UNIVERSIDADE FEDERAL DE MINAS GERAIS

DOUGLAS MOURA MIRANDA

METAHEURÍSTICAS PARA AS VARIANTES DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS: CAPACITADO, COM JANELA DE TEMPO E COM TEMPO DE VIAGEM ESTOCÁSTICO.

Dissertação apresentada à Escola de Engenharia da Universidade Federal de Minas Gerais para obtenção do título de Mestre em Engenharia de Produção.

Belo Horizonte
Setembro, 2011

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHEARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

DOUGLAS MOURA MIRANDA

METAHEURÍSTICAS PARA AS VARIANTES DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS: CAPACITADO, COM JANELA DE TEMPO E COM TEMPO DE VIAGEM ESTOCÁSTICO.

Dissertação apresentada à Escola de Engenharia da Universidade Federal de Minas Gerais para obtenção do título de Mestre em Engenharia de Produção.

Orientador: Prof. Samuel Vieira Conceição.

MESTRADO EM ENGENHARIA DE PRODUÇÃO

LINHA DE PESQUISA: SISTEMAS DE PRODUÇÃO E LOGÍSTICA

Belo Horizonte
Setembro, 2011

DOUGLAS MOURA MIRANDA

METAHEURÍSTICAS PARA AS VARIANTES DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS: CAPACITADO, COM JANELA DE TEMPO E COM TEMPO DE VIAGEM ESTOCÁSTICO.

Esta dissertação foi julgada e aprovada para a obtenção do grau de Mestre em Engenharia de Produção no Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Minas Gerais

.

Belo Horizonte, 13 de Setembro de 2011

BANCA EXAMINADORA

Prof. Dr. Samuel Vieira Conceição Universidade Federal de Minas Gerais Orientador	Prof. Dr. João Antônio Vasconcelos Universidade Federal de Minas Gerais
Prof. Dr. Leonardo Pereira Santiago Universidade Federal de Minas Gerais	Prof. Dr. Martín Gómez Ravetti Universidade Federal de Minas Gerais

AGRADECIMENTOS

Faço questão de agradecer a algumas pessoas e entidades que de algum modo tiveram importante contribuição para a concretização deste trabalho:

Deus, pela graça, amor e cuidado.

Minha querida esposa Mariana, pela paciência e apoio incondicional.

Meus pais (Celso e Joana) e minha irmã (Débora) por me apoiarem e desejarem o melhor pra mim.

Meu orientador (Samuel) por me motivar, instruir e confiar em meu trabalho.

Meus professores, pelos conhecimentos adquiridos.

Finamori e Djair da CIPI, pela maestria e sabedoria em seus conselhos.

Amigos que fiz durante o mestrado, em particular, o Bruno e o Juliano, pelo apoio no decorrer do curso e por terem me apresentado ao tênis.

Colegas de curso, pela convivência e trocas de experiências.

FAPEMIG pela bolsa de estudos concedida.

RESUMO

A atribuição e o planejamento de rotas de veículos é um problema crucial da administração de cadeias de suprimentos. No ambiente real é comum encontrar problemas que envolvam uma quantidade muito grande de clientes e que conseqüentemente fogem do alcance de métodos exatos. Neste contexto, este trabalho visa desenvolver metaheurísticas capazes de resolver algumas das mais importantes variantes do problema de roteamento de veículos (PRV): o PRV capacitado, o PRV capacitado com máxima distância e o PRV com janelas de tempo. As metaheurísticas desenvolvidas combinam a força de estratégias bem sucedidas na literatura como *Tabu Search, Guided Local Search* e *Adaptive Memory Procedure* dentro de uma estrutura que utiliza o *Iterated Local Search* e o *Variable Neighborhood Descent*.

O ambiente real também possui dados probabilísticos por natureza, como o tempo de viagem entre dois clientes. Isto faz com que um modelo de roteamento que considere as incertezas envolvidas nestes dados seja mais apropriado. Neste contexto, o presente trabalho também desenvolve uma metaheuristica para resolver um PRV com janela de tempo no qual o tempo de viagem entre os clientes é conhecido apenas probabilisticamente. Este problema é conhecido como PRV com Janelas de Tempo e Tempo de Viagem Estocástico. Um método inédito na literatura é desenvolvido não só para estimar o tempo de chegada nos clientes, mas também para calcular a probabilidade dos veículos atenderem os clientes dentro de suas respectivas janelas de tempo. O algoritmo encontra a rota de menor custo e ao mesmo tempo garante um nível mínimo de serviço aos clientes. Simulação Estocástica é utilizada para mostrar que o método proposto supera outros conhecidos métodos da literatura.

Palavras-chaves. Roteamento de Veículos, Tempo de Viagem Estocástico, Janela de Tempo, Busca Tabu, *GLS*, *ILS*, Memória Adaptativa.

ABSTRACT

The assigning and the planning of vehicle's routes is a crucial problem in the supply chain management. In the real-life environment it is common to find problems in which there is a large number of customers. Those problems can not be solved by exact methods in reasonable time. In this context, this work aims to develop metaheuristics that are able to solve some of the most important versions of the vehicle routing problems (VRP): the Capacitated VRP, the Capacitated VRP with maximum distance and the VRP with Time Windows. The developed metaheuristics combine the strengths of the successful strategies such as Tabu Search, Guided Local Search and Adaptive Memory Procedure into an Iterated Local Search and Variable Neighborhood Descent structure.

The real-life environment is also made of probabilistic data by nature, as the travel time between two customers. This fact makes the models that consider the environment uncertain more appropriate. In this context, the present work develops a metaheuristic capable to solve a VRP with Time Windows in which the travel time among the customers is known just probabilistically. This problem is known as Stochastic Vehicle Routing Problem with Time Windows (SVRPTW). This problem combines stochastic travel times with VRP with Time Windows (VRPTW). A methodology is developed to estimate the vehicle arriving time at each customer location and also to estimate the vehicle`s probability to respect the customer`s time window. To our knowledge, this method is unprecedented in the literature. The algorithm finds the best route with minimum expected cost while it guarantees that certain levels of service are met. Simulation results are used to demonstrate that the proposed methodology outperformed other recent methods in the literature.

Keywords. Vehicle Routing, Stochastic Travel Time, Time Windows, Tabu Search, Guided Local Search, Adaptive Memory Procedure, Iterated Local Search, Variable Neighborhood Descent.

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

1.1- CONTEXTUALIZAÇÃO E MOTIVAÇÃO

Em um mundo empresarial globalizado, as empresas que trabalham com distribuição própria de seus produtos ou prestam serviços de entrega ou coleta de mercadorias de qualquer tipo enfrentam diversos desafios como: elevada concorrência, necessidade de lidar com um elevado número de clientes com baixo consumo e pequenos volumes de entrega e, ainda, clientes mais exigentes tanto com a qualidade como também com o prazo de entrega dos produtos.

Para serem competitivas neste mercado, convém que as empresas ofereçam aos seus clientes características diferenciadoras, adotando processos mais rápidos e adaptados ao perfil do cliente. Groër *et al.* (2009) mencionam que para obter tais características, algumas empresas oferecem serviços como: rastreamento on-line de produtos, garantia de entrega no prazo acordado e de coleta dos produtos.

Ao longo do tempo, diversas iniciativas neste setor tentam otimizar a utilização dos veículos e rotas para a conseqüente minimização dos custos de distribuição. Este tipo de problema é estudado como o Problema de Roteamento de Veículos, ou simplesmente, PRV.

Cheng & Wang (2009) consideram que o PRV tem um impacto econômico vital em um centro de distribuição. Na prática, a resolução deste problema contribui diretamente para redução de custos no sistema logístico. Como o problema é complexo, a solução geralmente fica nas mãos de uma pessoa experiente com conhecimento geográfico do local e rota.

Segundo Toth & Vigo (2002), por mais experiente que a pessoa responsável pelo planejamento dos processos de distribuição seja, procedimentos computadorizados têm permitido uma importante redução dos custos totais de transporte. Kant *et al.* (2008) apresentaram em seu trabalho um estudo de caso sobre a implementação de um software de roteamento de veículos na Coca-Cola ® nos Estados Unidos da América. Este

projeto resultou em ganhos significativos para a empresa. O desafio foi a elaboração de um programa que resolvesse um PRV com restrições padrões, tais como, capacidade do veículo e duração máxima da rota; e também algumas restrições mais complexas como: veículo e equipamento adequado conforme pedido e localização do cliente, janela de tempo e tipo de tráfego para evitar engarrafamentos.

O PRV é uma variação do Problema do Caixeiro Viajante (PCV), um dos problemas mais abordados na área da Otimização Combinatória (Laporte *et al*., 1992), onde os problemas apresentam a característica comum de que a solução ótima é procurada em um conjunto finito de possibilidades. Segundo Solomon & Desrosiers (1998), este problema ainda não possui um algoritmo capaz de solucioná-lo de maneira exata para um elevado número de clientes, sendo difícil encontrar a solução ótima em tempo computacional não proibitivo.

O desenvolvimento dos computadores, tanto em hardware como em softwares, têm sido um fator importante no desenvolvimento de modelos matemáticos e de algoritmos que cada vez mais têm abordado aplicações do mundo real no que diz respeito aos problemas de roteamento.

Deste modo, estamos num contexto que possui algumas características que motivam este trabalho. A primeira é que embora o PRV tenha cerca de 50 anos, por ser um problema de alta complexidade computacional e com muitas variações possíveis, está longe de ser um tema esgotado. Um segundo ponto trata-se de uma variante do problema ainda pouco estudada e de forte relevância prática que diz respeito à natureza estocástica dos tempos de viagem. Uma terceira característica é o fato do PRV ter um método de solução complexo que está fora do alcance de compreensão de muitos profissionais que trabalham em empresas de logística, e portanto, sua aplicação prática num ambiente real depende fortemente do uso da tecnologia de informação.

1.2- OBJETIVOS

Dentro do contexto já descrito, este trabalho possui os seguintes objetivos.

1.2.1- Objetivo geral

Desenvolver uma metaheurística capaz de resolver eficientemente o Problema de Roteamento de Veículos com Janela de Tempo e Tempo de Viagem Estocástico.

1.2.2- Objetivos específicos

Para suportar o cumprimento do objetivo geral, são propostos os seguintes objetivos específicos:

1) Desenvolver metaheurísticas eficientes a partir de técnicas e algoritmos relatados na literatura que obtiveram reconhecido sucesso na solução do PRV determinístico: *Iterated Local Search, Variable Neighborhood Descent, Tabu Search, Memory Adaptive Procedure* e *Guided Local Search.*

2) Validar os modelos desenvolvidos em instâncias reconhecidas da literatura.

3) Estudar novas formas de exploração do espaço de busca.

4) Transformar o algoritmo desenvolvido para o PRV determinístico num algoritmo capaz de resolver a versão estocástica do problema no que diz respeito ao tempo de viagem.

5) Propor novos métodos para resolver o PRV com Tempo de Viagem Estocástico.

Convém esclarecer que o foco principal desta dissertação está no desenvolvimento de metaheurísticas dentro do contexto da Pesquisa Operacional. Durante a execução do presente trabalho o autor teve interesse em estudar como fazer a integração de um modelo de roteamento aos processos de negócios de uma empresa real de distribuição. Este estudo foi separado do presente trabalho e especificamente abordado no trabalho intitulado "Uso da modelagem de processo de negócio para integração de solução de roteamento em uma empresa de distribuição" por Pessoa & Miranda (2011), delineado como um estudo de caso. Trata-se de um trabalho de conclusão de curso orientado pelo autor desta dissertação em que ele implementou a

heurística de roteamento e o aluno desenvolveu o banco de dados e a interface com o usuário.

1.3- CONTRIBUIÇÃO CIENTÍCA DO ESTUDO

Este trabalho faz-se relevante pelos seguintes motivos:

1) Integra de modo eficiente algumas das técnicas de maior sucesso utilizadas para resolver o PRV Capacitado e o PRV com Janela de Tempo.
2) Investiga com profundidade as dificuldades de resolver adequadamente o PRV com Janela de Tempo e com Tempo de Viagem Estocástico.
3) Identifica uma lacuna na literatura quanto ao modo de solução do PRV com Tempo de Viagem Estocástico e desenvolve um método inédito para preencher parte desta lacuna e gerar uma contribuição às pesquisas já existentes.

1.4- METODOLOGIA

Esta sessão apresenta a forma como foi executada a pesquisa e a metodologia adotada. A presente pesquisa é classificada, sob o ponto de vista do delineamento adotado, como pesquisa experimental, uma vez que experimentos foram realizados para testar os algoritmos manipulando-se variáveis para criar situações de interesse com o propósito de analisar e avaliar os métodos desenvolvidos. A pesquisa é de natureza quantitativa, uma vez que as variáveis envolvidas foram quantificadas segundo um escala de intervalo ou de razão. Do ponto de vista do seu objetivo, o presente estudo pode ser classificado como uma pesquisa explicativa, o qual buscou aprofundar-se no conhecimento dos fenômenos, identificar relações de causa e efeito, descobrir o motivo de dificuldades e propor soluções.

Os procedimentos técnicos envolveram o estudo da arte, baseado em pesquisa bibliográfica sobre o tema do estudo e outras etapas descritas a seguir:

1) Estudo dos Problemas de Roteamento de Veículos.
2) Estudo com foco nos métodos existentes para resolver as variantes de interesse do problema.
3) Selecionar os métodos a serem implementados.

4) Implementar e testar as metaheurísticas utilizadas para resolver o VRP Capacitado Determinístico.

5) Implementar e testar as metaheurísticas utilizadas para resolver o VRP com Janela de Tempo Determinístico.

6) Implementar e testar as metaheurísticas e os métodos utilizados para resolver o VRP com Janela de Tempo e Tempo de Viagem Estocástico.

A ferramenta de programação escolhida para implementar os algoritmos deste trabalho foi o Matlab. Embora seja conhecido que o tempo computacional deste software seja maior que o de outras linguagens, o foco deste trabalho está na qualidade das soluções. Além disto, o Matlab tem benefícios importantes como maior agilidade na programação, no uso de gráficos e de ferramentas estatísticas. Isto mostrou-se importante para entender e implementar os conceitos com que esta pesquisa deseja trabalhar. Uma heurística foi implementada em Delphi, num caso específico, descrito ao longo do trabalho, em que o custo computacional foi considerado importante.

1.5- ORGANIZAÇÃO DO ESTUDO

O trabalho divide-se em 9 capítulos. No capítulo 1 são abordados aspectos relativos à contextualização e apresentação do problema de pesquisa, relevância, contribuição científica, justificativa da pesquisa e metodologia utilizada.

Os capítulos 2, 3 e 4 são destinados à revisão da literatura de modo a trazer um estudo da arte sobre os assuntos envolvidos na pesquisa, tratando respectivamente os seguintes problemas: PRV Capacitado, PRV com Janela de Tempo e PRV Estocástico.

Os capítulos 5 e 6 abordam os algoritmos desenvolvidos no trabalho. O capítulo 5 diz respeito aos algoritmos destinados a resolver o PRV Capacitado e o PRV com Janela de Tempo, ambos determinísticos. O capítulo 6 é destinado a explicar os algoritmos desenvolvidos para resolver o PRV com Janela de Tempo e Tempo de Viagem Estocástico.

Os capítulos 7 e 8 mostram os experimentos realizados e resultados obtidos. O primeiro diz respeito aos problemas determinísticos e o segundo refere-se ao problema estocástico.

Finalmente, no capítulo 9, apresenta-se as conclusões finais da pesquisa, a síntese dos resultados, a adequação aos objetivos propostos e recomendações para trabalhos futuros.

Esclarecemos que o objetivo principal do trabalho está no PRV Estocástico. Porém, antes de abordar este problema foi preciso também estudar o PRV Determinístico já que os mecanismos de melhoria das soluções são basicamente os mesmos. A organização do estudo dedica capítulos específicos para o problema determinístico e estocástico e convém que o leitor esteja ciente de que o assunto abordado no problema determinístico é bastante diferente do estocástico.

1.6- DELIMITAÇÃO DA PESQUISA

Todos os algoritmos implementados neste trabalho assumem a existência de um único depósito. Também são assumidos que a frota é homogênea, a distância entre os clientes é euclidiana e simétrica.

Nos algoritmos que resolvem o PRV com Janela de Tempo e Tempo de Viagem Estocástico, o tempo de viagem entre os clientes é o único parâmetro estocástico e assume-se que ele tenha função densidade de probabilidade normal.

CAPÍTULO 2

PROBLEMA DE ROTEAMENTO DE VEÍCULOS (PRV)

Considere que uma frota de veículos esteja disponível para o transporte de mercadorias demandadas ou ofertadas por um conjunto de clientes. Considere que cada veículo esteja inicialmente situado em um depósito. O Problema de Roteamento de Veículos (PRV) consiste em determinar a rota a ser percorrida por cada veículo, de modo que a demanda de todos os clientes seja satisfeita, e que cada veículo regresse ao depósito de origem ao final do período considerado. O objetivo é minimizar o custo total, definido pela soma dos custos das rotas.

O Problema de Roteamento de Veículos Capacitado (PRVC) é a versão mais conhecida do problema. Nela, cada cliente possui demanda determinística, ou seja, conhecida previamente, e que deve ser atendida integralmente por apenas um veículo. Todos os veículos são semelhantes e partem de um único depósito. Somente uma restrição de capacidade é imposta ao problema. Essa restrição estabelece que a soma das demandas de todos os clientes pertencentes a uma rota não deve superar a capacidade do veículo a ela designado.

Lenstra (1981) faz um estudo da complexidade computacional de resolver-se o PRV. O autor mostra que como uma generalização do Problema do Caixeiro Viajante (PCV), o PRV pertence à classe de problemas NP - difícil e um algoritmo em tempo polinomial para encontrar a solução ótima não é conhecido. Para uma análise aprofundada sobre complexidade computacional, fazemos referência a Garey & Jonhson (1979) e Ziviani (2002).

Segundo Laporte (2009), o PRV foi introduzido 50 anos atrás por Dantzig & Ramser (1959) com o titulo "*The truck dispatching problem*". Alguns anos depois Clarke & Wright (1964) propuseram uma heurística gulosa que melhorou a abordagem de Dantzig & Ramser. Seguindo estes 2 trabalhos, considerados pioneiros na modelagem do problema, muitos modelos exatos e heurísticos foram propostos para encontrar soluções ótimas e aproximadas para o PRV.

Um critério de classificação do problema é visto em Desrochers *et al.* (1990). Laporte & Nobert (1987) apresentam uma extensa revisão da literatura totalmente

dedicada a métodos exatos. Outras revisões da literatura, agora abordando modelos aproximados, foram apresentadas por Christofides *et al.* (1979), Fisher (1995), Toth & Vigo (1998) e Golden *et al.* (1998).

Bergeglia *et al*. (2010) conclui em seu trabalho que o volume de artigos publicados sobre o PRV tem aumentado significativamente na última década. Apesar disto, ainda permanecem questões em aberto sobre o assunto, em virtude principalmente das inúmeras variações que o problema possui para se aproximar das situações do mundo real.

O PRVC deu origem a diversos outros problemas, motivo pelo qual iremos estudá-lo em primeiro lugar, apresentando em seguida algumas de suas variações.

2.1 – PRV CAPACITADO (PRVC)

Baseando-se em Vieira (2009), o PRVC pode ser definido da seguinte forma: seja um grafo $G = (V, A)$ completo, em que *A* é um conjunto de arcos, que representam os caminhos que ligam os clientes entre si e estes ao depósito, e $V = 0, \dots, n$ denota um conjunto de *n+1* vértices. Convencionamos que o vértice *0* representa o depósito e os outros simbolizam os clientes. A cada arco *(i, j)* é associado um custo não negativo, c_{ij}, que representa o custo de viagem do vértice *i* ao vértice *j*. Neste trabalho, c_{ij}é dado pela distância euclidiana entre os clientes.

Na maioria dos casos, os custos dos arcos satisfazem a desigualdade triangular, $c_{ik} + c_{kj} \geq c_{ij}$ para $i, k, j \in V$. Quando o custo do arco *(i, j)* é igual ao custo do arco *(j, i),* dizemos que o problema é simétrico. Caso contrário, ele é dito assimétrico.

Seja um conjunto *K* de veículos idênticos, com capacidade *C*, onde $k \in K$. A cada cliente *i* é associado uma demanda não negativa, m_i. Para o depósito, definimos $m_0 = 0$.

O PRVC consiste em encontrar um conjunto de rotas, cada uma percorrida por um veículo, de modo a minimizar o custo total de transporte e a satisfazer as seguintes restrições:

1) Cada rota deve ter início e fim no depósito;
2) Cada cliente deve ser visitado apenas uma vez e somente por um veiculo;

3) A soma das demandas dos clientes incluídos em uma rota não deve exceder a capacidade do veículo.

Para definir o problema, utilizamos as variáveis binárias de decisão, dadas por:

$x_{ijk} = 1$, se o veículo k perfaz o arco *(i, j)* ou $x_{ijk} = 0$, caso contrário; para $i, j \in \{0, 1, \dots, n\}, i \neq j$ e $k \in \{1, \dots, K\}$. A formulação matemática do PRVC é apresentada abaixo.

$$Min\ M \sum_{k \in K} \sum_{j=1}^{n} x_{0jk} + \sum_{i=0}^{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} \sum_{k \in K} c_{ij} x_{ijk} \tag{2.1}$$

$s.a$:

$$\sum_{j=1}^{n} x_{0jk} = \sum_{j=1}^{n} x_{j0k} = 1;\ \ k = 1, \dots, K \tag{2.2}$$

$$\sum_{k \in K} \sum_{j=1}^{n} x_{ijk} = 1;\ i = 1, \dots, n \tag{2.3}$$

$$\sum_{j=0}^{n} x_{ijk} - \sum_{j=0}^{n} x_{jik} = 0;\ \ k = 1, \dots, K;\ i = 1, \dots, n \tag{2.4}$$

$$\sum_{k \in K} \sum_{i \in S} \sum_{j \in S} x_{jik}\ \leq |S| -\ v(S);\ \forall\ S\ \subseteq V \backslash \{0\};\ |S| \geq 2 \tag{2.5}$$

$$\sum_{i=1}^{n} m_i \sum_{\substack{j=0 \\ j \neq i}}^{n} x_{ijk} \leq C;\ k = 1, \dots, K \tag{2.6}$$

$$x_{ijk} \in \{0,1\};\ i = 1, \dots, n\ ;\ \ j = 1, \dots, n\ ;\ \ k \in K \tag{2.7}$$

Na equação (2.1), *M* é um número muito grande para forçar a função objetivo a priorizar a redução do número de veículos. A equação (2.2) assegura que cada rota tenha início e fim no depósito. As restrições (2.3) e (2.4) garantem que cada cliente seja visitado exatamente uma vez, e que o veículo que chega a um cliente saia dele. A restrição (2.5) evita que seja formado um subciclo que não inclua o depósito. Nesta restrição, *v(S)* representa o número mínimo de veículos necessário para atender o conjunto de clientes *S*.

Por exigir que o número de veículos usados para atender os clientes do conjunto *S* não seja inferior a *v(S)*, o mínimo necessário, a família de restrições (2.5) assegura, indiretamente, que a capacidade dos veículos não seja violada. Entretanto, para dar maior clareza à formulação do problema e facilitar as referências futuras, preferiu-se manter explícita a restrição de capacidade dos veículos, motivo pelo qual introduziu-se (2.6).

O problema capacitado de roteamento possui várias outras formulações, muitas das quais incluindo apenas n^2 variáveis. De fato, pode-se notar que, como todos os veículos possuem a mesma capacidade e como as rotas só se encontram no depósito, não é necessário indicar qual veículo percorrerá cada rota. Desta forma, poderia ser utilizado x_{ij} simplesmente, em lugar de x_{ijk}.

Cabe notar que a adoção da formulação (2.1)-(2.7), que tem mais variáveis e restrições do que o estritamente necessário, não provocará perda de eficiência nos algoritmos propostos, uma vez que não trabalharemos com um método exato, mas com uma metaheurística.

2.2 – EXTENSÕES DO PRVC

Percebe-se que o PRVC não é capaz de representar todas as situações cotidianas enfrentadas pelos setores de logística das empresas de distribuição de mercadorias e serviços. Assim, muitas vezes é necessário introduzir neste problema algumas restrições associadas aos clientes, aos veículos ou aos depósitos, para que ele reflita as características dos problemas reais.

Como exemplo, na prática, os clientes podem requerer que:

• diferentes tipos de produtos sejam entregues;

• apenas um determinado subconjunto dos veículos seja usado para serví-los, em função, por exemplo, de limitações de acesso, ou do tipo de mercadoria transportada;

• o serviço de carregamento ou descarregamento das mercadorias seja executado em um tempo predefinido;

• a entrega dos produtos seja feita em um período determinado do dia ou mês (janela de tempo), de maneira a respeitar os horários de funcionamento de seus

estabelecimentos, as limitações de tráfego e os prazos de entrega exigidos pelos consumidores por eles atendidos;

• produtos em excesso sejam devolvidos, quer por terem perdido a validade, como ocorre com jornais e revistas, quer por não terem sido bem recebidos pelos consumidores (exemplo: problemas de qualidade).

Os veículos, por sua vez:

• podem ser de diferentes tipos, ou possuírem capacidades diferentes;

• podem ser subdivididos em compartimentos, permitindo o transporte de diferentes produtos em diversas quantidades;

• podem suportar um tempo máximo de operação, antes de serem submetidos a revisão.

Além disso, é possível que exista mais de um depósito, de modo que:

• a rota de cada veículo possa ter início e fim em um depósito específico, diferente do utilizado por outros veículos;

• cada veículo possa partir de um depósito e terminar sua rota em outro depósito.

Muitos objetivos, freqüentemente conflituosos, podem ser considerados para o problema. Alguns típicos são (Toth & Vigo, 2002b):

- Minimização do custo total de transporte.
- Minimização do número de veículos requeridos para servir todos os clientes.
- Balanceamento de rotas, seja em relação ao tempo de viagem ou ocupação dos veículos.
- Minimização das penalidades associadas ao atendimento parcial dos clientes.

Em algumas aplicações é necessário considerar versões estocásticas ou dinâmicas do problema, ou seja, problemas em que a priori existe apenas um conhecimento parcial da demanda dos clientes ou do tempo de viagem entre clientes da rota.

Outras variantes do PRVC podem ser obtidas se considerarmos, por exemplo, que:

• o número de veículos utilizados não precisa ser previamente determinado (ou seja, há um grande número de veículos a disposição).

• a demanda é estocástica.

Descrevemos abaixo os principais problemas surgidos a partir da combinação de algumas dessas variações e bastante estudados na literatura:

1- *VRP with time Windows (VRPTW)*: O veículo deve chegar ao consumidor *i* dentro do intervalo [e_i, l_i]. Em muitos casos é permitido que o cliente seja visitado antes do *lower bound* e_i, existindo um tempo de espera até que a janela seja aberta. Algumas variações também permitem que o cliente chegue após o *upper bound* l_i, implicando então em alguma penalização a ser considerada na função objetivo. Mais detalhes deste tipo de problema serão vistos no capítulo 3.

2- *VRP with time deadlines* (*VRPTD*): é um caso especial do *VRPTW* onde as janelas de tempo são substituídas apenas por um limite máximo de atendimento. Mais precisamente, não existe o *lower bound* e_i para cada cliente *i*, apenas o *upper bound* l_i.

3- *Time-dependent VRP (TVRP):* visa modelar situações mais realísticas por considerar que o tempo de viagem entre 2 clientes depende não apenas da distância entre eles, mas também da hora do dia. Por exemplo, leva-se mais tempo para ir de um cliente para outro nos horários de pico do tráfego. O *TVRP* leva características como esta em consideração.

4- *Periodic VRP (PVRP):* o *VRP* periódico é uma variação do *VRP* onde o horizonte de tempo extende-se por um certo número de períodos. Rotas são construídas para cada período e cada cliente é visitado uma vez ou mais ao longo do horizonte de planejamento, conforme suas necessidades. Detalhes podem ser vistos em Cordeau (1997) e em Francis *et al.* (2008).

5- *VRP with heterogeneous fleet (VRPHF):* no *VRP* com frota heterogênea, os veículos não possuem necessariamente as mesmas características. Por exemplo, podem ter diferentes capacidades.

6- *Multiple Depot VRP (MDVRP):* no *VRP* com múltiplos depósitos existe mais de 1 depósito e os veículos podem começar e terminar suas rotas em quaisquer destes depósitos. Um importante estudo na literatura é visto em Renaud *et al.* (1996).

7- *VRP with backhauls (VRPB):* alguns clientes requerem entregas (*linehauls*) enquanto outros requerem carregamentos (*backhauls*). Cada rota é então um mix de clientes do tipo *linehauls* e do tipo *backhauls*. Percebe-se que uma importante redução de custo pode ser obtida se for permitido que os veículos vazios que retornam das entregas façam a coleta dos produtos. Na indústria alimentícia, por exemplo, supermercados e fornecedores seriam os clientes *linehauls* e *backhauls*, respectivamente. Uma Revisão bibliográfica é feita nos estudos de Goetschalckx & Jacobs (1989).

8- *VRP with simultaneous delivery and pick-up (VRPSDP):* no *VRP* com entrega e coleta simultâneas, um mesmo cliente pode solicitar que alguns produtos lhe sejam entreguem a partir do depósito e que outros produtos sejam coletados e levados ao depósito. O modelo é estudado em Min (1989). Uma poderosa modelagem heurística é vista em Pisinger (2007), onde um único modelo *VRPSDP* é capaz de resolver 5 diferentes tipos de *VRP* com ótimos resultados em instâncias conhecidas da literatura.

9- *Pick-up and delivery problem (PDP):* no problema de coleta e entrega, cada cliente tem uma localização de coleta *(i+)* e uma localização de entrega de produtos *(i-).* Ambos os locais i+ e i- devem pertencer à mesma rota, e necessariamente, o local *i+* deve ser visitado antes do local *i-*. É possível ainda existirem janelas de tempo para as coletas e entregas. Um estudo de referência é dado em Savelsbergh & Sol, M. (1995).

Os principais métodos encontrados na literatura para resolver os problemas de roteamento de veículos são divididos em duas grandes categorias: métodos exatos e métodos heurísticos. Os métodos exatos garantem a solução ótima do problema, mas, em geral, apresentam alto custo computacional. Os métodos heurísticos não garantem a solução ótima, mas geralmente resultam em soluções sub-ótimas de qualidade e um esforço computacional menor que os métodos exatos.

2.3 – TÉCNICAS EXATAS DE SOLUÇÃO PARA O PRVC

Desde a década de 60, muitos métodos exatos têm sido apresentados. Graças a estes esforços, o tamanho de problemas que possam ser resolvidos por métodos exatos têm aumentado significativamente. Embora o foco do presente trabalho seja em técnicas aproximadas (heurísticas), a seguir são descritos alguns dos métodos exatos. Para maior aprofundamento, uma extensa revisão da literatura é vista em Toth & Vigo (2002b).

2.3.1 – Algoritmos *Branch-and-Bound* para o PRVC

O método *Branch-and-Bound* tem sido usado freqüentemente nas últimas décadas para resolver o PRVC e algumas de suas variantes. Em muitos casos, como para o problema assimétrico e para o problema com distância máxima (*Distance-Constrained CVRP*) estes algoritmos representam a base de muitas pesquisas que abordam os métodos de solução exata. Numa extensa revisão bibliográfica focada em métodos exatos, Laport & Nobert (1987) fazem uma completa e detalhada análise dos algoritmos *branch-and-bound* propostos a partir da década de 80.

Mais recentemente, métodos baseados em Relaxações Lagrangeanas têm aumentado significativamente o tamanho dos problemas que podem ser resolvidos de forma exata. Em particular, para o VRP assimétrico (custo de ir de um cliente ao outro é diferente de perfazer o caminho inverso), excelentes resultados encontrados na literatura baseiam-se na técnica "*Additive Bounding*" proposta por Fischetti *et al.* (1994). Para o VRP simétrico (custo de ir de um cliente ao outro é o mesmo em ambas as direções), excelentes *bounds* são encontrados por Relaxações Lagrangeanas propostas por Fisher (1994) e por Miller (1995).

2.3.2 – Algoritmos *Branch-and-Cut* para o PRVC

Uma relaxação linear de um problema linear inteiro *IP* é o problema linear *LP* obtido pela remoção da condição de que todas as variáveis precisam ser inteiras. Então o valor ótimo *Zlp* da relaxação do problema linear é um limite inferior para o valor ótimo *Zip* do problema inteiro, ou seja, *Zlp* é menor que *Zip*.

Se o número de restrições de um problema linear inteiro é pequeno o suficiente para que sua relaxação linear possa ser inserida em um *solver LP*, um método clássico

para resolvê-lo é um *branch-and-bound*. Ou seja, primeiro resolve-se a relaxação linear. Se a solução ótima encontrada é inteira então esta é a solução. Caso contrário, arredonda-se o valor para cima e para baixo, obtendo 2 valores, então a partir destes, 2 novos programas lineares são construídos. Daí, procede-se por um clássico *branch-and-bound* no qual os limites são dados pelos valores ótimos dos programas lineares associados com os nós da árvore de busca.

Quando o número de restrições lineares do *IP* é grande, ou quando a relaxação linear é apertada adicionando-se algumas desigualdades válidas, que tipicamente possuem tamanho exponencial, então o sistema de restrições não pode ser colocado em um *solver* LP e uma técnica de plano cortante tem que ser usada para resolver o programa linear.

Branch-and-cut tem tido sucesso em resolver muitos problemas de otimização combinatória (ver Caprara & Fischetti, 1997), no entanto ele pode ter um desempenho ruim em algumas situações como: não se ter um bom algoritmo para executar a fase de plano cortante, o número de iterações da fase de plano cortante ser muito alta, o programa linear tornar-se insolúvel devido ao seu tamanho, a árvore gerada pelo procedimento *branching* tornar-se muito grande, entre outros.

Para uma compreensão mais ampla do método, bons estudos são feitos por Caprara & Fischetti (1997) e por Jtinger (1998). Outro estudo particularmente importante é Fukasawa *et al.* (2006) que aplicaram o método *branch-and-cut-and-price* para resolver o PRVC e conseguiram encontrar a solução ótima de instâncias com até 135 clientes, quantidade considerada alta na literatura.

