Georreferenciadas* tendo em vista que nos bancos de dados espaciais todas as informações contextuais, imagens, ortoimagens e aerofotogrametrias são atributos georreferenciados.

4.3 Sistema Gerencial de Banco de Dados Espacial

Um Sistema de Gerenciamento de Banco de Dados (SGBD) pode ser entendido como um conjunto de programas de computador (*softwares*) responsáveis pelo gerenciamento de repositórios de dados. Seu principal objetivo é abstrair das aplicações, a responsabilidade de gerenciar o acesso, a manipulação e a organização dos dados. Enquanto um SGBD trabalha com tipos de dados, numéricos e caracteres, uma extensão Espacial geralmente confere ao SGBD índices e tipos de variáveis espaciais, funções para realização de operações geométricas e geográficas e bibliotecas de projeções cartográficas associadas aos modelos matemáticos *geodésicos* (*datum*). Um Sistema Gerencial de Banco de Dados Espacial (*SGBDE*) serve não só como repositório de dados espaciais para o SIG, mas também viabiliza a manipulação e análise destes dados. *Microsoft SQL Server* e *Oracle Spatial* representam soluções proprietárias para servir como repositório de dados espaciais, porém, a utilização de arquiteturas proprietárias promove a dependência de uma única empresa que acaba por impor valores elevados para manter e atualizar esta estrutura, e as restrições por elas impostas acabam por inviabilizar uma possível ampliação da estrutura para atender as necessidades que podem surgir no decorrer do desenvolvimento de um sistema.

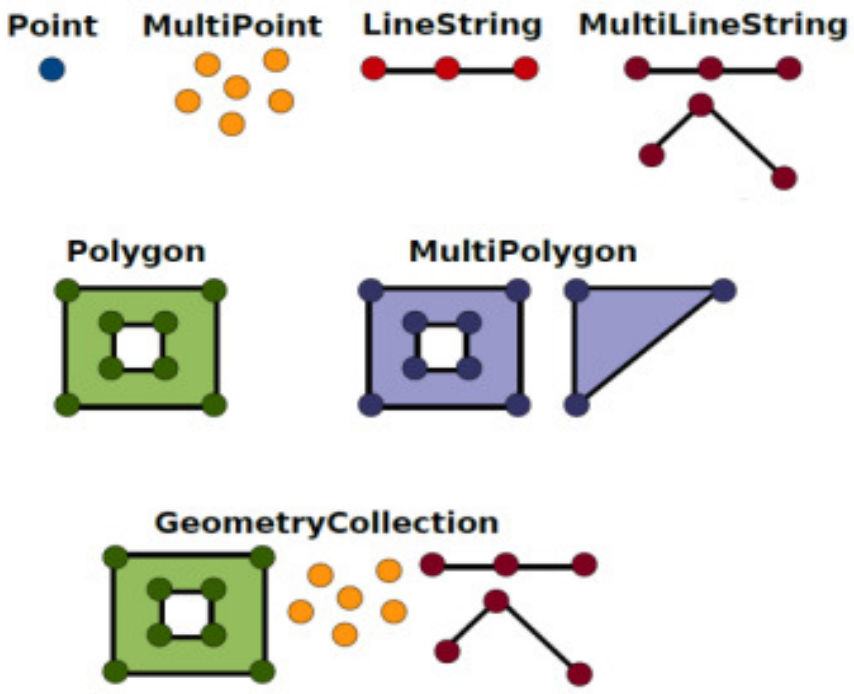
Consoante a estas desvantagens, empresas de Tecnologia da Informação resolveram investir em um modelo de negócios baseado em *Software* Livre (*SL*). Surgiram o banco de dados PostgreSQL e o *PostGIS*, o módulo espacial do SGBD *PostgreSQL*. Desenvolvida pela empresa canadense *Refractions*, este módulo apresenta um número expressivo de funções espaciais/topológicas utilizadas para subsidiarem operações espaciais. O algoritmo de indexação utilizado pelo *PostGIS* é o GiST (*Generalised Search Tree*), uma variação das *Árvores R*. Entende-se por Árvore R como sendo uma estrutura de dados utilizada em métodos de acesso espacial visando indexar informações multidimensionais, como por exemplo, as coordenadas longitudinais e latitudinais (X, Y) de pontos situados em um mapa digital. Como pode ser observado na Figura 4.8, a Árvore R divide o espaço por meio de sobreposição de retângulos de limites mínimos (caixas limitantes - *bounding boxes*) e aninhamento hierárquico.

Figura 4.8 – Objetos espaciais em Árvore R.



Visando o intercâmbio entre as diversas geotecnologias, os responsáveis pelo desenvolvimento do *PostGIS* decidiram seguir a *Simple Feature Specifications* (SFS), do consorcio *OpenGIS*, e a ISO (*International Organization for Standardization*). Esse conjunto de especificações fornece o modelo de armazenamento comum de dados geoespaciais no formato *Well-Know Text* (WKT). Os tipos permitidos pela SFS para criar e representar um objeto espacial são *Point*, *MultiPoint*, *LineString*, *MultiLineString*, *Polygon*, *MultiPolygon* e *GeometryCollection*. A Figura 4.9 ilustra estes objetos espaciais e seu equivalente geométrico.

Figura 4.9 – Objetos espaciais WKT e seus equivalentes geométricos.



Versões mais recentes do *PostGIS* estendem os comandos de *SQL* do Banco de Dados *PostgreSQL* com aproximadamente 900 funções, gerando um total de aproximadamente 1.200 funções do tipo *SQL*. Dentre as muitas funções do *PostGIS* utilizadas neste trabalho, pode-se mencionar como as mais usadas:

ST_Envelope: retorna uma geometria válida representando a caixa delimitadora da geometria;

ST_Contains: analisa se uma das duas geometrias dadas está contida na outra;

ST_Intersects: analisa se duas geometrias possuem alguma interseção, retornando *verdadeiro* caso isto ocorra;

ST_IsValid: recebe uma geometria como entrada e verifica se a mesma é valida, ou seja, é uma daquelas descritas na Figura 4.9;

ST_Distance: calcula a distância cartesiana mínima, com base na projeção, entre duas geometrias em unidades projetadas;

ST_MakeLine: cria uma geometria do tipo *Linestring* a partir de outras geometrias do tipo *Linestrings* ou *Point*.

Há outras alternativas para se armazenar mapas fora os SGBDEs. Por exemplo, um conjunto de arquivos denominados *Shape Files*, da ESRI, é o mais popular. Criado na década de 1990, hoje serve como um formato de intercâmbio entre vários programas. O próprio *PostGIS* apresentar rotinas para importar e até criar *Shape Files*. Um *Shape File* é composto basicamente por três arquivos com as seguintes extensões: *.shp*, contendo a informação vetorial, ou seja, as coordenadas georreferenciadas dos objetos desta geometria; *.dbs*, contendo informações sobre os objetos georreferenciados; e *.shx*, contendo os índices que relacionam os vetores do arquivo *.shp* às informações do arquivo *.dbs*. Em comparação com um SGBDE, a grande desvantagem do *Shape* é a dificuldade no equacionamento das questões de otimização de consultas, gerência de transações e controle de integridade e de concorrência (OLIVEIRA, 2005).

4.4 Ambiente de Resolução Georreferenciado do PCV

O Ambiente de Resolução Georreferenciado para o PCV (*ARG-PCV*) proposto segue uma premissa acadêmica, de servir como ferramenta de facilitação, para o estudo a cerca da aplicação prática do PCV no segmento de roteamento que permite:

a) modelar uma determinada malha viária em um grafo orientado e valorado;

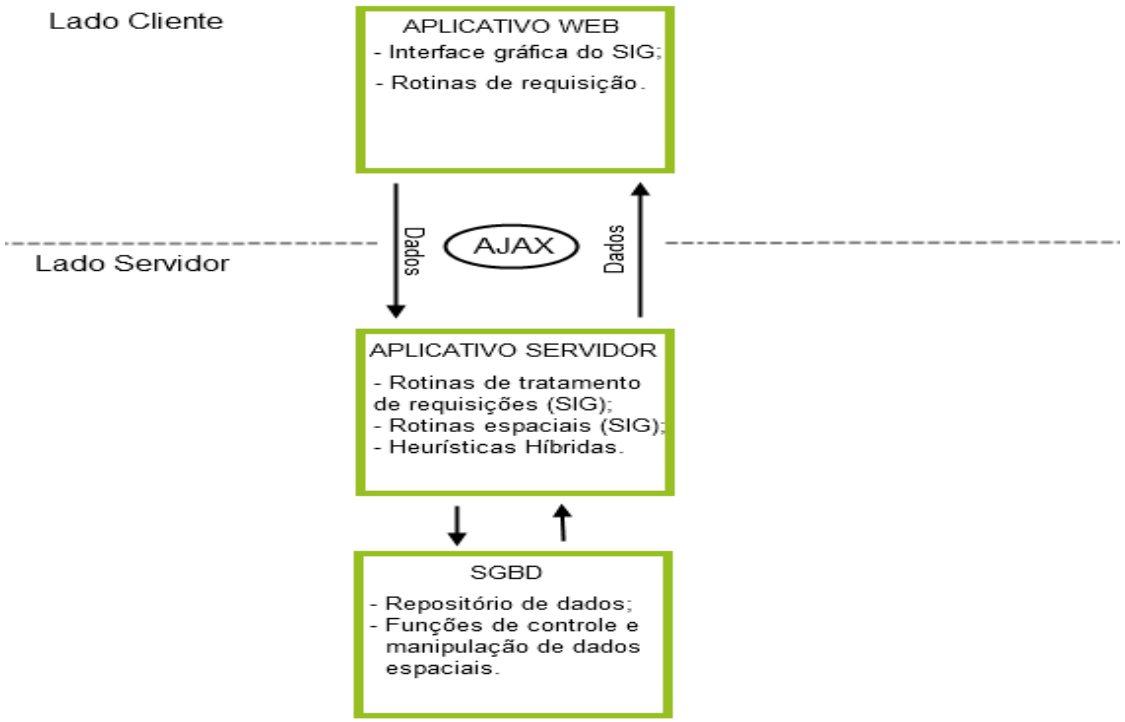
b) criar instâncias georreferenciadas do PCV através do cadastro de pontos sobre a malha viária;

c) resolver as instâncias criadas do problema a partir da seleção de um ou mais métodos heurísticos implementados neste ambiente;

d) produzir um subgrafo, a partir da solução proposta para a resolução do problema, com o intuito de visualizar e analisar graficamente esta solução;

e) acompanhar visualmente o processo de construção de rotas.

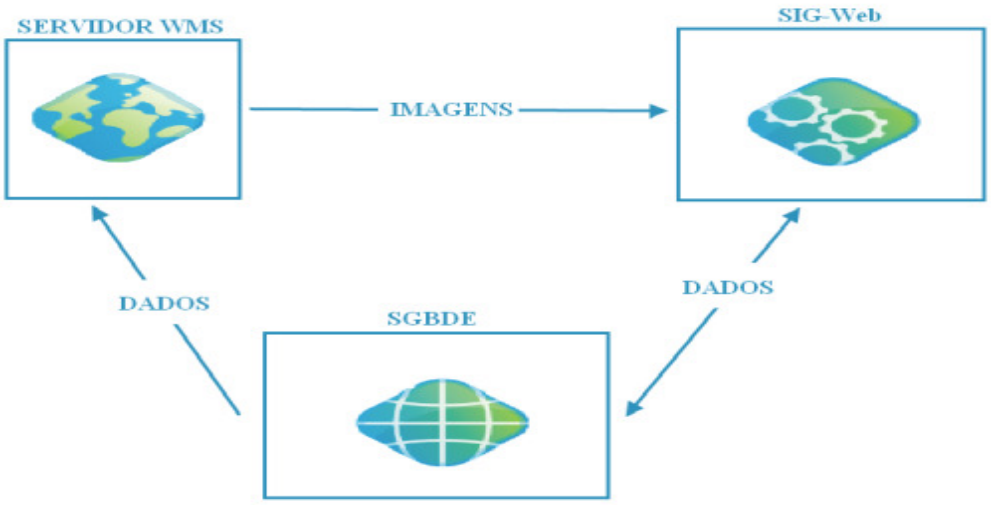
Figura 4.10 – Arquitetura *Cliente-Servidor* do *ARG-PCV*.



O ARG-PCV pode funcionar dentro de uma rede interna ou externa de computadores (internet/intranet) com a arquitetura *Cliente-Servidor*. Battisti (2001) conceituou a arquitetura *Cliente-Servidor* como aquela onde o processamento da informação é dividido em módulos ou processos distintos. Um processo é responsável pela manutenção da informação (*Servidor*), enquanto o outro é responsável pela obtenção dos dados (*Cliente*). Neste trabalho, o lado *Cliente* consiste na interface gráfica do SIG que conta com rotinas para visualizar e interagir com mapas e realizar as requisições ao lado do *Servidor*. Todas as rotinas de requisições foram implementadas em linguagem *JavaScript* utilizando AJAX (*Asynchronous JavaScript and XML*), um conjunto de tecnologias do *JavaScript* que permite, dentre outras funcionalidades, atualizar partes de uma página web sem recarregar a página inteira através de requisições assíncronas ao lado *Servidor*. No lado do *Servidor*, constam as Heurísticas Híbridas propostas, descritas no Capítulo 3, e as rotinas do SIG, de acesso a banco de dados e de geração de grafos, programadas em linguagem *Java* seguindo o paradigma da Programação Orientada a Objetos. A Figura 4.10 acima ilustra esta arquitetura.

O SIG é assistido por um SGBDE e um Servidor WMS (*Web Map Service*). A principal função desempenhada por um Servidor WMS é a de fornecer mapas a partir da interpretação do componente espacial de objetos persistidos em algum tipo de repositório suportado por este Servidor WMS. Como já mencionado, o SGBDE escolhido foi o *PostgreSQL/PostGIS.* Neste SGBDE, foi criado um banco de dados denominado *GRAPH,* para persistir todos os dados do ARG-PCV utilizando o *datum planimétrico* SIRGAS. Já o Servidor WMS selecionado foi o *GeoServer*, que é bastante utilizado para se trabalhar com o PostgreSQL/PostGIS. Além disso, ele utiliza o *GeoTools*, uma biblioteca Java de analise e processamento de componentes geoespaciais, e os padrões da *Open Geospatial Consortium* (OGC), possibilitando a interoperabilidade entre diversos bancos de dados espaciais, serviços e aplicações.

Figura 4.11 – A comunicação entre o SIG-Web, Servidor WMS e o SGBDE.

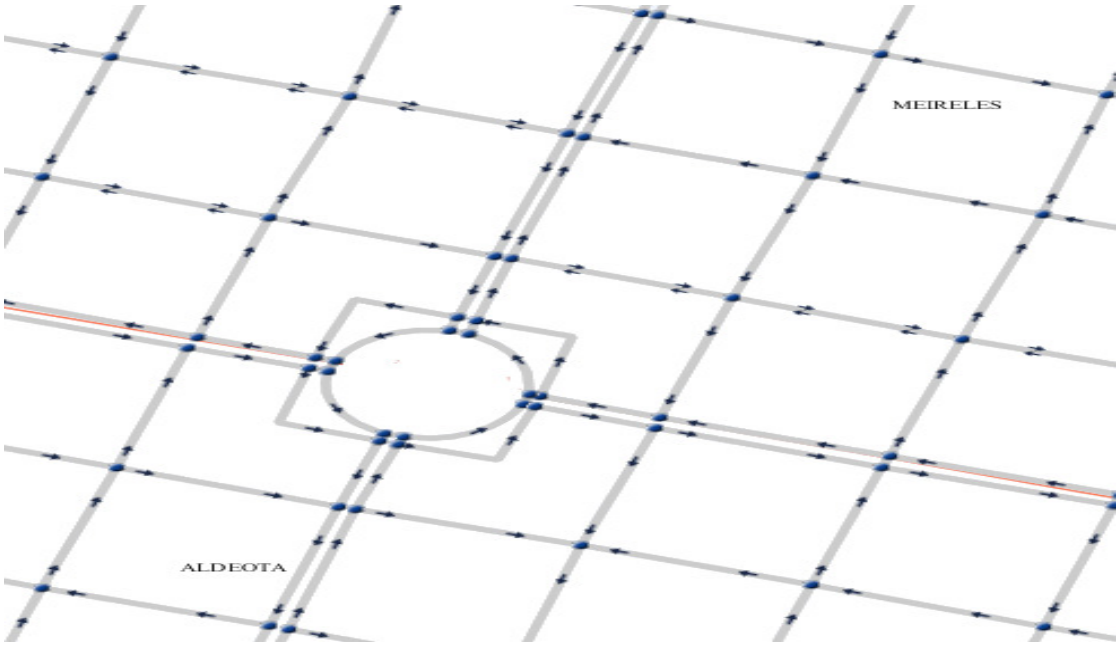


Para executar o *GeoServer* é necessário primeiro instalar um servidor de aplicações *Web* Java. Neste trabalho, optou-se pelo servidor *web Tomcat*. Para facilitar a implementação da interface gráfica do SIG-*Web* e abstrair os processos de requisições ao Servidor *WMS,* optou-se pelo *OpenLayers*, um *framework JavaScript Open Source* concebido para tal fim e criado pela *MetaCarta,* uma organização americana do segmento de geoprocessamento fundada por Jhon R. Frank e financiada por, dentre outras organizações, a também americana DARPA (*Defense Advanced Research*

Projects Agency). A comunicação entre o SIG-Web, Servidor WMS e o SGBDE é ilustrada na Figura 4.11, dada anteriormente.

Além de permitir a visualização dos objetos geoespaciais que compõem o ARG-PCV, outras funções importantes do SIG são gerar o grafo sobre o mapa de uma malha viária e a criação de instâncias a partir do cadastro de pontos. O processo de geração do grafo, baseia-se na topologia *arco-nó.* Nesta topologia, os nós (vértices) do grafo são desenhados a partir das interseções entre as ruas, e os arcos (arestas) são representados pelos trechos das ruas delimitados pelos nós. Técnicas semelhantes sobre a representação de malhas viárias em grafos podem ser encontrados em Assad e Sano (1998) e Lisboa, Iochpe e Borges (2001). Neste trabalho, os dados da malha viária remetem a cidade de Fortaleza-CE. Uma amostra do grafo gerado sobre os dados referentes a esta malha viária é ilustrado na Figura 4.12.

Figura 4.12 – Amostra do grafo da malha viária de Fortaleza gerado pelas rotinas do SIG.



Observe, pela Figura 4.12, que cada rua deste tipo de mapa é formada pela a união de segmentos, onde um segmento determina um quarteirão da rua. Cada segmento tem duas extremidades georreferenciadas denominadas início e fim. Dois atributos do segmento determinam como um veículo pode se movimentar nele através do sentido (Único ou Duplo) e da direção (Norte-Sul, Sul-Norte, Leste-Oeste ou Oeste-Leste).

Assim, é possível construir um grafo de forma bastante simples. Vale ressaltar que nem todos os dados georreferenciados de ruas possuem essa tipologia, foi verificado que mesmo em outras tipologias, diferente da que se apresenta, ainda é possível deixá-la nesta arquitetura.

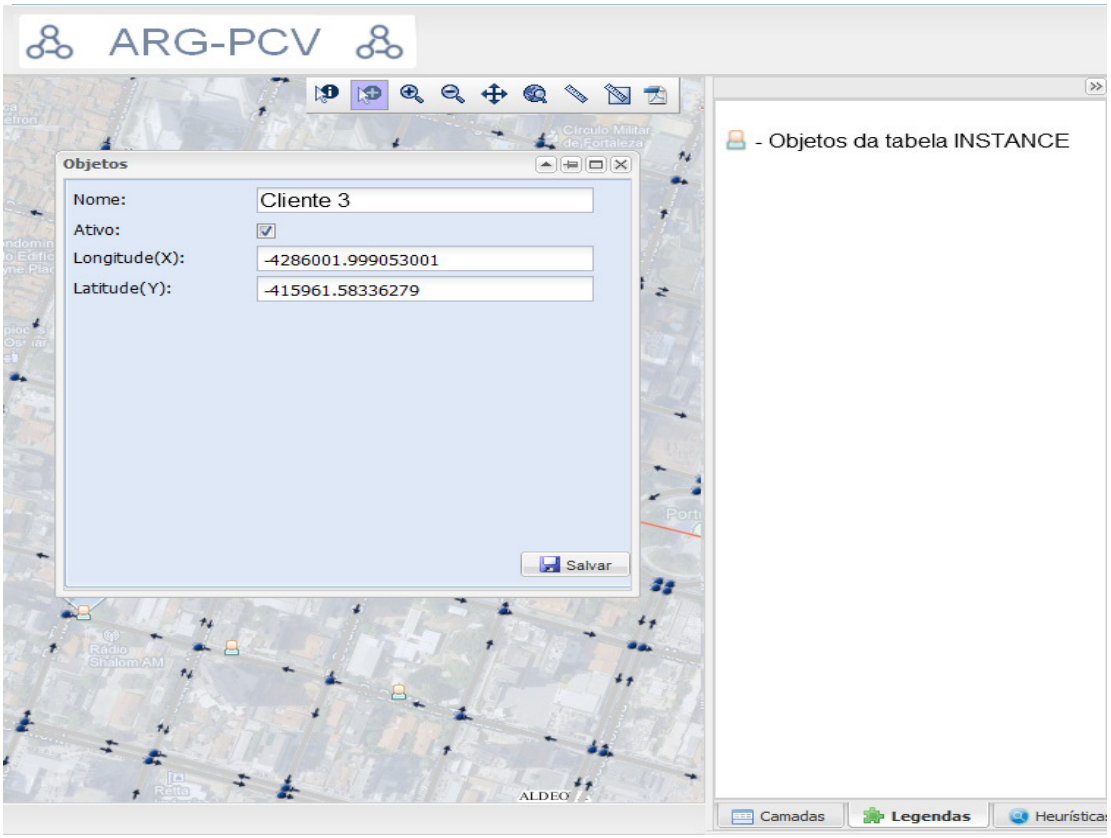
A primeira operação do algoritmo de geração do grafo proposto é criar uma lista de pontos de interseção entre as ruas dispostas no mapa. As interseções são obtidas através da função *ST_Intersects* do *PostGIS*. Concluída a lista de interseções, os elementos dela são inseridos na tabela *NODE* do banco de dados, e com isso é concluído o processo de geração dos nós do grafo. Um procedimento foi desenvolvido para evitar repetição de pontos com as mesmas coordenadas georreferenciadas, haja vista que quando uma ou mais ruas se interceptam, isto acontece nas extremidades dos segmentos. A próxima operação do algoritmo é gerar as arestas do grafo a partir dos nós. Nesta arquitetura, usada por nos e descrita no parágrafo anterior, os segmentos fazem exatamente o papel das arestas do grafo.