2.4 – HEURÍSTICAS PARA O PRVC

Nicholson (1971) define heurística como um procedimento para resolver problemas através de um enfoque “intuitivo”, em geral racional, no qual a estrutura do problema possa ser interpretada e explorada inteligentemente para obter uma solução razoável. Reeves (1993) define heurística como uma técnica que busca boas soluções (perto da ótima) com um custo operacional razoável, sem garantir soluções ótimas e, em muitos casos, não é capaz de afirmar quão próximo uma solução factível está da solução ótima. Para Ribeiro (2004), uma heurística é qualquer método aproximado projetado com base nas propriedades estruturais ou nas características das soluções dos problemas,

com complexidade reduzida em relação à dos algoritmos exatos e fornecendo, em geral, soluções viáveis de boa qualidade. Sendo assim, os métodos heurísticos também podem ser chamados de métodos aproximados em oposição aos métodos exatos.

Embora importante progresso venha sendo feito com o desenvolvimento de novos métodos exatos, os maiores problemas que podem ser consistentemente resolvidos por estes métodos possuem cerca de 50 clientes, e instâncias maiores podem ser resolvidas apenas em casos particulares (Renaud & Boctor, 2002). Devido ao limitado sucesso de métodos exatos, pesquisas têm sido desenvolvidas no sentido de desenvolver heurísticas capazes de encontrar boas soluções para instâncias de maior escala.

Além da teoria da complexidade computacional representar uma forte justificativa para a utilização de métodos heurísticos na solução do PRV, outro forte argumento apresentado por Reeves (1993) corresponde à possibilidade de modelar o problema real com maior precisão, uma vez que as heurísticas são mais flexíveis e aptas a operar com funções objetivo e/ou restrições mais complicadas e mais realistas do que os algoritmos exatos.

Ribeiro (2004) cita 3 categorias para os métodos heurísticos: métodos construtivos, busca local e metaheurísticas. Laporte (2009) coloca as heurísticas construtivas, heurísticas de duas fases e os métodos de melhorias (buscas locais) dentro de uma categoria chamada de “Heurísticas Clássicas”.

Assim, dentro do contexto do PRV, para melhor entendimento, ilustramos estas categorias de métodos de solução na figura 2.1.

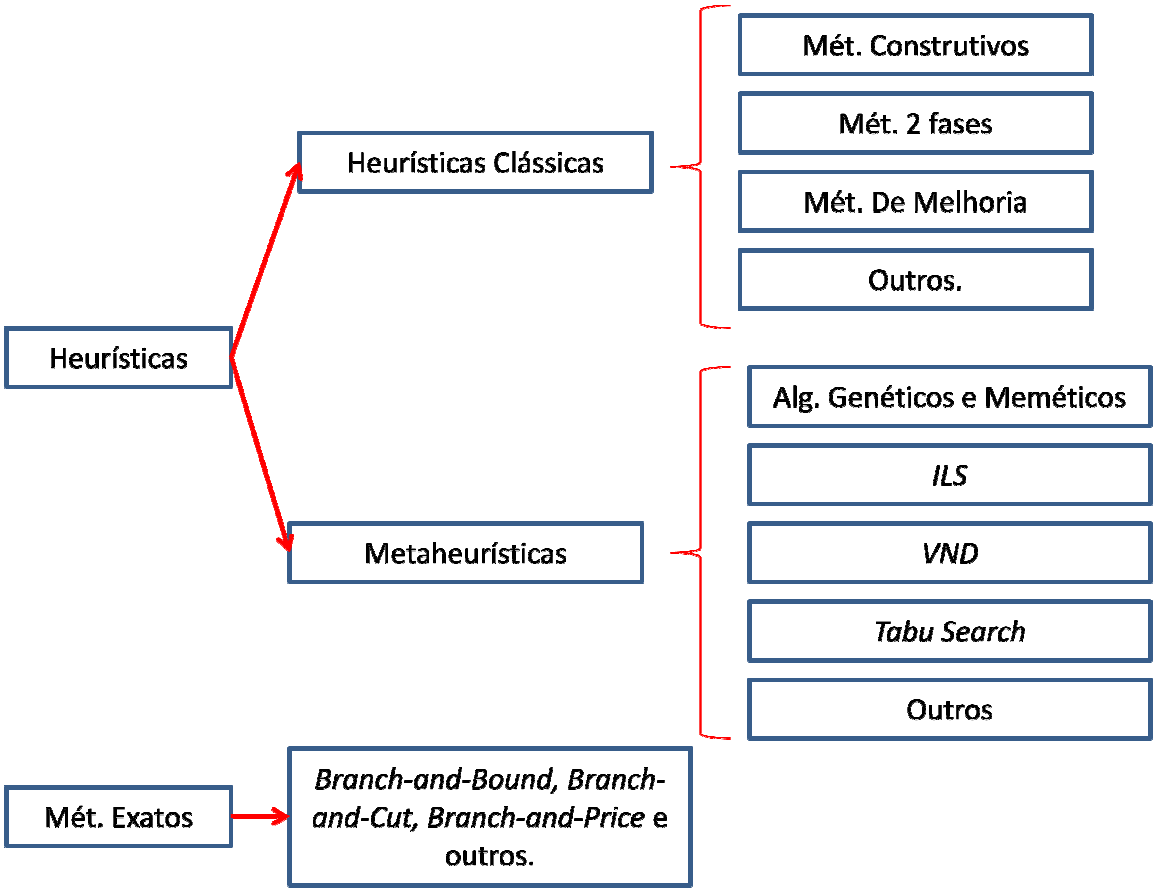


Figura 2.1: Categorias de Métodos de Solução para o PRV

2.4.1 – Heurísticas Clássicas para o PRVC

Desenvolvidas principalmente entre as décadas de 60 e 90, estas heurísticas exploram o espaço de busca de forma relativamente limitada e normalmente produzem soluções de boa qualidade e com tempos computacionais não proibitivos.

Tais heurísticas podem ser divididas em 3 categorias:

1) Heurísticas Construtivas: gradualmente constroem uma solução viável e simultaneamente buscam uma melhor solução. Elas não possuem uma fase de melhoria.

2) Heurísticas de 2 fases: o problema é decomposto em seus 2 componentes naturais: a clusterização de vértices em rotas viáveis e a construção da rota. É possível haver *loops* entre os 2 estágios. Esta categoria ainda pode ser dividida em 2 tipos: clusterizar primeiro e rotear depois, e analogamente, rotear primeiro e clusterizar depois. No primeiro tipo, os vértices são primeiramente organizados em grupos viáveis e uma rota é construída para

cada um deles. No segundo tipo, uma rota é inicialmente construída unindo todos os vértices e então é segmentada em rotas viáveis.

3) Métodos de Melhoria ou Buscas Locais: tentam melhorar uma dada solução viável perfazendo uma seqüência de trocas de arcos ou vértices dentro de uma mesma rota ou entre diferentes rotas.

Algumas clássicas heurísticas construtivas são mostradas em 2.4.1.1 e 2.4.1.2. Uma famosa heurística de 2 fases é apresentada em 2.4.1.3 enquanto que métodos de melhoria são introduzidos em 2.4.1.4 e 2.4.1.5.

2.4.1.1 – Heurística de Clarke & Wright

O algoritmo Clarke & Wright (1964) é talvez a heurística mais conhecida do PRV. É baseada no conceito de *"*economias*"*, também usado em muitas outras heurísticas. Quando 2 rotas *(0,..., i, 0) e (0, j,,..., 0)* podem ser viavelmente unidas em uma única rota *(0,..., i, j,,..., 0),* a economia obtida é dada pela equação 2.8:

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \text{ com } i, j = 1, \dots, n \text{ sendo } i \neq j \quad (2.8)$$

Onde: s_{ij} é a economia obtida ao conectar os clientes *i* e *j*; c_{i0} é o custo do cliente *i* ao depósito; c_{0j} é o custo do depósito ao cliente *j*; c_{ij} é o custo do cliente *i* ao cliente *j*; *n* é o número de clientes ou vértices.

Este algoritmo aplica-se naturalmente a problemas em que o número de veículos também é uma variável. Pode ser usado tanto para grafos direcionados como para grafos não direcionados. No artigo de Vigo (1996) é relatado que o comportamento do método piora significativamente no caso direcionado.

Há uma versão seqüencial e paralela do algoritmo. A versão paralela tem melhor desempenho que a seqüencial (Clarke & Wright, 1964). A versão paralela funciona da seguinte forma:

1) Calcular as economias: aplicar equação 2.8 para calcular as economias de todos os arcos *(i, j)* do grafo. Ordená-los em ordem decrescente.

2) Unir as rotas: começando do topo da lista de economias, dado a economia $s_{ij,}$ determinar se existem 2 rotas, uma contendo o arco *(0,j)* e outra com o arco *(i,0)* que podem ser unidas respeitando a restrição de capacidade.

Então, combinar estas 2 rotas deletando-se *(0,j)* e *(i,0)* e adicionando-se *(i,j)*.

O algoritmo tende a gerar boas rotas no início de sua execução e à medida que caminha para o final, as rotas tendem a perder qualidade, por exemplo, rotas circulares em torno do depósito. Para tratar isto, Gaskell (1967) e Yellow (1970) propuseram uma sutil mudança na forma de calcular as economias. A equação 2.8 é alterada para 2.9:

$$s_{ij} = c_{i0} + c_{0j} - \lambda * c_{ij} \text{ com } i, j = 1, \dots, n \text{ sendo } i \neq j \quad (2.9)$$

Onde λ é um parâmetro de forma da rota. Quanto maior o valor do parâmetro λ, maior é a ênfase dada à distância entre os vértices a serem conectados. Golden *et al*. (1977) relatam que valores para o parâmetro λ entre 0.4 e 1.0 alcançam bons resultados, levando-se em conta o número de rotas e o custo total da solução.

2.4.1.2 – Heurística de Inserção Seqüencial de Mole & Jameson

A heurística de Mole & Jameson usa 2 parâmetros λ e μ para expandir a rota em construção. Ver equações 2.10 e 2.11.

$$\alpha(i, k, j) = C_{ik} + C_{kj} - \lambda * C_{ij} \quad (2.10)$$

$$\beta(i, k, j) = \pi * C_{0k} - \alpha(i, k, j) \quad (2.11)$$

O algoritmo é descrito a seguir:

1) Inicialização de uma nova rota: se todos os vértices pertencem a uma rota, pare. Caso contrário, inicie uma nova rota *(0, k, 0),* onde *k* é algum vértice ainda não inserido na solução.

2) Escolha do próximo vértice: calcular para cada vértice *k* ainda não inserido na solução o custo $\alpha^*(i_k, k, j_k) = min\{\alpha(r, k, s)\}$ para todos os vértices adjacentes *r* e *s* da rota em construção, onde i_k e j_k são os 2 vértices que obtêm o custo α^*. Se nenhuma inserção é viável, retorne ao passo 1. Caso contrário, o melhor vértice k^* a ser inserido na rota em construção é o vértice com $\beta^*(i_{k^*}, k, j_{k^*}) = max\{\beta(i_k, k, j_k)\}$ dentre todos os vértices *k* que podem ser viavelmente inseridos na solução. Inserir k^* entre i_{k^*} e j_{k^*}.

3) Otimização da rota: otimizar a rota corrente utilizando, por exemplo, um procedimento 3-*Opt* (ver Lin, 1965), e retorne ao passo 2.

Várias regras de inserção são controladas pelos parâmetros λ e μ. Por exemplo, se λ=1 e μ=0 o algoritmo irá inserir o vértice que implica na menor distância extra. Se λ=0 e μ=0, o vértice a ser inserido corresponde àquele com a menor soma das distâncias entre 2 vizinhos. $\mu=\infty$ e $\lambda>0$, o vértice mais distante do depósito será inserido.

2.4.1.3 – Algoritmo de Varredura *(Sweep)*

Grupamentos viáveis são inicialmente formados pela construção de um raio centrado no depósito. Uma rota é então obtida para cada grupamento resolvendo-se um PCV (Problema do Caixeiro Viajante). Algumas simples implementações incluem uma fase de pós-otimização na qual vértices são trocados entre grupamentos (*clusters*) adjacentes. A primeira citação deste método foi feita em Wren & Holliday (1972), mas ele foi popularizado com os trabalhos de Gillett & Miller (1974). Uma simples implementação é dada a seguir. Assuma que cada vértice i seja representado por suas coordenadas polares (θ_i e ρ_i) onde θ_i é o ângulo e ρ_i é o tamanho do raio. Atribua um valor θ^*_i=0 a um vértice arbitrário i^* e calcule os ângulos dos demais vértices a partir *(0,i*).* Ordene os vértices em ordem crescente de θ_i.

1) Inicialização da rota: escolha um veículo k ainda não utilizado.
2) Construção da rota: Partindo do vértice com o menor ângulo atribua vértices (clientes) ao veículo k enquanto a rota permanecer viável.
3) Otimização da rota: otimize a rota de cada veículo separadamente, resolvendo seu PCV equivalente (por um método exato ou aproximado).

A figura 2.2 ilustra as duas fases desta estratégia. Na célula A, os grupos de vértices são organizados a partir da varredura circular. Na célula B, as rotas são construídas a partir da solução do PCV para cada um dos grupos escolhidos na primeira fase.

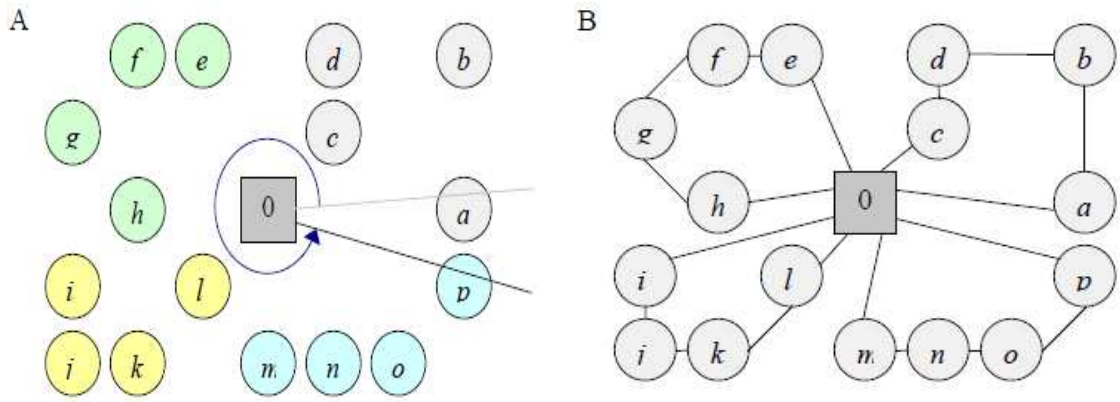


Figura 2.2: As 2 fases do Algoritmo de Varredura

2.4.1.4 – Algoritmos de Melhoria Intra-Rotas

Estas heurísticas buscam fazer melhorias em uma única rota, sem realizar qualquer mudança nas demais rotas. Assim, a maioria dos métodos de melhoria atribuídos ao PCV podem ser utilizados (ver Lin, 1965). Nestes métodos, λ arcos são removidos da rota e reconectados de todas as formas possíveis. Se alguma reconexão apresentar melhoria no custo (a primeira ou a melhor), a mudança é efetivada. O procedimento pára em um ótimo local, quando nenhuma melhoria pode ainda ser obtida. Testar a λ-otimalidade da solução tem um custo computacional da ordem $O(n^{\lambda})$. A figural 2.3 ilustra um movimento λ-*Opt* com λ=2, ou seja um *2-Opt.* As arestas *(2, 4)* e *(1, 3)* foram substituídas por *(1, 2) e (3, 4).*

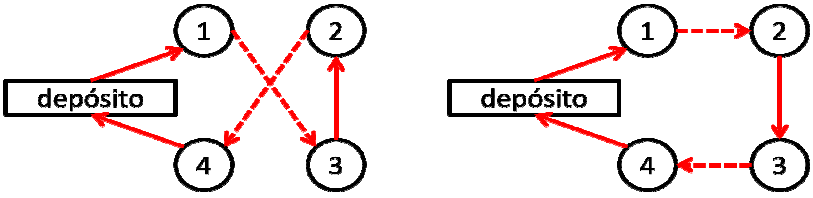


Figura 2.3: movimento 2-opt.

Várias modificações desta idéia principal têm sido desenvolvidas. Lin & Kernighan (1973) modificam λ dinamicamente durante a busca. Or (1976) propôs o método *Or-Opt* que consiste em mover seqüências de 1,2 ou 3 vértices consecutivos para outro local da rota. Isto cai num caso particular de mudanças λ-*Opt* para λ=3. Checar a Or-otimalidade requer tempo $O(n^2)$.

2.4.1.5 – Algoritmos de Melhoria Inter-Rotas

Estes algoritmos também chamados de algoritmos de melhoria multirotas, modificam mais de 1 rota simultaneamente buscando melhorar a solução. Por exemplo, o método λ-Opt citado em 2.4.1.4 pode ser analogamente aplicado trocando-se λ arcos entre diferentes rotas.

Um método muito utilizado é chamado "Realocação". Nele, um cliente é removido de uma rota r_1 e inserido numa rota r_2. A figura 2.4 ilustra este tipo de movimento para uma solução com 3 rotas.

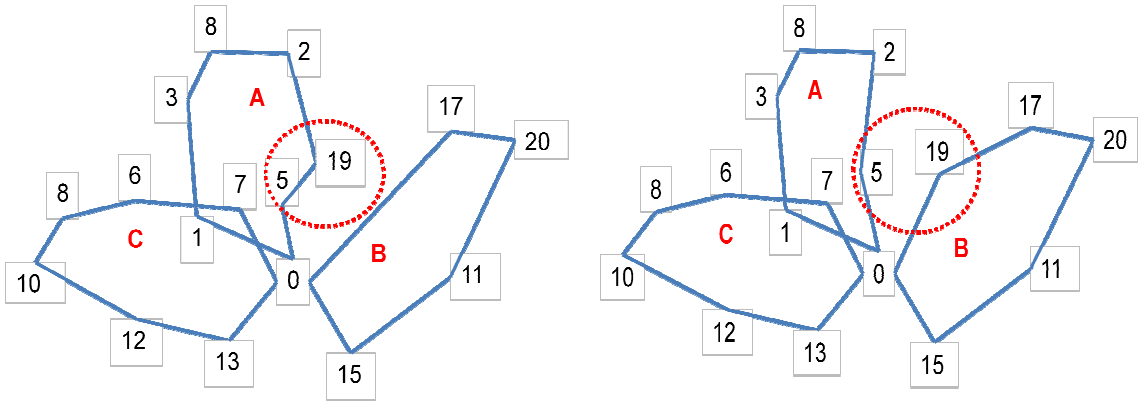


Figura 2.4: Realocação. Cliente 19 é movido da rota A para rota B

Outro método também muito utilizado é chamado "Intercâmbio". Nele x clientes de uma rota r_1 são movidos para r_2 e y clientes de r_2 são movidos para r_1. A figura 2.5 ilustra um movimento para $x=1$ e $y=1$.

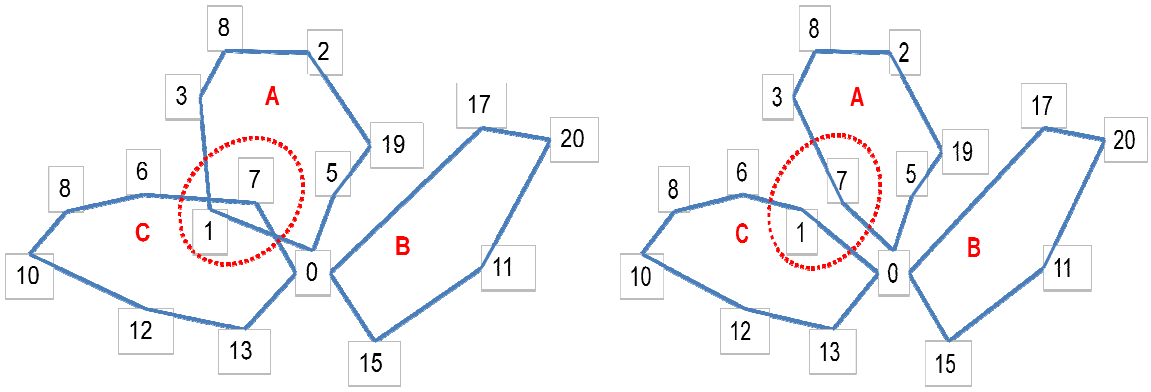


Figura 2.5: Intercâmbio. Cliente 7 de C para A e cliente 1 de A para C

Os métodos "Realocação" e "Intercâmbio" são derivados do método λ-*Interchange*, desenvolvido por Osman (1993). Dadas duas rotas distintas, em cada uma

delas seleciona-se no máximo λ nós (a quantidade de nós pode diferir entre uma rota e outra). Em seguida, troca-se os nós entre as rotas.

Thompson & Psaraftis (1993) e Kindervater & Savelsbergh (1997) fornecem detalhes de trocas de arcos multirotas para o PRV. O artigo de Thompson & Psaraftis (1993) descreve um *b-cycle* genérico, uma estratégia na qual uma permutação circular de *b* rotas é considerada e *k* clientes de cada rota são deslocados para a próxima rota da permutação cíclica. Os autores mostram que a aplicação de seqüências específicas com b=2 ou variável e com k=1 ou 2 implica em resultados interessantes.

2.4.2 – Metaheurísticas para o PRVC

Assim como as heurísticas clássicas, as metaheurísticas têm como objetivo explorar apenas parte do espaço solução. Entretanto, geralmente, isto é feito de um modo mais abrangente, fazendo com que as soluções encontradas sejam de melhor qualidade. Contrariamente às heurísticas clássicas, as metaheurísticas são mais gerais e têm capacidade de sair de ótimos locais, como será visto adiante.

São muitas as metaheurísticas aplicadas ao PRV. Algumas das principais são: *Simulated Annealing (SA), Deterministic Annealing (DA), Tabu Search (TS), Genetic Algorithms (GA), Ant Systems (AS), Neural Networks (NN), Greedy Randomized Adaptive Search Procedures (GRASP), Adaptive Memory Procedure (AMP)* e o *Guided Local Search (GLS), Iterated Local Search (ILS) e VND (Variable Neighborhood Descent).*

Todos estes tipos de metaheurísticas podem ser combinados entre si ou ainda com outros tipos não citados. Também é comum que metaheurísticas se utilizem de heurísticas clássicas, usando métodos construtivos para gerar soluções iniciais e de métodos de melhoria em pontos específicos do algoritmo.

Algoritmos como o *AS*, *DA* e *TS*, começam de uma solução inicial x_1 e a cada iteração t, movem de x_t para uma solução x_{t+1} na vizinhança $N(x_t)$ de x_t, até que um critério de parada seja satisfeito. Se $f(x)$ denota o custo de x, então $f(x_t+1)$ não é necessariamente menor que $f(x_t)$. Como conseqüência, cuidados devem ser tomados para evitar ciclagem (retorno sistemático ao mínimo local anteriormente alcançado).

O *GA* ou Algoritmo Genético examina a cada iteração uma população de soluções. Cada iteração ou geração é derivada da geração anterior, pela combinação de

suas melhores soluções (indivíduos). *Ant Systems* ou Colônias de Formigas é uma abordagem construtiva na qual novas soluções são criadas a cada iteração utilizando-se algumas das informações obtidas em gerações anteriores. Como descrito em Taillard *et al.* (1998), o *Tabu Search* (Busca Tabu), algoritmos genéticos e colônias de formigas são métodos que armazenam ao longo das iterações informações sobre as soluções encontradas e usam tais informações para obter melhores soluções.

O *GRASP* é um algoritmo *multi-start* que consiste em criar uma solução inicial e depois efetuar uma busca local para melhorar a qualidade da solução. Seu diferencial para outros métodos está na geração desta solução inicial feita através de um método construtivo que se utiliza de uma função gulosa para avaliar os elementos a serem inseridos na solução. Uma vez realizada essa valoração, é construída uma lista de elementos candidatos, de onde escolhe-se um elemento aleatoriamente dentre os melhores avaliados da lista, para ser inserido na solução. Devido a este misto entre a função gulosa na construção de uma lista e a decisão aleatória sobre qual elemento dessa lista utilizar, diz-se que o *GRASP* é semi-guloso.

Iremos focar o *TS*, *AMP, GLS, ILS* e VND, por serem as metaheurísticas que mais contribuíram para o presente trabalho. Sugestões de artigos reconhecidos na literatura são dadas a seguir, para um maior aprofundamento dos demais tipos:

- *Simulated Annealing (Recozimento Simulado):* Osman (1993).
- *Deterministic Annealing (Recozimento Determinístico):* Dueck and Scheurer (1990).
- *Algoritmos Genéticos:* Holland (1975), Baker & Ayechew (2003)
- *Ant Systems (AS):* Bullnheimer et al. (1999)
- *Neural Networks (NN):* Hopfield & Tank (1985), um artigo pioneiro na aplicação da técnica ao VRP, e Potvin & Robillard (1995).
- *GRASP:* Feo & Resende (1995), Assis (2007) e Pereira (2010).

2.4.2.1 – Busca Tabu (*Tabu Search - TS*)

Segundo Gendreau *et al.* (1994), a Busca Tabu foi proposta por Glover em 1977, e dois dos primeiros trabalhos associadas ao PRV são de autoria de Willard

(1989) e de Pureza (1990). Depois disso, uma série de outros autores publicaram os esforços de suas pesquisas, conseguindo resultados expressivos diante de outros métodos para o PRV.

Inicialmente, vamos imaginar um cenário onde aplicamos uma heurística rápida qualquer, visando encontrar uma solução de partida para uma instância do PRV, por exemplo. Em seguida, podemos aplicar alguns procedimentos de melhoria, como apresentados anteriormente, em busca de um resultado de maior qualidade. Tais procedimentos conseguem melhorar a solução até encontrar um mínimo local. Se quisermos continuar com o processo de busca, a primeira coisa a fazer é realizar um movimento de piora, digamos o que causa menor aumento no custo, de maneira que comecemos a escapar desse mínimo local. A partir dessa situação, podemos pesquisar dentre os movimentos disponíveis, um que realiza uma melhora na solução.

Existe a possibilidade de que tal movimento faça com que a solução retorne ao mínimo local anteriormente alcançado. Caso isso ocorra, o passo seguinte indicaria novamente o movimento que causa o menor aumento no custo, e na seqüência poderíamos novamente retornar ao mínimo local. Essa situação ilustra o fenômeno da ciclagem, capaz de comprometer o processo em busca de melhores soluções. Precisamos, portanto, de um elemento restritivo na busca, capaz de impedir a reversão dos movimentos, fazendo com que o processo siga procurando novas soluções por regiões ainda não pesquisadas.

Poderíamos armazenar todos os movimentos anteriormente efetivados para realizar comparações, mas isso implicaria em maior utilização de memória e esforço de processamento. Além do mais, após sucessivas alterações, a possibilidade de retorno a uma determinada solução diminui. Isso sugere que a restrição feita a certos movimentos pode ser relaxada ao longo do processo. Caso não exista mais o risco de ciclagem, tal relaxação é até desejável, pois os movimentos liberados oferecem maior flexibilidade ao método.

Os aspectos anteriormente referidos constituem os dois elementos básicos da Busca Tabu: a restrição da busca, devido a proibição de movimentos contidos na lista tabu (movimentos proibidos); e a liberação dos movimentos, através de uma função de curto prazo ou após certo número de iterações, ou ainda, pelo próprio tamanho máximo da lista , que provoca um esquecimento conveniente (Pureza, 1990).

Na construção da lista tabu, para aliviar requerimentos de memória e tempo requeridos, é comum armazenar um atributo da solução tabu ao invés de toda a solução propriamente dita. Há trabalhos na literatura que escolhem um atributo diferente da solução para armazenar. Isto muda a estrutura de dados da Lista Tabu. Por exemplo: em Campos & Mota (2000) o atributo escolhido é uma tripla (u_i, u_j, r_1) no caso de um movimento intra-rota, e uma quádrupla (u_i, u_j, r_1, r_2) no caso de um movimento inter-rotas. Na tripla, temos a informação de que os vértices u_i e u_j foram trocados, ambos pertencendo à rota r_1. Na quádrupla, temos a informação de que o vértice u_i da rota r_1 foi trocado com o vértice u_j da rota r_2. Em Pureza (1990), são armazenadas as arestas envolvidas no movimento. Cria-se duas listas, uma com as arestas adicionadas e outra com as arestas removidas. Em Taillard *et al*. (1997), a lista possui apenas o valor da função objetivo após a execução do movimento.

De qualquer forma, é consenso que durante o processo de busca, os índices dos atributos envolvidos em cada movimento efetivado são incluídos no início das listas correspondentes, deslocando os demais índices para o final da lista.

A decisão sobre o tamanho da Lista Tabu faz-se importante. Listas pequenas podem não evitar a ciclagem, enquanto listas grandes podem impedir a exploração de novas e interessantes soluções.

Segundo Laporte *et al.* (2000), o mecanismo básico da Busca Tabu pode ser melhorado por diversas características como diversificação e estratégias de intensificação, como descrito por Hertz *et al.* (1997), e por Glover & Laguna (1993, 1997), por exemplo.

Intensificação Regional: é uma estratégia promovida por funções de memória de médio prazo que comparam e armazenam características comuns de um certo número de boas soluções, coletadas durante um período específico da busca. Tais características são tomadas como um atributo regional de boas soluções. De posse dessas informações, a estratégia procura outras soluções que apresentem tais características, impondo restrições ou penalizações aos movimentos disponíveis.

Diversificação Global: É uma estratégia promovida por funções de memória de longo prazo que estimulam a procura de soluções em regiões ainda não exploradas do espaço de busca. Um exemplo simples seria contar o número de vezes que cada aresta aparece nas rotas geradas. Uma penalização proporcional pode ser aplicada a esse

número, fazendo com que as arestas que mais participaram dos movimentos de intercâmbio, não sejam tão utilizadas como foram anteriormente. Com isso, novas regiões passam a ser exploradas.

Outros conceitos básicos são descritos em Campos & Mota (2000). Estrutura de vizinhança: é definida por todas as soluções que podem ser obtidas a partir da solução corrente pela execução de um movimento. Ou seja, é possível ir da solução *A* para a solução *B* em 1 movimento. Movimento: consiste de uma troca entre 2 vértices ou da remoção/inserção de 1 vértice em 1 rota. Movimento admissível: se as 2 rotas envolvidas no movimento preservam sua viabilidade e o movimento não é tabu, ou se o movimento implicou numa solução melhor que a encontrada até o momento (critério de aspiração). Movimento tabu: Se o cliente *p* na rota *i* é movido para a rota *j,* então mover *p* para a rota *i* é declarado tabu.

A seguir, é descrito alguns estudos que obtiveram sucesso na aplicação dos conceitos da Busca Tabu.

O algoritmo *Taburoute* de Gendreau *et al.* (1994) apresenta algumas características inovadoras. O processo de busca pode examinar soluções inviáveis com relação às restrições de capacidade e distância máxima, por exemplo. A função objetivo possui 2 termos de penalidade, neste caso, uma para cada uma das restrições citadas anteriormente. Cada termo de penalidade é ponderado por um coeficiente que é ajustado dinamicamente. Se todas as 10 soluções anteriores forem viáveis, o peso da penalidade é dividido por 2, se todas as 10 soluções anteriores foram inviáveis, o peso é multiplicado por 2. Outra característica é que o *Taburoute* não possui de fato uma lista tabu, mas um atributo tabu. Sempre que um vértice é movido da rota *r* para outra rota na iteração *t*, sua reinserção na rota *r* é proibida até a iteração $t+\theta$, onde θ é um inteiro aleatoriamente escolhido no intervalo [5,10]. Também utiliza-se uma estratégia de diversificação que visa penalizar vértices movidos freqüentemente com o objetivo de aumentar a probabilidade de se testar também vértices com baixa freqüência de movimentos. Neste caso a função objetivo é artificialmente aumentada, adicionando-se um termo proporcional à freqüência de movimentação do vértice corrente. Uma última característica relevante é a utilização de "*false starts*". Inicialmente, várias soluções são geradas e uma pesquisa (busca de melhoria) limitada (rápida) é feita em cada uma delas. A melhor solução é então selecionada como ponto de partida do algoritmo.

O algoritmo *Granular Tabu Search (GTS)* tem um conceito interessante. Foi introduzido por Toth & Vigo (1998b) e obteve excelentes resultados no PRV. A principal idéia que permeia o *GTS* vem da observação de que arcos maiores (compridos) do grafo têm pequena probabilidade de fazer parte da solução ótima. Então, eliminando-se todos os arcos cujo tamanho excede um limite de granularidade (*granularity threshold*), muitas soluções não promissoras nunca serão consideradas no processo de busca. Os autores do artigo sugerem usar para o limite de granularidade um valor $\nu=\beta*\mu$, onde β é um coeficiente escolhido no intervalo [1,2], e μ é a média do comprimento dos arcos calculado a partir de uma solução gerada por uma heurística rápida. Os resultados do artigo mostram excelentes soluções em um baixo tempo computacional.

2.4.2.2 – *Adaptive Memory Procedure* (Memória Adaptativa)

Um conceito interessante que tem tido ótimos resultados foi desenvolvido por Rochat & Taillard (1995). Uma memória adaptativa é um conjunto de boas soluções que são dinamicamente atualizadas através de processos de busca. Periodicamente alguns elementos destas soluções são extraídos do conjunto e combinados para produzir uma nova boa solução. No PRV, rotas selecionadas de várias soluções são usadas como ponto de partida. O processo de extração dá um maior peso para as rotas pertencentes às melhores soluções. No momento de selecionar as rotas, deve-se ter o cuidado de evitar a inclusão repetida de um cliente. Isto significa que o processo de seleção freqüentemente termina com uma solução parcial que deverá ser completada usando-se alguma heurística de construção. Este procedimento obteve 2 novas melhores soluções nas 14 instâncias clássicas de Christofides *et al.* (1979).

Tarantilis (2005) apresentou um algoritmo denonimado SEPAS (*Solutions' Elite PArts Search*) fortemente baseado no conceito de memória adaptativa. Ele obteve excelentes resultados particularmente quando aplicado a instâncias de larga escala.

2.4.2.3 – *Guided Local Search (GLS)*

O *Guided Local Search* (Voudouris, 2003) é uma metaheurística que conduz algoritmos de busca local pelo espaço de busca. Toda vez que o algoritmo encontra um mínimo local, uma característica considerada indesejável desta solução (digamos o valor

do arco de maior comprimento) é selecionada e soluções que possuem esta característica têm seu custo penalizado. A função objetivo $f(s)$ é substituída por $h(s) = f(s) + \lambda \Sigma(p_i * I_i(s))$ em que p_i é a penalidade associada à característica i, $I_i(s)$ é uma variável binária que assume valor 1 caso a solução possua a característica i e valor 0 caso contrário, e λ é um parâmetro do algoritmo.

Inicialmente, todas as penalidades são ajustadas em 0. Ao se deparar com um mínimo local s*, um valor $u_i(s^*) = I_i(s^*)c_i/(1+p_i)$, que define a utilidade de se penalizar cada característica i, é calculado. A característica com maior valor u tem sua penalidade aumentada por 1 unidade.

Esse mecanismo penaliza características com alto custo na solução corrente e que não tenham sido muito penalizadas durante a execução do algoritmo.

David & Olli (2005) desenvolveram um algoritmo combinando a força do *GLS* com uma metaheurística que possui 2 estágios iterativos. O algoritmo recebeu o nome de *AGES* (*Active-Guided Evolution Strategies*). A metaheurística consiste de 2 fases. O objetivo da primeira fase é criar uma solução inicial para a segunda fase. A solução inicial é gerada pela criação de s soluções a partir da aplicação de uma heurística de inserção mais barata híbrida descrita em Mester *et al.* (2005) e então selecionando-se a melhor solução encontrada. A segunda fase é dividida em 2 estágios. O primeiro estágio faz uso do *GLS*. O *GLS* funciona aumentando a função objetivo com um termo de penalidade baseado em características particulares da solução, como descrito anteriormente. A característica escolhida pelos autores é o arco mais comprido de uma determinada solução. O *GLS* é usado para guiar um procedimento composto de 3 a 5 heurísticas de melhoria local. Quando estas buscas não encontram melhorias durante um certo número de iterações consecutivas, o segundo estágio da segunda fase é iniciado. O segundo estágio opera perfazendo uma série de operações de remoção e re-inserção de arcos na vizinhança do ótimo local encontrado pela fase anterior. Este estágio continua até que nenhuma melhoria é feita durante uma quantidade de iterações consecutivas e então retorna ao primeiro estágio. Quando ambos os estágio não encontrarem melhorias, o algoritmo é finalizado.

No primeiro estágio da segunda fase, o *GLS* trabalha penalizando sempre um único arco toda vez que o algoritmo fica preso em um ótimo local. O arco escolhido para penalização é o arco da solução corrente (ótimo local) para o qual o maior valor da função "utilidade" foi obtido. A função "utilidade" (U) é dada na equação 2.12:

$$U = {C_{ij}}/{1 + p_{ij}} \tag{2.12}$$

$$C_{ij}^{*} = C_{ij} + p_{ij}\lambda L \tag{2.13}$$

Onde p_{ij} é o contador de penalidades do arco (i,j) que armazena o número de vezes que o arco já foi penalizado. Quando avalia-se um movimento dentro do procedimento de busca local, o novo custo C_{ij}^{*} do arco penalizado é calculado conforme equação 2.13, onde L é a média do comprimento dos arcos na solução inicial e λ é um parâmetro.

2.4.2.4 – *Iterated Local Search (ILS)*

Dado o espaço de soluções viáveis S e uma função $BL : S \rightarrow S$, tal que dado $s \in S$, e $s^* = BL(s)$ tem-se $f(s) \geq f(s^*)$. A idéia básica do *ILS* (Lourenço, 2003) é fazer uma busca estocástica no espaço $S^* = BL(S)$. Isto é, se a função *BL* é um procedimento de busca local, o método *ILS* realiza uma busca pelo espaço de mínimos locais dado pela função. O *ILS* mantém uma solução corrente $s^* \in S^*$ e, a cada passo, encontra uma nova solução $s_p \in S^*$ pela geração de uma solução perturbada $s^{*\prime} \in S$ a partir de s^* seguido pela aplicação de *BL* em $s^{*\prime}$ para gerar $s_p^* \in S^*$. Nesse ponto, s_p^* pode ou não ser aceita como solução corrente. Perturbações muito fortes da solução corrente farão com que o espaço de soluções S* seja explorado de maneira aleatória, levando a um algoritmo *Random Restart*, enquanto perturbações fracas podem fazer com que a busca volte à solução corrente e poucas novas soluções sejam exploradas. O algoritmo básico do *ILS* é mostrado a seguir:

Algoritmo 2.1: *ILS*

```
1: s0 = SolucaoInicial
2: s*= BL(s0)
3: repita
4:         s*' = Perturbacao(s*)
5:         sp* = BL(s*')
6:         se CriterioDeAceitacao(s*; sp*) então
7:                 s* = sp*
8:         fim se
9: até que o critério de parada seja satisfeito
```

2.4.2.5 – *Variable Neighborhood Descent (VND)*

O VND foi desenvolvido por Hansen & Mladenovic (2003). É uma variante da estratégia *Variable Neighborhood Search (VNS*), cuja idéia básica é a troca sistemática de vizinhanças. Trata-se de uma estratégia na qual a idéia básica consiste na combinação de busca local (*Descents*) em diversas vizinhanças, baseada no fato de que um mínimo local em uma vizinhança não é necessariamente um mínimo local em outra vizinhança. Assim, o *VND* realiza a troca sistemática de vizinhanças durante a busca local. Ao fim do procedimento, o *VND* alcança uma solução que é um mínimo local em todas as vizinhanças usadas e, conseqüentemente, com boas chances de ser uma boa solução.

O Algoritmo 2.2 exibe o procedimento básico do *VND* (Pereira, 2010).

Algoritmo 2.2: VND(s)

```
1: k = 1;
2: enquanto k ≤ k_max faça
3:         s_0 = melhor vizinho de s em N_k
4:         se s_0 não for melhor que s então
5:                 k = k + 1;
6:         senão
7:                 k = 1;
8:         fim se
9: fim enquanto
```

Seja K a quantidade de estruturas de vizinhanças e $N_1,N_2,...,N_k$ as estruturas de vizinhança (diferentes procedimentos de busca local). Antes de iniciar o *VND* defini-se aleatoriamente a seqüência com que as estruturas serão exploradas através de um vetor de índices de tamanho K.

2.5 – DISCUSSÃO FINAL

Segundo Laport *et al.* (2000), os métodos aproximados para o VRP tiveram importante progresso desde a década de 60. Os mais poderosos algoritmos são agora capazes de resolver instâncias de média escala de forma ótima ou próxima do ótimo. Atualmente, a Busca Tabu emerge como a abordagem que tem obtido maior sucesso, até por poder ser combinada com outros conceitos muito satisfatórios como memória adaptativa, por exemplo.

Toth & Vigo (2002b) ao examinarem resultados obtidos por diferentes metaheurísticas, concluem que procedimentos baseados em Algoritmos Genéticos puros e em Redes Neurais têm sido superados, enquanto aqueles baseados no *Simulated Annealing* e *Ant Systems* não têm se mostrado tão competitivos. No entanto, todos mostram-se promissores quando implementados em algoritmos híbridos. Os trabalhos de Prins (2004) e Nagata (2007) provam que algoritmos genéticos híbridos (meméticos) são capazes de obter excelentes resultados.

Durante o estudo da arte, não se encontrou nenhum trabalho na literatura que buscasse integrar as técnicas *ILS*, *VND*, Busca Tabu, Memória Adaptativa e *GLS* numa única metaheurística. Como tais técnicas obtiveram sucesso no contexto de outras metaheurísticas, faz-se objetivo deste trabalho implementar uma metaheurística que una todas estas técnicas de modo satisfatório.

Com o objetivo de aprofundar-se em problemas de roteamento mais próximos de situações reais, será estudado a seguir os problemas com janela de tempo.

CAPÍTULO 3

PRV COM JANELAS DE TEMPO (PRVJT)

Segundo Azi *et al.* (2010), embora o número de estudos tenha crescido, o PRV com janela de tempo ainda não tem recebido muita atenção na literatura em relação à sua importância prática.

O Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) é a generalização do problema capacitado na qual se associa, a cada cliente, um período de tempo no qual algum veículo deve começar a atendê-lo. A esse intervalo dá-se o nome de janela de tempo.

Segundo Solomon & Desrosiers (1988), o problema de roteamento de veículos com janelas de tempo também é *NP-hard* (NP-difícil) por ser uma extensão do PRV.

Como no PRVC, o problema é representado por um grafo $G = (V, A)$ em que A é o conjunto de arcos e $V = 0, \ldots, n$ é o conjunto de vértices. O vértice *0* indica o depósito, enquanto os demais nós representam os clientes.

O conjunto de clientes tem sua demanda satisfeita por uma frota de veículos que partem do depósito. Todos os veículos possuem uma capacidade constante *C*, de forma que a frota é homogênea.

A cada cliente *i* associa-se:

1) Uma demanda m_i;
2) Um tempo de serviço s_i;
3) Um instante de início da janela de tempo e_i;
4) Um instante para o final da janela de tempo l_i;

O depósito também possui uma janela de tempo $[e_0, l_0]$, indicando o momento a partir do qual os veículos podem começar trafegar e o momento em que o último veículo deve estar de volta.

Em virtude da existência de janelas de tempo, é necessário associar a cada arco *(i, j),* além do custo c_{ij}, o escalar t_{ij}, que representa o tempo necessário para ir do

vértice *i* ao vértice *j*. Por simplicidade, sem perda de generalidade, adotaremos $c_{ij} = t_{ij}$, o que corresponde a atribuir um valor unitário à velocidade dos veículos. Além disso, supomos sempre que a desigualdade triangular é respeitada.

Embora existam problemas em que a janela de tempo pode ser violada mediante pagamento de penalidades (classe conhecida como *Soft Time Windows*), trabalharemos aqui com o caso em que isso não é permitido (classe conhecida como *Hard Time Windows*). Por outro lado, um veículo pode chegar ao endereço de um cliente *j* antes do início de sua janela de tempo. Neste caso, o veículo deve permanecer parado em espera até o início do serviço.

O objetivo primário do problema é minimizar o número de veículos e o objetivo secundário é minimizar a distância total percorrida, neste caso, o tempo total de viagem. Assim como no PRVC, a variável binária x_{ijk} indica se o veículo *k* percorreu ou não o arco que liga o nó *i* ao nó *j*. Por sua vez, a variável real $b_i \; com \; i = 1, \dots, n$, indica o instante de início do atendimento ao cliente *i*.

A formulação matemática do problema é dada a seguir, sendo baseada no trabalho de Vieira (2009) em que a função objetivo foi modificada para hierarquizar os objetivos:

$$Min \;\; M \sum_{k \in K} \sum_{j=1}^{n} x_{0jk} + \sum_{i=0}^{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} \sum_{k \in K} c_{ij} x_{ijk} \tag{3.1}$$

$s.a$:

$$\sum_{j=1}^{n} x_{0jk} = \sum_{j=1}^{n} x_{j0k} = 1; \;\; k = 1, \dots, K \tag{3.2}$$

$$\sum_{k \in K} \sum_{j=1}^{n} x_{ijk} = 1; i = 1, \dots, n \tag{3.3}$$

$$\sum_{j=0}^{n} x_{ijk} - \sum_{j=0}^{n} x_{jik} = 0; \;\; k = 1, \dots, K; \; i = 1, \dots, n \tag{3.4}$$

$$\sum_{k \in K} \sum_{i \in S} \sum_{j \in S} x_{jik} \; \leq |S| - \; v(S); \; \forall \, S \; \subseteq V \backslash \{0\}; \; |S| \geq 2 \tag{3.5}$$

$$\sum_{i=1}^{n} m_i \sum_{\substack{j=0 \\ j \neq i}}^{n} x_{ijk} \leq C;\ k = 1, \ldots, K \tag{3.6}$$

$$\sum_{k \in K} \sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ijk}\left(b_i + s_i + t_{ij}\right) \leq b_j;\ j = 1, \ldots, n \tag{3.7}$$

$$e_i \leq b_i \leq l_i;\ \ i = 0, \ldots, n \tag{3.8}$$

$$x_{ijk} \in \{0,1\};\ i = 1, \ldots, n\ ;\ \ j = 1, \ldots, n\ ;\ \ k \in K \tag{3.9}$$

A função objetivo (3.1) e as restrições (3.2)-(3.6) e (3.9) foram herdadas do PRVC. A restrição (3.7) relaciona o instante de início do atendimento de dois clientes visitados consecutivamente por um mesmo veículo. Ela evita que o intervalo entre os instantes de início do atendimento desses clientes seja inferior à soma do tempo gasto no atendimento do primeiro cliente com o tempo consumido na viagem entre os dois clientes. Por sua vez, (3.8) impede que o início do atendimento do cliente *i* ocorra fora de sua janela de tempo.

A folga da restrição (3.7) corresponde ao tempo de espera do veículo que atende o cliente *j*, ou seja, o tempo consumido entre o instante de chegada do veículo ao endereço do cliente e o início do atendimento deste.

3.1 – TÉCNICAS EXATAS DE SOLUÇÃO PARA O PRVJT

Os principais métodos exatos utilizados na resolução do PRVJT são o *Branch-and-Bound*, o *Branch-and-Cut* e o *Branch-and-Price*. Todos eles utilizam alguma técnica de relaxação do problema de programação inteira, combinada, dependendo do método, com o uso de planos de corte e de geração de colunas.

3.1.1 – Algoritmos Branch-and-Bound para o PRVJT

Considerando o problema de roteamento com janela de tempo, ou seja, um problema de programação inteira mista *0–1*, o método tem início pela solução do problema no qual todas as restrições de integralidade são relaxadas. Em seguida, a técnica é decomposta em duas partes. Na fase de *branching* (ramificação), uma variável binária é fixada, ora em 0, ora em 1, gerando, assim, dois subproblemas que devem ser

resolvidos separadamente. O valor da função objetivo da solução ótima de cada um desses subproblemas é um limitante inferior para a solução ótima do problema original de programação inteira.

Ao conjunto de subproblemas relaxados associamos uma árvore que tem como raiz o problema em que todas as restrições de integralidade são relaxadas. Cada mudança de nível desta árvore está associada à ramificação de alguma variável. Os subproblemas ainda não resolvidos formam os nós pendentes, ou seja, os ramos não explorados dessa árvore.

Sempre que a solução ótima de um subproblema, x^*_k, satisfaz todas as restrições de integralidade do problema original, dá-se início a fase de *bounding*, que consiste na eliminação dos nós pendentes da árvore de subproblemas que têm função objetivo com valor pior que aquele correspondente a x^*_k.

Como exemplo da aplicação do *Branch-and-Bound* ao PRVJT citamos: Baker (1982) e Kolen *et al*. (1987).

3.1.2 – Algoritmos *Branch-and-Cut* para o PRVJT

O método de *Branch-and-Cut* consiste na combinação do procedimento de planos de corte com a técnica de *Branch-and-Bound*. Deste modo, após a fase em que o um problema relaxado é resolvido, o método de planos de corte é aplicado com o propósito de reduzir ainda mais a região factível dos subproblemas derivados desse nó da árvore. Essa redução é obtida através da adição de restrições que geram cortes no polítipo factível do problema.

Bard *et al.* (2002) apresentam detalhes da aplicação deste método ao PRVJT. Neste trabalho, limites inferiores são obtidos resolvendo-se uma série de problemas relaxados. Uma grande variedade de cortes são introduzidos para apertar o problema relaxado obtido do programa inteiro misto original. Soluções viáveis ou limites superiores são obtidos com a ajuda de uma heurística *GRASP*. Resultados são obtidos para instâncias de Solomon (1987) de 50 a 100 clientes.

3.1.3 – Algoritmos *Branch-and-Price* para o PRVJT

O método *Branch-and-Price* é basicamente um método *Branch-and-Bound* com geração de colunas. Em cada nó da árvore de busca encontrada pelo método *Branch-and-Bound*, utiliza-se o método de geração de colunas com o propósito de encontrar novas soluções viáveis.

O método de geração de colunas representa uma generalização da decomposição de Dantzig- Wolfe (1960) na qual o problema original é dividido em duas partes: um problema principal e um subproblema. Como o conjunto de todas as soluções factíveis é muito grande, apenas uma pequena parte desse conjunto de soluções é considerada no problema principal. Em cada iteração principal, determina-se se existe uma rota que possa reduzir a distância total. Isto é conseguido resolvendo o subproblema. Se tal rota existe, sua coluna correspondente é adicionada ao modelo e o procedimento é repetido até que a solução ótima seja encontrada.

Como exemplo da aplicação do *Branch-and-Price* ao PRVJT citamos: Desrochers *et al*. (1992) e Jepsen *et al*. (2006).

3.2 – TÉCNICAS APROXIMADAS DE SOLUÇÃO PARA O PRVJT

Como já citado, o PRVJT enquadra-se na classe dos problemas NP – difíceis. Sua alta complexidade faz dos métodos heurísticos uma interessante alternativa quando é preciso encontrar boas soluções em um tempo limitado.

A grande maioria dos métodos aproximados vistos no capítulo 2 podem ser adaptados aos problemas com janela de tempo. A seguir, alguns dos métodos mais utilizados na resolução do PRVJT e suas adaptações serão apresentados.

3.2.1 – Heurísticas Construtivas para o PRVJT

Solomon (1986) descreve um método do tipo “rotear primeiro e agrupar depois”. Inicialmente, uma única rota unindo todos os clientes é gerada, e então dividida em rotas menores. A rota inicial pode, por exemplo, ser gerada como em um PCV sem considerar-se as restrições de capacidade e tempo. Nenhum resultado computacional é fornecido no artigo quanto a esta heurística.

Várias heurísticas são descritas em Solomon (1987). Aqui será descrito brevemente duas delas, e uma terceira (de melhores resultados) será descrita com mais detalhes.

Um primeiro método trata-se de uma extensão do *Clarke-and-Wright* visto no capítulo 2. Uma lista de economias constituída por pares de nós é gerada e então as rotas começam a ser formadas de forma seqüencial ou paralela. Para lidar com as restrições tanto espaciais quanto temporais, Solomon determina um limite para o tempo de espera da rota.

Um segundo método, trata-se de uma estratégia do vizinho mais próximo orientado no tempo. Uma rota é iniciada encontrando-se o cliente mais próximo do depósito que ainda não faça parte da solução. A cada iteração, a heurística procura pelo cliente mais próximo do último adicionado a uma rota e adiciona-se este cliente no final da rota corrente. Uma nova rota é iniciada quando o processo não consegue encontrar um cliente a ser inserido na rota corrente de forma viável.

A terceira heurística é a que apresentou o melhor desempenho, sendo chamada no artigo de *I1*. Uma rota é inicializada com um cliente "semente" e os demais clientes que ainda não fazem parte da solução são adicionados até que a rota viole uma das restrições de tempo ou capacidade. Se ainda existem clientes não roteados, o processo de inicialização de uma nova rota e inserção de clientes continua até que todos os clientes sejam servidos. Os clientes sementes são selecionados encontrando-se aqueles mais distantes do depósito ou aqueles com o tempo de atendimento mais cedo. Após inicializar uma rota com um cliente semente, o método usa 2 critérios para selecionar o cliente u a ser inserido entre os clientes i e j.

Seja $(i_0, i_1, i_2, \ldots, i_m)$ a rota corrente, onde i_0 e i_m representam o depósito. Para cada cliente não roteado u, primeiro calcula-se o melhor custo de inserção da rota conforme equação 3.10.

$$c_1\big(i(u), u, j(u)\big) = min\big\{c_1\big(i_{p-1}, u, i_p\big)\big\} \text{ onde } p = 1, \ldots, m \tag{3.10}$$

A seguir, o melhor cliente u^* a ser inserido na rota é aquele para o qual o problema 3.11 é resolvido.

$$c_2\big(i(u^*), u^*, j(u^*)\big) = max\left\{\begin{matrix} c_2\big(i(u), u, j(u)\big) | u \text{ não } foi \\ selecionado \text{ } e \text{ } a \text{ } rota \text{ } permança \text{ } viável \end{matrix}\right\} \tag{3.11}$$

O cliente u^* é então inserido na rota entre $i(u^*)$ e $j(u^*)$. Quando não é mais possível encontrar um cliente a ser inserido de forma viável, inicia-se uma nova rota, a não ser que todos os clientes já façam parte da solução. Mais precisamente, $c_1(i,u,j)$ é calculado conforme equação 3.12.

$$c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j) \tag{3.12}$$

Onde $\alpha_1 + \alpha_2 = 1;\ \alpha_1 \geq 0;\ \alpha_2 \geq 0;$

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij};\ \mu \geq 0; \tag{3.13}$$

$$c_{12}(i,u,j) = b_j^{new} - b_{j;} \tag{3.14}$$

Os parâmetros d_{iu}, d_{uj}e d_{ij} são as distâncias entre os clientes *i* e *u*, *u* e *j* e *i* e *j* respectivamente. O parâmetro μ controla o cálculo da economia em termos de distância, e b_j^{new}denota o novo tempo de início de atendimento do cliente *j*, dado que *i* foi inserido na rota, e b_j é o início do atendimento do cliente *j* antes da inserção.

O critério $c_2(i,u,j)$ é calculado como a seguir:

$$c_2(i,u,j) = \lambda d_{0u} - c_1(i,u,j);\ \lambda \geq 0; \tag{3.15}$$

O parâmetro λ é usado para definir o quanto o local de melhor inserção para o cliente não roteado depende de sua distância em relação ao depósito, por outro lado, dita o quanto sua melhor posição depende da distância extra e do tempo extra requeridos para visitar o cliente na rota corrente.

As três heurísticas citadas por Solomon (1987) possuem complexidade $O(n^2 \log n^2)$. A heurística *I1,* que apresentou resultados melhores que as demais, na maioria dos casos obteve soluções a mais de 10% da ótima. De qualquer forma, por ser uma heurística rápida, a qualidade da solução pode ser considerada adequada como uma solução inicial de uma metaheurística.

Dullaert (2000) e Dullaert & Bräysy (2003) argumentam que o critério $c_{12}(i,u,j)$ subestima o tempo adicional necessário para inserir um novo cliente *u* entre o depósito e o primeiro cliente na rota corrente em construção. Isto pode fazer com que o critério de inserção selecione uma posição de inserção que não é a melhor. Então, uma rota com um número relativamente pequeno de clientes pode ter um tempo de planejamento maior que o necessário. O autor modifica este critério de inserção e tem uma importante melhoria nos resultados.

Ainda em Solomon (1987) é proposto uma heurística chamada de "Heurística de Varredura Tempo-Orientada". Na primeira etapa (fase de grupamento) atribui-se clientes aos veículos conforme Gillet & Miller (1974), descrita no capítulo 2. Na segunda fase, é feito a construção das rotas aplicando-se uma heurística de construção, por exemplo, *I1*, a cada um dos grupamentos. Entretanto, nesta fase, alguns clientes dos grupamentos podem não estar inclusos na solução (em função da janela de tempo). Se isso ocorrer, o processo de *Gillett & Miller* é reiniciado usando apenas os clientes não roteados.

3.2.1.1 – Algoritmos de Melhoria

A vizinhança de uma rota *r* consiste em um conjunto de rotas que podem ser obtidas a partir de *r* através da substituição de *k* de seus arcos por outro conjunto de *k* arcos. Tais substituições são chamadas de *k*-trocas e uma rota que não pode ser melhorada por uma *k*-troca é dita *k*-ótima. Verificar a *k*-otimalidade requer um tempo $O(n^k)$.

Russel (1977) é um dos primeiros trabalhos em um PRVJT com o uso de uma heurística de melhoria *k*-ótima. A chamada abordagem "M-Rotas" foi capaz de resolver problemas com poucos clientes possuindo restrições de janelas de tempo. Para um problema com 163 clientes, onde 15 % deles possuíam janelas de tempo, uma solução foi encontrada em menos de 90 segundos, o que para a época é satisfatório.

Potvin & Rousseau (1995) comparam diferentes heurísticas de troca para o PRVJT (2-Opt, 3-Opt e Or-Opt) e introduz um novo 2-Opt* cuja principal idéia é combinar 2 rotas de modo que os últimos clientes de uma dada rota são introduzidos após os primeiros clientes da outra rota, de modo a preservar a orientação das rotas. Este operador é ilustrado na figura 3.1, onde os quadrados em branco são os depósitos (duplicados) e os círculos em preto são os clientes. Os arcos *(i, i+1)* e *(j, j+1)* são substituídos por *(i, j+1) e (j, i+1),* ou seja, a parte final de ambos são trocados. Como um caso especial, é possível combinar 2 rotas em uma única rota se o arco *(i, i+1)* é o primeiro em sua rota e o arco *(j, j+1)* é o último sem sua rota ou vice-versa. Uma abordagem híbrida baseada no 2-Opt* e Or-Opt mostra-se particularmente poderosa. Esta abordagem oscila entre as 2 vizinhanças trocando o operador cada vez que um ótimo local é encontrado. Os autores também testam uma implementação onde os 2

operadores são unidos. As soluções iniciais são criadas com o algoritmo *I1* de Solomon (1987).

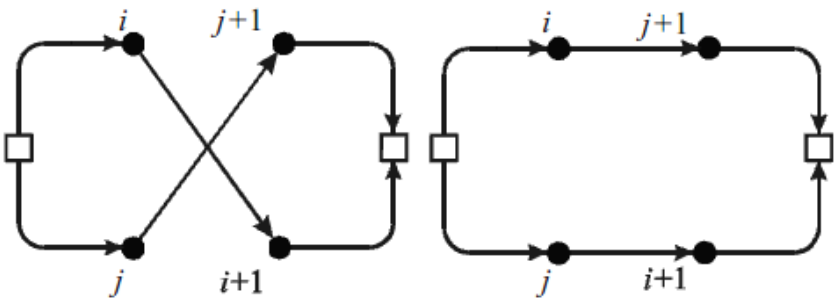


Figura 3.1: 2-Opt* (Potvin & Rousseau, 1995)

Outro método muito útil em problemas com janela de tempo é o *Cross Exchange* (Taillard, 1997). O método é ilustrado pela figura 3.2.

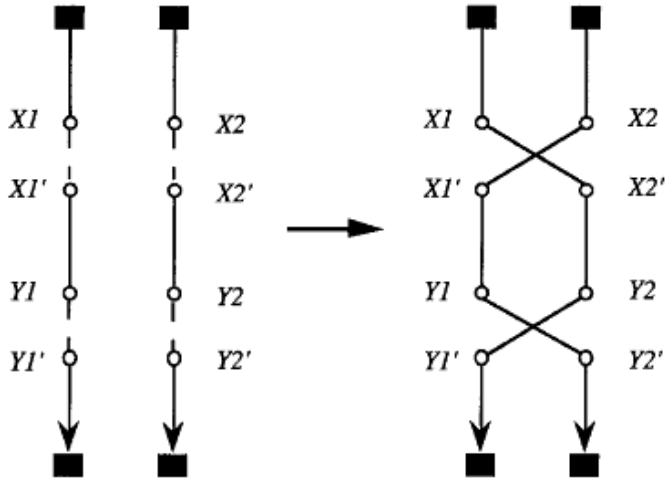


Figura 3.2: *Cross Exchange* (Taillard, 1997)

Trata-se de um método que realiza movimentos inter-rotas. Na figura 3.2, temos duas rotas, onde os quadrados em preto representam o depósito (duplicado para simplificar o desenho) e os círculos brancos são os clientes. Inicialmente os 2 arcos *(X1, X1')* e (Y1, Y1') são removidos da primeira rota enquanto os arcos (X2, X2') e *(Y2, Y2')* são removidos da segunda rota. Então os segmentos *X1'-Y1* e *X2'-Y2*, os quais possuem uma quantidade arbitrária de clientes entre eles, são trocados pela introdução dos novos arcos *(X1, X2'), (Y2, Y1'), (X2, X1')* e *(Y1, Y2')*.

Note que as restrições de janela de tempo definem uma orientação implícita em cada rota. No exemplo da figura 3.2, *Y1* é visitado após *X1'*. O segmento *X1'-Y1* tem a mesma orientação após o movimento. O mesmo ocorre para *Y2* e *X2'*.

O caso *2-Opt** ilustrado na figura 3.1 é um caso especial do *Cross Exchange*. O *2-Opt** troca 2 arcos tomados de 2 diferentes rotas, estando os arcos sempre diretamente conectados ao depósito. O *Or-Opt* também é um caso especial do *Cross Exchange*, onde há 3 ou menos clientes entre os 2 pontos de corte de cada uma das rotas envolvidas.

3.2.2 – Metaheurísticas para o PRVJT

Todas as metaheurísticas citadas no capítulo 2 podem e são adaptadas por muitos pesquisadores para resolver o PRVJT. É preciso obviamente incluir nos algoritmos testes que garantam a viabilidade da solução no que diz respeito às restrições de janela de tempo dos clientes.

Trabalhar com esta restrição no algoritmo tem suas particularidades. Tanto na construção de uma solução como em buscas de melhoria e refinamento, movimentos são realizados modificando a solução corrente. Uma particularidade é como analisar se o movimento irá gerar uma solução factível. Para sermos mais específicos, definamos uma rota parcial factível ($i_0, i_1,..., i_m$), na qual i_0 e i_m representam o depósito. Seja u um cliente ainda não adicionado a uma rota. Suponha, ainda que, o nó u será incluído entre os nós i_{p-1} e i_p que pertencem à rota. Observe que isso só poderá ocorrer se a rota continuar factível. Para a análise de viabilidade, entretanto, não basta verificar as restrições de capacidade e de janela de tempo para o nó que será inserido, uma vez que, a partir do nó i_p, os instantes de início de atendimento, b_{ik}, para $k = p,..., m$; podem ter seus valores alterados.

Com a finalidade de facilitar a verificação de viabilidade de uma rota, após a inserção de um cliente, Solomon (1987) criou um procedimento denominado *Push Forward*, que tem como objetivo transferir aos clientes posteriores a u, o acréscimo de tempo que a inclusão deste acarreta nos próximos atendimentos. Assim, se $b_{i_p}^{new}$ é o novo instante de início de atendimento do cliente i_p, definimos o *Push Forward* deste cliente como:

$$PF_{i_p} = b_{i_p}^{new} - b_{i_p} \tag{3.16}$$

Para os próximos clientes da rota, esse aumento no tempo é dado por

$$PF_{i_k} = max\{0, PF_{i_{k-1}} - w_{i_k}\}, k = p+1, \dots, m; \tag{3.17}$$

Onde $w_{i_j=} b_{i_j} - b_{i_{j-1}} - s_{i_{j-1}} - w_{i_{j-1}i_j}$ é o tempo de espera para o início do atendimento do cliente i_k. Note que podemos ter PF = 0. Isto ocorre quando o tempo de espera é maior ou igual ao aumento ocorrido no PF anterior. Neste caso, não haverá alterações no início de atendimento dos próximos clientes. A inclusão do depósito ao final da rota tem a finalidade de garantir que o veículo retorne dentro do intervalo de tempo especificado. Esta é uma forma eficiente encontrada por Solomon (1987) para testar a viabilidade ao se adicionar um novo cliente em uma rota parcial, sendo bastante útil ao realizar movimentos tanto nas metaheurísticas quanto nas heurísticas.

Uma outra particularidade do PRVJT que muda sua abordagem em relação ao PRVC num método aproximado é o fato de que num PRVC a direção em que os clientes de uma rota são percorridos não importa, ou seja, a rota pode ser percorrida tanto no sentido horário quanto no anti-horário e as restrições do problema continuarão a ser satisfeitas. Já num PRVJT, a direção em que a rota é percorrida é importante, existindo uma correlação entre a janela de tempo dos clientes e a ordem em que os clientes aparecem nas rotas. Esta característica pode ser útil em estratégias de exploração do espaço de busca (Nagata, 2007).

Metaheurísticas têm sido o foco de muitos trabalhos recentes envolvendo o PRVJT. Taillard *et al.* (1997) descreveu uma metaheurística baseada na Busca Tabu para o *VRP with Soft Windows*. Trata-se de um tipo de PRVJT em que é permitido atender um cliente mesmo após o fim da janela de atendimento, porém em contrapartida, há uma penalidade atribuída à solução, proporcional ao tamanho da violação da restrição. Porém, o mesmo modelo do autor pode ser utilizado para situações em que nenhuma violação é permitida, aumentando-se fortemente as penalidades pelo atraso no atendimento ao cliente. A metaheurística é baseada no conceito de memória adaptativa introduzida por Rochat & Taillard (1995) e no procedimento de decomposição e reconstrução proposto por Taillard (1993). A memória adaptativa utiliza-se de um conjunto de rotas provenientes de boas soluções encontradas durante a busca. Ao selecionar-se rotas deste conjunto de soluções para formar uma nova solução, obtém-se uma solução parcial que é completada utilizando a heurística de inserção *I1* proposta por Solomon (descrita no tópico 3.2.1). A cada iteração, uma solução é construída através de um processo de seleção aleatório onde rotas pertencentes a soluções com melhor avaliação da função objetivo têm maior probabilidade de serem selecionadas na lista de rotas que compõem a memória adaptativa. Esta solução é então melhorada através de

repetidos chamados a uma heurística de Busca Tabu. As rotas desta solução melhorada são armazenadas na memória adaptativa se a mesma não estiver completa ou se a solução for melhor que as soluções que já estavam na lista da memória adaptativa. Então o processo continua até que um critério de parada seja satisfeito.