Assim, cada trecho (ou segmento) é armazenado na tabela *EDGE* do banco de dados, com a descrição georreferenciada do nó inicial e final deste segmento, levando em consideração os dois nós associados cadastrado na tabela *NODE* com as mesmas coordenadas geográficas. Quando se quer saber quem são os nós sucessores ou antecessores de um nó *i* específico do grafo, basta ver quem são os nós iniciais e finais dos registros cadastrados na tabela *EDGE* com as mesmas coordenadas geográficas do nó *i*, através de uma consulta simples usando comandos SQL. Se o mapa da malha viária informar o sentido das ruas, o grafo gerado é orientado, caso contrário, o grafo produzido é não orientado. Como os dados obtidos sobre a malha viária utilizada possuem um atributo informando o sentido e a direção das ruas, através de seus segmentos, foi possível gerar um grafo orientado, o que acabou por atribuir um grau de realidade maior ao processo de resolução do PCV.

O processo de criação das instâncias no ARG-PCV ocorre a partir do cadastramento de pontos georreferenciados no banco de dados do SIG. Para viabilizar esse processo foi criado na interface gráfica do SIG, uma ferramenta que permite plotar um ponto no mapa clicando sobre o mesmo numa posição específica. Há um procedimento que

determina e armazena a coordenada do ponto específico, depois disso ela é enviada para uma função que cria um objeto tipo *Point* na tabela *INSTANCE* do banco de dados. Todo ponto cadastrado em INSTANCE possui um atributo *booleano* denominado *ativo*. Se um ponto possuir o atributo *ativo* configurado como *verdadeiro* então este ponto será interpretado como uma localidade a ser visitada durante o processo de construção da rota. O SIG conta também com ferramentas para o tratamento de dados tais como editar, atualizar, consultar ou remover os objetos cadastrados na tabela *INSTANCE.* A Figura 4.13 ilustra o cadastro de um ponto na tabela *INSTANCE* por meio da ferramenta desenvolvida.

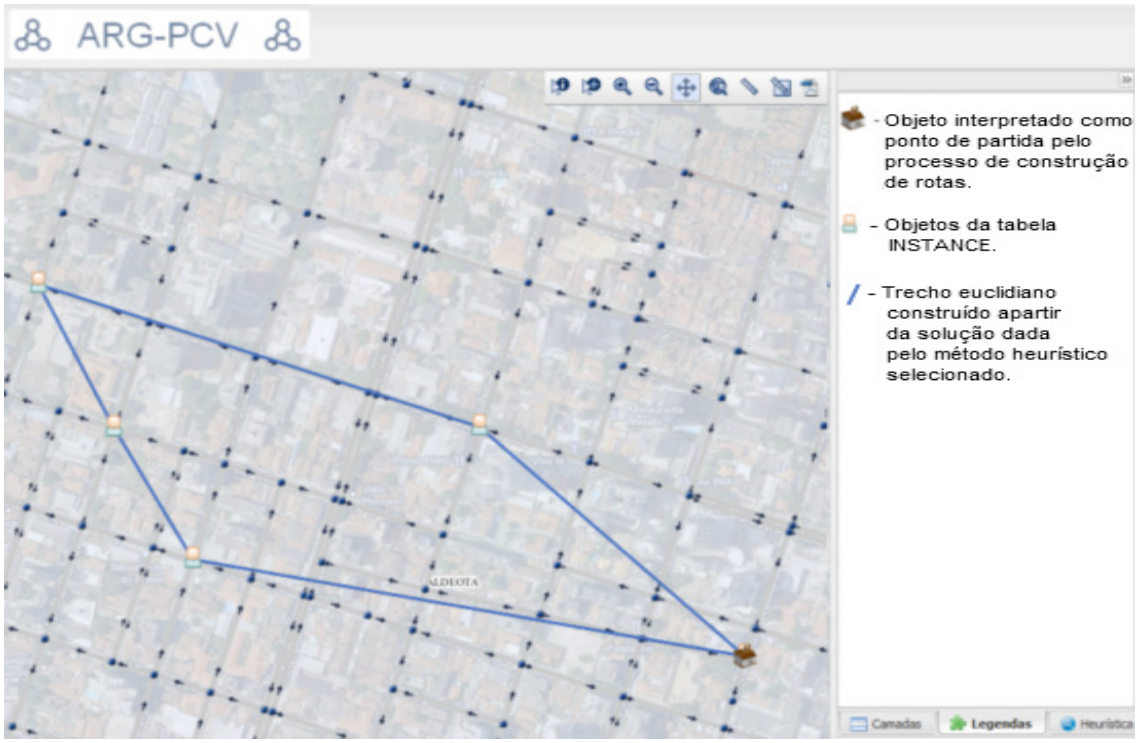
Figura 4.13 – Cadastro de um ponto na tabela INSTANCE através da ferramenta desenvolvida.



Para possibilitar a resolução das instâncias do PCV, cadastradas no banco de dados do SIG, foi criado um formulário no próprio ambiente, apropriado para acionar uma ou mais das Heurísticas desenvolvidas e implementadas, neste ambiente, usada especificamente para a resolução do PCV. Depois de acionada uma dessas Heurísticas, a

primeira operação é criar uma matriz de custo *D* dos pontos ativos da tabela *INSTANCE* que servirá como entrada de dados da Heurística selecionada. É considerado como ponto de partida e, consequentemente, de conclusão da rota, o objeto de *INSTANCE* que possuir código igual a 1. Os custos são calculados a partir da função *ST_Distance* do *PostGIS*. Esta função calcula a distância euclideana entre dois pontos considerando fatores *geodésicos* como as irregularidades da superfície terrestre. A tabela *HISTORY* foi criada no banco de dados do SIG para armazenar informações sobre a execução dos métodos heurísticos. Desta forma, torna-se possível analisar os dados da última execução da Heurística selecionada como tempo de execução, custo da melhor rota apresentada para uma instância, rota sugerida, etc.

Figura 4.14 – Solução dada por uma das heurísticas selecionada.

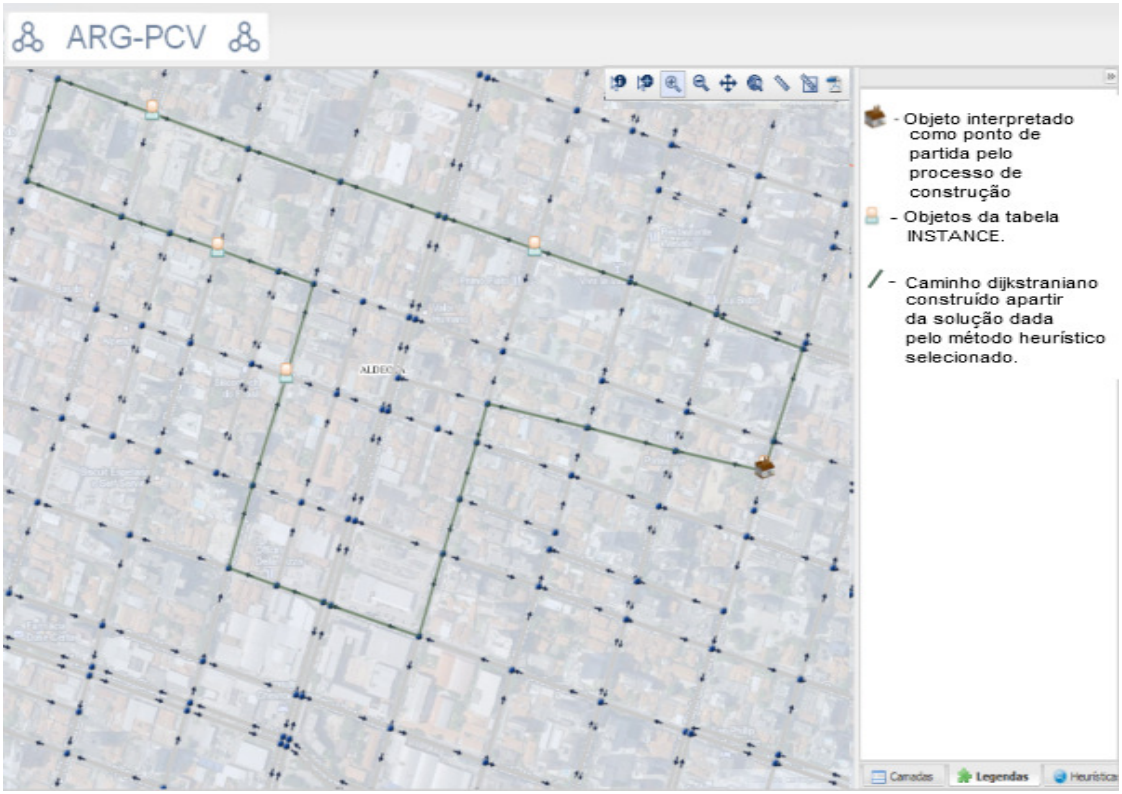


Como pode ser visto na Figura 4.14, a solução dada pelo método heurístico selecionado é baseada em distância euclidiana. Com base na rota dada pela Heurística tem-se que visitar cada um dos pontos ativos. Estes pontos e suas ordens de visitas vão gerar n trajetos, cada um com um ponto (vértice) inicial e outro final. O Algoritmo de *Dijkstra* foi implementado e adicionado a este ambiente para construir uma rota real a partir da

solução heurística considerando a estrutura do grafo modelado. Criado pelo pesquisador holandês Edsger Wybe Dijkstra, em 1956 e publicado em 1959, este algoritmo baseia-se no algoritmo BFS e em uma estratégia gulosa, o problema de caminhos mais curtos em um grafo ponderado orientado $G = (V, E)$ para o qual todos os pesos das arestas são não negativos.