Os chamados feitos à heurística de Busca Tabu são orientados por um mecanismo de decomposição e reconstrução de rotas que particiona, através de um procedimento de varredura, a solução corrente em um número de subconjuntos disjuntos de rotas. Cada subconjunto é tratado por um outro procedimento Tabu e as melhores rotas para cada subconjunto são unidas para formar uma nova solução para o próximo passo do processo de decomposição e construção. Estes passos são repetidos por um certo número de iterações e a decomposição muda de uma iteração para a outra, escolhendo diferentes ângulos de partida para a geração de novas soluções no processo de varredura. A Busca Tabu implementada é clássica, e consiste em escolher a cada iteração a melhor solução não tabu na vizinhança da solução corrente. Esta vizinhança é criada por um procedimento de troca chamado *Cross Exchange* que troca seqüências de clientes consecutivos entre 2 rotas. Este operador aplica os conceitos do 2-Opt* (Potvin and Rousseau, 1995) e do *Or-Opt* (Or, 1976) que são casos especiais do *λ-interchanges* (Osman, 1993) desde que os subconjuntos de clientes escolhidos para troca em cada rota sejam consecutivos. Para otimizar rotas individuais, a vizinhança é aumentada pela inclusão de trocas tipo *Cross* aplicadas a uma mesma rota: 2 arcos são removidos de 1 rota e o segmento entre os 2 arcos é movido para outra localização dentro da mesma rota.

Alvarenga (2005) desenvolveu um algoritmo populacional para resolver o VRPJT nas versões: estática e dinâmica. Há também uma versão que utiliza a distância total percorrida como único objetivo e outra que utiliza a redução do número de veículos como primeiro objetivo e a distância total percorrida somente como segundo. Uma seleção baseada em um processo de torneio utilizando critérios hierárquicos foi proposta para o algoritmo genético. O trabalho obteve bons resultados e interessantes métodos para eliminação de rotas foram desenvolvidos.

3.3 – PROBLEMAS TESTE DE SOLOMON

A heurística para o PRVJT proposta nesta dissertação foi validada utilizando-se os problemas propostos por Solomon (1987). Estes problemas são baseados em alguns dados de problemas utilizados por Christofides *et al.* (1979) para o PRV capacitado. Trata-se de diferentes classes de instâncias, cada qual com características geográficas e de restrições específicas. Os problemas foram propostos nas dimensões 25, 50 e 100 consumidores. São também divididos em seis grupos: R1, R2, C1, C2, RC1 e RC2. Em todos os casos, os consumidores estão geograficamente distribuídos em um quadrado no plano XY com dimensões 100x100.

Algumas características que afetam o desempenho dos algoritmos aplicados ao PRVJT são: distribuição geográfica dos clientes, número de clientes atendidos por um veículo, proporção de clientes com janelas de tempo e o tamanho das janelas de tempo.

Nas classes R1 e R2 os clientes estão uniformemente distribuídos. Nas classes C1 e C2, os clientes estão agrupados. E nas classes RC1 e RC2 há mistura entre as duas classes anteriores. As classes R1, C1 e RC1 possuem um horizonte de tempo curto (janela do veículo ou depósito) e permitem poucos clientes por rota (aproximadamente de 5 a 10). Diferentemente, as classes R2, C2 e RC2 possuem um horizonte de tempo mais longo permitindo que muitos clientes (mais de 30) sejam atendidos por um mesmo veículo.

Uma variação importante é a diminuição ou aumento da janela de tempo dos consumidores. Se por um lado, pequenos intervalos como janelas de tempo diminuem significativamente o espaço de busca, o que pode ser desejável por um lado, requer complexos operadores de obtenção de vizinhança nas heurísticas de busca local (Alvarenga, 2005).

As figuras 3.3 a 3.5 permitem visualizar bem as diferenças das classes R, C e RC quanto à distribuição geográficas dos clientes.

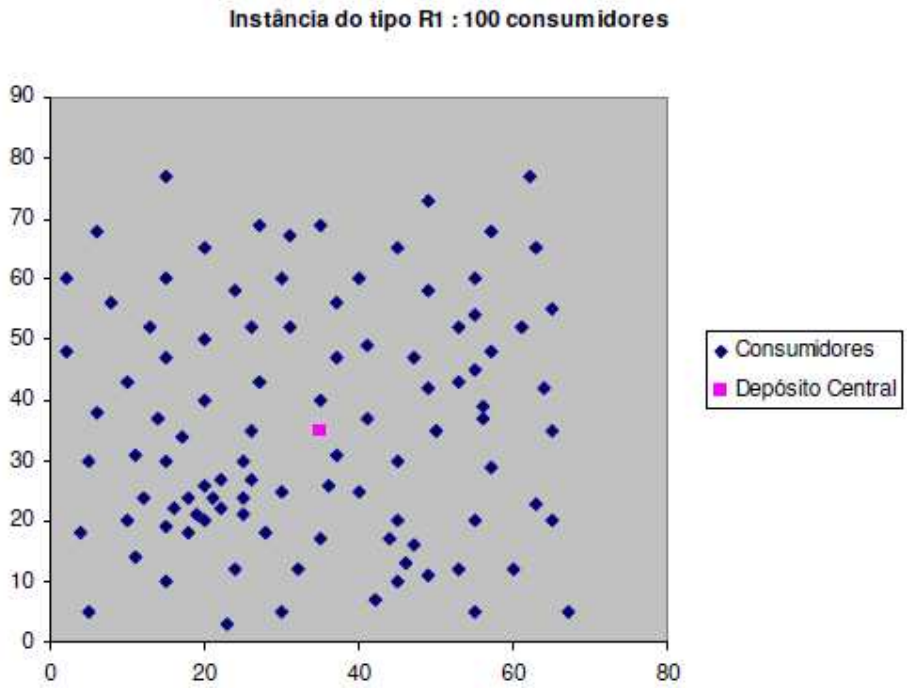


Figura 3.3: Disposição espacial para as classes R1 e R2 (Alvarenga, 2005)

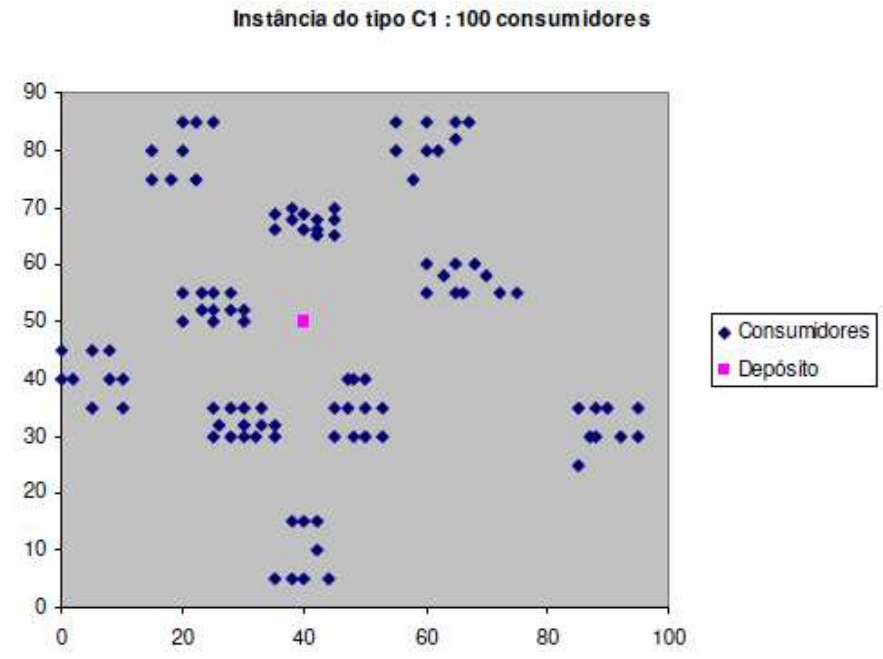


Figura 3.4: Disposição espacial para as classes C1 e C2 (Alvarenga, 2005)

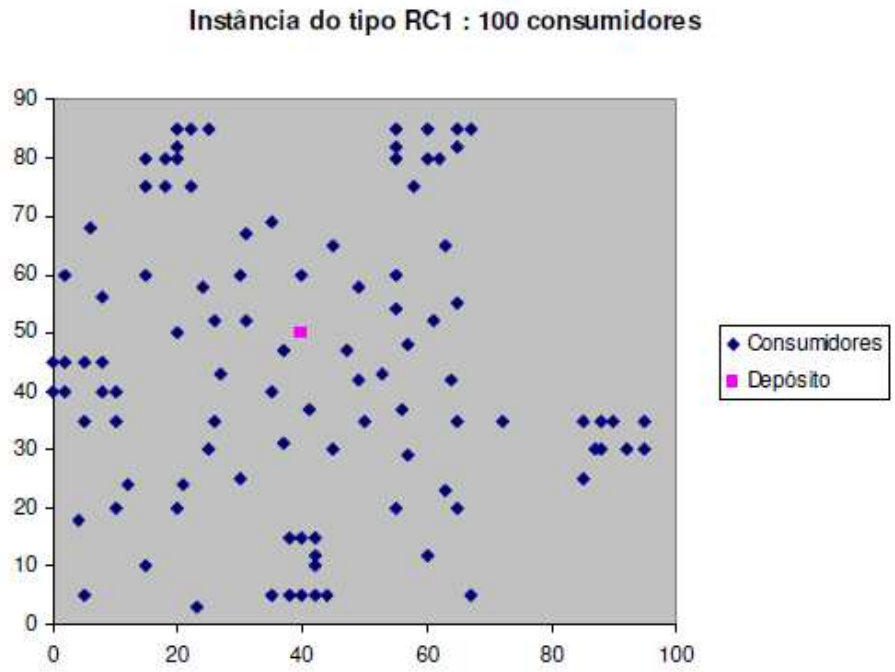


Figura 3.5: Disposição espacial para as classes RC1 e RC2 (Alvarenga, 2005)

As diferentes classes de problemas de Solomon têm como objetivo estabelecer cenários distintos para que diferentes métodos de solução possam ser avaliados de forma mais abrangente possível.

3.4 – DISCUSSÃO FINAL

Algoritmos exatos usando *branching* e *cutting* nas soluções obtidas através da decomposição Dantzig – Wolfe têm guiado a área de pesquisa dos métodos exatos. Seu sucesso reforça o uso de desigualdades válidas para apertar os *lower bounds* para o VRPTW. Estes avanços devem gerar maior interesse em resolver de forma ótima também instâncias de maior escala, como Fukasawa *et al.* (2006) já provaram ser possível.

Faz-se fato que as metaheurísticas têm aumentado muito seu escopo de atuação e encontrado soluções cada vez mais próximas do ótimo em tempos cada vez mais curtos e para instâncias cada vez maiores, como mostrado em diversos trabalhos como: Rochat & Taillard (1995), Homberger & Gehring (1999), e Cordeau *et al.*(2000), entre outros.

O presente trabalho considerou importante aprofundar-se no estudo e implementação do PRVJT, pois como no VRP Capacitado, não encontrou-se na

literatura algoritmos que utilizassem numa mesma metaheurística as técnicas: *ILS*, *VND*. Busca Tabu, Memória Adaptativa e *GLS*. Também nota-se que o PRVJT aproxima-se mais da realidade que o PRVC e ainda é uma importante ponte para uma implementação satisfatória do PRV Estocástico, visto a seguir.

CAPÍTULO 4

PRV ESTOCÁSTICO

No PRV clássico, é assumido que parâmetros do problema como custos, distâncias, demanda dos clientes, tempo de viagem dos veículos entre outros, são todos conhecidos (ditos determinísticos) e não mudam (ditos estáticos). No mundo real, esta abordagem falha em capturar alguns aspectos essenciais do problema. Um aspecto dos problemas do mundo real que mostram a necessidade de modificar o modelo clássico é a incerteza. Isto significa que um ou mais parâmetros do modelo não são inicialmente conhecidos ou são conhecidos apenas probabilisticamente. Isto dá origem ao Problema de Roteamento de Veículos Estocástico (PRVE).

Outra situação ocorre quando algumas entradas do modelo tornam-se conhecidas apenas quando a execução do modelo já começou, originando o Problema de Roteamento de Veículos Dinâmico (PRVD). Devido a recentes avanços nas tecnologias de informação e telecomunicação, estas classes de problemas têm se tornado importantes, dado a existência de potenciais ganhos financeiros em incluir modelos estocásticos e dinâmicos em aplicações do mundo real. O PRVD não é foco do presente trabalho. Para estudos mais aprofundados citamos: Psaraftis (1980), Psaraftis (1995) e Larsen (2000).

Até aqui temos então condições de compreender uma classificação genérica para os problemas de roteamento. Eles podem ser do tipo determinístico ou estocástico. E também podem ser do tipo estático ou dinâmico.

Resumidamente:

Problemas de Roteamento Estáticos: Assume-se que todas as informações relevantes são conhecidas antes do início do processo de roteamento. Não há alteração das informações relevantes depois de iniciado o processo de roteamento.

Problemas de Roteamento Dinâmicos: Em oposição aos estáticos, assume-se que não são conhecidas todas as informações relevantes antes do início do processo de roteamento. As informações relevantes depois de iniciado o processo de roteamento podem sofrer alterações.

Problemas de Roteamento Determinísticos: todos os parâmetros do modelo, como tempo de viagem, demanda dos clientes e outros são assumidos conhecidos e constantes.

Problemas de Roteamento Estocásticos: Diferentemente do determinístico, o modelo assume que pelo menos 1 dos parâmetros são conhecidos ou estimados apenas probabilisticamente.

Nos capítulos 2 e 3, os modelos descritos eram essencialmente estáticos e determinísticos. E agora no capítulo 4, expandimos nossa visão para os problemas estocásticos.

4.1 – APROFUNDANDO-SE NO PRV ESTOCÁSTICO (PRVE)

O PRVE introduz elementos de incerteza nos parâmetros que são dados de entrada do modelo matemático. Isto faz com que algumas propriedades do PRV não sejam as mesmas para o PRVE e em geral, os métodos de solução do PRVE são mais complicados. Uma revisão da literatura sobre o PRVE é dado em Gendreau *et al.* (1996).

O PRVE pode ser brevemente classificado conforme os seguintes critérios:

1) Natureza da incerteza no problema: a incerteza pode estar presente em diferentes partes do problema. Podemos ter o PRV com clientes estocásticos, o PRV com demanda estocástica, o PRV com tempo de viagem estocástico e o PRV com tempo de serviço estocástico.

2) Método de modelagem: o método de modelagem pode ser um critério para se classificar o PRVE. O problema pode ser abordado usando-se programação estocástica, por exemplo: *Chance Constrained Program (CCP)* e o *Stochastic Program with Recourse (SPR)*. Outra possível abordagem para o modelo é vê-lo como um processo de decisão de Markov.

3) Técnica de solução: diferentes técnicas de solução são uma conseqüência da natureza da incerteza no problema e do método de modelagem. Normalmente implicam em 2 categorias: métodos exatos e métodos heurísticos.

Programas estocásticos são modelados em 2 estágios. No primeiro, uma solução "*a priori*" é determinada de modo a revelar os valores dos parâmetros associados a variáveis aleatórias. Então no segundo estágio uma ação corretiva (recurso) é aplicada ao primeiro estágio da solução. O recurso geralmente cria uma economia ou um custo que deve ser considerado.

Por exemplo, seja um PRVE com demanda estocástica, onde o parâmetro determinístico d_i (demanda do cliente i) é substituído pela variável aleatória ε_i. O primeiro estágio consiste em encontrar as m rotas de veículos onde todos os clientes sejam visitados uma vez. Após o primeiro estágio ser executado, a verdadeira demanda de cada cliente torna-se conhecida. Então pode ser que realizar as rotas como planejado inicialmente tenha se tornado impossível, pois alguma restrição pode ter sido violada, como a capacidade do veículo de uma determinada rota, por exemplo. Uma possibilidade para o segundo estágio seria percorrer uma certa rota até que sua capacidade fosse violada. Então retornar ao depósito e depois continuar a visitar os clientes da rota inicial.

Como dito anteriormente, um programa estocástico é comumente modelado como um *CCP* ou *SPR*. Um *CCP* força que a probabilidade de satisfazer as restrições com parâmetros estocásticos sejam superiores a um valor previamente estabelecido, contudo, os custos de ações corretivas não são considerados em sua formulação. Ao contrário, um *SPR* incorpora o custo esperado das ações corretivas do segundo estágio na função objetivo do modelo. Isto trás consideráveis dificuldades, porém é mais plausível.

Para um dado problema, ações corretivas (ou políticas recursivas) podem ser feitas de diversas formas. Para mais informações sobre as diferentes políticas que podem ser adotadas, ver Dror *et al*.(1989).

A seguir são descritos duas importantes classes de PRV estocásticos: uma com demanda e clientes estocásticos e outra com tempo de viagem estocástico.

4.1.1 – PRV com demanda e clientes estocásticos

Um PRV é dito com demanda estocástica quando pelo menos 1 dos clientes possui uma demanda associada a uma variável aleatória. Tillman (1969) propôs o primeiro algoritmo para este tipo de problema. Penalidades eram aplicadas sempre que

um veículo estava vazio ou com ocupação acima de sua capacidade máxima. O modelo foi baseado numa modificação no cálculo das economias proposto por Clarke & Wright (1964), visto no capítulo 2. Outra importante contribuição para este tipo de problema foi dada por Bertsimas (1992). O autor considera o caso em que a demanda é 1 com probabilidade p_i e em que a demanda é *0* com probabilidade $1-p_i$.

Um PRV com cliente estocástico é aquele em que pelo menos 1 cliente tem sua presença associada a uma variável aleatória, mas sua demanda é determinística. Numa abordagem *SPR*, os clientes não presentes no estágio são "pulados". Muitos artigos tratam casos em que a demanda é unitária. Waters (1989) considera um caso em que a demanda não é binária e faz uma comparação entre 3 políticas propostas por ele. Uma em que o veículo segue a rota inicialmente proposta sem "saltar" clientes que revelaram-se ausentes. Outra em que tais clientes são "saltados" pelo veículo. E uma terceira política em que a rota restante é re-otimizada sempre que um cliente é confirmado como "ausente".

Duas propriedades interessantes podem ser aplicadas a ambos os problemas: demanda estocástica e clientes estocásticos. Na primeira propriedade, mesmo que os custos de viagem sejam simétricos, o custo total da solução é dependente da direção em que a rota é percorrida (Dror & Trudeau, 1986). Uma segunda propriedade diz que quanto maior a capacidade dos veículos maior será o custo da solução (Jaillet & Odoni, 1988).

4.1.2 – PRV com tempo de viagem estocástico

O PRV com tempo de viagem estocástico descreve um ambiente de incerteza intrínseco das condições de tráfego das vias. Kao (1978) foi o primeiro a propor uma solução a partir de uma heurística baseada em programação dinâmica e com enumeração implícita para o PCV com tempo de viagem estocástico.

Laporte *et al*. (1992) propuseram 3 modelos para abordar o problema de forma exata: um primeiro modelo *CCP*, um segundo modelo *SPR* de 3 índices e um terceiro *SPR* com 2 índices. No problema abordado, há uma duração máxima para as rotas de forma que uma penalidade proporcional é aplicada ao veículo cujo tempo ultrapassar o limite estabelecido. Tais problemas ocorrem freqüentemente no contexto onde os motoristas devem ser pagos por horas extras de trabalho após o período normal.

Também é o caso quando veículos são usados para coletar dinheiro das filiais para a matriz, onde todas as transações registradas após um certo horário do dia são creditados no dia seguinte, implicando em perda financeira pelos juros envolvidos. No modelo *CCP* a função objetivo consiste em minimizar uma combinação linear dos custos das rotas e dos veículos e ao mesmo tempo assegurar que a probabilidade de que o tempo de duração de uma rota exceda um valor B seja no máximo igual a um dado valor α. Foi apresentado um algoritmo *branch-and-cut* para os 3 modelos. Resultados mostraram que o segundo modelo pode ser usado para resolver de forma exata problemas de tamanho moderado.

Miranda & Conceição (2010) fazem um estudo de um problema de roteamento com tempos de viagem estocásticos e janelas de tempo, incluindo penalidades por violação na janela de tempo. Isto permitiu avaliar as rotas baseando-se na incerteza envolvida. O modelo com tempos de viagem estocásticos é comparado com um modelo determinístico simulando-se o funcionamento de uma empresa prestadora de serviços em manutenção. Também são realizados experimentos comparando instâncias capacitadas e não capacitadas, com janela de tempo e sem janela de tempo.

Comparado com os modelos com clientes e com demanda estocásticos, o modelo com tempo de viagem estocástico tem recebido menos atenção, embora não seja menos importante e sim, possivelmente, o mais interessante entre suas variantes (Gendreau, 2010). Segundo Li (2010), dentre os poucos trabalhos que tratam o tempo de viagem estocástico, a grande maioria não considera a presença de janelas de tempo para os clientes. Esta classe especial de problemas pode ser classificada como PRVJTTVE (Problema de Roteamento de Veículos com Janelas de Tempo com Tempo de Viagem Estocástico) que será melhor estudada a seguir.

4.1.3 – PRVJTTVE

O PRVJTTVE é o foco da modelagem estocástica realizada neste trabalho. Nota-se que nos problemas determinísticos com janelas de tempo pode-se facilmente classificar as rotas entre viáveis e não viáveis, pois assume-se conhecer o momento em que o veículo chegará em cada cliente, sendo simples checar se isto ocorre antes da abertura da janela (portanto a rota é viável, há espera e o tempo de início de atendimento fica sendo a abertura da janela), dentro da janela (portanto a rota é viável, não há espera

e o tempo de início de atendimento é o mesmo do tempo de chegada) e após o fechamento da janela (rota não viável).

O PRVTVE (PRV com tempo de viagem estocástico) pode ter 3 tipos principais. O primeiro é o PRVTVETM (PRV com tempo de viagem estocástico e tempo máximo) em que não há janelas de tempo para os clientes, apenas para o veículo. O segundo é o PRVTVE *with time deadlines,* um tipo de problema em que só existe o tempo de fechamento das janelas, ou seja, o tempo de abertura de todas as janelas é zero. O terceiro é o PRVJTTVE (PRV com janela de tempo e tempo de viagem estocástico), onde o tempo de abertura da janela de tempo de um ou mais clientes é diferente de zero.

No caso do PRVTVE *with time deadlines* e PRVJTTVE, uma rota é considerada viável se a probabilidade do tempo de chegada do veículo num dado cliente ser menor que o final da janela deste cliente for maior ou igual ao nível de serviço desejado, ou seja, a probabilidade do veículo não violar o final da janela deve ser maior ou igual ao nível de serviço desejado. A probabilidade de violar a janela é calculada para cada um dos clientes da rota, ou seja, cada cliente tem seu próprio nível de serviço. O nível se serviço da rota é o menor nível de serviço dos clientes. No PRVTVETM uma rota é considerada viável se a probabilidade do veículo retornar ao depósito antes do tempo máximo desejado for maior ou igual ao nível de serviço desejado.

Normalmente, para estes 3 tipos de problemas, assume-se conhecida, através de dados históricos coletados em campo, a função distribuição de probabilidade do tempo de viagem entre cada um dos clientes e os parâmetros desta distribuição. Assume-se ainda que os tempos de viagem entre os clientes são independentes e identicamente distribuídos (Laporte *et. al*., 1992).

No PRVTVETM e no PRVTVE *with time deadlines,* assumindo-se as condições descritas no parágrafo anterior, a função distribuição de probabilidade do tempo de chegada em cada um dos clientes de uma dada rota é a mesma dos arcos. Ou seja, se o tempo de viagem entre os clientes segue uma distribuição Normal, então o tempo de chegada nos clientes também seguirá uma distribuição Normal. Devido a suposição de independência entre os arcos é possível somar as médias e as variâncias ao longo da rota. Conhecendo-se a distribuição do tempo de chegada num dado cliente e os parâmetros desta distribuição é possível calcular a probabilidade do veículo violar o

fechamento da janela de tempo (seja de um cliente ou do depósito) e então verificar o nível de serviço.

No PRVJTTVE, a existência de janelas de tempo com tempo de abertura diferente de zero torna o cálculo do nível de serviço mais difícil. Nesta situação, diferentemente das outras duas, existe tempo de espera dos veículos. O tempo de espera depende das janelas de tempo e do tempo de chegada nos clientes. Uma vez que o tempo de chegada é uma variável aleatória, o tempo de espera também é uma variável aleatória. Porém ela não terá necessariamente a mesma função distribuição de probabilidade dos arcos. Isto ocorre devido a existência de não linearidade causada pelas janelas de tempo. Sendo assim, mesmo supondo conhecida a distribuição de probabilidade dos arcos e mesmo assumindo-se que os arcos são independentes, há 2 dificuldades neste tipo de problema: conhecer o tipo de distribuição de probabilidade do tempo de chegada nos clientes e conhecer os parâmetros desta distribuição.

No contexto do PRVJTTVE três importantes estudos foram realizados por Li *et al*. (2010), Jula *et al*. (2006) e Chang *et al.* (2009). Esta dissertação investiga estes 3 estudos, sugere mudanças nos métodos utilizados e compara os resultados obtidos entre os 4 trabalhos (os 3 citados e este).

4.1.4 – Estudos realizados por Li *et al.* (2010)

O trabalho usa simulação estocástica para calcular as probabilidades envolvidas. Utiliza uma modelagem do tipo *CCP* e outra para o tipo *SPR*.

O modelo *CCP* (*Chance Constrained Programming Model*) no contexto do PRVJTTVE consiste em planejar um conjunto de rotas que está sujeito a restrições de probabilidade com o objetivo de assegurar que a probabilidade de falha na rota seja menor que um nível desejado. Similar a outros PRVs determinísticos, o modelo *CCP* do PRVJTTVE tem funções objetivos hierarquizadas, onde o objetivo primário é a redução do número de veículos e o objetivo secundário é a minimização da distância (ou tempo) total de viagem. Isto significa que uma solução com menor número de veículos e maior distância total é considerada melhor que uma solução com maior número de veículos e menor distância total.

O modelo *CCP* utilizado assim como a notação da formulação matemática são dados a seguir:

M: número de veículos utilizados (pode ser um parâmetro ou uma variável de decisão).

K: conjunto de veículos, definido como *K= {1, 2,..., m};*

Q: capacidade do veículo; aqui a frota é considerada homogênea.

B: tempo máximo de duração admissível para as rotas.

Q_i: demanda do cliente $i \in V$, onde *V* é o conjunto de nós. $V_0 = V \backslash \{0\}$ denota os clientes.

d_{ij}: distância de viagem entre os vértices *i* e *j*, sendo $i,j \in V$;

t_{ij}: tempo de viagem entre os vértices *i* e *j*. No artigo, t_{ij} é assumido ser uma variável aleatória contínua com distribuição normal truncada à esquerda no ponto zero.

δi: tempo de serviço no cliente *i*. Também assumido ser uma variável aleatória contínua com distribuição normal.

s_{ik}: tempo de chegada do veículo *k* no cliente *i*.

$[e_i, l_i]$: janela de tempo disponível para que o veículo atenda o cliente *i*.

w_i: tempo de espera do veículo no cliente *i*.

$$\min F_0(x) = M.f \sum_{j \in V_0} \sum_{k \in K} x_{0jk} + \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} d_{ij}.x_{ijk} \tag{4.1}$$

$$s.a. \sum_{j \in V} \sum_{k \in K} x_{ijk} = 1 \quad \forall i \in V_0 \tag{4.2}$$

$$\sum_{j \in V} x_{0jk} = 1 \quad \forall k \in K \tag{4.3}$$

$$\sum_{i \in V} x_{i0k} = 1 \quad \forall k \in K \tag{4.4}$$

$$\sum_{i \in V_0} x_{ijk} - \sum_{i \in V_0} x_{jik} = 0 \quad \forall j \in V_0, k \in K \tag{4.5}$$

$$\sum_{i \in V_0} q_i \sum_{j \in V} x_{ijk} \leq Q \quad \forall k \in K \tag{4.6}$$

$$P\left\{ \sum_{i \in V} \sum_{j \in V} t_{ij}.x_{ijk} + \sum_{i \in V_0} (\delta_i + w_i) \sum_{j \in V} x_{ijk} \leq B \right\} \geq \beta \quad \forall k \in K \tag{4.7}$$

$$P\{s_{ik} \leq l_i\} \geq \alpha, \qquad i \in V, \qquad k \in K \tag{4.8}$$

$x_{ijk} \in \{0,1\}\ \forall i,j \in V, k \in K$ (4.9)

No modelo acima, x_{ijk} é uma variável de decisão: x_{ijk}= 1 se o arco $(i,j) \in E$ (conjuntos dos arcos) é usado pelo veículo *k*; e x_{ijk}= 0 caso contrário.

Na equação 4.1, o parâmetro *M* é um número muito grande ($M = 10^{10}$). Ele é utilizado para assegurar que o número de veículos tenha prioridade na função objetivo. A restrição 4.2 indica que cada cliente deve ser visitado apenas 1 vez e por 1 veículo. Equações 4.3 a 4.5 determinam o fluxo no caminho a ser seguido pelo veículo *k*, ou seja, o veículo deve sair e retornar ao depósito, e deve sair de um nó quando o visita. A equação 4.6 assegura o respeito à capacidade do veículo. A equação 4.7 assegura o atendimento ao nível de serviço desejado para o tempo máximo de duração das rotas. Um baixo valor de β significa que o motorista do veículo tem maior probabilidade de fazer horas extras. A equação 4.8 assegura o respeito ao nível de serviço desejado para atender o cliente dentro de sua janela de tempo.

As equações 4.7 e 4.8 são restrições que afetam fortemente a viabilidade das soluções. Para muitas instâncias, valores altos para os parâmetros α e β podem fazer com o problema não tenha solução viável.

O autor prova que para o PRVTVE *with time deadlines,* se o tempo de viagem entre os arcos são independentes e identicamente distribuídos*,* é possível converter as restrições probabilísticas 4.7 e 4.8 em suas correspondentes determinísticas. Isto não é possível para o PRVJTTVE devido à existência do tempo de espera como uma variável aleatória.

O artigo também elabora um modelo *SPR* (*Stochastic Programming model with Recourse*). No modelo *CCP* é necessário que o nível de serviço de cada rota seja superior a um valor α. Isto significa que quando uma rota tem nível de serviço igual a α, sendo viável, existe uma probabilidade $(1-\alpha)$ de violação da janela de tempo. Então o modelo *SPR* considera um custo para esta probabilidade $(1-\alpha)$ e insere este termo na função objetivo como um termo de penalidade.

O autor desenvolveu uma heurística de roteamento baseada na Busca Tabu para resolver o problema. A verificação da viabilidade da rota quanto às restrições probabilísticas é feita através de simulação estocástica. Cada rota era simulada pelo menos 1000 vezes para se coletar os dados do tempo de chegada em cada cliente e então

contabilizar a quantidade de violações em cada cliente, estabelecendo o nível de serviço da rota.

Embora este método apresente uma boa qualidade para as soluções encontradas, o custo computacional é muito alto, de modo que mesmo utilizando heurísticas, apenas problemas de pequena instância podem ser resolvidos em tempo praticável.

4.1.5 – Estudos realizados por Jula *et al*. (2006)

Os autores investigam o PRV com tempo de viagem estocástico em 3 diferentes situações. Na primeira, o tempo de serviço e as janelas de tempo são desconsiderados. Na segunda, o tempo de serviço é uma variável aleatória, sendo independente e identicamente distribuída entre os clientes. Na terceira situação, considera-se a existência de janelas de tempo e seu efeito é estudado. Esta é a nossa situação de interesse, pois é a presença do tempo de espera como variável aleatória que torna o problema difícil de ser tratado.

Neste cenário, o autor sugere meios de se estimar a média e a variância do tempo de chegada do veículo nos clientes. Assume-se mais uma vez que o tempo de viagem entre os arcos são conhecidos probabilisticamente, ou seja, conhece-se a distribuição de probabilidade e seus parâmetros, sendo considerados independentes e identicamente distribuídos.

Seja $g(y_i^r)$ uma função não linear que denota o tempo de partida como função do tempo de chegada y_i^r em cada nó *i* de uma dada rota *r*.

Dado o tempo de chegada Y_i^r no nó *i* na rota *r,* o tempo de partida W_i^r é dado por:

$$W_i^r = g(Y_i^r) \tag{4.10}$$

Seja $f_{Y_i^r}(y_i^r)$ a função distribuição de probabilidade (FDP) da variável aleatória Y_i^r. O primeiro momento (média) do tempo de partida $W_i^r = g(Y_i^r)$ pode ser obtido por:

$$E[W_i^r] = \int_0^\infty g(y_i^r)\, f_{Y_i^r}(y_i^r) dy_i^r \tag{4.11}$$

Como é difícil obter a FDP da variável aleatória Y_i^r, estimativas são obtidas para aproximar a função $g(Y_i^r)$ na vizinhança de $E[Y_i^r]$ usando expansão por Série de Taylor.

Detalhes do desenvolvimento podem ser vistos no artigo. O autor chega às seguintes expressões para a média e variância do tempo de partida respectivamente:

$$E[W_i^r] = g(E[Y_i^r]) \tag{4.12}$$

$$var(E[Y_i^r]) = (g')^2\,(E[Y_i^r])\,var(Y_i^r) \tag{4.13}$$

As equações 4.12 e 4.13 conduzem a estimativas da média e variância do tempo de partida W_i^r num cliente *i* em termos da função $g(.)$ e da média e variância do tempo de chegada Y_i^r. Contudo deve ser notado que o uso das equações 4.12 e 4.13 pode gerar um problema: uma pequena diferença na média do tempo de chegada pode conduzir a uma grande diferença na variância do tempo de partida. A figura 4.1 ilustra o problema:

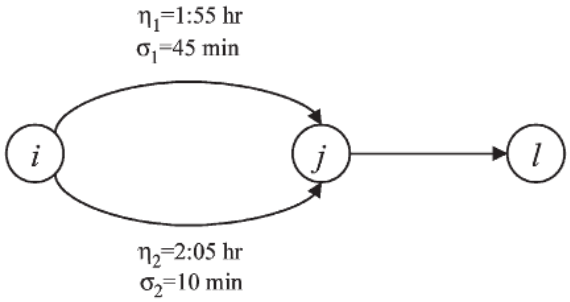


Figura 4.1: 2 rotas com diferentes características estocásticas (Jula, 2006)

Sejam as 2 rotas possíveis entre *i* e *j* descritas na figura 4.1. Assume-se que a média e o desvio-padrão do tempo de viagem entre os nós *i* e *j* na rota 1 são: η_1 =01h55min e σ_1 =45 min. Na rota 2 são: η_2 =02h05min e σ_2 =10 min. Assume-se também que a janela de tempo do cliente *j* seja [11h00min, 12h30min] e que o tempo de partida do cliente *i* seja 09h00min.

De acordo com as equações 4.12 e 4.13, ao percorrer a rota 1, a média e variância do tempo de partida será $E[W_j^1]$ = 11h00min e $var(W_j^1)$ = 0. Ao percorrer a rota 2 temos: $E[W_j^2]$ = 11h05min e $var(W_j^2)$ = 100. Ou seja, uma pequena diferença na media do tempo de chegada conduziu a uma grande diferença na variância do tempo de partida.

Para reduzir o problema, as equações 4.12 e 4.13 devem ser usadas apenas se as condições abaixo forem asseguradas:

$$\left\{ \begin{matrix} \left|E\left[Y_j^r\right] - a_j\right| \gg \sigma\left(Y_j^r\right) \\ \left|E\left[Y_j^r\right] - b_j\right| \gg \sigma\left(Y_j^r\right) \end{matrix} \right\} \tag{4.14}$$

Para os casos em que as desigualdades em 4.14 não são válidas, o cálculo da variância deve utilizar a equação 4.15:

$$\sigma\left(W_j^r\right) \cong \frac{1}{2} \int_{E\left[Y_j^r\right]-\sigma\left(Y_j^r\right)}^{E\left[Y_j^r\right]+\sigma\left(Y_j^r\right)} g'(x)dx \tag{4.15}$$

Para o exemplo da figura 4.1, as desigualdades em 4.14 não são satisfeitas. Utilizando a equação 4.15 temos: $\sigma\left(W_j^1\right)$ = 20 minutos e $\sigma\left(W_j^2\right)$ = 7.5 minutos. Isto mostra que percorrer a rota 2 é mais vantajoso se a variância for o foco da tomada de decisão.

Após propor estes métodos para estimar a média e a variância, o autor estuda a utilização de 2 métodos para estimar a probabilidade de violação da janela de tempo: Desigualdade de Chebyshev e Desigualdade de Chernoff.

Embora estes métodos não sejam descritos no artigo, uma breve descrição é dada no tópico a seguir.

O autor realizou experimentos com algumas rotas geradas a partir de um VRP com 5 nós e comparou as estimativas de média e variância com os resultados obtidos via simulação. Segundo o autor os resultados apresentaram um erro pequeno, sendo considerados satisfatórios. Os testes também mostraram que um aumento no nível de serviço desejado implicou num aumento do custo total da solução. Também se observou que a utilização da Desigualdade de Chernoff forneceu limites inferiores para o nível de serviço mais apertados que a Desigualdade de Chebyshev.

4.1.5.1 – Desigualdade de Chebyshev

Seja uma variável aleatória com média m_x e variância σ_X^2 finitas. Para todo número positivo δ:

$$P[|X - m_x| \geq \delta] \leq \frac{\sigma_X^2}{\delta^2} \tag{4.16}$$

Na desigualdade, o termo $P[|X - m_x| \geq \delta]$ é a probabilidade da variável aleatória X ter um valor na região sombreada da figura 4.2.

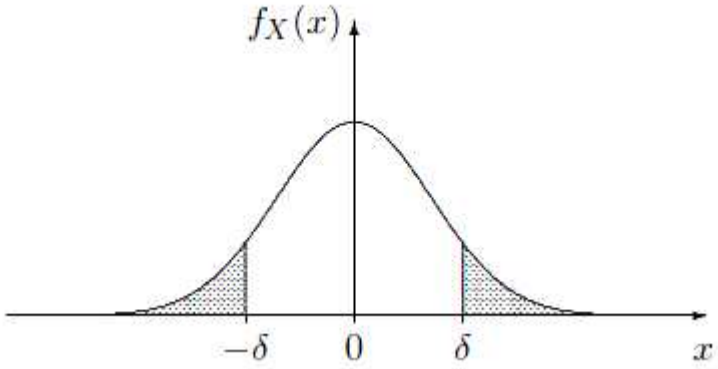


Figura 4.2: região da Desigualdade de Chebyshev

A Desigualdade de Chebyshev assume conhecimento da média e da variância da função distribuição de probabilidade, mas não precisa do conhecimento da forma ou do tipo da distribuição. Este é o caso ocorrido nos problemas com tempo de viagem estocástico em que existe tempo de espera. Portanto, uma vez desenvolvido um estimador para a média e variância, é plausível utilizar esta desigualdade para fornecer um limitante inferior para o nível de serviço de uma rota.

4.1.5.2 – Desigualdade de Chernoff

Para uma variável aleatória *X* e uma constante *c*:

$$P[X \geq c] \leq \min_{s \geq 0} e^{-sc} \phi X(s) \tag{4.17}$$

Na equação 4.17, o termo $\phi X(s)$ é a Função Geratriz de Momento da variável aleatória X. A desigualdade fornece limitantes mais apertados que Chebyshev, mas para isto, precisa conhecer a função densidade de probabilidade da variável aleatória.

4.1.6 – Estudos realizados por Chang *et al.* (2009)

Os autores estudam um problema do caixeiro viajante com janela de tempo, e com tempo de viagem estocástico e dinâmico. Para calcular o nível de serviço de uma rota, o autor se depara com as dificuldades citadas anteriormente. A primeira delas é

estimar a média e a variância do tempo de saída do veículo. Para isto, ele estuda a relação entre o tempo de chegada e a janela de tempo, sendo este um ponto de interesse desta dissertação.

Seja Y_i uma variável aleatória conhecida do tempo de chegada do veículo no cliente *i*. Seja *[l_i , u_i]* a janela de tempo. Seja também $\overline{\gamma}$ a probabilidade máxima admissível de violação da janela. Se $P(Y_i \geq u_i) > \overline{\gamma}$ a rota é descartada. Caso a rota seja considerada viável, considera-se que o veículo chegue definitivamente antes da abertura da janela se $P(Y_i \geq l_i) \leq \underline{\gamma}$, onde $\underline{\gamma}$ é uma constante pré-especificada, pequena e positiva. Neste caso, o tempo de início de atendimento no cliente *i* é dado por l_i. Caso contrário, se $P(Y_i \geq l_i) > \underline{\gamma}$ o veículo pode ou não chegar antes da abertura da janela. Então o tempo de início de atendimento é dado por $max(Y_i, l_i)$ e o tempo de partida do veículo é $max(Y_i, l_i) + S_i$, onde o tempo de atendimento do cliente *i* é $S_i \sim \aleph(v_i, \theta_i^2)$, ou seja, segue uma distribuição normal com média v_i e variância θ_i^2.

Há 3 cenários a serem considerados:

Cenário 1: O veículo chega mais cedo, ou seja, há tempo de espera. O início de atendimento começa em *li*, então o tempo de partida é $X_i \sim \aleph(l_i + v_i, \theta_i^2)$. Note que o tempo de início de atendimento é uma constante, e que a variância acumulada do tempo de chegada é zerada.

Cenário 2: O veículo chega tarde. $P(Y_i \geq u_i) > \overline{\gamma}$. Rota é considerada inviável, sendo descartada.

Cenário 3: O veículo chega dentro da janela de tempo do cliente. Seja $p_e = P(Y_i < l_i)$ a probabilidade de haver espera. Considerando que $P(Y_i \geq u_i)$ seja pequena, o veículo chega dentro da janela de tempo com probabilidade $P(l_i \leq Y_i \leq u_i) \approx P(l_i \leq Y_i \leq \infty) = 1 - p_e$. Então, $X_i = \begin{Bmatrix} l_i + s_i, & se\ Y_i < l_i \\ Y + S_i\ c.c. & \end{Bmatrix}$, ou seja, $max(Y_i, l_i) + S_i$, onde $P(Y_i < l_i) = p_e$. Assume-se que Y_i tem distribuição Normal.

A média do tempo de saída do cliente *i* é dada por:

$$E[X_i] = E[(max\{Y_i, l_i\} + S_i)] = \ l_i P(l_i > Y) + E\left(Y1_{\{Y>l\}}\right) + E(S_i) =$$

$$= l_i^2 \phi\left(\frac{l_i - E(Y_i)}{\sqrt{V(Y_i)}}\right) + \ E[Y_i]\left[1 - \phi\left(\frac{l_i - E(Y_i)}{\sqrt{V(Y_i)}}\right)\right] + \sqrt{V[Y_i]}\varphi\left(\frac{l_i - E(Y_i)}{\sqrt{V(Y_i)}}\right) +$$

$$+ E(S_i) \tag{4.18}$$

A variância do tempo de saída do cliente *i* é dada por:

$$V[X_i] = E[(max\{Y_i, l_i\} + S_i)^2] - E^2(X_i) = \ E[(max\{Y_i, l_i\})^2] + 2\mathrm{E}(S_i)\ E[max\{Y_i, l_i\}]$$

$$+\mathrm{E}\left(S_i^2\right) - E^2(X_i) = l_i^2 \phi\left(\frac{l_i - E(Y_i)}{\sqrt{V(Y_i)}}\right) + \left(V[Y_i] + E^2(Y_i)\right)\left[1 - \phi\left(\frac{l_i - E(Y_i)}{\sqrt{V(Y_i)}}\right)\right]$$

$$+ \ (E(Y_i) + l_i)\sqrt{V(Y_i)\varphi\left(\frac{l_i - E(Y_i)}{\sqrt{V(Y_i)}}\right)} +$$

$$+2\mathrm{E}(S_i)\left\{l_i \phi\left(\frac{l_i - E(Y_i)}{\sqrt{V(Y_i)}}\right) + E(Y_i)\left[1 - \phi\left(\frac{l_i - E(Y_i)}{\sqrt{V(Y_i)}}\right)\right] + \sqrt{V[Y_i]}\varphi\left(\frac{l_i - E(Y_i)}{\sqrt{V(Y_i)}}\right)\right\} +$$

$$+\mathrm{E}(S_i^2) - E^2(X_i) \tag{4.19}$$

O modelo desenvolvido no trabalho assume que o tempo de chegada no cliente segue uma distribuição normal. Uma rota com 12 clientes foi gerada e simulada. Realizou-se um teste de adesão nos dados coletados aplicando-se o teste de Kolmogorov-Smirnov, e o autor considerou que a adesão à distribuição normal foi satisfatória.

No artigo, realizou-se experimentos para a mesma instância de 12 clientes com o objetivo de comparar o modelo estocástico com o determinístico. Os resultados mostram que a distância total percorrida no modelo estocástico foi maior que no modelo determinístico. Além disto, 2% das soluções geradas no modelo determinísticos eram inviáveis enquanto que no modelo estocástico isto não ocorreu.

4.2 – DISCUSSÃO FINAL

De fato, poucos trabalhos na literatura investigam de forma profunda o PRVJT com tempo de viagem estocástico. A presença do tempo de espera como uma variável aleatória faz com que os tempos de chegada nos clientes sigam uma função densidade de probabilidade desconhecida e variável ao longo dos clientes de uma rota.

Li *et al.* (2010) resolve este problema via simulação estocástica. Os resultados quanto à qualidade da solução são bons, mas é fácil perceber que o custo computacional é muito alto. No modelo determinístico, quando deseja-se checar a viabilidade de uma rota quanto ao respeito às janelas de tempo, percorre-se o vetor com os clientes da rota uma única vez. No modelo estocástico, isto é feito pelo menos 1000 vezes para que a amostragem dos dados seja suficientemente grande. Num algoritmo aplicado a resolver um VRP, esta verificação é feita muitas vezes, então simular todas estas rotas faz com que seja possível resolver apenas problemas pequenos mesmo com o uso de heurísticas.

Para contornar este problema (alto custo computacional da simulação), Jula *et al.* (2006) e Chang *et al.* (2009) propuseram métodos para estimar a média e a variância do tempo de chegada nos clientes. Também foram propostos meios de calcular a probabilidade de violação das janelas sem a necessidade da simulação.

Em Jula *et al.* (2006), embora o autor tenha considerado o resultado dos experimentos satisfatórios, nota-se que os experimentos foram realizados a partir de uma única e pequena instância de 5 nós, com janelas de tempo largas (cerca de 40% da janela do depósito). Em Chang *et al.* (2009), problema similar ocorre. Os experimentos foram realizados a partir de uma instância para o PCV com 12 clientes. As janelas dos clientes eram largas (50% da janela do depósito). Além disto, realizou-se um teste de adesão para a distribuição normal que pode ser considerado frágil, por alguns motivos: a instância utilizada é muito simples, o nível de confiabilidade utilizado no teste de hipótese é baixo (85%, quando o mais utilizado é 90% ou 95%), além disto, o tempo de chegada em um dos clientes não teve adesão ao teste.

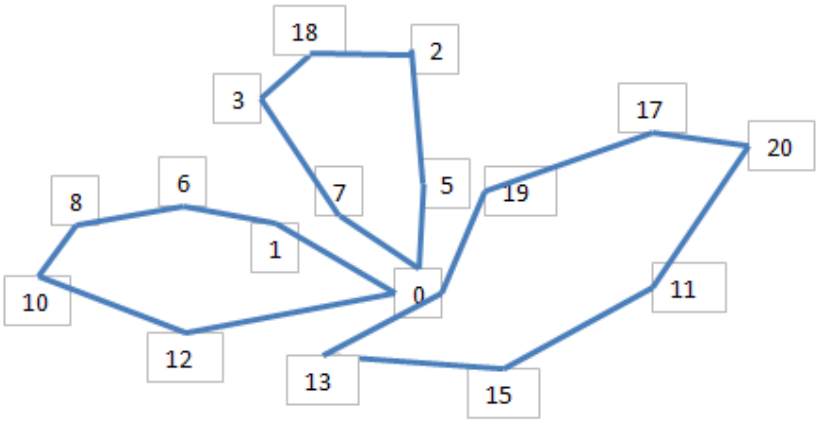
Desta forma, percebe-se que os trabalhos anteriores não conseguiram dar um tratamento adequado ao efeito do tempo de espera no tempo de chegada do veículo. Assim, uma pergunta continua em aberto: como calcular a probabilidade de violação da janela de tempo dos clientes sem o uso da simulação? Observa-se que uma lacuna na literatura foi encontrada. Mesmo estudos recentes propuseram métodos que não conseguiram resolver de forma satisfatória as dificuldades originadas pelo efeito das janelas de tempo no tempo de espera como uma variável aleatória. Sendo, assim, esta dissertação investiga alternativas e possíveis contribuições para avançar nesta questão.

CAPÍTULO 5

ALGORITMOS DESENVOLVIDOS PARA O PRV DETERMINÍSTICO

Neste capítulo, as metaheurísticas desenvolvidas são discutidas e testadas. Elas são baseadas em uma ou mais das seguintes metaheurísticas: *Variable Neighborhood Descent (VND), Iterated Local Search (ILS), Tabu Search (TS), Guided Local Search (GLS) e Adaptive Memory Procedure (AMP).* Nas seções seguintes, as heurísticas desenvolvidas são apresentadas e informações teóricas mais específicas, além das presentes nos capítulos anteriores, sobre os conceitos utilizados são mostrados conforme necessidade.

Em todos os algoritmos implementados neste trabalho, cada solução tem uma representação matricial e seqüencial, ou seja, cada solução é uma matriz em que cada linha é uma rota, e nesta rota os clientes são alocados na seqüência em que a rota é percorrida. Numa solução qualquer, digamos SOL_{ij}, o índice i refere-se à rota e o índice j refere-se ao cliente. Pela figura 5.1, fica subentendido que o elemento antes do primeiro cliente e após o último cliente de cada linha é o depósito. Esta representação é conhecida como *Multi-part Representation.* Esta e outras representações podem ser vistas em Potvin (2007).



1	6	8	10	12	0	0
5	2	18	3	7	0	0
19	17	20	11	15	13	0

Figura 5.1: Solução com 3 rotas e sua correspondente representação matricial

5.1 - PRV CAPACITADO: HEURÍSTICA H1

A primeira heurística desenvolvida é chamada aqui de *H1*. Consiste basicamente de um mecanismo *ILS* com fase de melhoria *VND*. A solução inicial do algoritmo é produzida através de duas heurísticas rápidas, a heurística de varredura de Gillet & Miller, e a heurística de economias de Clarke & Wright versão paralela. A seguir um procedimento rápido e conseqüentemente parcial de melhorias é feito nas soluções iniciais geradas, de forma que a melhor solução será escolhida para seguir adiante (conceito de "*false starts*" visto no tópico 2.4.2.1).

Então inicia-se o processo de melhoria (segunda fase do algoritmo) onde um conjunto de buscas locais é aplicado à solução para realizar melhorias segundo o conceito *VND* (descrito na sessão 2.4.2.5). O processo de melhorias do *VND* é executado até que mais melhorias não são mais alcançadas. Então um procedimento para perturbar a solução é iniciado. Este procedimento permite haver alguma piora na solução com o objetivo de fugir do ótimo local em que o procedimento anterior ficou preso. Isto é feito basicamente a partir de operadores genéticos.

A solução gerada por este procedimento de perturbação é então a entrada do procedimento de melhorias da segunda fase do algoritmo, caracterizando o mecanismo *ILS*. Todo este processo é repetido de forma que após certa quantidade de iterações sem melhorias, o algoritmo é encerrado. O pseudocódigo do corpo principal do algoritmo é dado a seguir:

Algoritmo 5.1: *H1*

```
01: s0:= MelhorSoluçãoInicial;
02: s:=s0;
03: s*:= s;
04: s':= BuscaLocal(s);
05: se s' for melhor do que s
06:         s:= s';
07:         retornar para passo 4;
08: fim se
09: se s for melhor do que s*
10:         s*:= s;
11:         conta:=0;
12: fim se
13: se conta > max
14:         Terminar e Retornar s*
15: fim se
16: s:= GerarPerturbação(s*);
17: conta:=conta+1;
18: retornar para linha 4;
```

Onde s_0 é a solução inicial ou "falsa solução inicial"; *s* é a solução corrente em que o algoritmo está trabalhando; *s'* é a solução retornada após os processos de buscas locais; *s** é a melhor solução encontrada pelo algoritmo; *conta* é um contador do número de iterações consecutivas em que não houve melhoria em *s** e *max* é o critério de parada, neste caso, número máximo de iterações consecutivas sem melhoria em s*. A função *BuscaLocal* retorna sempre a melhor solução encontrada, sendo composta por diferentes estruturas de vizinhança que serão descritas adiante.

O funcionamento de H1 também pode ser visualizado pela figura 5.2:

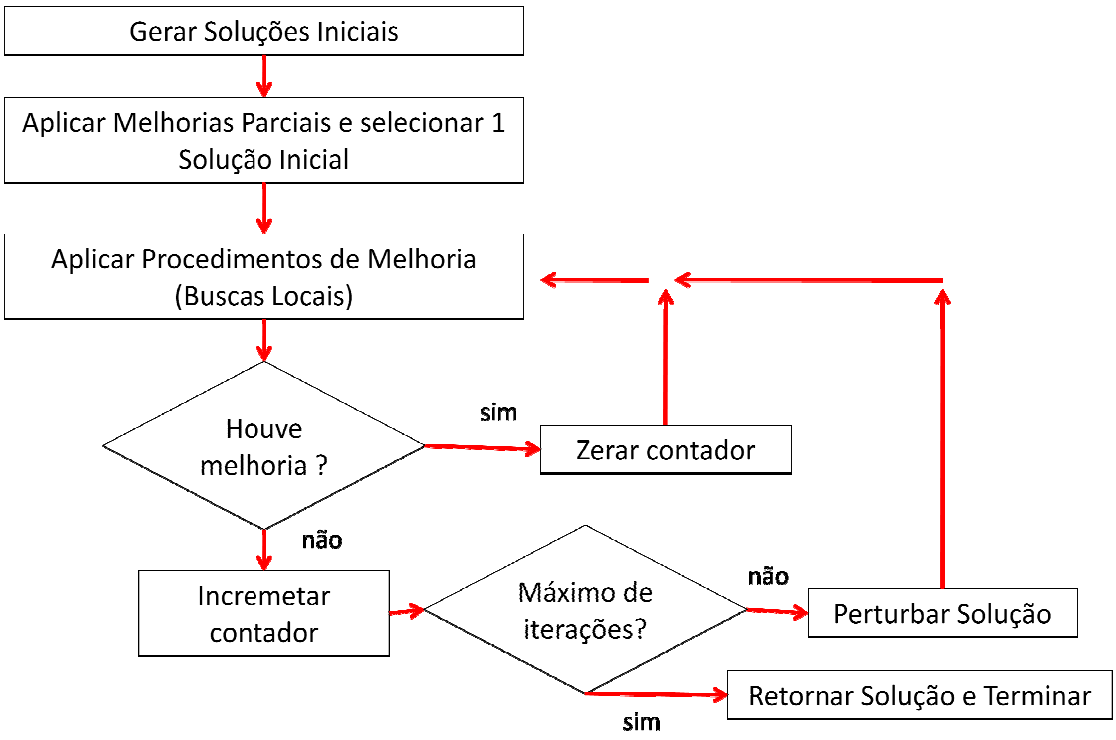


Figura 5.2: funcionamento de H1

5.1.1 – Solução Inicial

Uma solução inicial de boa qualidade pode acelerar o processo de busca local. Para gerar soluções iniciais utilizou-se as heurísticas: Gillet & Miller (1974) e Clarke & Wright (1964). Outras heurísticas construtivas para gerar soluções iniciais foram implementadas: Algoritmo do Vizinho Mais Próximo (Solomon, 1987), Fisher & Jaikumar (1981), Mole & Jameson (1976) e Gaskell (1967). Testes realizados mostraram que estas heurísticas não trouxeram benefícios significativos para a metaheurística. Embora tenham retornado boas soluções em alguns casos, a maioria mostrou-se muito sensível aos parâmetros de entrada utilizados. Portanto, selecionou-se apenas Gillet & Miller (1974) e Clarke & Wright (1964). Ambas foram utilizadas por serem rápidas e fornecerem soluções que sirvam como uma solução inicial de boa qualidade. As duas mostraram-se complementares no sentido de que a primeira mostra-se melhor em soluções em que há pouca "mistura" entre as rotas, e a segunda em soluções em que há maior "mistura" entre as rotas, ou seja, há mais cruzamentos entre diferentes rotas e/ou uma parte de uma rota está "dentro" de outra rota.

São geradas 3 soluções iniciais, 2 a partir do processo de varredura e 1 a partir da heurística de economias. Esta é feita segundo Clarke & Wright (1964), versão

paralela como já mencionado. No processo de varredura, uma vez criada a lista com os ângulos de cada um dos clientes em ordem crescente, é preciso escolher o cliente ou ângulo de partida, conforme Gillet & Miller (1974). Então é feito o seguinte:

1) Gera-se *N* ângulos de partida, onde cada ângulo de partida é um cliente localizado numa posição da lista de ângulos. Desta forma são geradas *N* soluções. Onde *N* é o número de clientes.
2) Aplica-se um processo de melhoria 2-Opt intra-rotas para cada uma das rotas de cada uma das soluções.
3) Escolhe-se as 2 melhores soluções.

Posteriormente, a cada uma das 3 soluções são aplicados 3 processos de buscas locais. Eles foram escolhidos por serem buscas rápidas e capazes de fazer melhorias significativas nas soluções iniciais. As 3 buscas locais empregadas foram: Eliminação de Rotas, Propagação de Vizinhança V1 e Realocação. A busca "Propagação de Vizinhança V1" será explicada neste tópico. As demais serão explicadas no tópico a seguir, juntamente com outras buscas. Após a aplicação destas buscas às soluções, escolhe-se a melhor para inicializar a fase de melhorias do algoritmo.

5.1.1.1 – Propagação de Vizinhança V1

A busca chamada "Propagação de Vizinhança V1" consiste em fazer trocas simultâneas entre as rotas de uma maneira circular. Vamos introduzir alguns conceitos para melhor explicação.

Seja uma rota *ra*. Define-se *r1* e *r2* como rotas vizinhas de *ra*, por serem aquelas com os centros de gravidades mais próximos do centro de gravidade de *ra*, entre todas as demais rotas. Na figura 5.3, vemos uma instância de Christofides *et al.* (1979) com 50 clientes. Há 5 rotas numeradas de 1 a 5. Também há uma semi-reta conectando o depósito ao centro de gravidade de cada rota e as respectivas coordenadas (*x, y*) do centro de gravidade de cada rota. Percebe-se claramente que para *ra*=4, temos *r1*=5 e *r2*=3.

Para estas mesmas rotas, suponha um movimento em que um cliente de *ra* é realocado para *r1* e um cliente de *r2* é realocado para *ra*. Podemos definir isto como um

movimento de propagação. Este movimento pode se propagar ao longo de todas as rotas formando um movimento circular de propagação como na figura 5.4.

Pela figura 5.3, podemos visualizar que os clientes da rota 4 vizinhos à rota 3 são: 38,9,30, 34 e 50. Analogamente, os clientes da rota 3 vizinhos à rota 4 são: 46, 5, 49, 10 e 39, ou seja, são os clientes compreendidos entre os segmentos de reta que ligam o depósito ao centro de gravidade das respectivas rotas.

Os movimentos sempre ocorrem entre clientes vizinhos. Desta forma, o número de trocas possíveis é reduzido. Jamais será considerada a inserção do cliente 26 da rota 1 em qualquer posição da rota 4, pois a rota 1 e 4 nem são vizinhas. Analogamente, não será considerada a possibilidade de inserir o cliente 45 da rota 3 em qualquer posição da rota 4, pois embora sejam vizinhos, o cliente 45 não está na vizinhança entre a rota 3 e rota 4. Por sua vez, a possível inserção do cliente 5 da rota 3 na rota 4 será considerada, pois este cliente faz parte da vizinhança da rota 3 em relação à rota vizinha 4.

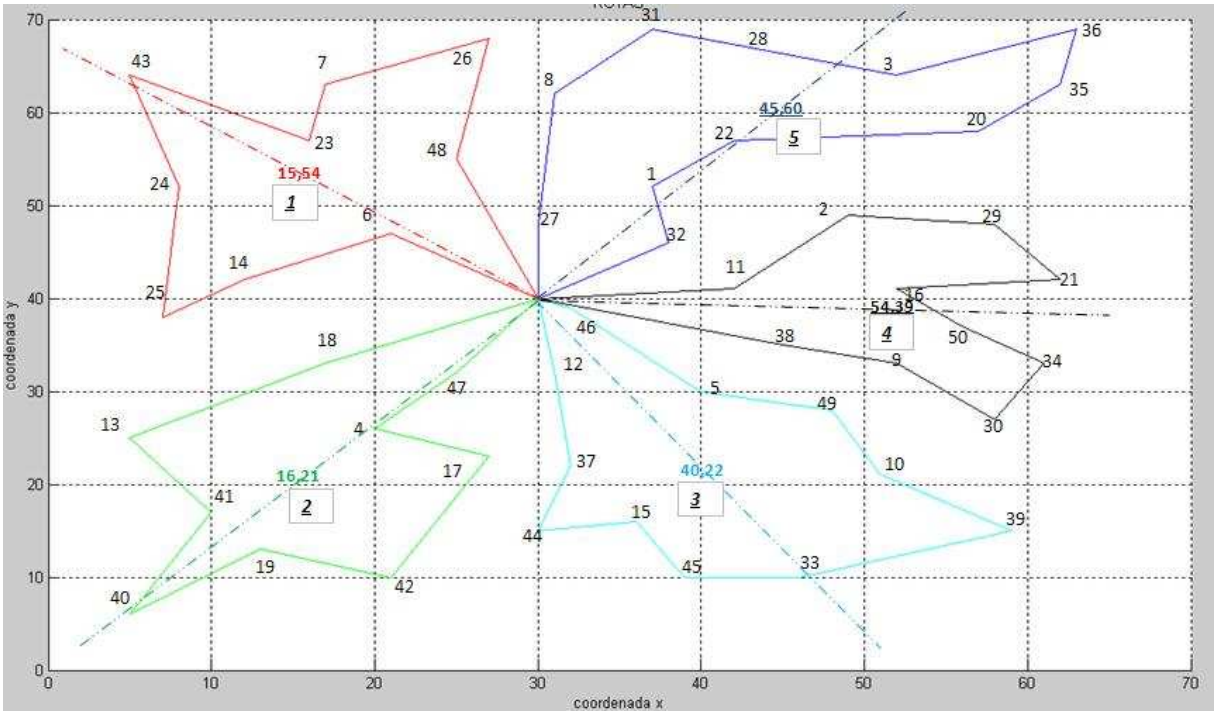


Figura 5.3: Exemplo de uma solução para 50 clientes

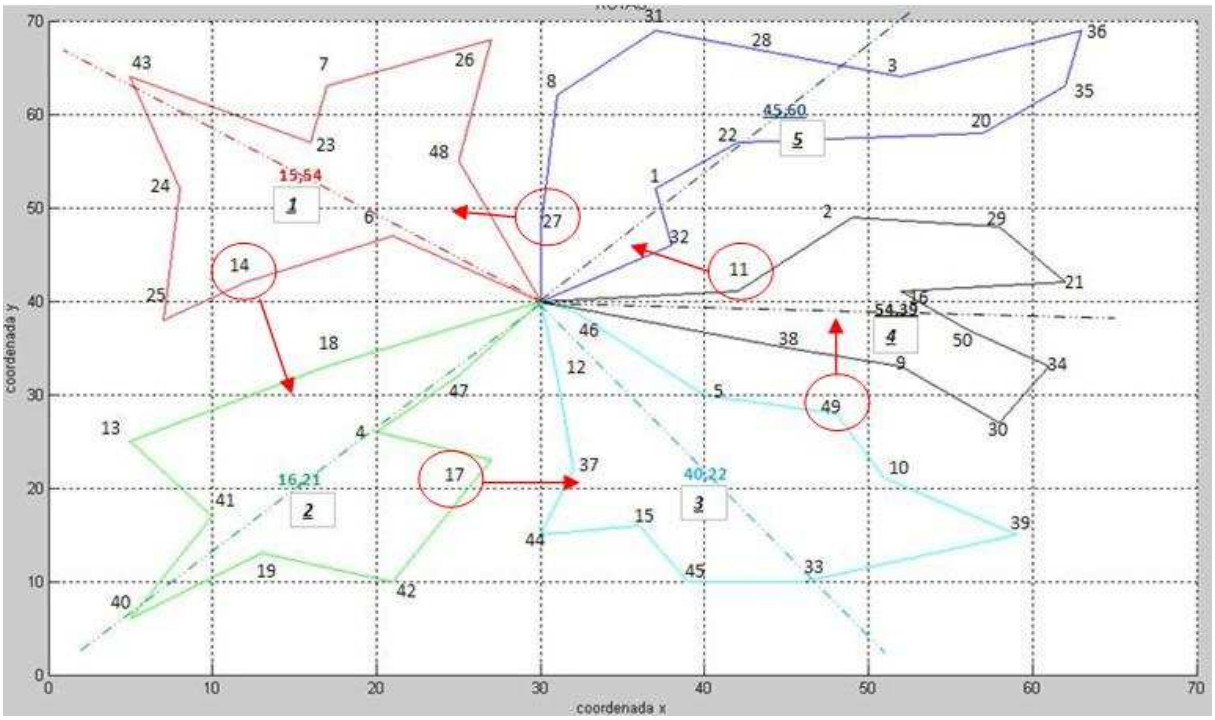


Figura 5.4: Exemplo de uma solução para 50 clientes

Os passos da busca são dados a seguir. Inicialmente *r1*=0 e o número de veículos da solução é dado por *nv*.

1) Fazer *r1*= *r1*+1, ou seja, selecionar uma rota da solução corrente. Se *r1*>*nv*, terminar.

2) Selecionar *r2*, ou seja, rota vizinha à *r1*.

3) Determinar *v1* (conjunto de clientes de *r1* vizinhos a *r2*) e *v2* (conjuntos de clientes de *r2* vizinhos a *r1*).

4) Para cada cliente de *v2* que pode ser inserido em *r1* (respeitando a restrição de capacidade), fazer inserção do cliente na melhor posição de *r1* e avaliar função objetivo.

5) Escolher a melhor inserção do passo 4.

6) Fazer *r1*=*r2*.

7) Se *r1*=*r1ini*, avaliar solução. Se melhor que a melhor solução corrente, armazenar solução. Ir para o passo 1. Caso *r1*≠*r1ini*, ir para passo 2.

No passo 7, se *r1*=1, significa que um ciclo foi completado. Então um novo ciclo com início em outra rota deve ser inicializado. No passo 6, *r2=r1*, significa que se está na última etapa antes de completar um ciclo, então este é o momento de avaliar a função objetivo, pois todo o movimento de propagação está completo. No passo 3, há duas particularidades. Uma ocorre quando *r1*=1. Então permite-se que a inserção de um cliente de *r2* em *r1* viole a capacidade de *r1*. A outra é na última etapa do ciclo, quando *r2=r1*. Então além de testar se a inserção de um cliente de *r2* em *r1* não viola a capacidade de *r1*, é também preciso testar se a remoção deste cliente de *r2* tornou a rota factível.

5.1.2 – Buscas Locais

Dada uma solução $s \in S$, em que S é o espaço de soluções viáveis, uma vizinhança $N(s)$ é composta de soluções obtidas pela aplicação de uma mudança elementar bem definida (movimento) em s. Uma solução $s \in S$ é um mínimo global (ótimo global) se não houver solução $s' \in S$ tal que $f(s') < f(s)$. Para a fase de busca local do algoritmo, usamos o *VND* (descrito na sessão 2.4.2.5).

Em Laporte *et al.* (2000) é descrito um trabalho de Van Breedam em que são descritas 2 estratégias de busca local. Uma chamada *First Improvement (FI)* e outra chamada *Best Improvement (BI)*. A primeira consiste em efetivar o primeiro movimento que melhora a função objetivo. A segunda consiste em avaliar todos os movimentos possíveis e efetivar o melhor movimento. Neste trabalho foi usada a estratégia *BI* em todas as buscas. Além disto, cada busca é executada enquanto houver melhoria.