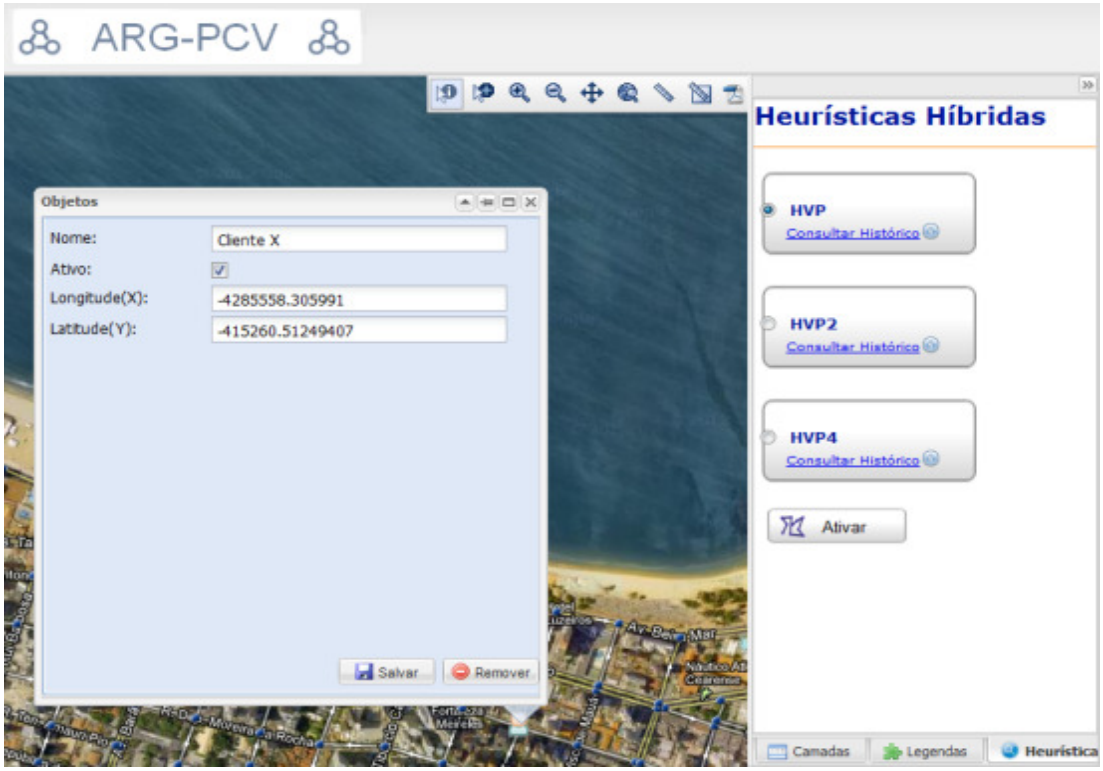
Em Cormen et. al. (2001) há exemplos e detalhes sobre o algoritmo de Dijkstra. Nesta versão implementada, o algoritmo calcula o custo mínimo para ir do vértice inicial ao vértice final em cada trajeto. Esses dois pontos inicial e final são colocados no grafo e seus nós sucessores e antecessores são determinados com base nas arestas cadastradas na tabela *EDGE* mais próximas aos pontos inicial e final, respectivamente, levando em consideração o sentido e a direção dos segmentos para a proximidade, ou seja, o que se quer é determinar quais os pontos de quinas mais próximo deles onde a partir dali haja uma boa locomoção. A Figura 4.15, a seguir, mostra uma rota gerada para o exemplo ilustrado na Figura 4.14, já usando esta implementação do Dijkstra.

Figura 4.15 – Rota gerada pelo algoritmo de Dijkstra.



A interface gráfica do SIG-Web conta com várias ferramentas que permitem o controle de visualização dos objetos, o cálculo de distância, o calculo de área, a conversão do mapa em arquivos PDF (Portable Document Format), a consulta em objetos específicos do mapa como cliente, ruas, segmentos de ruas, etc. e a visualização de legendas. Uma função capaz de acessar as imagens de satélites fornecida pelo *Google Maps* foi implementada para o SIG, com o intuito de melhorar a ambientação espacial do ARG-PCV. Uma prévia do ARG-PCV acessando as imagens do *Google Maps* pode ser vista na Figura 4.16.

Figura 4.16 – ARG-PCV com algumas funcionalidades ativadas.



Assim, conclui-se a descrição passo a passo para a construção de um ambiente computacional, através de um SIG, onde é possível realizar um estudo mais acurado sobre a forma de resolver o PCV tanto teórica, com instâncias consolidadas da literatura, quanto prática, com instâncias reais, como aquelas praticadas pelas empresas de cargas, de distribuição de produtos, de manutenção de elevadores, de serviços, dos correios, etc.

CAPÍTULO 5 – EXPERIMENTOS COMPUTACIONAIS

Neste capítulo são apresentados os experimentos computacionais realizados com os métodos heurísticos desenvolvidos. A primeira seção é dedicada a execução dos métodos heurísticos para diversas instâncias do PCV, simétrico e assimétrico disponíveis na TSPLIB. Já a seção 5.2 trata da aplicação prática dos métodos heurísticos em instâncias georreferenciadas criadas no ARG-PCV.

5.1 Aplicação aos problemas clássicos da literatura TSPLIB

Vários experimentos computacionais foram realizados para observar o desempenho das Heurísticas propostas no capítulo 3, sendo elas a Heurística Híbrida do Vizinho Mais Próximo (*HHVP*), Heurística do Vizinho mais Próximo com 2 caminhos paralelos (*HVP2*), Heurística do Vizinho mais Próximo com 2 caminhos paralelos com inserção de uma cidade por iteração (*HVP2B*), Heurística do Vizinho Mais Próximo com 4 caminhos paralelos (*HVP4*), Heurística da Inserção Mais Barata com rotas iniciadas por cada uma das n cidades (*HIMB*) e Heurística da Inserção Mais Barata Modificada (*HIMBM*). Estas heurísticas foram executadas em um microcomputador *PlugPC* (3,0 Ghz e 1 Gb de RAM), tendo o código implementado em linguagem C/C++, do compilador Dev C++ na versão 4.9.9.2.

Vale ressaltar que em todas as heurísticas descritas acima o procedimento *4-Opt* também foi executado. Assim, avaliou-se o desempenho das heurísticas propostas, descritas acima, de duas formas, uma com o critério *4-Opt* e outra sem o uso deste critério. As Tabelas dadas a seguir mostram o desempenho destas heurísticas sem e com o uso de *4-Opt*, respectivamente. Os resultados obtidos também foram comparados com uma boa Meta-Heurística para efeito de avaliação. Esta Meta-Heurística, denominada de *AG3*, é baseada na evolução dos algoritmos genéticos, descrita no trabalho de Silva, Soma e Viana (2004). Este algoritmo genético usou o procedimento *3-opt* como o operador mutação do método, ou seja, o algoritmo foi hibridizado. Ainda neste trabalho, têm-se os dados relativos ao desempenho de *AG3* sem o operador mutação, denominado simplesmente de *AG*.

Nas Tabelas 5.1 e 5.3, dadas a seguir, encontram-se: a referência da instância testada; o valor de *n*; e o desvio em relação à solução ótima, dado por 100*(z-z')/z', onde z é o valor encontrado pelo método e z' é a solução ótima.

Tabela 5.1 – Desempenho dos métodos sem hibridização.

Referência	*n*	AG	HHVP	HVP2	HVP2B	HVP4	HIMB
br17	17	61,54%	43,59%	2,56%	43,59%	7,69%	0,00%
gr17	17	22,35%	4,51%	3,12%	4,46%	1,82%	0,00%
gr21	21	58,29%	16,66%	8,35%	11,86%	10,01%	0,00%
gr24	24	45,60%	19,73%	8,41%	15,25%	4,48%	1,42%
ftv35	36	1,83%	13,17%	15,75%	15,27%	17,38%	7,81%
ftv38	39	8,82%	20,85%	20,72%	16,47%	14,12%	4,71%
Dantzig42	42	0,00%	27,75%	11,44%	17,31%	9,44%	15,16%
gr48	48	20,17%	32,08%	8,74%	10,74%	8,18%	1,70%
ry48	48	30,42%	8,60%	9,19%	9,58%	9,10%	4,36%
eil51	51	25,73%	25,59%	11,74%	16,67%	12,68%	3,76%
ftv55	56	17,04%	25,75%	26,55%	31,03%	26,62%	12,81%
Brazil58	58	20,43%	10,41%	8,03%	5,64%	5,61%	3,48%
ftv64	65	28,28%	26,86%	23,87%	33,44%	31,43%	10,66%
pr76	76	17,72%	40,13%	14,83%	11,79%	11,79%	13,30%
gr96	96	10,90%	20,43%	12,82%	9,31%	6,38%	4,43%
kroC100	100	22,53%	25,35%	7,15%	8,15%	12,62%	4,49%
kro124	100	13,96%	21,74%	19,71%	19,61%	22,59%	8,17%
pr107	107	26,11%	5,80%	4,05%	3,80%	13,45%	8,72%
Bier127	127	17,60%	17,66%	10,61%	10,03%	15,01%	8,13%
Kroa150	150	8,65%	27,16%	15,76%	18,29%	16,16%	6,35%
Tsp225	225	9,55%	23,11%	12,54%	16,50%	15,53%	10,62%
Rbg403	403	175,09%	42,68%	43,29%	67,34%	42,72%	5,40%
ali535	535	8,01%	32,20%	19,90%	19,11%	22,86%	2,60%
Média		**28,29%**	**23,12%**	**13,88%**	**18,05%**	**14,68%**	**6,00%**

Analisando os dados da Tabela 5.1 têm-se os seguintes comentários:

a) AG foi o método com o pior desempenho (28,29%), enquanto HIMB foi aquele com o melhor desempenho (6,00%), depois dele vem HVP2 (13,88%);

b) AG encontrou a solução ótima em apenas um caso e HIMB encontrou a solução ótima nas 3 primeiras instâncias;

c) Em apenas uma instância AG teve desempenho para valores do desvio maior que 0% e menor que 5%, enquanto HIMB obteve 9 instâncias dentro destes limites. Ainda neste quesito os demais métodos foram superiores a AG, exceto HHVP.

A Tabela 5.2 mostra o desempenho dos métodos no tempo computacional gasto para a apresentação da solução do problema.

Tabela 5.2 – Tempo gasto na obtenção da solução do problema.

Referência	*n*	HHVP	HVP2	HVP2B	HVP4	HIMB
br17	17	0	0	0	0	0
gr17	17	0	0	0	0	0
gr21	21	0	0	0	0	0
gr24	24	0	0	0	0	0
ftv35	36	0	0	0	3	0
ftv38	39	0	0	0	5	0
Dantzig42	42	0	0	0	7	0
gr48	48	0	0	0	17	0
ry48	48	0	0	0	17	0
eil51	51	0	0	0	2	0
ftv55	56	0	0	0	3	0
Brazil58	58	0	0	0	4	0
ftv64	65	0	0	0	8	0
pr76	76	0	0	0	21	0
gr96	96	0	1	1	84	0
kroC100	100	0	1	1	23	0
kro124	100	0	0	0	22	0
pr107	107	0	1	2	33	0
Bier127	127	0	3	4	97	0
Kroa150	150	0	7	9	79	0
Tsp225	225	0	31	42	389	0
a280	280	0	73	96	641	0
Rbg323	323	0	124	174	783	0
Rbg403	403	0	286	378	1025	0
ali535	535	1	706	1182	3207	2
Média		**0,0**	**53,6**	**82,1**	**281,3**	**0,1**

Analisando os dados da Tabela 5.2 fazem-se os seguintes comentários:

a) Os valores iguais a zero traduz-se que a execução levou menos de 1 segundo;

b) Os tempos de AG não foram apresentados no artigo de Silva, Soma e Viana (2004);

c) As heurísticas HHVP e HIMB gastaram, em média, menos de 1 segundo, mostrando que elas são muito rápidas;

d) HVP2B e HVP4 foram os métodos que gastaram mais tempo. Contudo, HVP2B e HVP4 responderam em média a menos de 2 e 5 minutos, respectivamente.

Tabela 5.3 – Desempenho dos métodos propostos com hibridização.

Referência	*n*	AG3	HHVP	HVP2	HVP2B	HVP4	HIMB	HIMBM
br17	17	0,00%	0,00%	0,00%	0,00%	2,56%	0,00%	0,00%
gr17	17	0,24%	0,00%	1,68%	0,00%	1,82%	0,00%	0,00%
gr21	21	0,00%	16,66%	4,65%	10,75%	4,62%	0,00%	0,00%
gr24	24	0,00%	13,84%	2,04%	2,67%	4,48%	1,34%	1,34%
ftv35	36	0,95%	9,03%	6,59%	13,44%	13,71%	7,81%	3,80%
ftv38	39	0,92%	15,69%	15,42%	13,33%	10,46%	3,01%	3,53%
Dantzig42	42	0,00%	22,32%	5,72%	11,73%	4,01%	13,73%	4,86%
gr48	48	0,18%	21,68%	7,89%	9,89%	6,36%	1,70%	1,51%
ry48	48	0,31%	7,07%	5,82%	6,30%	6,09%	4,36%	1,82%
eil51_e	51	0,70%	23,47%	9,62%	14,55%	10,33%	3,76%	1,88%
ftv55	56	1,18%	24,13%	23,69%	25,12%	20,58%	12,69%	3,11%
Brazil58	58	0,00%	8,35%	5,65%	3,58%	4,49%	3,39%	0,99%
ftv64	65	2,01%	24,42%	17,89%	27,57%	27,68%	9,24%	7,67%
pr76	76	1,75%	32,58%	13,57%	16,07%	10,60%	11,60%	5,80%
gr96	96	1,08%	14,97%	10,09%	6,77%	4,23%	4,04%	4,04%
kroC100	100	3,19%	24,62%	6,28%	7,17%	9,64%	4,47%	3,10%
kro124	100	4,37%	20,36%	17,95%	17,58%	20,64%	7,74%	7,74%
pr107	107	0,29%	4,70%	3,58%	2,71%	12,79%	8,46%	8,46%
Bier127	127	2,18%	16,98%	10,08%	9,41%	13,76%	7,72%	7,72%
Kroa150	150	1,54%	25,20%	15,01%	17,95%	15,40%	6,06%	6,06%
Tsp225	225	0,32%	22,63%	10,85%	16,16%	13,71%	9,83%	9,83%
Rbg403	403	0,00%	41,70%	41,95%	63,37%	41,62%	4,99%	4,99%
ali535	535	1,78%	31,76%	19,80%	18,96%	22,76%	2,40%	2,11%
Média		**1,00%**	**18,35%**	**11,12%**	**13,70%**	**12,28%**	**5,58%**	**3,93%**

Analisando os dados da Tabela 5.3 têm-se os seguintes comentários:

a) AG3 foi o método com o melhor desempenho (1,00%), enquanto HIMBM foi o segundo melhor (3,93%), ficando bem próximo de AG3;

b) AG3 encontrou a solução ótima em 6 instâncias e HIMBM e HIMB encontraram a solução ótima nas 3 primeiras instâncias. Exceto HVP4, as demais heurísticas também chegaram a encontrar a solução ótima em pelo menos uma instância;

c) AG3 teve desempenho inferior a 5,00% para todas as 23 instâncias testadas, enquanto HIMBM obteve 16 das 23 instâncias;

d) Todas as heurísticas melhoraram seus desempenhos quando executaram o procedimento *4-Opt*, sendo HIMB aquele que obteve menos êxito que os demais;

e) Nas instâncias ftv38, Dantzig42 e pr107 as heurísticas HIMB, HVP4 e HVP2B (da mesma forma HHVP e HVP2) obtiveram melhor resultado que HIMBM, respectivamente.

Na Tabela 5.4, dada a seguir, é apresentado o tempo que cada método hibridizado gastou para determinar a solução do problema.

Tabela 5.4 – Tempo gasto, com hibridização, na obtenção da solução do problema.

Referência	*n*	AG3	HHVP	HVP2	HVP2B	HVP4	HIMB	HIMBM
br17	17	2	0	0	0	0	0	0
gr17	17	2	0	0	0	0	0	0
gr21	21	3	0	0	0	0	0	0
gr24	24	3	0	0	0	0	0	0
ftv35	36	4	0	0	0	3	0	3
ftv38	39	4	0	0	0	5	0	4
Dantzig42	42	5	1	1	1	8	0	6
gr48	48	8	1	1	1	18	0	13
ry48	48	8	1	1	1	18	0	13
eil51	51	10	0	0	0	2	0	20
ftv55	56	13	0	0	0	3	0	33
Brazil58	58	14	0	0	0	4	0	40
ftv64	65	17	0	0	0	8	0	77
pr76	76	32	0	0	0	22	0	203
gr96	96	65	6	7	8	91	0	315
kroC100	100	132	1	1	1	23	0	0
kro124	100	132	1	1	1	22	0	0
pr107	107	197	3	3	3	35	0	1
Bier127	127	344	5	5	5	102	1	1
Kroa150	150	496	4	11	12	82	2	2
Tsp225	225	1417	12	34	57	402	14	24
Rbg403	403	6312	20	306	398	1027	255	350
ali535	535	7200	23	729	1210	3263	511	630
Média		**713,9**	**3,4**	**47,8**	**73,8**	**223,4**	**34,0**	**75,4**

Analisando os dados da Tabela 5.4 fazem-se os seguintes comentários:

a) Os valores iguais a zero traduz-se que a execução levou menos de 1 segundo;

b) AG3 foi o método que gastou mais tempo, seguidos por HVP4, HIMBM, HVP2B, HVP2, HIMB e HHVP;

c) Todas as heurísticas não seguem uma ordem crescente de tempo quando n cresce devido ao salto colocado no procedimento *4-Opt*;

d) HHVP, HIMB e HIMBM foram os métodos que gastaram menos tempo em média;

e) HIMBM gastou 10,5 minutos na instância com *n*=535, enquanto AG3 gastou 120 minutos, quase 10,5 vezes mais que HIMBM. Isto mostra que HIMBM, mesmo com um bom desempenho, é mais eficaz que AG3.

5.2 Aplicação em Instâncias Georreferenciadas no ARG-PCV

Depois da avaliação realizada, onde constatou-se principalmente a eficácia (eficiência apenas HIMBM) dos métodos propostos para resolver o problema, resolveu-se implementá-los no ARG-PCV. A Tabela 5.5, a seguir, mostra a evolução de três instâncias criadas especificamente neste ambiente para *n*=50, 100 e 200 clientes que devem ser visitados. Nesta tabela tem-se o valor de *n*, as heurísticas aplicadas na solução dessas instâncias (HHVP, HVP2, HVP2B, HVP4, HIMB e HIMBM), a solução inicial e final encontrada por cada método, o desvio entre estas soluções e o tempo gasto para obtenção da solução final.

Tabela 5.5 – Aplicação dos métodos propostos dentro do ARG-PCV.

n	Heurísticas	Sol. Inicial	Sol. Final	Desvio	tempo(s)
50	**HHVP**	304	232	31,03%	0,112
	HVP2	308	195	57,95%	0,187
	HVP2B	265	202	31,19%	0,219
	HVP4	332	192	72,92%	20,469
	HIMB	203	195	4,10%	0,021
	HIMBM	198	**188**	5,32%	19,053
100	**HHVP**	335	305	9,84%	0,562
	HVP2	327	274	19,34%	1,922
	HVP2B	338	276	22,46%	2,437
	HVP4	377	278	35,61%	141,969
	HIMB	264	241	9,54%	0,325
	HIMBM	260	**238**	9,24%	25,213
200	**HHVP**	468	422	10,90%	9,015
	HVP2	455	377	20,69%	27,891
	HVP2B	454	381	19,16%	49,875
	HVP4	524	365	43,56%	1566,184
	HIMB	308	295	4,41%	9,076
	HIMBM	308	**294**	4,76%	68,527

Analisando os dados contidos nesta tabela constatou-se que:

a) HIMBM obteve o melhor desempenho, pois apresentou a melhor solução final para cada problema. Depois dela veio HIMB;

b) HVP4 obteve a melhor evolução, em se tratando de uma solução inicial para a solução final;

c) O tempo gasto na prática com os métodos foram muito bons, em se tratando de aplicações práticas, em geral menos de 2 minutos. Exceção feita a HVP4;

d) HVP4 foi a que gastou mais tempo, depois dela ficou HIMBM com um tempo bem distante de HVP4. Isto mostra a eficácia e eficiência de HIMBM quando comparada com as demais heurísticas.