As estruturas de vizinhança exploradas são:

- Eliminação de Rotas.
- Propagação Parcial de Rotas V1.
- Propagação Parcial de Rotas V2.
- Realocação.
- Intercâmbio.
- Inter Rotas Probabilístico.
- Operador Genético.

O primeiro procedimento a ser executado é o "Eliminação de Rotas", pois prioriza-se a redução do número de veículos. Os demais procedimentos visam a redução da distância total percorrida e são executados numa seqüência aleatoriamente permutada. Todas estas buscas são inter-rotas (envolvem movimentos com clientes pertencentes a diferentes rotas) e possuem mecanismos de buscas intra-rotas (movimentos com clientes da mesma rota) no final de seus respectivos procedimentos.

5.1.2.1 – Eliminação de Rotas

O procedimento implementado tem uma característica dinâmica que parte da observação de que procedimentos estáticos só eliminam rotas nos casos mais triviais. Aqui, procedimentos ditos estáticos são aqueles que testam a realocação de cada cliente de uma determinada rota (que deseja-se eliminar ou rota alvo) para todas as demais possíveis rotas, sem considerar movimentos entre as demais rotas. O procedimento chamado aqui de dinâmico, ao testar a realocação de um cliente da rota alvo para uma outra determinada rota e identificar que isto não é possível, considera ainda algum remanejamento na rota candidata a receber o cliente. Ou seja, considera trocas entre a rota candidata e suas rotas vizinhas para tentar criar uma condição em que torna-se possível receber o cliente da rota alvo.

Tenta-se ainda realizar estes movimentos sem "destruir" a solução corrente, já que existe, nestes movimentos, a possibilidade de que a solução final com 1 rota a menos tenha uma avaliação da função objetivo pior que a rota original, com 1 cliente a mais. Isto é feito limitando o remanejamento de clientes às rotas com centro de gravidade mais próximas.

Em nosso algoritmo, este procedimento é o primeiro a ser executado no processo de buscas locais. Esta decisão foi baseada em pesquisas de Bent & Van Hentenryck (2004) e de Bräysy *et al*. (2004) que indicam que a minimização do número de rotas e a minimização da distância total percorrida devem ser diferenciadas; de modo que a minimização do número de rotas deve anteceder a minimização da distância total percorrida.

As etapas do procedimento são descritas a seguir:

1) Selecionar rota com capacidade mais ociosa (menor ocupação). Esta será a rota candidata à eliminação.

2) Calcular a soma da demanda ociosa das demais rotas. Se este valor for maior que a demanda total da rota candidata, o procedimento segue adiante, caso contrário é finalizado. Isto é feito para checar se há "espaço" nas demais rotas.

3) Identificar as 2 rotas vizinhas (*r1* e *r2*) à rota alvo (*ra*), ou seja, aquelas com centro de gravidade mais próximo de centro de gravidade de ra.

4) Determinar quais são os clientes de *ra* mais próximos de *r1* (conjunto de vizinhança *v1*). Analogamente, determinar *v2* para a segunda rota vizinha *r2*. Os conjuntos *v1* e *v2* são partições do conjunto formado pelos clientes de *ra*.

5) Incluir todos os clientes de *v1* em *r1*. Caso a capacidade de *r1* seja excedida, selecionar um conjunto de clientes de *r1* para serem removidos de modo a deixar a rota factível. Inserir estes clientes na rota vizinha de *r1,* diferente de *ra*. Repetir esta etapa até que uma das seguintes condições seja satisfeita: 1)Todos os vizinhos foram explorados e nenhuma solução factível foi encontrada. 2) Encontrou solução viável.

6) Incluir todos os clientes de *v2* em *r2*. Proceder de forma análoga ao passo 5.

Um fluxograma do método descrito acima é dado na figura 5.5. As etapas 5 e 6, para melhor entendimento, podem ser separadas numa rotina específica que chamamos de "Propagar", cujo fluxograma é dado na figura 5.6.

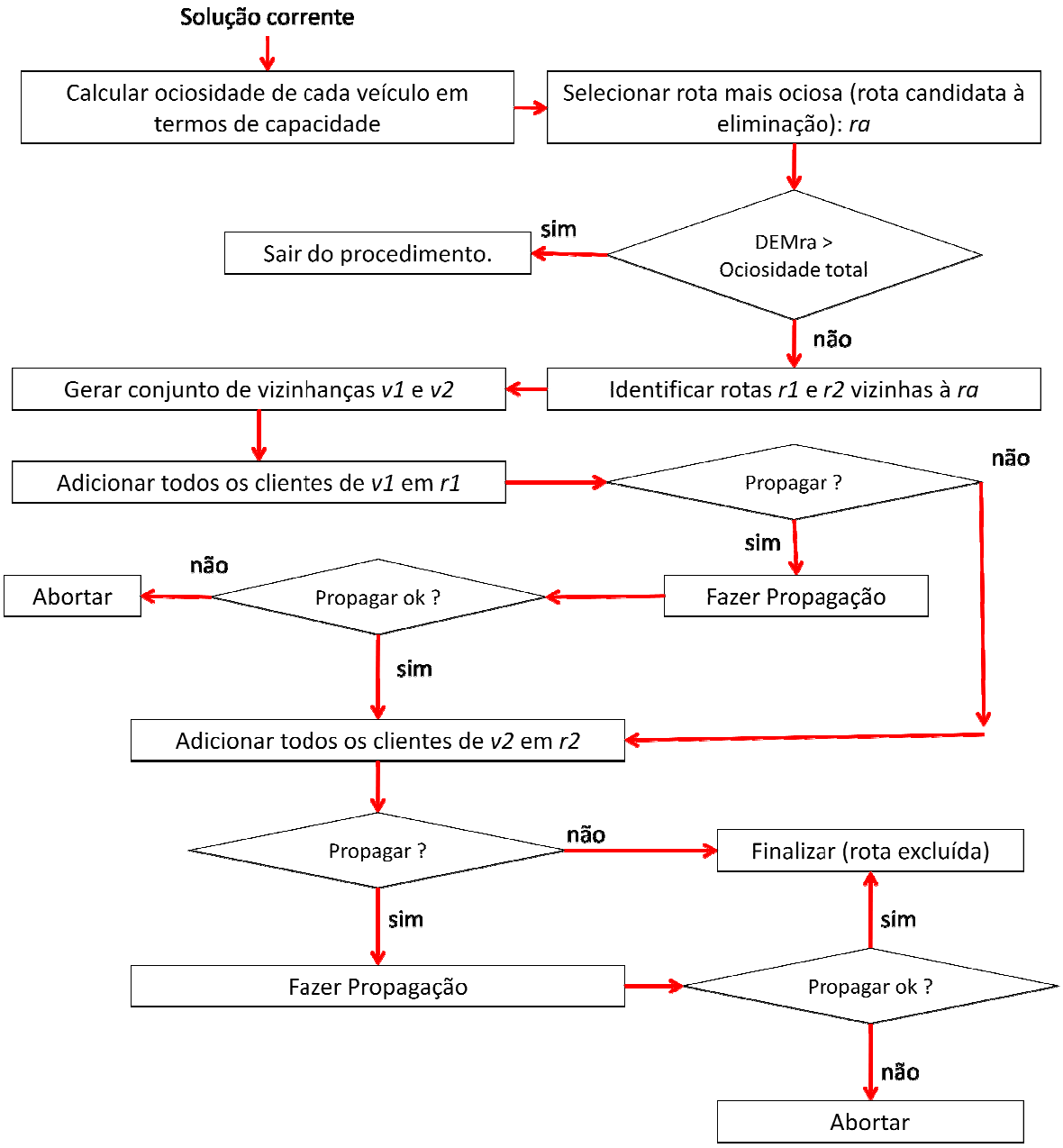


Figura 5.5: eliminação de Rotas

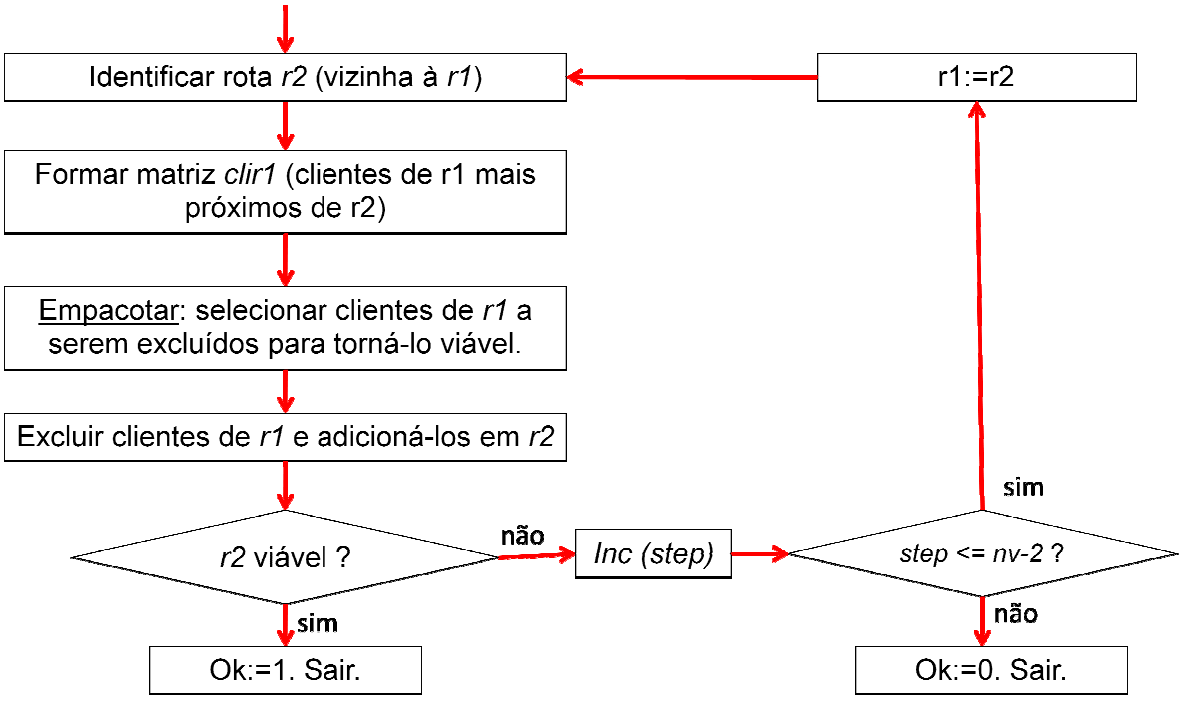


Figura 5.6: Propagar

Na etapa 5, a seleção dos clientes que devem ser removidos segue 3 critérios. 1: serem clientes próximos da rota vizinha (diferente de *ra* e ainda não explorada). 2: não remover muitos clientes para não alterar demais a solução. 3: a demanda total dos clientes removidos não deve ser muito maior que a demanda necessária a ser removida para tornar a rota factível (chamemos esta demanda de δ). Sendo *nc* o número de clientes da rota corrente, escolhe-se os clientes mais próximos da rota vizinha (conjunto *v*).

Então aplica-se um estratégia similar ao *Best Fit Decreasing*, heurística clássica do *Bin Packing Problem* (Bramel & David, 1997) da seguinte forma. Se há 1 ou mais clientes em *v* cuja demanda seja maior ou igual a δ, escolher o de menor demanda (mais próximo de δ). Caso contrário, escolher aquele de maior demanda (mais próximo de δ). Repete-se o processo de seleção até que a demanda total dos clientes selecionados seja maior ou igual a δ. Esta etapa pode ser chamada de "Empacotar", sendo seu fluxograma mostrado na figura 5.7:

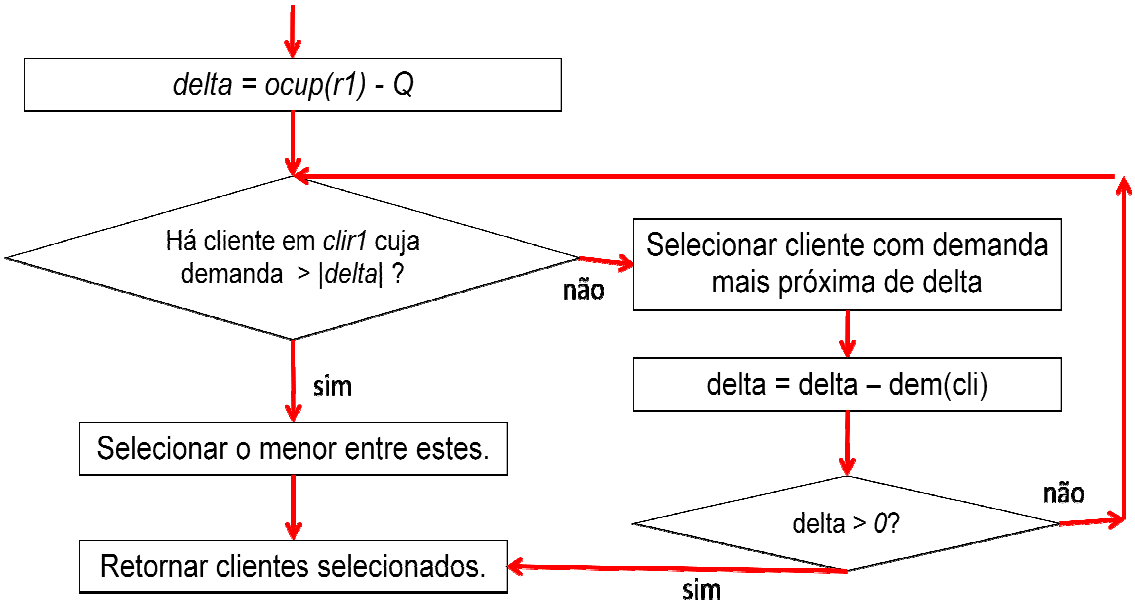
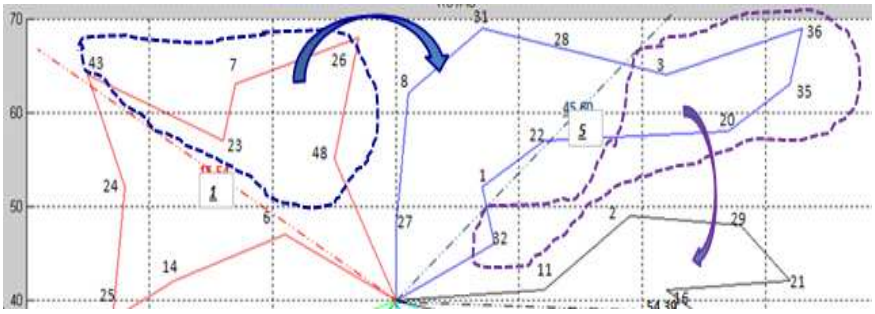


Figura 5.7: Empacotar

Vamos exemplificar a tarefa "Empacotar". A figura 5.8 mostra 2 rotas. A rota 1 é a rota a ser excluída. Os clientes circulados são movidos para a rota 5. Desta forma, a rota 5 passou a exceder sua capacidade em 57 unidades. Logo é preciso escolher quais clientes de 5 devem ser excluídos para que a rota se torne viável. Primeiro é formado o conjunto de clientes candidatos à exclusão (clientes circulados da rota 5, com demanda descrita na figura 5.8). Neste momento a tarefa "Empacotar" é iniciada.



cli	dem
32	22
20	19
35	15
36	12
3	21

Figura 5.8: Exemplo da tarefa "Empacotar"

Inicialmente temos Como não há cliente candidato com demanda maior que 57, escolhe-se o cliente com demanda mais próxima, neste caso o cliente 32 com demanda 22. Então temos . Dentre os clientes ainda não selecionados, nenhum tem a demanda maior que 35. Então escolhe-se aquele com demanda mais próxima, neste caso o cliente 3 com demanda 21. Então temos

. Dentre os clientes não selecionados, o cliente 35 tem demanda maior que 14, então ele é selecionado. Então temos . O processo é terminado. A exclusão

dos clientes selecionados torna a rota 5 novamente viável. Os clientes selecionados são movidos para a rota vizinha de 5 e uma nova tarefa “Propagar” é iniciada.

5.1.2.2 – Propagação Parcial de Rotas V1

A busca consiste em percorrer cada rota da solução em ordem decrescente de comprimento e realocar clientes destas rotas para outras rotas utilizando-se do mesmo conceito de propagação visto nas buscas anteriores. Os passos são:

1) Para cada rota da solução corrente, em ordem decrescente de comprimento:
2) Selecionar até 4 clientes da rota (*ra*), sendo os 2 mais distantes do depósito e os 2 mais distantes do centro de gravidade da rota.
3) Identificar a rota mais próxima de cada um dos clientes escolhidos no passo 2 (critério é escolher a rota que possui um cliente mais perto do respectivo cliente em *ra*).
4) Formar uma matriz de cenários, onde cada um é uma tentativa de remover 1 ou 2 clientes de *ra*, realocando-os para as rotas identificadas no passo 3. Cada cenário trata-se de uma combinação $C_{4,1}$ ou $C_{4,2}$ dos 4 clientes selecionados no passo 2. Então, numa rota com mais de 4 clientes, onde escolhe-se 4 clientes no passo 2, temos 10 cenários possíveis.
5) Para cada um dos cenários, inserir 1 cliente do cenário corrente na rota mais próxima a ele (selecionada no passo 3) e naturalmente, excluir este cliente da rota alvo (*ra*).
6) Caso a rota que recebeu o cliente tenha tido sua capacidade excedida, fazer a propagação, remanejando outras rotas conforme passo 5 do tópico 6.1.2.1.
7) Se achou solução viável e o cenário corrente tenha apenas 1 cliente, ir para passo 8. Se achou solução viável e cenário corrente tenha 2 clientes, ir para passo 5, repetindo processo para o segundo cliente. Caso não tenha encontrado solução viável, ir para passo 5, mas agora incrementando o número do cenário de modo a escolher outro cenário.
8) Avaliar função objetivo. Caso tenha havido melhoria, armazenar solução como a melhor.

Esta busca pode em determinadas situações eliminar uma rota. Caso isto ocorra, dá-se preferência pela solução que elimina uma rota, mesmo que ela tenha uma avaliação pior que a melhor solução da busca.

5.1.2.3 – Propagação Parcial de Rotas V2

Esta busca é idêntica à Propagação Parcial de Rotas V1. A diferença é que ela sempre busca o melhor movimento para melhorar a avaliação da função objetivo. Não dá preferência por uma solução que venha a reduzir rotas. Este é o motivo da busca ser executada apenas após a busca Propagação Parcial de Rotas V1, pois até este momento espera-se que todas as reduções possíveis no número de rotas já tenham sido feitas para que o algoritmo passe a focar a redução da distância total percorrida.

5.1.2.4 – Realocação

Esta busca foi explicada na sessão 2.4.1.5. Faz-se trocas do tipo *0x1*, ou seja, um cliente é removido de uma rota e inserido em outra rota. É testado a remoção de todos os clientes. Os passos são:

1) Determinar as *NR* rotas vizinhas de cada rota da solução
2) Para cada rota *ra* da solução corrente, para cada cliente desta rota, para cada rota *rv* vizinha à rota *ra*, testar movimento de inserção do cliente de *ra* em *rv*. Armazenar o melhor movimento.
3) Caso o melhor movimento do passo 2 tenha resultado numa melhoria da solução de entrada da busca, renomear esta solução como solução corrente e retonar ao passo 1.

As rotas vizinhas de uma dada rota são determindas pelo critério de centro de gravidade. O valor de *NR* pode variar de 1 até o número total de rotas da solução. Quanto menor *NR,* menor é o número de trocas possíveis, então, menor é o custo computacional.

No passo 2, faz-se a inserção do cliente na primeira posição da rota. Então aplica-se o método "2-opt intra-rota" para esta rota específica a fim de encontrar uma melhor posição para o cliente inserido na rota. Na remoção, o cliente anterior ao removido e o cliente posterior ao removido são conectados.

5.1.2.5 – Intercâmbio

Esta busca foi explicada na sessão 2.4.1.5. O procedimento é análogo ao "Realocação", porém aqui o movimento é do tipo *1x1*, ou seja, um cliente de uma rota é trocado com outro cliente de outra rota. Os passos são:

1) Determinar as *NR* rotas vizinhas de cada rota da solução
2) Para cada rota *ra* da solução corrente, para cada rota *rv* vizinha à rota *ra*, testar todas as possíveis trocas de clientes entre *ra* e *rv*. Armazenar o melhor movimento.
3) Caso o melhor movimento do passo 2 tenha resultado numa melhoria da solução, renomear esta solução como solução corrente e retonar ao passo 1.

No passo 2, faz-se a inserção do cliente na primeira posição da rota. Então aplica-se o método "2-opt intra-rota" para as duas rotas envolvidas na troca para encontrar uma melhor posição para os clientes inseridos.

5.1.2.6 – Inter-rotas Probabilístico

Esta busca possui vários parâmetros aleatórios. Inicialmente, decide-se quantas rotas devem ser perturbadas (1 ou 2) e depois escolhe-se quais serão estas rotas. Depois escolhe-se o tipo de troca a ser feito na rota escolhida. Há 6 tipos: 0x1 (significa que a rota selecionada doa 1 cliente para sua rota vizinha e não recebe nenhum), 1x0 (rota selecionada recebe 1 cliente da rota vizinha e não doa nenhum), 1x1 (rota selecionada recebe um cliente da rota vizinha e ao mesmo tempo doa 1 cliente), 1x2 (rota selecionada recebe um cliente e doa 2 clientes para rota vizinha), 2x1 (rota selecionada recebe 2 clientes e doa 1 cliente) e 2x2 (rota selecionada doa e recebe 2 clientes).

As etapas são dadas a seguir:

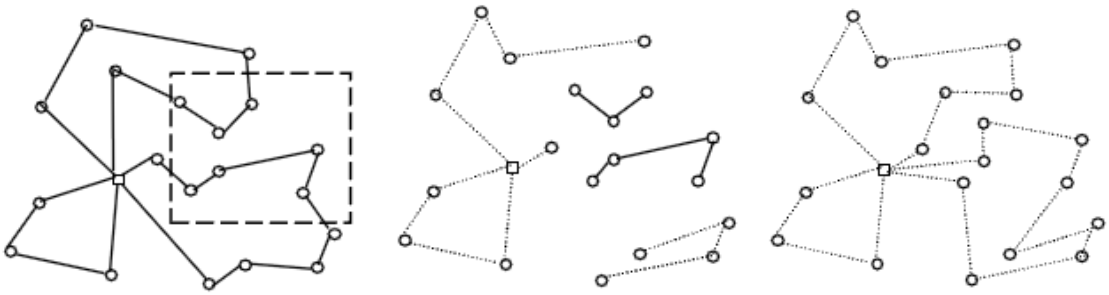
1) Escolher o número de rotas a serem perturbadas (*numper*): escolher aleatoriamente 1 ou 2.
2) Selecionar a rota a ser perturbada *(r1)*. É um número aleatório de 1 ao número de rotas da solução.

3) Determinar rota vizinha (*r2*) em relação a *r1* pelo critério de centro de gravidade.
4) Determinar o conjunto de clientes *v1* e *v2* que forma as vizinhanças entre *r1* e *r2* como descrito no tópico 5.1.1.1.
5) Escolher o tipo de troca a ser realizado: aleatório entre 1 e 6.
6) Para o tipo escolhido, testar todas as combinações possíveis de troca envolvendo os clientes de *v1* e de *v2*. Armazenar o melhor movimento. Caso não encontre uma troca viável, escolher aleatoriamente outro tipo de troca e repetir testes. Caso não encontre nenhuma troca viável, retornar ao passo 2.
7) Avaliar a função objetivo e comparar com a melhor solução. Incrementar *np* (contador de perturbações realizadas). Caso *numper* tenha sido superado por *np*, ir para passo 8. Caso contrário, ir para o passo 2.
8) Incrementar o contador de movimentos. Caso o número de movimentos tenha atingido um valor máximo preestabelecido, finalizar busca, caso contrário retornar ao passo 1.

Escolher *numper*=2 significa que 2 rotas diferentes serão perturbadas. Neste caso, no passo 7, ao efetivar a troca na primeira rota selecionada, retorna-se ao passo 2 para testar trocas na segunda rota selecionada.

5.1.2.7 – Cruzamento Natural

O operador implementado aqui baseia-se em Potvin (2007). Na figura 5.9, segundo o autor, a área demarcada por um quadrado tracejado em “Pai 1” pode ter as mais variadas formas geométricas e tamanhos.



Pai 1 Filho Pai 2

Figura 5.9. Operador Cruzamento Natural (Potvin, 2007)

Naturalmente as duas entradas principais da busca são os pais. Aqui o *pai1* é a solução corrente do algoritmo e o *pai2* é a melhor solução que o algoritmo tem no momento corrente. As etapas básicas do operador implementado são:

1) Escolher o número de filhos desejados (*nfdes*): metade do número de veículos do *pai1*. Inicializar número de filhos gerados: *nf*=1.
2) Fazer *nf*=*nf*+1. Is *nf* > *nfdes*, ir para o passo 8.
3) Escolher região: aleatório de 1 a 8.
4) Selecionar em *pai1* a parte da solução dentro da região escolhida que tenha no mínimo um cliente adjacente também dentro da região (ou seja, 1 arco dentro da região delimitada).
5) Armazenar os clientes selecionados em *filho*, número "*nf*".
6) Copiar a solução de *pai2*, sem os clientes selecionados no passo 4.
7) Reparar o *filho* com o procedimento "Eliminação de Rotas", visto no tópico 5.1.2.1 e retornar ao passo 2.
8) Escolher o *filho* com melhor avaliação da função objetivo.
9) Melhorar o *filho* selecionado com as buscas "Realocação" e "Intercâmbio" (tópicos 5.1.2.4 e 5.1.2.5).
10) Avaliar solução corrente e comparar com a melhor solução.

Há 8 regiões possíveis. Elas são criadas em função da localização do depósito, do cliente de maior coordenada *x*, do cliente de maior coordenada *y,* do cliente de menor coordenada *x* e do cliente de menor coordenada *y*. Conforme pode ser visto nas

figuras 5.10 e 5.11, estes 4 clientes seriam: 36, 31, 13, 40 respectivamente, num exemplo de uma instância de Christofides de 50 clientes. As interseções das retas que passam por cada um destes clientes formam 4 pontos extremos chamados de *e1*, *e2*, *e3* e *e4*. Os 4 segmentos de reta que unem o depósito a cada um destes extremos formam as 4 primeiras regiões, vistas na figura 5.10.

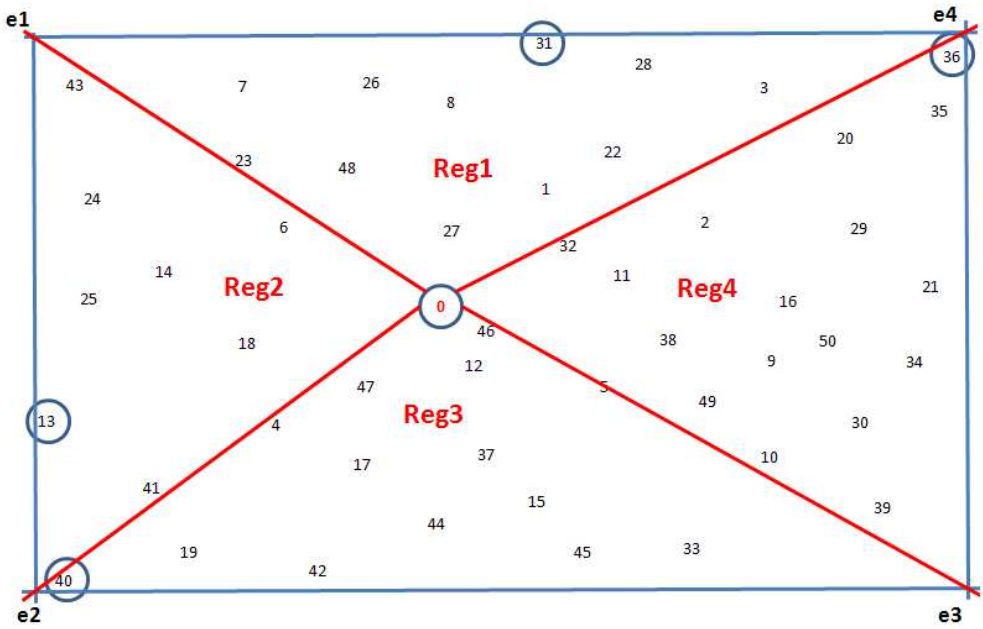


Figura 5.10: regiões 1 a 4

Na figura 5.11, podemos ver as demais 4 regiões. Elas são formadas pelos 4 segmentos de reta que partem do depósito e vão até os 4 pontos médios calculados a partir dos 4 pontos extremos. Desta forma as 8 regiões mudam conforme a distribuição dos clientes e a localização do depósito.

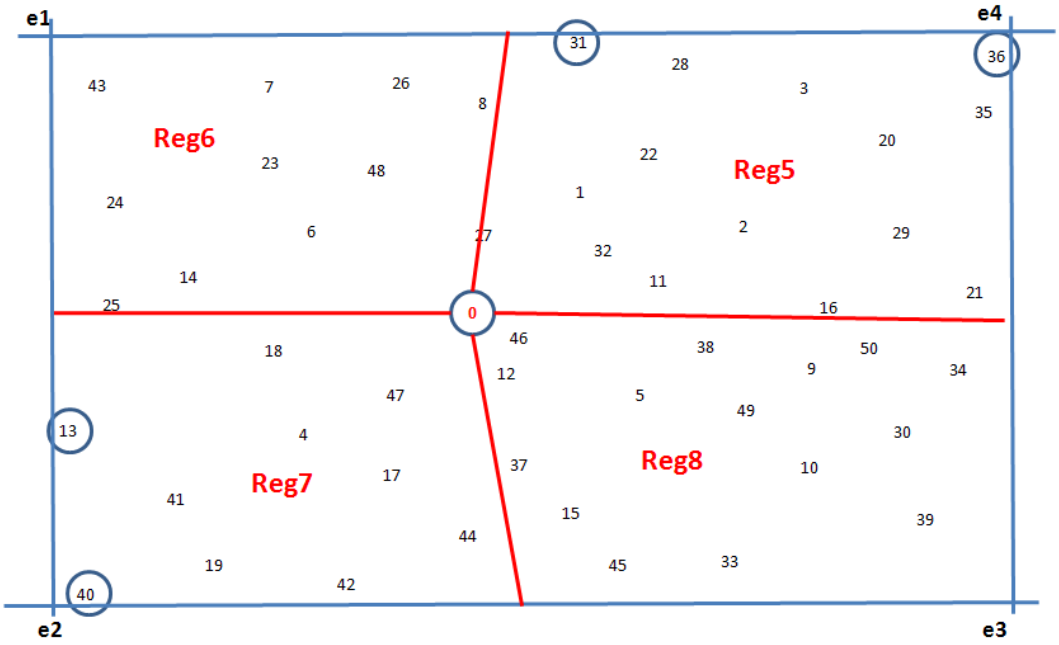


Figura 5.11: regiões 5 a 8

A solução contida na área selecionada tem origem no *pai1* porque desta forma a maior parte da solução terá origem no *pai2* que é a melhor solução obtida pelo algoritmo até o momento. Assim, a busca mostra alguma preferência por esta solução.

5.1.3 – Procedimentos de Perturbação

Como mencionado anteriormente, a perturbação é gerada a partir de operadores genéticos. Das 3 soluções iniciais geradas no algoritmo, as outras 2 soluções não são desperdiçadas, sendo utilizadas no algoritmo quando um movimento de piora é necessário. De fato, o algoritmo sempre mantém uma lista de 3 soluções (ou indivíduos num contexto evolucionário) disponíveis para quando houver necessidade, chamemos esta lista de "*IniSol*" pois inicialmente esta lista é composta das 3 soluções iniciais geradas no algoritmo. A melhor solução que o algoritmo encontrou até o momento é chamada "*BesSol*". Conforme algoritmo 5.1, o procedimento de perturbação é executado quando o algoritmo está estacionado em um ótimo local. O procedimento de perturbação é composto dos seguintes passos:

1) Checar se solução que acabou de sair da fase de busca é repetida no conjunto "*IniSol*". Se sim, aplicar operador *Shift* (explicado adiante em 5.1.4.4) e substituir solução repetida.

2) Selecionar 1 indivíduo de "*IniSol*" conforme critério de aspiração;

3) Executar 3 cruzamentos genéticos entre o indivíduo selecionado no passo 2 (*pai1*) e o *BesSol* (*pai2*). Estes 3 cruzamentos genéticos são: Cruzamento Natural, Cruzamento B2 e Cruzamento B4 (o primeiro foi descrito no tópico 5.1.2.7 e os demais serão explicados adiante). Incluir os 3 filhos na lista de candidatos.

4) Os 3 *filhos* dos cruzamentos e o próprio *pai1* formam uma lista de 4 soluções candidatas.

5) Escolher a melhor solução da lista de candidatos que não seja Tabu. Este indivíduo é incluído em *IniSol* em substituição ao indivíduo escolhido no passo 1 (pode ocorrer de ser ele mesmo).

6) O indivíduo selecionado no passo 5 retorna para a fase de melhoria do algoritmo, conforme figura 5.12.

A fase de perturbação é ilustrada pelo fluxograma da figura 5.12. Observa-se que a função *Shift* tem o papel de manter as soluções do conjunto *IniSol* diferentes. Também observa-se que quando uma solução é adicionada no conjunto *IniSol*, a função objetivo destas soluções é armazenada numa lista Tabu. Sempre que uma solução é gerada para substituir alguma solução de *IniSol* é testado se esta solução é Tabu. Isto é importante para evitar que soluções já exploradas anteriormente na fase de busca sejam novamente exploradas.