As Figuras 5.1 e 5.2, dadas a seguir, mostram as ilustrações das soluções propostas pelos métodos HIMBM e HVP2, respectivamente, no ambiente ARG-PCV para *n*=50.

Figura 5.1 – Solução prática de HIMBM para *n*=50.

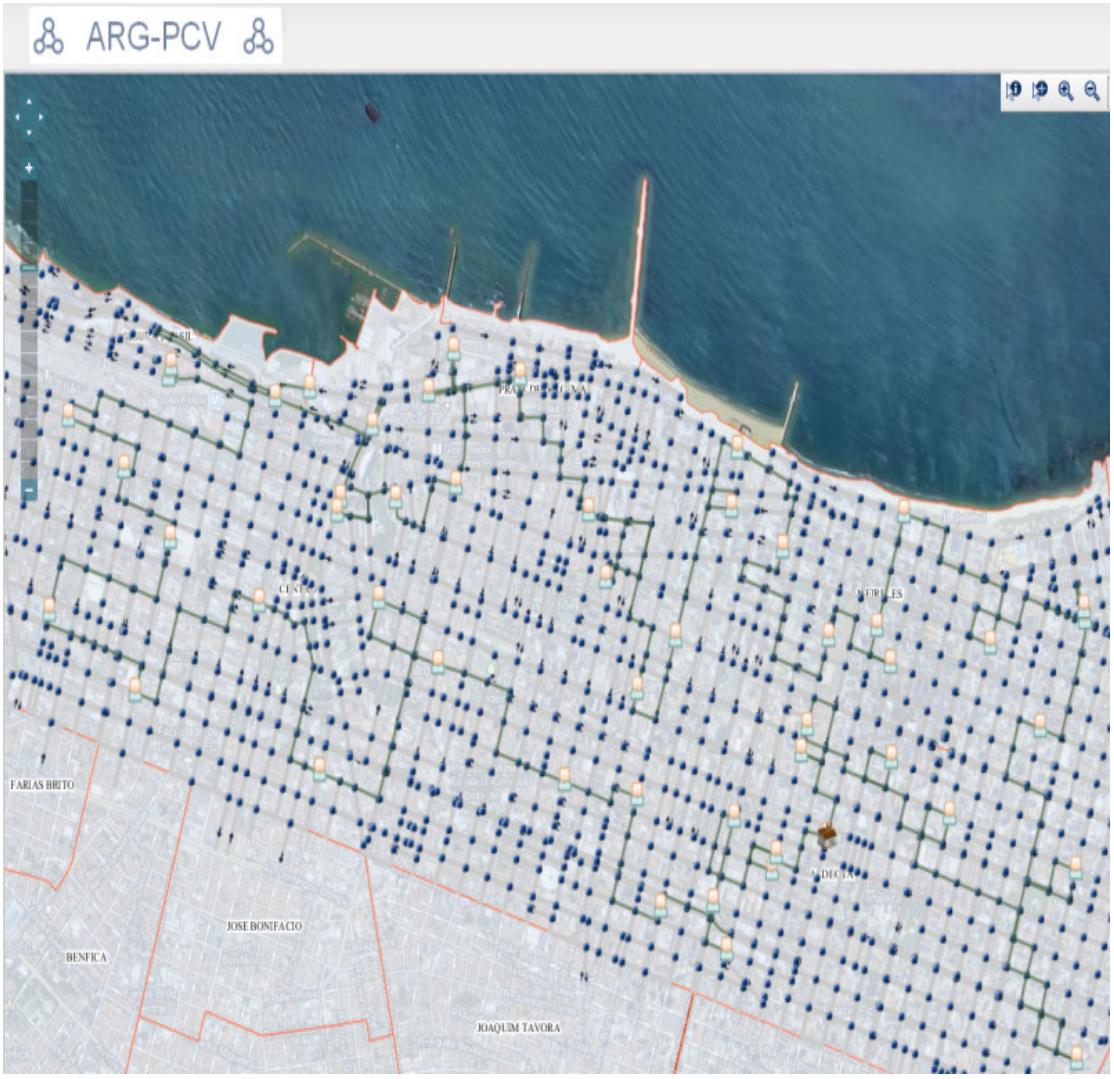


Figura 5.2 – Solução prática de HVP2 para *n*=50.

CAPÍTULO 6 – CONSIDERAÇÕES FINAIS

6.1 Conclusões

Neste trabalho, um dos objetivos foi propor métodos heurísticos para resolver o Problema do Caixeiro Viajante de forma eficaz e eficiente e sem um consumo demasiado de recursos computacionais. Para isso, foram implementadas heurísticas híbridas a partir da junção de heurísticas construtivas e de melhoria de roteiros. Para validar a eficiência dos métodos heurísticos propostos, os mesmos foram testados com instâncias clássicas do PCV dispostas na biblioteca TSPLIB. Na bateria de testes, foi observado como vantagem das heurísticas construtivas, a produção de boas soluções em intervalos mínimos de tempo. Porém, notou-se como desvantagem destes métodos, escolhas ruins para composição da rota no fim do processo de execução em detrimento do caráter guloso empregado por estes métodos. No processo de desenvolvimento da heurística de melhoria de roteiros *4-Opt* foi observado que mais permutações demandam mais tempo de execução, porém, a qualidade das soluções geradas por este tipo de algoritmo tende a melhorar com a operação de um número maior de permutações construídas e avaliadas. Os resultados encontrados através dos métodos heurísticos desenvolvidos neste trabalho para as instâncias escolhidas equiparam-se aos encontrados na literatura. Assim pode-se afirmar que a metodologia utilizada obteve bons resultados com eficiência e rapidez para resolver o PCV.

O principal objetivo deste trabalho foi desenvolver um sistema que permitisse o estudo de aplicações práticas do PCV no segmento de roteirização. Para isso foi desenvolvido o Ambiente de Resolução Georreferenciado para o problema citado. Este ambiente é produto da integração entre os métodos heurísticos propostos e um Sistema de Informação Georreferenciada. Para o SIG, foram desenvolvidas rotinas para geração de grafos a partir de um mapa de malha viária, criação de instâncias georreferenciadas do PCV e ativação de uma das heurísticas para resolução da instância criada. A escolha da topologia arco-nó para representação de malhas viárias contidas em mapas facilitou o estudo e tornou possível a aplicação dos métodos de resolução propostos às instâncias georreferenciadas. O uso de *softwares* e linguagens livres para desenvolver o SIG foi primordial para que todos os subproblemas decorrentes da implementação pudessem ser

resolvidos por meio da expansão ou adequação dos módulos destes *softwares* as necessidades correntes. Ao final do desenvolvimento do ARG-PCV, o mesmo foi submetido a testes e análises a partir da resolução de instâncias criadas de médio e grande porte georreferenciadas. Neste ambiente foi possível ver o sucesso desta ferramenta computacional que pode atender, de forma bastante satisfatória, às empresas que usam a roteirização com racionalidade na distribuição dos seus produtos ou serviços aos seus clientes. Ainda nesta ferramenta, o controle e o acompanhamento desta distribuição pode ser vista em tempo real.

6.2 Sugestões para trabalhos futuros

O presente trabalho teve como foco a resolução do PCV no segmento de roteirização. Considerando a grande relevância deste tema apontada na Introdução deste trabalho, tanto para o meio acadêmico quanto para o meio comercial, seguem abaixo algumas sugestões para ampliação do estudo ora realizado acerca de roteamento:

a) Aplicar *4-Opt* em cada solução gerada pelas heurísticas híbridas desenvolvidas, assim como fez AG3, pois HHVP, HVP2, HVP2B, HVP4, HIMB e HIMBM usam o procedimento *4-Opt* somente uma vez e na melhor solução encontrada;

b) Desenvolver e implementar um procedimento computacional, em ARG-PCV, para avaliar a retirada e a inserção de aresta manualmente na rota dentro do ambiente gráfico;

c) Adaptar o ARG-PCV para tratar o problema com janela de tempo para entrega/coleta de produtos ou serviços com frota heterogênea de veículos, ou seja, preparar a ferramenta gráfica para tratar o Problema de Roteamento de Veículos com Janela de Tempo e Frota Heterogênea de Veículos;

d) Paralelizar as heurísticas HVP4 e HIMBM quando usadas sem salto e aplicando *4-Opt*, em todas as soluções geradas, com o intuito de controlar o tempo de execução das mesmas;

e) Desenvolver, implementar e testar HIMBM com mais de uma sub-rota em paralelo, semelhante ao que foi feito com HVP2 e HVP4.

REFERÊNCIAS BIBLIOGRÁFICAS

Andrade, E. L. **Introdução à Pesquisa Operacional: métodos e modelos para a análise de decisão**. Rio de Janeiro: LTC, 2000, 277p.

Applegate, D.; Bixby, R.; Chvátal, V.; Cook, W.. **Finding cuts in the TSP (A preliminary report)**. DIMACS Technical Report. 1995.

_____. **On the Solution of Traveling Salesman Problems**. In: Documenta Mathematica, vol. Extra Volume ICM III, pp. 645-656. 1998.

_____. **The Traveling Salesman Problem**: **A Computational Study**. Princeton University Press. 2006.

Applegate, D. L.; Bixby, R.E.; Chvátal, V.; Cook, W. J.; Espinoza, D. G.; Goycoolea, M.; Helsgaun, K.. **Certification of an optimal TSP tour through 85900 cities**. In: Operations Reearch Letters., Vol. 37, No. 1, pp. 11–15. 2009.

Assad, E. D.; Sano, E. E.. **Sistema de informações geográficas: aplicações na agricultura**. Brasília, DF: EMBRAPA, 2ª edição, 1998.

Avner P.; Bruls T.; Poras I.; Eley L.; Gas S.; Ruiz P.; Wiles M. V.; Sousa-Nunes R.; Kettleborough R.; Rana A.; Morissette J.; Bentley L.; Goldsworthy M.; Haynes A.; Herbert E.; Southam L.; Lehrach H.; Weissenbach J.; Manenti G.; Rodriguez-Tome P.; Beddington R.; Dunwoodie S.; Cox R. D. . **A radiation hybrid transcript map of the mouse genome**. Nature Genetics 29: 194–200. 2001.