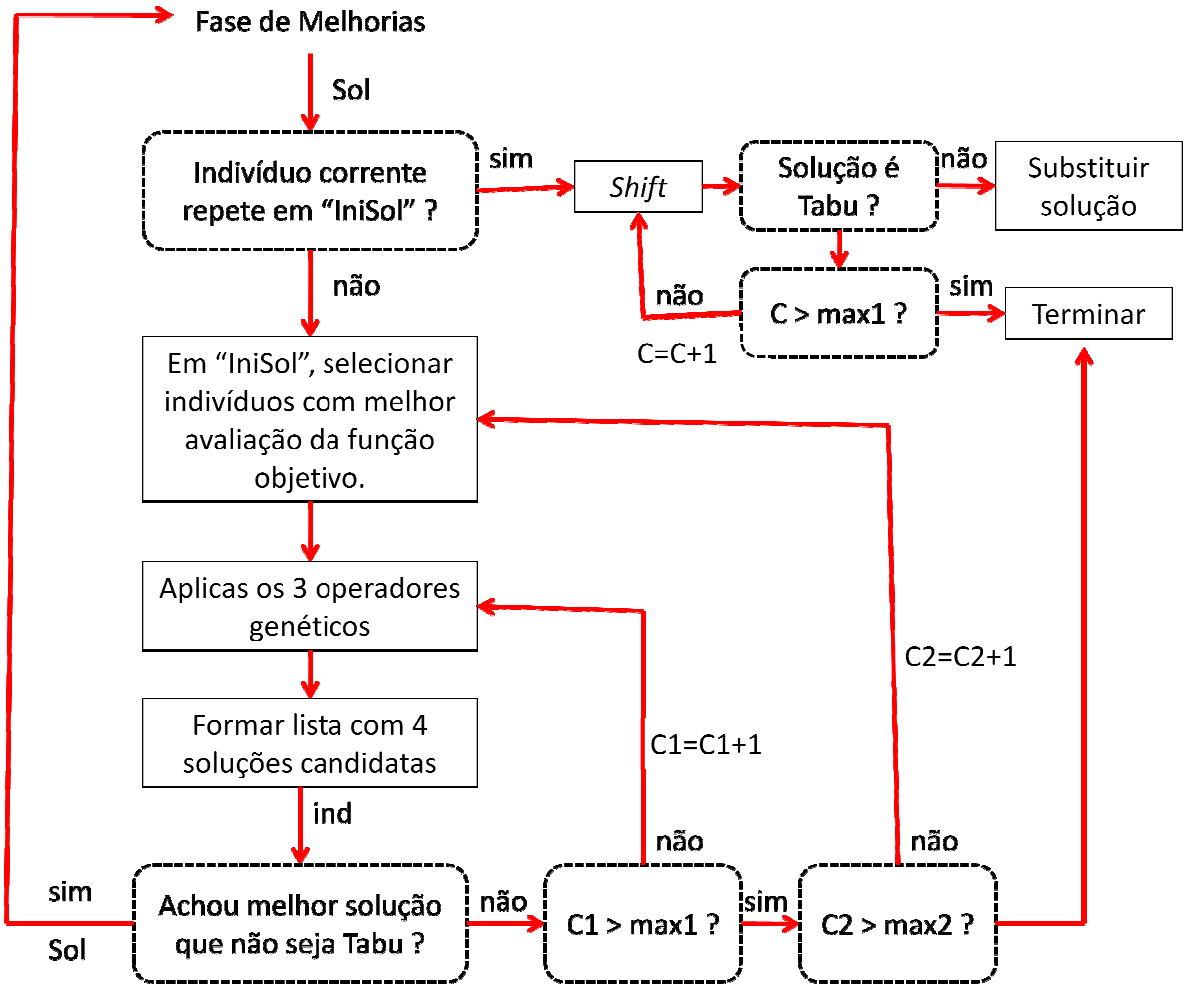


Figura 5.12: Fase de Perturbação

O conjunto *IniSol* trata-se de uma memória adaptativa, sendo importante para manter uma diversidade das soluções encontradas na fase de perturbação. Ao mesmo tempo que mantém esta diversidade, trata-se de um conjunto de soluções de boa qualidade que não são descartadas pelo algoritmo. O procedimento *Shift* juntamente com o conjunto *IniSol* dão à fase de perturbação maior poder para explorar soluções com diferentes características e ao mesmo tempo de boa qualidade, de forma que as soluções geradas não têm um efeito de "restart" no algoritmo.

Os cruzamentos genéticos utilizados também contribuem para encontrar soluções diferentes e de qualidade. Diferentes devido a seus parâmetros aleatórios e por ter flexibilidade de mudar um dos pais que participa do cruzamento devido ao conjunto *IniSol*. E de boa qualidade, pois um dos pais é a melhor solução encontrada, e o segundo pai faz parte de *IniSol*.

A figura 5.13 mostra um gráfico com a convergência da melhor solução (círculos) e solução gerada pelo mecanismo de perturbação (linha contínua) numa instância qualquer.

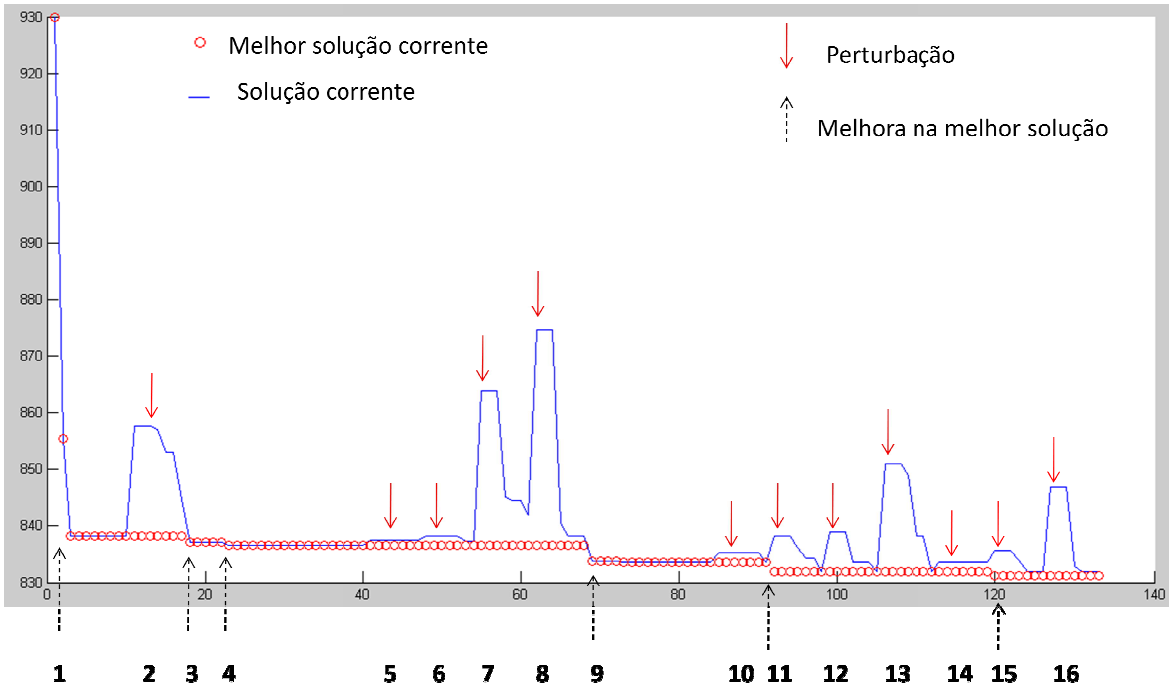


Figura 5.13: Convergência e Perturbações

A figura 5.13 mostra que o algoritmo convergiu rapidamente após a solução inicial. No ponto 2 vê-se uma perturbação e em seguida o algoritmo converge para uma solução melhor que aquela tida como a melhor até então. Nos pontos 5 e 6 observa-se perturbações pequenas e nos pontos 7 e 8 perturbações maiores. A perturbação do ponto 8 permitiu que o algoritmo melhorasse a melhor solução corrente, sendo um clássico exemplo de fuga do ótimo local em que se encontrava. É possível ver como as perturbações ajudaram o algoritmo a sair de um ótimo local e continuar convergindo.

5.1.3.1 – Cruzamento B2

Este operador foi baseado em Hjorring (1995). As etapas são descritas a seguir:

1) Escolher o número de filhos desejados (*nfdes*): metade do número de veículos do *pai1*. Inicializar número de filhos gerados: *nf*=1.

2) Fazer *nf*=*nf*+1. Is *nf* > *nfdes*, ir para o passo 8.

3) Escolher aleatoriamente uma rota de *pai1* (rota *semente*).
4) Armazenar a rota *semente* em *filho,* número "*nf*".
5) Copiar a solução de *pai2*, sem os clientes selecionados no passo 4.
6) Reparar o *filho* com o procedimento "Eliminação de Rotas", visto no tópico 5.1.2.1. e retornar ao passo 2.
7) Escolher o *filho* com melhor avaliação da função objetivo.
8) Melhorar o filho selecionado com as buscas "Realocação" e "Intercâmbio" (tópicos 5.1.2.4 e 5.1.2.5).
9) Avaliar solução corrente e comparar com a melhor solução.

Observa-se que apenas 1 rota é herdada de *pai1*, toda a solução tem origem no *pai2*, mostrando forte preferência por este indivíduo.

5.1.3.2 – Cruzamento B4

Este operador foi baseado em Alvarenga & Mateus (2004). As etapas são descritas a seguir:

1) Inicializar *filho*, escolhendo aleatoriamente 1 rota de *pai2*, copiando esta rota.
2) Se possível, escolher aleatoriamente uma rota de *pai1* que não possua qualquer cliente em *filho*.
3) Se possível, escolher aleatoriamente uma rota de *pai2* que não possua qualquer cliente em *filho*.
4) Repetir passos 2 e 3 até que um deles não seja mais possível.
5) Se existirem clientes não incluídos em *filho*, completar solução aplicando uma heurística rápida para estes clientes. Aqui isto foi feito aplicando *Gillet & Miller* e *Clarke & Wright*. Assim, obtem-se 2 filhos, cada um completado com uma heurística diferente.
6) Para cada *filho*,aplica-se 2-Opt intra-rotas para todas as rotas das 2 soluções.
7) Escolher o *filho* com melhor avaliação da função objetivo.

8) Melhorar o filho selecionado com as buscas “Eliminação de Rotas”, “Realocação” e “Intercâmbio” (tópicos 6.1.2.1, 6.1.2.4 e 6.1.2.5).

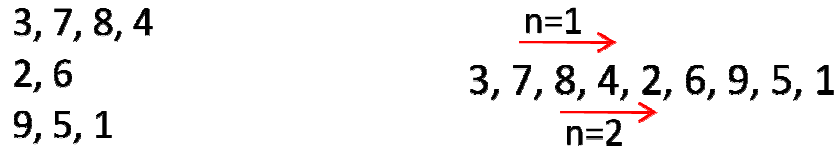
9) Avaliar solução corrente e comparar com a melhor solução.

5.1.3.3 – Processo de Reparação

Em todos os 3 operadores genéticos apresentados (Cruzamento Natural, Cruzamento B2 e Cruzamento B4) o processo de geração dos filhos é o mesmo da sua respectiva fonte de referência, porém o processo de reparação do filho é diferente. Normalmente o filho gerado é inviável, não em termos da restrição de capacidade, mas por existirem sub-rotas não conectadas ao depósito (ver figura 5.9), então um processo de reparação faz-se necessário. Aqui, a reparação é feita simplesmente conectando o primeiro e o último cliente de cada sub-rota ao depósito. Assim, obtem-se uma solução viável e que possui características dos pais. O efeito colateral imediato é que este processo normalmente aumenta o número de rotas da solução. Então, em todos os cruzamentos, o primeiro processo de busca a ser executado é o “Eliminação de Rotas”. Após esta busca, a solução volta a ter um menor número de rotas e então as buscas “Realocação” e “Intercâmbio” são aplicadas para melhorar a solução.

5.1.4.4 – *Shift*

O procedimento *Shift* tem como objetivo impedir que o conjunto “IniSol” composto de 3 soluções tenha alguma solução repetida. Isto ajuda a Fase de Perturbação a gerar soluções diferentes que possam convergir para um ótimo local diferente daqueles já obtidos pelo algoritmo. A idéia e análoga ao *Sweep* de Gillet & Miller (1974). O funcionamento é ilustrado pelas figuras 5.14 e 5.15.



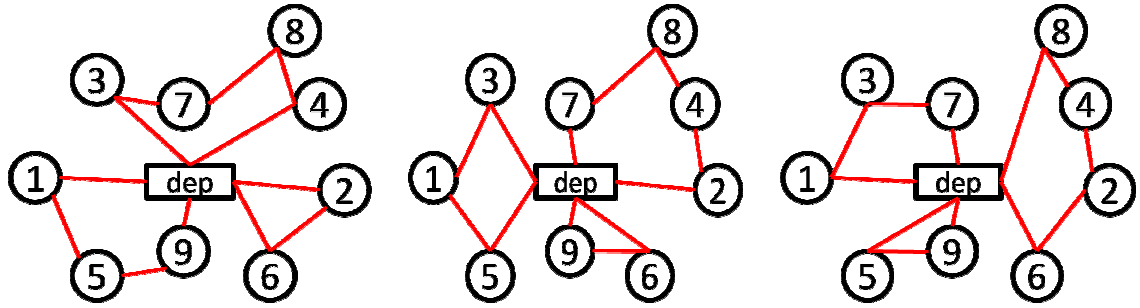
Matriz Solução Vetor *Shift*

Figura 5.14: Geração de novas soluções a partir do *Shift*

A figura 5.14 (lado esquerdo) mostra uma matriz de uma solução com 3 rotas e 9 clientes. A partir desta matriz forma-se o vetor *shift* no lado direito da figura. Este vetor é formado colocando os clientes de cada rota, uma rota após a outra respeitando a relação de vizinhança entre as rotas. Lembramos que rotas vizinhas tratam-se, no contexto deste trabalho, de uma rota que possui o centro de gravidade mais perto do centro de gravidade de outra rota. Formado o vetor *shift*, escolhe-se o ponto inicial de formação da nova solução. No exemplo, foi utilizado 2 possíveis pontos: após o primeiro cliente (shift nível 1) e a após o segundo cliente (shift nível 2).

O processo de geração de uma nova solução ocorre da seguinte forma: inicia-se uma rota com o cliente do ponto de partida. Então os clientes seguintes no vetor *shift* vão sendo adicionados à rota até que alguma restrição seja violada. Então, inicia-se uma nova rota. O processo continua até que todos os clientes estejam na solução.

A figura 5.15 mostra as soluções de nível 1 e 2 geradas a partir da solução cuja matriz foi mostrada na figura anterior.



Solução Original Solução nível 1 Solução nível 2

Figural 5.15: soluções obtidas a partir do procedimento *shift*

É possível a utilização de vários outros pontos de partida. Neste trabalho, utilizou-se 10 pontos de partida, escolhidos aleatoriamente entre 1 e *N*, onde *N* é o número de clientes da solução. As 10 soluções são avaliadas pela função objetivo e a solução selecionada é escolhida aleatoriamente entre as 5 melhores. Isto caracteriza um processo semi-guloso similar ao utilizado pelo *GRASP*, brevemente descrito na sessão 2.4.2. Naturalmente, o número máximo de pontos de partida é *N*.

5.2 - PRV CAPACITADO: HEURÍSTICA H2

A segunda heurística desenvolvida é chamada aqui de *H2*, que basicamente é a heurística *H1* com a inclusão da Busca Tabu. Na fase onde as buscas locais são aplicadas, as vizinhanças exploradas são:

- Eliminação de Rotas.
- Propagação Parcial de Rotas V1.
- Propagação Parcial de Rotas V2.
- Busca Tabu
- Inter Rotas Probabilístico Tabu.
- Operador Genético.

Tratam-se dos mesmos procedimentos de buscas locais utilizados em H1, porém as duas buscas: "Realocação" e "Intercâmbio" foram susbtituídas pelo procedimento "Busca Tabu". Outra mudança é que o procedimento "Inter Rotas Probabilístico" foi substituído por sua versão tabu, aqui chamada de "Inter Rotas Probabilístico Tabu". Estas mudanças foram realizadas para aumentar a capacidade do algoritmo de "fugir" de ótimos locais. As novas buscas, com a implementação de conceitos *Tabu Search* vistos em 2.4.2.1 são descritas a seguir.

5.2.1 – Procedimento "Busca Tabu"

Neste procedimento, dado uma solução, obtém-se toda a vizinhança desta solução a partir de movimentos do tipo *0x1* e *1x1*. O melhor movimento que não seja tabu é avaliado. Caso este movimento implique numa melhoria, ele é inserido na Lista Tabu e o processo de busca recomeça. Os passos são:

1) Determinar as *NR* rotas vizinhas de cada rota da solução.
2) Para cada rota *ra* da solução corrente, para cada cliente desta rota, para cada rota *rv* vizinha à rota *ra*, se o movimento não for tabu, testar movimento de inserção do cliente de *ra* em *rv*. Armazenar o melhor movimento (tipo *0x1*).

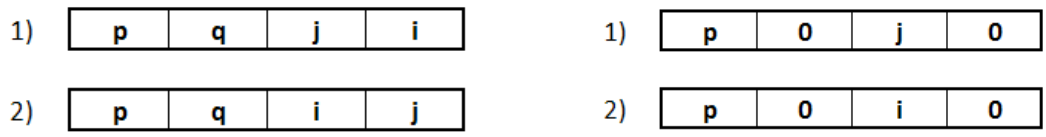
3) Para cada rota *ra* da solução corrente, para cada rota *rv* vizinha à rota *ra* , testar todas as possíveis trocas de clientes entre *ra* e *rv* que não sejam um movimento da lista tabu. Armazenar o melhor movimento (tipo 1x1).

4) Caso o melhor movimento dos passos 2 e 3 tenha resultado numa melhoria da melhor solução corrente, inserir este movimento na Lista Tabu, renomear esta solução como solução corrente. Retonar ao passo 1.

Pelo fato desta busca realizar internamente tanto movimentos do tipo *0x1* como *1x1*, ela substituiu as buscas "Realocação" e "Intercâmbio" usadas em H1.

A Lista Tabu utilizada trata-se de uma matriz *T x 4* em que cada linha corresponde a um movimento, e as 4 colunas de cada linha caracterizam o movimento da lista de tamanho *T*. A figura 5.16 ilustra a forma como o movimento é armazenado. No caso "A" temos um movimento do tipo *1x1*, onde em "1" tem-se o movimento realizado pela troca e em "2" tem-se o correspondente movimento a ser inserido na Lista Tabu. Em "1" temos que o cliente *p* foi inserido na rota *j* e o cliente *q* foi inserido na rota *i*. Então, devemos proibir o movimento inverso que é descrito em "2", ou seja, proibimos um movimento em que o cliente *p* seja reinserido em *i* e o cliente *q* seja reinserido em *j*.

No caso "B" a situação é análoga, porém para um movimento do tipo *0x1*. Em "1", temos o movimento realizado, ou seja, o cliente *p* foi inserido na rota *j*. E em "2" temos o movimento oposto classificado como tabu, ou seja, proibimos a inserção de *p* em *i*.



(Caso A) troca 1x1 (Caso B) troca 0x1

Figura 5.16: estrutura da Lista Tabu

Testes realizados mostraram que uma Lista Tabu de tamanho 5 obteve bons resultados, sendo este o tamanho utilizado nos algoritmos deste trabalho. Tamanhos superiores a 10 mostraram-se inadequados, provavelmente devido à proibição de uma grande quantidade de movimentos. Este estudo não é detalhado aqui devido ao escopo extenso do presente trabalho.

5.2.2 – Procedimento “Inter Rotas Probabilístico Tabu”

A busca “Inter Rotas Probabilístico Tabu” é a mesma busca descrita em 6.1.2.6 adaptada para utilizar a Lista Tabu.

As etapas são dadas a seguir:

1) Escolher o número de rotas a serem perturbadas (*numper*): escolher aleatoriamente 1 ou 2.
2) Selecionar a rota a ser perturbada *(r1)*. Gerar aleatório de 1 ao número de rotas da solução.
3) Determinar rota vizinha (*r2*) em relação a *r1* pelo critério de centro de gravidade.
4) Determinar os conjuntos de clientes *v1* e *v2* que formam as vizinhanças entre *r1* e *r2* como descrito no tópico 5.1.1.1.
5) Escolher o tipo de troca a ser realizado: aleatório entre 1 e 6.
6) Para o tipo escolhido, testar todas as combinações possíveis de troca envolvendo os clientes de *v1* e de *v2*. Dentre todos os movimentos que não estejam na Lista Tabu, armazenar o melhor movimento. Caso não encontre uma troca viável, escolher aleatoriamente outro tipo de troca e repetir testes. Caso não encontre nenhuma troca viável, retornar ao passo 2.
7) Avaliar a função objetivo e comparar com a melhor solução. Se o melhor movimento resultou numa melhoria, efetivar o movimento e inserí-lo na Lista Tabu. Incrementar *np* (contador de perturbações realizadas). Caso *numper* tenha ultrapassado o valor de *np*, ir para passo 8. Caso contrário, ir para o passo 2.
8) Incrementar o contador de movimentos. Caso o número de movimentos tenha atingido um valor máximo pré-estabelecido, finalizar busca, caso contrário retornar ao passo 1.

Nesta busca, há 6 movimentos possíveis: *0x1, 1x0, 1x1, 1x2, 2x1 e 2x2*. Os últimos 3 tipos tratam na verdade de 2 movimentos elementares realizados simultaneamente. Ou seja, o movimento *1x2* trata-se de dois movimentos: *0x1* e *1x1*; o

movimento *2x1* trata-se dos movimentos: *1x0* e *1x1*; e o movimento *2x2* trata-se dos movimentos *1x1* e *1x1*. Então quando estes movimentos são inseridos na Lista Tabu, na verdade eles são decompostos em seus movimentos elementares e depois inseridos. Da mesma forma, quando um movimento é checado na Lista Tabu, considera-se o movimento tabu apenas se os seus dois movimentos elementares forem proibidos.

5.3 - PRV CAPACITADO: HEURÍSTICA H3

A terceira heurística desenvolvida é chamada aqui de *H3*, que basicamente é a heurística *H2* com a inserção da Memória Adaptativa, cujo conceito foi visto no tópico *2.4.2.2*.

Uma primeira diferença é o fato do algoritmo trabalhar com uma lista de boas soluções (memória adaptativa) que chamaremos aqui de *IniSol*. Esta lista tem tamanho *B*. No algoritmo *H1*, vimos que durante a fase de geração das soluções inicias, geram-se 3 soluções, duas por *Gillet* & *Miller* (1974) e uma solução por *Clarke* & *Wright* (1964), conforme explicado em 5.1.1. No algoritmo *H3*, além destas soluções outras soluções irão compor o conjunto *IniSol*. Este conjunto será inicialmente formado por uma solução de *Clarke* & *Wright* (1964), e por *B*-1 soluções de *Gillet* & *Miller* (1974).

Uma segunda diferença é que na fase de melhoria, onde as buscas locais são aplicadas, há a adição do procedimento "Operador Adaptativo", de modo que as vizinhanças exploradas são:

- Eliminação de Rotas.
- Propagação Parcial de Rotas V1.
- Propagação Parcial de Rotas V2.
- Busca Tabu
- Inter Rotas Probabilístico Tabu.
- Operador Genético.
- Operador Adaptativo.

A seguir será explicado o funcionamento do "Operador Adaptativo" onde também será compreendido como o conjunto de soluções *IniSol* é utilizado pelo algoritmo.

5.3.1 – Procedimento "Operador Adaptativo"

Basicamente, o procedimento avalia todas as rotas de cada uma das soluções do conjunto *IniSol*. As rotas destas soluções compõem a chamada "Lista Adaptativa". Vale observar que as rotas da melhor solução corrente conseguida pelo algoritmo (*BestSol*) também fazem parte desta lista. As melhores rotas são escolhidas para formar novas soluções. Estas novas soluções não apenas são comparadas com *BestSol*, mas também são comparadas com todas as soluções de *IniSol*, de modo a substituir as soluções deste conjunto que forem pior avaliadas que as novas soluções geradas. Assim, o procedimento renova o conjunto de soluções, que pode ser visto no contexto como uma renovação dos indivíduos desta população.

A seleção das rotas é feita segundo o "Método da Roleta" (ver Vasconcelos, 2008b), ou seja, rotas com melhor avaliação (função de desempenho *fitness*) possuem maior probabilidade de serem selecionadas para compor as novas soluções. A *fitness* é formada por dois critérios: o valor da distância total da solução em que a rota faz parte e o tamanho do arco de maior comprimento da rota. Quanto maior o valor da *fitness* da rota, melhores são as características desta rota, então maior é a probabilidade desta rota ser selecionada para compor novas soluções. Estes dois critérios foram escolhidos pois no tópico 2.4.2.2 em que foram vistos conceitos da Memória Adaptativa, estudos da literatura mostraram que rotas pertencentes a melhores soluções devem ter alguma preferência no processo de seleção, além disto, estudos citados no tópico 2.4.2.3 quando foi visto o *Guided Local Search*, mostraram que arcos compridos dificilmente fazem parte da solução ótima. Portanto a *fitness* busca quantificar estas características de forma a favorecer a seleção de rotas que pertençam a soluções melhor avaliadas quanto à distância total percorrida pelos veículos e que ao mesmo tempo não possuam arcos compridos quando comparado com as demais rotas.

A *fitness* tem sua estrutura baseada no Método da Inversão (ver Vasconcelos, 2008), conforme equação 5.1:

$$d(x) = \frac{1}{\lambda + h(x)} \tag{5.1}$$

Onde *d(x)* é a função *fitness* para uma dada rota *x,* λ é uma constante (ver eq. 5.2) e *h(x)* é uma composição entre as duas características da rota (ver eq. 5.3).

$$\lambda = 10^{-n} - h(x^*) \quad (5.2)$$

Onde *n* é uma constante, de modo que quanto maior seu valor, maior será a diferença entre as *fitness* da rota melhor avaliada e da rota pior avaliada; h(x*) é a rota de menor valor h(x).

$$h(x) = c(x) + \mu * a(x) \quad (5.3)$$

Onde *c(x)* é custo da solução em que a rota *x* pertence; *a(x)* é o valor do arco de maior comprimento da rota *x*; e μ é uma constante para dar peso ao valor do arco.

O procedimento é descrito a seguir:

1) Gerar uma matriz para armazenar os valores *c(x)* e *a(x)*. Então calcular *h(x)* e *d(x)* para cada rota e armazenar nesta mesma matriz, em que cada linha corresponde a uma rota. As rotas de todas as soluções de *IniSol* são avaliadas.

2) Fazer a seleção das rotas pelo "Método da Roleta". Cria-se um vetor de seleção (*vs*) de tamanho *nr* (número total de rotas), com os índices das respectivas rotas. Rotas melhor avaliadas pela *fitness* tendem a ter mais cópias neste vetor que rotas pior avaliadas.

3) Escolher aleatoriamente uma posição em *vs,* para selecionar o índice de uma rota. Salvar a rota correspondente a este índice numa matriz *filho*. Repetir processo enquanto possível, de modo que cada rota escolhida não deve ter clientes que já façam parte de *filho*.

4) Se *filho* tiver *N* clientes, então a solução está pronta. Caso contrário, aplicar uma heurística construtiva para completar a solução *filho*.

5) Retornar ao passo 2 até que *M filhos* tenham sido gerados.

6) Comparar cada *filho* com *BestSol* e com cada solução de *IniSol* e fazer substituições conforme avaliação da função objetivo de cada solução.

Uma terceira diferença em H3 em relação a H2 está na geração da solução perturbada a ser escolhida quando o algoritmo estaciona num ótimo local. Em verdade, o procedimento de perturbação é exatamente o mesmo descrito em 5.1.3, mas agora há mais opções para se escolher a solução que irá retornar para a fase de melhoria do algoritmo, pois o conjunto *IniSol* é maior.

Desta forma, o "Operador Adaptativo" está constantemente tentando gerar uma nova melhor solução a partir das rotas das soluções de *IniSol*. Mesmo quando não gera uma nova melhor solução, renova parte das soluções de *IniSol*, melhorando a qualidade destas soluções e ajudando no processo de perturbação.

5.4 - PRV CAPACITADO: HEURÍSTICA H4

A quarta heurística desenvolvida é chamada aqui de *H4*, que basicamente é a heurística *H3* com a inserção do *Guided Local Search* (ver tópico *2.4.2.3*) como critério de seleção da solução perturbada que deve retornar à fase de melhorias do algoritmo.

Nas heurísticas anteriores, utilizou-se um critério simples para escolher a solução perturbada. Conforme descrito em 5.1.3, escolhe-se inicialmente a melhor solução de *IniSol* que seja diferente de *BestSol.* Esta solução é inserida numa lista de soluções candidatas ou perturbadas. Aplica-se à solução escolhida três operadores (Cruz. Natural, B2 e B4) gerando-se mais 3 soluções que também são inseridas na lista de soluções perturbadas. Então a melhor solução dentre estas 4 soluções retorna para a fase de melhorias do algoritmo para tentar fugir do ótimo local em que no presente momento o algoritmo encontra-se preso. Nas heurísticas anteriores, o critério de seleção é dito "simples", ou seja, escolhe-se entre as candidatas, aquela que tiver melhor avaliação da função objetivo (menor distância total).

Na heurística H4, o critério de seleção muda, e passa-se a escolher a solução candidata que tiver a melhor avaliação da função pseudo-objetivo *h(x),* conforme equação 5.4.

$$h(x) = c(x) + \mu * a(x) \tag{5.4}$$

Onde *c(x)* é a avaliação da função objetivo da solução *x*; *a(x)* é o valor do arco de maior comprimento da solução *x*; e μ é uma constante para dar peso ao valor do arco.

5.5 –PRV COM DISTÂNCIA MÁXIMA

Os experimentos realizados na sessão 7.1.5 (será visto adiante) mostraram que a heurística H4 foi capaz de fornecer melhores resultados. Desta forma, decidiu-se utilizar aqui a heurística H4 modificada para resolver os problemas com a restrição de distância máxima das rotas.

A alteração é sutil. Sempre que um movimento inter-rotas (entre 2 rotas) é realizado, a função objetivo deve ser avaliada para verificar se as 2 novas rotas satisfazem a restrição de máxima distância. Ou seja, deve-se checar se as 2 novas rotas possuem comprimento inferior à distância máxima admissível (parâmetro de entrada do problema). Caso a nova rota viole a restrição, o movimento deve ser rejeitado.

No caso dos movimentos intra-rotas (troca de clientes de uma mesma rota) não é necessário realizar o teste para verificar se a restrição de distância máxima foi atendida. Isto ocorre, pois o algoritmo sempre trabalha com soluções viáveis. Uma vez que um procedimento que realize um movimento intra-rotas como o 2-opt, por exemplo, receba uma rota viável como entrada, o procedimento só irá efetivar algum movimento nesta rota caso haja melhoria, ou seja, caso a distância total da rota seja reduzida e nunca aumentada.

Na geração das soluções iniciais, tratando-se de heurísticas construtivas, sempre que a inserção de um cliente na rota corrente é testada, além de verificar a capacidade deve-se verificar também a distância.

5.6 –PRV COM JANELA DE TEMPO

Foi implementada uma heurística baseada na H4 pelo mesmo motivo já descrito no tópico anterior. Porém, as modificações para tornar H4 capaz de resolver problemas com janelas de tempo foram bem maiores que as modificações para torná-la capaz de resolver problemas com distância máxima. Como nos modelos anteriores, a função objetivo primária é minimizar o número de veículos, e a secundária é minimizar a distância total percorrida. Neste trabalho chamamos a heurística para resolver esta classe de problemas de HJT. A figura 5.17 mostra a estrutura básica do algoritmo (a mesma usada em H1 a H4), colocada aqui para facilitar referências.

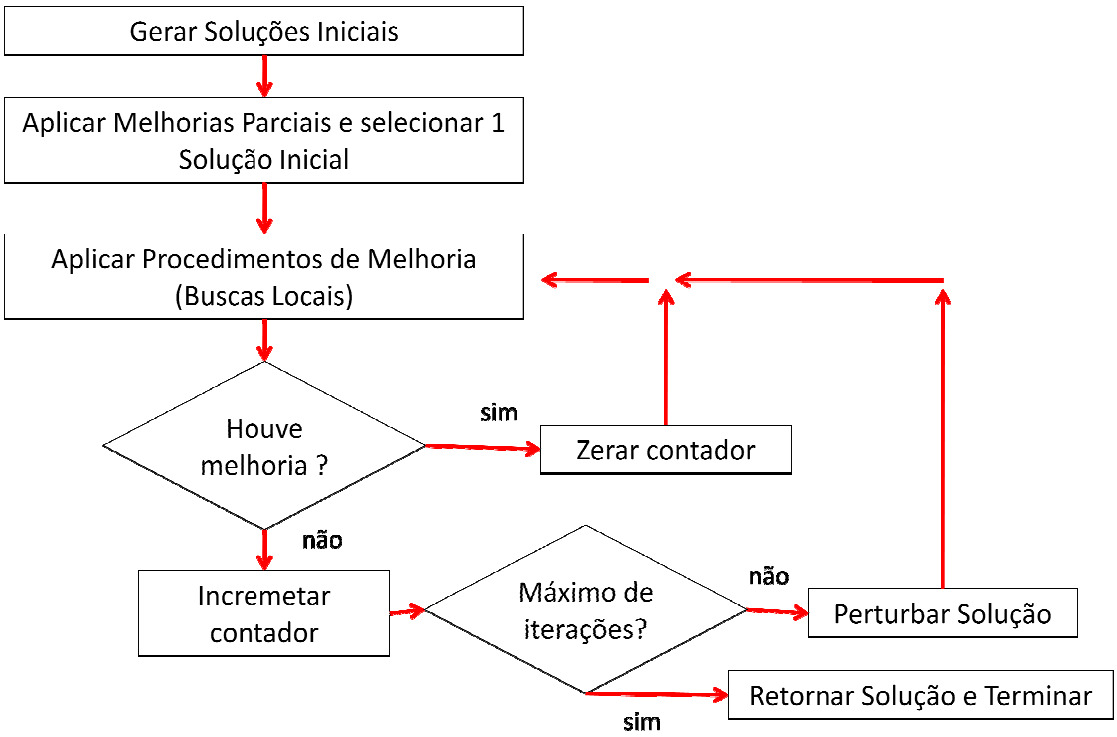


Figura 5.17: estrutura principal de HJT

A etapa "Gerar Soluções Iniciais" é dada pela heurística construtiva *I1* de Solomon, implementada conforme descrito no tópico 3.2.1. Gera-se 20 soluções iniciais utilizando diferentes valores para os parâmetros α_1, α_2, λ e μ. Uma bateria de testes foi realizada para encontrar os parâmetros que fornecessem boas soluções para todas as instâncias testadas sem necessidade de definir parâmetros específicos para cada instância.