Balas, E.; Toth, P.. **Branch and bound methods. In The Traveling Salesman Problem**. Edited by Lawler, E. L.; Lenstra, J. K.; Rinnooy Kan, A. H. G. and Shmoys, D. B. John Wiley & Sons Ltd, 361-401, 1985.

Ballou, Ronald.H.. **Gerenciamento da Cadeia de Suprimentos: Planejamento, Organização e Logística Empresarial**. Porto Alegre:Bookman, 2001.

Baraglia, R.; Hidalgo, J. I.; Perego, R.. **A Parallel Hybrid Heuristic for the TSP**. In: Proceedings of EvoCOP2001, the First European Workshop on Evolutionary Computation in Combinatorial Optimization. 2001.

Battisti, J.. **SQL Server 2000: Administração e Desenvolvimento – Curso Completo.** 2ª Edição. Rio de Janeiro: Axcell Books, 2001.

Bektas, T.. **The multiple traveling salesman problem: an overview of formulations and solution procedures**. Omega. 34, pp. 209–219, 2006.

Bellmore, M.; Nemhauser, G. L.. **The traveling salesman problem: a survey**. Operations Research. Vol. 16, 538-558. 1968.

Berbeglia, G. ; Cordeau, J.; Laporte, G. . **Dynamic pickup and delivery problems**. In: European Journal of Operational Research, 2009.

Bezerra, F. P. **Um algoritmo genético aplicado no problema da roteirização periódica de veículos com caso prático**. Universidade Federal do Ceará. Dissertação (Mestrado em Engenharia de Sistemas Logísticos). 2012.

Bland, R. E.; Shallcross, D. F.. **Large Traveling Salesman Problem Arising from Experiments in X-ray Crystallography: a Preliminary Report on Computation**, Relatório Técnico No. 730, School of OR/IE, Cornell University, Ithaca, New York, 1987.

Blum, C. ; Roli, A. . **Metaheuristics in combinatorial optimization: Overview and conceptual comparison**. ACM Computing Surveys, Vol. 35, pp. 268–308. 2003.

Bodin, L.D.; Golden, B.. **Classification in Vehicle Routing and Scheduling**. Networks, v.11, n.2, p.97-108, 1981.

Bodin, L. D.; Golden, B.; Assad, A.; Ball, M.. **Routing and scheduling of vehicle and crews: The state of the art**. Computers & Operations Research, v.10, n.2, p.63-211, 1983.

Brião, W.. **Métodos de Exploração de Espaço de Projeto em Tempo de Execução em Sistemas Embarcados de Tempo Real Soft Baseados em Redes-Em-Chip**.

Universidade Federal do Rio Grande do Sul. Tese de Doutorado (Programa de Pós-Graduação em Computação). 2000.

Boaventura Netto, P. O. . **Grafos: teoria, modelos e algoritmos**. São Paulo: Edgard Blucher. 2003.

Bosch, R.; Herman, A. . **Continuous line drawings via the traveling salesman problem**. Operations Research Letters, Vol.32, n.4, pp. 302-303. 2004.

Caldas, M. A. F.; Santos, D. A. .**Uma heurística para roteamento de frota de veículos para a distribuição de bebidas em uma região urbana**. In: XLSBPO - Simpósio Brasileiro de Pesquisa Operacional, 2008.

Campelo Júnior, J. U. . **Proposta de otimização da roteirização dos distritos dos carteiros: Um estudo de caso no centro de entrega de encomendas de Fortaleza**. Universidade Federal do Ceará. Dissertação (Mestrado em Engenharia de Sistemas Logísticos). 2010.

Campello, R. E.; Maculan, N.. **Algoritmos e Heurísticas. Desenvolvimento e Avaliação de Performance**. Editora da Universidade Federal Fluminense. 1994.

Carbajal, S. G.; Corne, D. W.; Reid, F.. **Solving Monalisa TSP Challenge with Parallel Ant Colony Optimization**. Computer Science Department. University of Oviedo, Gijon, Spain. 2010. Disponível em: http://www2.epcc.ed.ac.uk/europa/ HPCE2/Reports/0160_GarciaCarbajal.pdf, ultimo acesso em setembro, 2012.

Cerny, V.. **Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm**. Journal of Optimization Theory and Applications, Vol. 45, pp. 41-51. 1985.

Chakhlevitch, K.; Cowling, P. I.. **Hyperheuristics: Recent developments**. In: Cotta C, Sevaux M, S*f*orensen K (eds) Adaptive and Multilevel Metaheuristics, Studies in Computational Intelligence, Vol. 136, Springer, pp. 3-29. 2008.

Chen, Y.; Zhang, P.. **Optimized annealing of traveling salesman problem from the nth-nearest-neighbor distribution**. Physica A: Statistical Mechanics and its Applications, Vol. 371, Issue 2, Pages 627-632. 2006.

Chen, P.; Kendall, G.; Berghe, G. V.. **An Ant Based Hyper-heuristic for the Travelling Tournament Problem**. Computational Intelligence in Scheduling. vol., no., pp.19-26. 2007.

Clarke, G.; Wright, J. W.. **Scheduling of vehicles from a central depot to a number of delivery points**. Operations Research. Vol. 12, No. 4, pp. 568-581. 1964.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C.. **Introduction to Algorithms**. Second Edition, 1184 pages. The MIT Press, Cabridge, MA, USA. 2001.

Cook, S. A. **The complexity of theorem-proving procedures**. In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, ACM, New York, NY, USA, pp. 151-158, 1971.

Cotta, C.; Talbi, E. G.; Alba, E. . **Parallel hybrid metaheuristics**. In E. Alba ed. Parallel Metaheurist ics, a New Class of Algorithms. pp. 347-370. John Wile. 2005.

Croes, G. A. . **A Method for Solving Traveling-Salesman Problems**. Operations Research, Vol. 6, pp. 791-812. 1958.

Crowder, H.; Padberg, M. W.. **Solving Large-Scale Symmetric Traveling Salesman Problems to Optimality**. Management Science, vol. 26, pp. 495-509, 1980.

Crowder, H.; Johnson, E.L.; Padberg, M.W. . **Solving Large Scale Zero One Linear Programming Problems**. In: Operational Research, Vol. 31, pp. 808-834, 1983.

Cunha, C. B.. Aspectos práticos da aplicação de modelos de roteirização de veículos a problemas reais", **Revista Transportes da ANPET – Associação Nacional de Pesquisa e Ensino em Transportes**, Vol.8, n.2, p.51-74, nov. 2000.

D'Alge, J. C. L.. **Coordenadas geodésicas e sistemas de informação geográfica**. In: Congresso e Feira para Usuários de Geoprocessamento da América Latina (GISBRASIL'99), 5., Salvador, 1999. Anais. São José dos Campos: INPE, 1999.

Desrochers, M.; Lenstra, J.K.; Savelsbergh, M.W.P.. **A classification scheme for the vehicle routing and scheduling problems**. European Journal of Operational Research, v.46, n.3, p.322-332, 1990.

Detofeno, T. C. ; Steiner, M. T. A.. **Optimizing Routes for the Collection of Urban Solid Waste: a Case Study for the city of Joinville**. In: IJIE (Iberoamerican Journal of Industrial Engineering), v. 2, pp. 124-136, 2010.

Dorigo, M.; Maniezzo, V.; Colorni, A. . **Ant System: Optimization by a Colony of Cooperating Agents**. IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol. 26(1), 29–41. 1996.

Drexl, Michael. **Rich vehicle routing in theory and practice**. In: Logistics Research, Volume 5, Issue 1-2 , pp. 47-63 , 2012.

Dumitrescu, I.; Stützle, T. **A survey of methods that combine local search and exact algorithms**. Technical Report AIDA-03-07, FG Intellektik, FB Informatik, TU Darmstadt, Germany, 2003.

Erdogan, G.; Cordeau, J.-F.; Laporte, G. . **A Branch-and-Cut Algorithm for the Non-Preemptive Capacitated Swapping Problem**. In: 24th European Conference on Operational Research, Lisbon, Portugal. 2010.

Farkuh Neto, A.; Lima, R. S. . **Roteirização de veículos de uma rede atacadista com o auxílio de Sistemas de Informações Geográficas (SIG)**. XXV Encontro Nac. de Eng. de Produção, Porto Alegre, RS, Brasil. 2005.

Feo, T. A.; Resende, M. G. C.. **A probabilistic heuristic for a computationally difficult set covering problem**. Operations Research Letters, Vol. 8, pp. 67–71. 1989.

Garey, M. E., Johnson, D. S.. **Computers and intractability: a guide to the theory of NP-completeness**. Freeman, San Francisco. 1979.

Gavett, J. W.. **Three heuristic rules for sequencing jobs to a single production facility**. Management Science, Vol. 11B, pp. 166-176. 1965.

Gemael, C.. **Introdução ao ajustamento de observações: aplicações geodésicas**. Curitiba: Editora da UFPR. 1994.

Glover, F.. **Ejection chains, reference structures and alternating path methods for traveling salesman problems**. Discrete Applied Mathematics, Vol. 65, pp. 223–253. 1996.

Gonçalves, D. S.. **GRASP e Otimização Global Contínua**. Departamento de Matemática Aplicada, IMECC, UNICAMP. 2009. Disponível na Internet em http://www.ime.unicamp.br/~chico/mt852/douglas.pdf, acessado em 29 Julho de 2012.

Gomes, R. F. S.. **Aplicação da Meta-Heurística Tabu Search na otimização de rotas de manutenção preventiva em campo**. Universidade Federal do Ceará. Dissertação (Mestrado em Engenharia de Sistemas Logísticos). 2011.

Gonnet, G.; Korotensky, C.; Benner, S.. **Evaluation measures of multiple sequence alignments**. In: Journal of Computational Biology 7(1-2), 261-276. 2000.

Grötschel, M.. **Polyedrische Charakterisierungen kombinatorischer Optimierungs-probleme**. Anton Hain Verlag, Meisenheim/Glan. 1977.

Guimarães, M. A. N.. **Reconfiguração de sistemas de distribuição de energia elétrica utilizando algoritmos de Busca Tabu**. Universidade Estadual de Campinas. Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação. 2005.

_____. **Plataforma integrada para o planejamento de sistemas de distribuição de energia elétrica utilizando Meta-Heurísticas**. Universidade Estadual de Campinas. Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação. 2009.

Held, M.; Karp, R.M. . **The traveling-salesman problem and minimum spanning trees: part II**. Mathematical Programming, Vol. 1, pp. 6-25. 1971.

Helsgaun, K. . **General k-opt submoves for the Lin-Kernighan TSP heuristic**. Mathematical Programming Computation, Vol. 1(2-3), pp. 119–163. 2009.

Hillier, F. S.; Lierberman, G. J.. **Introdução à pesquisa operacional**. 8ª. ed. Porto Alegre: AMGH, 2010.

Hoffman, A. J.; Wolfe, P.. **History in The Traveling Salesman Problem.** Lawler, Lenstra, Rinooy Kan and Shmoys, eds., Wiley, 1-16, 1985.

Hoffman, K. ; Padberg, M. . **Traveling Salesman Problem**. Karla Hoffman home page, Encyclopedia Articles. Disponível na Internet em http://iris.gmu.edu/ %7Ekhoffman/papers/trav_salesman.html, acessado em 21 maio de 2012.

IBGE. **Introdução a Geodesia**. Disponível em: <http://http://www.ibge.gov.br/home/ geociencias/geodesia/>. Ultimo acesso em outubro 2012.

Ignacio, A. A. V. ; Ferreira Filho, V. J. M. ; Galvão, R. D. . **Métodos heurísticos num entorno paralelo**. In: Simpósio Brasileiro de Pesquisa Operacional, 32. Anais. Viçosa: Ufv, pp. 769-788. 2000.

Jfinger, M.; Reinelt, G.; Thienel, S.. **Provably good solutions for the traveling salesman problem**. Zeitschrift für Operations Research, Vol. 40, pp. 183-217. 1994.

Karg, R.L.; Thompson, G.L. . **A Heuristic Approach to Solving Travelling Salesman Problems**. Management Science, Vol. 10, pp.225-248. 1964.

Karp, R. . **Reducibility among combinatorial problems**. In: Miller, R.E.; Thatcher, J.W. (Ed.) *Complexity of Computer Computations*. Nova York: Plenum Press, 1972, p. 85-193.

Kendall, G.; Li, J. . **Competitive travelling salesmen problem: A hyper-heuristic approach**. In: Journal of the Operational Research Society, Vol. 64, 208–216. 2012.

Keller, R. E.; Poli, R. . **Cost-Benefit Investigation of a Genetic-Programming Hyperheuristic**. Artificial Evolution, Lecture Notes in Computer Science, Vol. 4926, pp 13-24. 2008.

Kirkpatrick, S.; Gelatt, C.C.; Vecchi, M.P. . **Optimization by Simulated Annealing**. Science 220, pp. 671–679. 1983.

Korte, B. H. **Applications of combinatorial optimization**. In: Mathematical Programming:Recent Developments and Applications, M. Iri and K. Tanabe, Eds., Kluwer, Dordrecht, p.1-55. 1989.

Kötzing, T.; Neumann, F.; Röglin, H.; Witt, C.. **Theoretical Properties of Two ACO Approaches for the Traveling Salesman Problem**. Swarm Intelligence, Lecture Notes in Computer Science, Vol. 6234, pp. 324-335. 2010.

Kurz, M. E. . **Heuristics for the Traveling Salesman Problem**. Wiley Encyclopedia of Operations Research and Management Science. 2011.

Land, A.H.; Doig, A.G.. **An automatic method for solving discrete programming problems**. Econometrica 28, 497–520. 1960.

Laporte, G.. **The Traveling Salesman Problem: An overview of exact and approximate algorithms**. European Journal of Operational Research, Vol. 59(2), pp. 231-247, 1992.

Lawler, E. L.; Rinnooy-Kan, A. H. G.; Lenstra, J. K.; Shmoys, D. B.. **The Traveling salesman problem: a guided tour of combinatorial optimization.** [S.l.]: Edição de Wiley, 1985.

Lin, S.. **Computer solutions of the traveling-salesman problem**. Bell System Technology Journal, Vol. 44, pp. 2245-2269. 1965.

López-Ibáñez, M.; Blum, C.. **Beam-ACO for the travelling salesman problem with time windows**. Computers & Operations Research, Vol. 37, pp. 1570–1583. 2010.

Lisboa, J.; Iochpe, C.; Borges, K. A. V.. **Reutilização de Esquema de Banco de Dados em Aplicações de Gestão Urbana**. I Latin American Conference on Pattern Languages of Programming (Sugar LoafPLOP) , Rio de Janeiro. 2001.

Lucchesi, C. L. . **Introdução à Teoria dos Grafos**. IMPA-CNPq, Rio de Janeiro. 1979.

Maredia, A.. **History, Analysis, and Implementation of Traveling Salesman Problem (TSP) and Related Problems**. Department of Computer and Mathematical Sciences. University of Houston-Downtown. 2010.

Marinakis, Y.; Migdalas, A.; Pardalos, P. M.. **Expanding neighborhood search–GRASP for the probabilistic traveling salesman problem**. Optimization Letters, Vol. 2, Issue 3 , pp. 351-361. 2008.

Martin, O.; Otto, S.W.; Felten, E.W. . **Large-step Markov chains for the traveling salesman problem**. Complex Systems, Vol. 5 , pp. 299–326. 1991.

Menger, K. . **Das Botenproblem**. In: K. Menger, ed., Ergebnisse eines Mathematischen Kolloquiums 2, 11–12. 1932.

Merz*, P.* **Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies**. Parallel Systems Research Group. Department of Electrical Engineering and Computer Science. University of Siegen. Ph.D. Thesis. 2000.

Mestria, M. **Meta-Heurísticas Híbridas para a Resolução do Problema do Caixeiro Viajante com Grupamentos**. Universidade Federal Fluminense. Tese de Doutorado (Instituto de Computação). 2011.

NASA. **Gravity Anomaly Maps and The Geoid**. Disponível em: <http://earth observatory.nasa.gov/Features/GRACE/page3.php>. Ultimo acesso em outubro 2012.

Nguyen, H. D. ; Yoshihara, I.; Yamamori, K.; Yasunaga, M. . **Implementation of an Effective Hybrid GA for Large-Scale Traveling Salesman Problems**. In: Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on , Vol.37, no.1, pp. 92-99. 2007.

Oliveira, R. A.. **Desenvolvendo Aplicações Livres de Modelagem Objeto/Relacional com Banco de Dados Orientado a Objetos**. 2005.

OR-Library. In:http://people.brunel.ac.uk/~mastjjb/jeb/info.html

Padberg, M.; Rinaldi, G.. **Optimization of a 532-city symmetric traveling salesman problem by branch and cut**. Operations Research Letters, Vol. 6 , pp. 1-7. 1987.

Reeves, C. R. **Modern Heuristic Techniques for Combinatorial Problems,** McGraw-Hill, London, 1995.

Rei, Gerhard. **The Traveling Salesman: Computational Solutions for TSP Applications**. Springer-Verlag. 1994.

Rich, E.; Knight, K. **Inteligência Artificial**. 2. ed. São Paulo, Makron Books. 1993.

Ronen, D. **Perspectives on practical aspects of truck routing and scheduling**. European Journal of Operational Research, v.35, n.2, p.137-145, 1988.

Rosa, R.. **Cartografia Básica**. Universidade Federal de Uberlândia, Instituto de Geografia, Laboratório de Geoprocessamento. 2004. Disponível em: www.ig. ufu.br /lgeop/apostilas/Cartografia.pdf. Ultimo acesso em outubro de 2012.

Schepke, C.; Schopf, E. C.; Silva, M. L.; Silva, P. F.. **Um Estudo da Aplicação de Heurísticas Construtivas e de Melhoramento para um Problema de PRV**. In: III Simpósio de Informática da Região Centro do RS, 2004, Santa Maria. Anais - SIRC/RS 2004 - III Simpósio de Informática da Região Centro do RS, 2004.

Schneider, J.. **The time-dependent traveling salesman problem**, Physica A. . Vol. 314, pp. 151–155. 2002.

Silva, J. L. C.; Soma, N. Y.. **Uma heurística para problemas de otimização combinatória permutacional**. In: XXIII Simpósio Brasileiro de Pesquisa Operacional (SBPO), Campos do Jordão-SP, pp.1298-1306. 2001.

Silva, J. L. C.; Soma, N. Y.; Viana, G. V. R.. **Um algoritmo genético híbrido construtivo para problemas de otimização combinatór ia permutacional**. In: XII CLAIO. Congreso Latinoamericano de Investigación de Operaciones, 2004, Havana, anales del CLAIO, 2004.

Silva, J. L. C. . **Notas de aula**. 2010.

Sirenko, S. I. . **Classification of heuristic methods in combinatorial optimization**. In: International Journal "Information Theories & Applications". Vol.16, No. 4. 2009.

Solomon, M. M. . **Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints**. Operations Research Vol. 35, pp. 254–265. 1987.

Sousa, A. S. . **Tomada de decisão fuzzy e Busca Tabu aplicada ao planejamento da expansão de sistemas de transmissão.** São Paulo. Escola de Engenharia de São Carlos. Dissertação (Mestrado em Engenharia Elétrica), 2009.

Subramanian, A.; Penna, P. H. V.; Uchoa, E.; Ochi, L. S.. **A Hybrid Algorithm for the Fleet Size and Mix Vehicle Routing Problem**. International Conference on Industrial Engineering and Systems Management. 2011.

Süral, H. ; Özdemirel, N. E.; Önder, Ý.; Turan, M. S. . **An Evolutionary Approach for the TSP and the TSP with Backhauls**. Computational Intelligence in Expensive Optimization Problems. Adaptation Learning and Optimization Volume 2, pp 371-396. 2010.

Szwarccfiter, J. L. . **Grafos e Algoritmos Computacionais**. Editora Campus Ltda. 1984.

Toriello, A. . **A Dynamic Traveling Salesman Problem with Stochastic Arc Costs**. In: INFORMS Annual Meeting. Phoenix, Arizona. 2012.

Tsai, C.; Tsai, C.; Tseng, C. . **A new hybrid heuristic approach for solving large traveling salesman problem**. Information Sciences-Informatics and Computer Science: An International Journal, Vol.166, n.1-4, p.67-81. 2004.

TSP-Library. In: *http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.*

Viana, Gerardo V. R. **Meta-heurísticas e programação paralela em otimização combinatória**. Fortaleza: UFC, 1998.

Vittes, Fernando J. **Optimizing the perfomace of a chip shooter machine**. Faculty of the Virginia Polytechnic Institute and State University. Dissertação (Master of Science In Industrial and Systems Engineering). 1999.

Whitley, D. ; Hains, D. ; Howe, A. . **A Hybrid Genetic Algorithm for the Traveling Salesman Problem Using Generalized Partition Crossover**. Parallel Problem Solving from Nature, PPSN XI Lecture Notes in Computer Science, Vol. 6238, pp. 566-575. 2010.

Wolsey, L.A. **Integer Programming**. John Wiley & Sons, 1998.