A etapa "Aplicar Melhorias Parciais" consiste na aplicação do método *2-opt* a todas as 20 rotas. A seguir, o procedimento de eliminação de rotas descrito na sessão 5.6.2 é aplicado às três melhores soluções iniciais. O passo seguinte é aplicar a busca local "Realocação" a estas soluções. Então, seleciona-se a melhor delas para ser entrada da fase de melhorias do algoritmo.

Na etapa "Aplicar Procedimentos de Melhoria" utiliza-se os seguintes operadores: Eliminar Rotas JT (descrito adiante em 5.6.2), Inter Rotas Probabilístico Tabu (descrito em 5.2.2), Cruzamento Natural (descrito em 5.1.2.7), Realocação Cross (descrito a seguir em 5.6.1) e Operador Adaptativo (descrito em 5.3.1). O procedimento de eliminação de rotas é sempre executado em primeiro lugar. A seqüência dos demais procedimentos é definida aleatoriamente.

Naturalmente, todos os procedimentos citados foram modificados para verificar a violação das janelas de tempo das soluções, sendo aplicado tanto em movimentos inter-rotas como também em movimentos intra-rotas.

A busca local "RealocaçãoCross" e o procedimento "Eliminar Rotas JT" foram especificamente implementados para serem utilizados no modelo com janela de tempo, sendo descritos a seguir.

5.6.1 – Realocação Cross

A busca "Realocação Cross" utiliza o método "Realocação" descrito em 5.1.2.4 e também o método *Cross* (Taillard, 1997) descrito no tópico 3.2.1.1. O método *Cross* preserva a orientação das rotas, o que é uma boa característica para problemas com janelas de tempo.

Neste procedimento, dado uma solução, obtém-se toda a vizinhança desta solução a partir de movimentos do tipo *0x1* e *Cross*. O melhor movimento que não seja tabu é avaliado. Caso este movimento implique numa melhoria, ele é inserido na Lista Tabu e o processo de busca recomeça. Os passos são:

5) Determinar as *NR* rotas vizinhas de cada rota da solução.
6) Para cada rota *ra* da solução corrente, para cada cliente desta rota, para cada rota *rv* vizinha à rota *ra*, se o movimento não for tabu, testar movimento de inserção do cliente de *ra* em *rv*. Armazenar o melhor movimento (tipo *0x1*).
7) Para cada rota *ra* da solução corrente, para cada rota *rv* vizinha à rota *ra* , testar todos os possíveis movimentos tipo *Cross* entre *ra* e *rv* que não sejam um movimento da lista tabu. Armazenar o melhor movimento (tipo 1x1).
8) Caso o melhor movimento dos passos 2 e 3 tenha resultado numa melhoria da melhor solução corrente, inserir este movimento na Lista Tabu, renomear esta solução como solução corrente. Retornar ao passo 1.

5.6.2 – Eliminação de Rotas JT

O processo de redução de rotas nos problemas com janela de tempo mostrou-se mais difícil que nos problemas puramente capacitados. Nos problemas com janelas de tempo a exploração de rotas e clientes geograficamente mais próximos já não é tão importante quanto nos problemas puramente capacitados.

O algoritmo 5.2 descreve o novo procedimento desenvolvido especificamente para o PRVJT. Inicialmente escolhe uma rota alvo candidata à eliminação (rota com capacidade ociosa), identificada como *ra*. Então, tenta-se mover 1 cliente de cada vez para outras rotas. Quando não consegue, força a inserção de 1 cliente numa determinada rota, chamada *r2*. Isto faz com que a rota *r2* seja violada. Então a função *PropagarJT* tenta tornar a rota novamente viável fazendo tentativas de mover 1 cliente da rota violada para alguma outra rota. Quando este processo falha, tenta-se mover 2 clientes adjacentes de cada vez para outras rotas, pois talvez seja possível que a inserção de 2 clientes com a remoção de apenas 1 outro deixe a rota receptora viável. Caso este processo falhe, força-se a inserção de 2 clientes e a função *PropagarJT* tenta tornar a rota receptora novamente viável realocando clientes para outras rotas. A função *PropagarJT* é descrita adiante no algoritmo 5.3.

O objetivo é continuar este processo de remoção de clientes da rota alvo, 1 de cada vez, até que a rota seja excluída. Caso o processo de exclusão de uma rota alvo não tenha êxito, muda-se a rota alvo e o processo recomeça.

A descrição de algumas variáveis é dada a seguir. *r*: matriz solução, onde cada linha é uma rota e as colunas são os clientes; *nv*: número de veículos da solução; *e, l, s* e *dis* são respectivamente: vetor com abertura das janelas dos clientes, vetor com final das janelas dos clientes, vetor com o tempo de serviço dos clientes, matriz com distância entre os nós. Temos ainda *ncra*: número de clientes da rota de índice *ra*; *excluiu* e *facr2*: variáveis binárias; *r2aux, r1aux, raux* são vetores que possuem os clientes das rotas de índices *r2, r1 e ra* respectivamente.

A função *RotasVizinhas* calcula *viz*: um vetor com os índices das rotas vizinhas de *ra* já ordenado pela proximidade; e *nviz*: quantidade de rotas vizinhas. A função *2optAdap* possui 3 dados de entrada. Os 2 primeiros são 2 rotas e um terceiro parâmetro de entrada pode ter valores 1 ou 2. Se valor 1, então significa que 1 cliente foi adicionado à rota *raux2*. Então o objetivo da função é realizar movimentos 2-opt intra-

rotas para tentar “acomodar” o cliente adicionado. Caso tenha sucesso, retorna *facr2*=1, e também *raux2* atualizado. Se valor 2, então 2 clientes foram adicionados à *raux2*. Então o objetivo da função é mais uma vez buscar “acomodar” os clientes. Aqui há 3 situações: uma em que ambos os clientes são incluídos com sucesso (rota é viável), outra em que nenhum cliente pode ser incluído (retorna *facr2*=0), e uma terceira situação em que 1 dos 2 clientes pode ser adicionado de modo que a rota permaneça viável (quanto à capacidade e janelas). Neste caso, 1 cliente é adicionado à *raux2*, outro é adicionado à *raux* e *facr2* também tem valor 1, pois conseguiu inserir pelo menos 1 cliente com sucesso.

Algoritmo 5.2: *[r]* = **EliminarRotasJT** (*r, nv, e, l, s, dis*)

```
01:  Ordenar rotas pela capacidade ociosa em ordem decrescente;
02:  para (i = 1, 2, ..., ⌈nv/3⌉) faça
03:      ra= i-ésima rota mais ociosa;
04:      excluiu=1;
05:      [viz,nviz]=RotasVizinhas(r,ra);
06:      enquanto (ncra) > 0) & (excluiu=1) faça
07:          excluiu =0;  clira=r(ra,1);
08:          para (k = 1, 2, ..., nviz) faça
09:              r2= viz(k); r2aux=r(r2,:); raux=r(ra,:);
10:              Adicionar clira em r2aux;
11:              [r2aux, raux , facr2] = 2optAdap(r2aux, raux, 1);
12:              se facr2=1;
13:                  r(r2,:)=r2aux; excluiu =1;
14:              ou então
15:                  rr = r; rr(r2,:)=r2aux;
16:                  (rr,excluiu)= PropagarJT (rr,ra,r2,1);
17:                  se excluiu=1 então
18:                      r=rr;
19:                      ncra = ncra -1;
20:                      Retornar para linha 6;
21:                  fim;
22:              fim;
23:          fim;
24:          clira1=r(ra,1); clira2=r(ra,2);
25:          para (k = 1, 2, ..., nviz) faça
26:              r2= viz(k); r2aux=r(r2,:); raux=r(ra,:);
27:              Adicionar clira1 e clira2 em r2aux;
28:              [r2aux, raux , facr2] = 2optAdap(r2aux, raux, 2);
29:              se facr2=1;
30:                  r(r2,:)=r2aux; excluiu =1;
31:                  r(ra,:)=raux;
32:              ou então
33:                  rr = r;
34:                  rr(r2,:)=r2aux; raux=r(ra,:);
35:                  (rr, excluiu)=PropagarJT(rr,ra,r2,2);
36:                  se excluiu=1 então
37:                      r=rr;
38:                      ncra = ncra -1;
39:                      Retornar para linha 6;
40:                  fim;
41:              fim;
42:          fim;
43:      fim;
44:  fim;
```

O algoritmo 5.3 detalha a função *PropagarJT*. A função *Combinação* retorna *cen*: uma matriz de cenários, onde cada linha é uma tentativa de exclusão (realocação) de clientes a ser realizada. Quando *tipo*=1, cada linha tem apenas 1 cliente. Quando *tipo*=2, cada linha tem 2 clientes; *ncen* é o número de linhas ou cenários da matriz *cen*. Os tipos 1 e 2 tratam-se de combinações de n clientes tomados 1 a 1 e 2 a 2 respectivamente. Aqui, a função *2optAdap,* além do descrito anteriormente, também retorna o escalar *vio*: tamanha da violação e *M*: uma matriz com o cliente e sua posição na rota em que há a menor violação, seja da capacidade ou da janela de tempo.

Algoritmo 5.3 *[rr, excluiu]* = **PropagarJT**(*rr, ra,r2,tipo*)

```
01:   r1= r2
02:   para (i = 1, 2, … , nv) faça
03:      [cen, ncen]=Combinação(r1,tipo);
04:      vclira=cen (i,:);
05:      minvio=10^10;
06:      para (j = 1, 2, … , ncen) faça
07:         [viz,nviz]=RotasVizinhas(rr,r1);
08:         para (k = 1, 2, … , nviz) faça
09:            r2= viz(k); onde r2≠ra
10:            r2aux=r(r2,:);
11:            Adicionar clientes de vclira em r2aux;
12:            [r2aux, raux, facr2, M, vio] = 2optAdap(r2aux, raux, tipo);
13:           se vio<minvio então
14:                minvio=vio; Maux=M;
15:            fim;
16:            se facr2=1 então
17:                Excluir clientes em vclira de r1.
18:                excluiu=1; rr(ra,:)=raux;
19:                terminar;
20:            fim;
21:         fim;
22:      fim;
23:      Excluir clientes de vclira de rr( r1,:);
24:      Inserir clientes de M em rr(r2,:);
25:      r1=r2;
26:   fim;
```

5.6.3 – Eliminação de Rotas JT Modificado

Na fase de perturbação, uma mudança relevante foi a substituição do procedimento *Shift* utilizado no modelo capacitado por um procedimento muito similar ao "Eliminar Rotas JT". Esta mudança ocorreu, pois testes mostraram que o movimento circular feito no procedimento *Shift* não se mostrou adequado devido às restrições de

janela de tempo. No movimento do procedimento *Shift*, ocorre muitas vezes de tentar-se inserir o último cliente de uma rota no início de outra rota. Numa instância com janelas apertadas isto normalmente resulta numa rota inviável.

Então o papel de impedir que o conjunto "*IniSol*" tenha rotas repetidas é feito por uma adaptação do procedimento "Eliminar Rotas JT". Aqui o objetivo não é realocar os clientes de uma rota alvo, um de cada vez, para outras rotas até eliminar a rota. O objetivo é excluir 1 cliente de 1 ou 2 rotas, realocando estes clientes para outras rotas através da função "PropagarJT". Esta função pode causar modificações em mais de uma rota, gerando, portanto uma solução diferente apta a substituir uma solução repetida do conjunto "*IniSol*". A escolha por excluir apenas 1 cliente de 1 ou 2 rotas foi feita para não provocar muitas mudanças na solução original.

Aqui há um *trade-off* a ser considerado. Se mudar pouco a solução, ela não irá ajudar a criar indivíduos diferentes nos cruzamentos genéticos. Então as soluções iriam convergir facilmente para ótimos locais já encontrados. Ao mesmo tempo, mudanças drásticas na solução original iriam levar o algoritmo a um simples "*restart'*.

A figura 5.18 mostra a convergência do algoritmo para uma certa instância com janela de tempo.

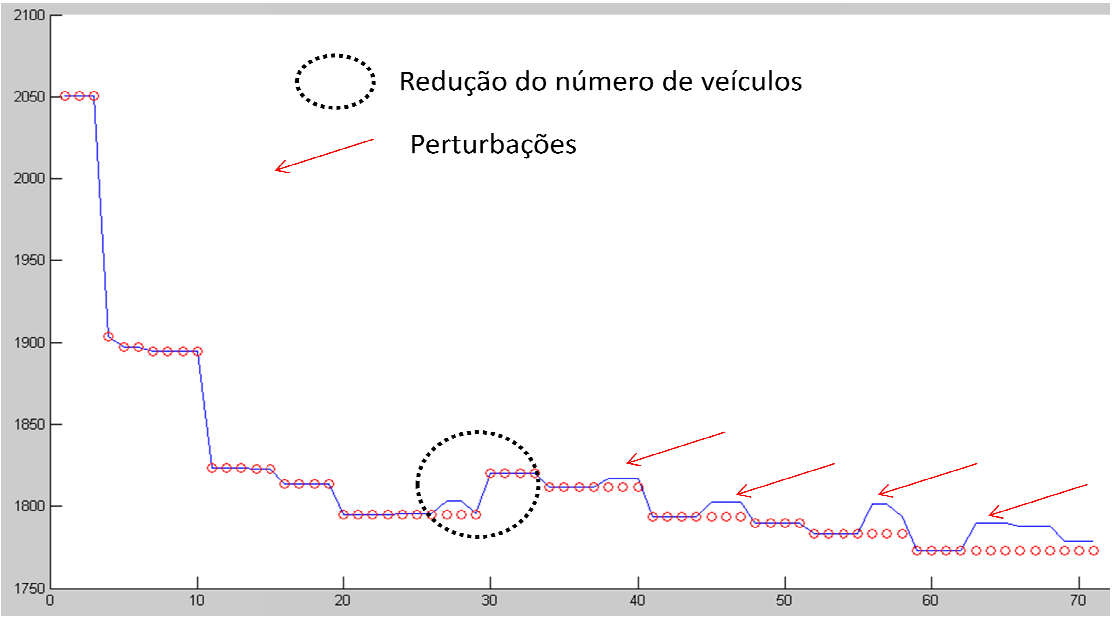


Figura 5.18: Convergência e perturbações

Observa-se que na região circulada há uma piora na avaliação da distância total percorrida. Neste momento houve a redução de 1 veículo da solução com um

aumento da distância percorrida. Logo a seguir observa-se que o algoritmo voltou a convergir, sendo possível ver as perturbações geradas. Antes da região circulada não houve pioras na solução corrente, pois não houve perturbações. Neste momento, a fase de buscas locais foi executada enquanto houve convergência.

5.7 – DISCUSSÃO FINAL

Este capítulo dedicou-se aos problemas de roteamento determinísticos, sem e com janela de tempo. A figura 5.19 resume as principais diferenças entre os algoritmos desenvolvidos para o PRV Capacitado.

	Busca	H1	H2	H3	H4
1	Eliminação de Rotas	x	x	x	x
2	Propagação de Vizinhanças V1				
3	Propagação Parcial de Rotas V1	x	x	x	x
4	Propagação Parcial de Rotas V2	x	x	x	x
5	Realocação	x			
6	Intercâmbio	x			
7	Inter Rotas Probabilístico	x			
8	Cruzamento Natural	x	x	x	x
9	Busca Tabu		x	x	x
10	Inter Rotas Probabilístico Tabu		x	x	x
11	Operador Adaptativo			x	x
12	Fase de Perturbação com GLS				x

Figura 5.19: Resumo das diferenças entre os algoritmos

Neste capítulo foram desenvolvidas 4 heurísticas. A primeira tratou-se de uma estratégia *ILS* e *VND* com estruturas de vizinhanças clássicas, baseadas em movimentos de realocação e intercâmbio.

Com a heurística H1, foi possível aprender como a solução inicial é importante para o algoritmo. Diferentes heurísticas clássicas foram implementadas como: Algoritmo do Vizinho Mais Próximo (Solomon, 1987), Fisher & Jaikumar (1981), Mole & Jameson (1976), Gaskell (1967), Gillet & Miller (1974) e Clarke & Wright (1964). Testes mostraram que as duas últimas apresentaram melhores resultados. Estes testes não tiveram os resultados incluídos na pesquisa devido ao escopo extenso do presente

trabalho. Além disto, a sensibilidade de metaheurísticas à solução inicial é algo bastante conhecido da literatura. Ainda assim, os testes foram úteis para que o autor do trabalho conhecesse na prática o grau de sensibilidade do algoritmo diante diferentes estratégias de solução inicial.

H1 também foi importante para amadurecer o mecanismo de funcionamento da fase de perturbação. Esta fase mostrou-se muito útil para escapar de ótimos locais. Também foi importante para entender o funcionamento do *VND* e a importância de variar as estruturas de vizinhança para realizar as buscas locais. Observou-se que no início da execução, nas primeiras iterações do algoritmo, mais de uma busca conseguia melhorar a solução. Quando o algoritmo já tinha convergido para soluções melhores, observou-se que em várias situações, num mesmo *loop* da fase de buscas locais, apenas uma estrutura de vizinhança conseguia melhorar a solução, porém esta variava entre todas as estruturas disponíveis. Em H1, as buscas que realizaram mais melhorias foram: Realocação, Intercâmbio e Propagação Parcial de Rotas V2. A eliminação de qualquer das estruturas de vizinhanças implicaram numa piora da qualidade da solução.

H2 foi importante para aprender a trabalhar com a Busca Tabu. Observou-se que trabalhar com listas maiores (digamos maior que 10) eram prejudiciais ao algoritmo. Então trabalhamos com listas menores para não ter o risco de proibir muitos movimentos.

H3 foi importante para entendermos o mecanismo de funcionamento da Memória Adaptativa. O funcionamento ocorre de modo similar a um algoritmo genético em que os indivíduos são as rotas e as melhores rotas têm maior probabilidade de serem selecionadas para formarem uma solução.

H4 foi útil para entender como o conceito *GLS* pode ajudar a selecionar soluções com maior potencial de melhora. Foi possível observar que, de fato, privilegiar a escolha de uma solução dentre outras considerando unicamente a avaliação da função objetivo não é a melhor estratégia. Características não promissoras de uma solução podem não ser expressas apenas pela distância total percorrida e número de veículos. Convém identificar tais características (como por exemplo, um arco muito comprido), avaliá-las e incluir este componente na avaliação da função objetivo.

O mecanismo de eliminação de rotas desenvolvido é inédito. Ele permitiu encontrar o número mínimo de veículos em todas as instâncias testadas. Tanto o

mecanismo usado nos problemas puramente capacitados como o utilizado nos problemas com janela de tempo funcionaram bem. O conceito de "propagação de vizinhanças" mostrou-se útil. Eliminar rotas nos problemas com janela de tempo mostrou-se mais difícil, provavelmente por esta restrição ser muito forte e fornecer uma quantidade menor de possibilidades que permitam reduzir o número de veículos.

A utilização do conjunto de solução, aqui chamado de *IniSol*, (uma memória adaptativa) também foi importante. Testes mostraram que a não utilização deste mecanismo piorou o resultado obtido. Convém observar que a memória adaptativa usado neste trabalho (*IniSol*) não é utilizada somente para gerar a "Lista Adaptativa" (conjuntos de rotas das soluções da memória adaptativa) usada no "Operador Adaptativo" das heurísticas H3 e H4 e descrito em 5.3.1 a partir das idéias de Rochat & Taillard (1995). O conjunto *IniSol* também é utilizado em todas as heurísticas desenvolvidas para melhorar o funcionamento da fase de perturbação do algoritmo, sendo que, este tipo de utilização da memória adaptativa não foi encontrado em outros trabalhos durante a revisão da literatura.

Não existiram dificuldades em adaptar H4 inicialmente implementado para o problema puramente capacitado para o problema com distância máxima. O mesmo não ocorreu com o problema com janela de tempo, onde mudanças foram necessárias em 3 pontos importantes do algoritmo: a adição do operador *Cross*, alteração na estratégia de eliminação de rotas e a substituição do procedimento *Shift* da fase de perturbação por outro procedimento similar ao de eliminação de rotas.

A integração dos conceitos do *GLS*, Tabu *Search*, Memória Adaptativa à estrutura do *ILS* e *VND* mostrou-se útil. Isto será claramente observado nos resultados dos experimentos realizados no capítulo 7. Esta integração também não foi encontrada em outros trabalhos de roteamento de veículos.

CAPÍTULO 6

ESTUDOS E ALGORITMOS DESENVOLVIDOS PARA O PRV ESTOCÁSTICO

Neste capítulo, explica-se os estudos realizados para entender a influência da janela de tempo no tempo de chegada do veículo. O objetivo é conseguir uma reposta mais adequada à seguinte questão ainda em aberto na literatura: como calcular a probabilidade de violação da janela de tempo dos clientes sem o uso da simulação?

Inicialmente iremos mostrar que a existência do tempo de espera como variável aleatória faz com que, em muitas situações, a função densidade de probabilidade da variável aleatória "tempo de chegada" não tenha adesão à distribuição normal. Então, estudam-se formas de calcular a média e o desvio-padrão desta variável e, ainda, meios de melhorar a estimativa da probabilidade de violação da janela de tempo. Finalmente, aplica-se o conhecimento adquirido nos passos anteriores para a solução do PRVJT com Tempo de Viagem Estocástico.

6.1 – SUPOSIÇÃO DE NORMALIDADE

Para muitos problemas práticos, o tempo de viagem dos arcos segue uma distribuição de probabilidade normal (Kenion & Morton, 2003; Jie, 2010). Foi visto no capítulo 4 que alguns trabalhos consideram que o tempo de chegada do cliente pode ser aproximado por uma distribuição normal, como Chang *et al*. (2009). Vamos investigar com profundidade se esta suposição é plausível através da rota descrita na tabela 6.1.

Nesta tabela, têm-se os dados de uma rota com 8 clientes. Na primeira coluna temos a descrição dos dados: abertura da janela, fechamento da janela, tempo de serviço (atendimento), tempo do arco (dado determinístico original da instância assumido aqui como a média teórica do tempo de viagem do cliente *i-1* ao cliente *i)*, desvio-padrão do arco (assumido como 20% do tempo do arco), tempo de chegada determinístico, tempo de chegada real, desvio-padrão do tempo de chegada real, tempo de início de atendimento real e desvio-padrão real do tempo de início de atendimento. Os valores ditos "reais" foram obtidos via simulação.

A rota foi obtida através da aplicação da heurística I1 de Solomon (1987) descrita no tópico 3.2.1 aplicada à instância RC101 de Solomon (1987). Os valores da janela de tempo de alguns clientes foram modificados em relação à instância original para melhor ilustração do fenômeno investigado. Considera-se a velocidade do veículo unitária, de modo que o tempo e distância passam a ter os mesmos valores. Assume-se que o tempo seja dado em minutos e que o tempo de serviço nos clientes seja determinístico.

	índice	1	2	3	4	5	6	7	8
Abert. Jan.	e	35	45	72	58	105	119	142	149
Fech. Jan.	l	65	55	102	88	110	124	172	173
T. Atend.	s	10	10	10	10	10	10	10	10
T. Arco	t_{ij}	35,36	3,00	5,39	2,00	9,43	5,00	7,07	11,18
Desv. Arco	σ_{ij}	7,07	0,60	1,08	0,40	1,89	1,00	1,41	2,24
T. Cheg. Det.	TCHi	35,36	48,36	63,74	84,00	103,43	120,00	137,07	163,18
T. Cheg. Real	TCHi	35,36	50,98	66,36	84,44	103,89	120,52	137,66	163,26
Desv. T. Cheg. Real	DEVCHi	7,07	4,29	4,43	1,65	2,50	1,75	2,19	2,32
T. Atend. Real	TATi	37,98	50,98	72,44	84,44	105,52	120,59	142,08	163,26
Desv. T. Atend. Real	DEVATi	4,25	4,29	1,60	1,65	1,43	1,67	0,64	2,32

Tabela 6.1: rota exemplo

A partir da simulação estocástica da rota exemplo (uma rodada de 10000 réplicas), foram coletados os dados do tempo de chegada e tempo de início de atendimento em cada cliente. A figura 6.1 mostra os histogramas do tempo de chegada do veículo em cada um dos 8 clientes da rota.

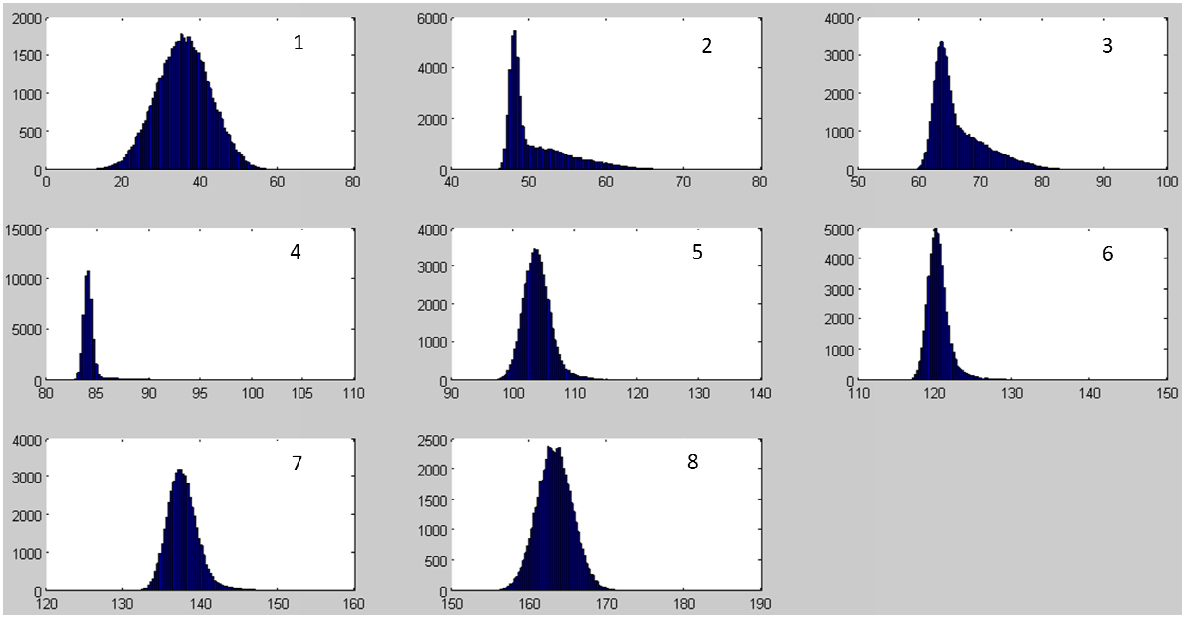


Figura 6.1: histogramas do tempo de chegada nos clientes

Observa-se claramente que a distribuição dos clientes 2 e 3 não são normais. Um teste de normalidade foi aplicado aos tempos de chegada de cada um dos clientes, sendo mostrado na figura 6.2.

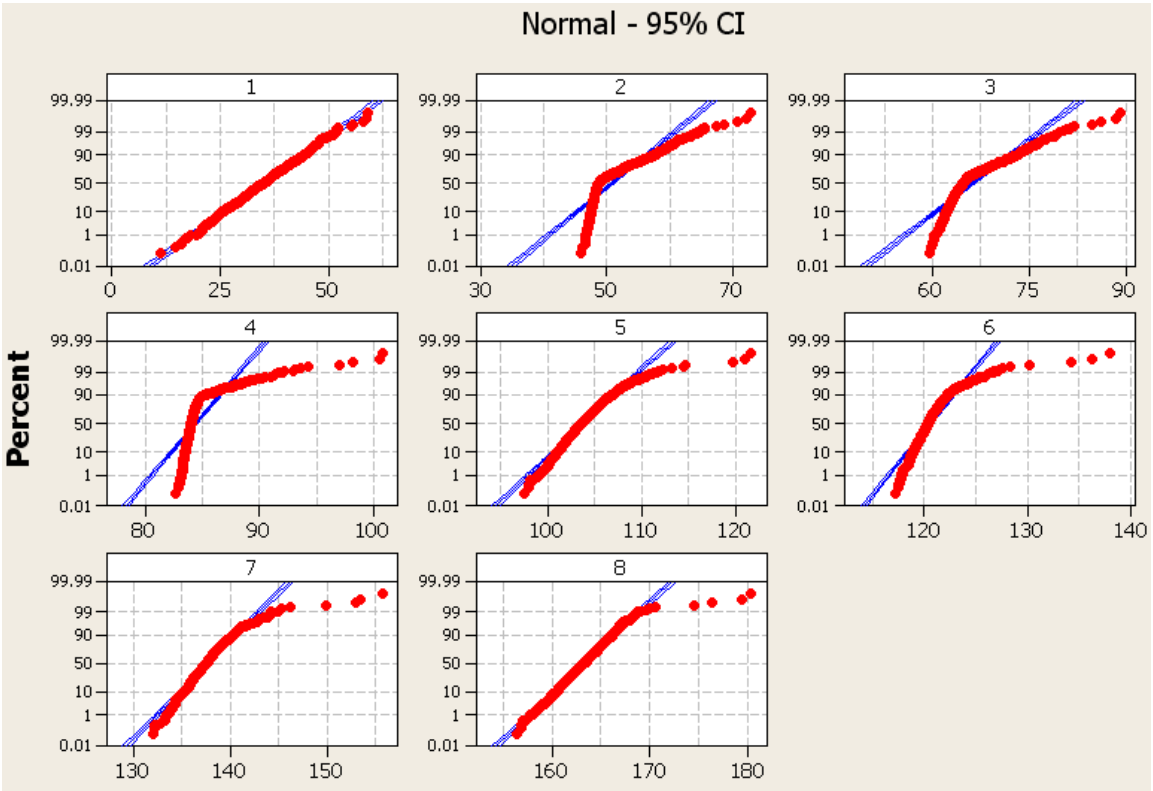


Figura 6.2: Teste de Normalidade usando Anderson-Darling.

Para um nível de confiança de 95%, o cliente 1 apresentou *P-value* de 0.579. Para todos os demais clientes, o *P-value* foi inferior a 0.005. Uma vez que o tempo de viagem dos arcos possui uma distribuição normal, naturalmente o tempo de chegada no primeiro cliente sempre seguirá uma distribuição normal, pois se assume que o tempo de partida do depósito é determinístico. Já para os demais clientes, neste exemplo, ficou claro que o tempo de chegada de todos eles não tem distribuição normal.

A figura 6.3 mostra o histograma do tempo de início de atendimento para cada um dos clientes do exemplo.

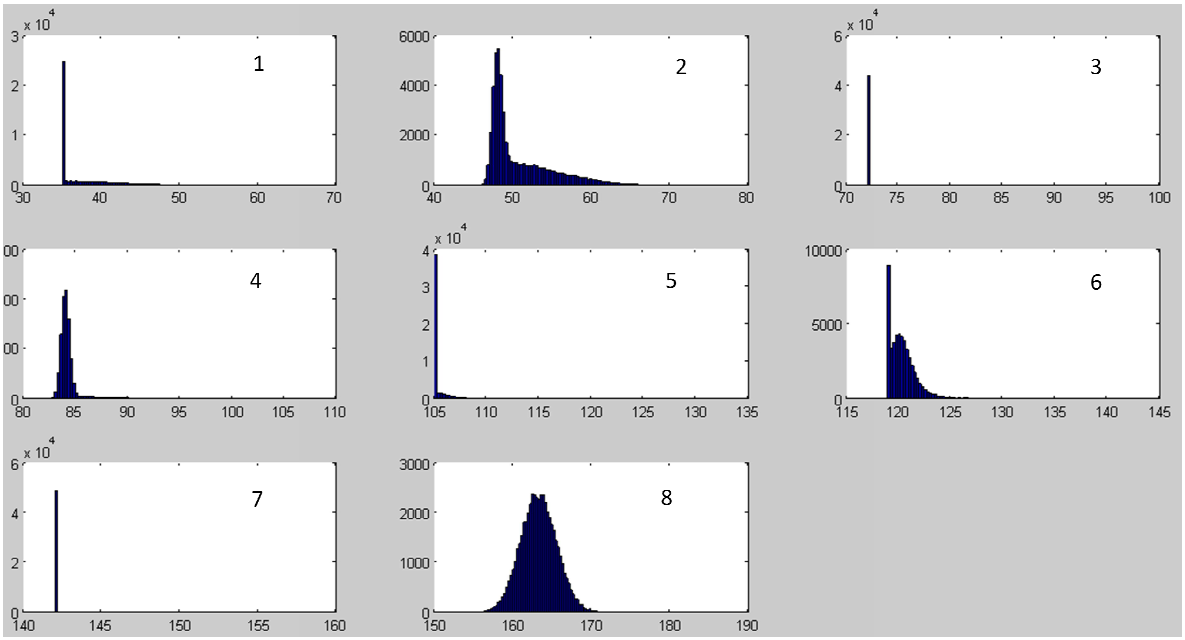


Figura 6.3: histogramas do tempo de início de atendimento nos clientes

Na figura 6.3 é possível perceber o efeito do tempo de espera na distribuição do tempo de início de atendimento. A média do tempo de chegada no cliente 1 é 35,36 minutos e o desvio-padrão de 7,07 minutos. A abertura da janela de tempo ocorre no minuto 35. Sendo a distribuição de probabilidade neste cliente normal, a probabilidade de que o veículo chegue antes do minuto 35 é de 47,8%. Ou seja, em 47,8% das oportunidades, o veículo chegará antes da abertura da janela, existindo tempo de espera, de modo que, para estes valores, o tempo de início de atendimento passa a ser a abertura da janela. Isto é representado no histograma do cliente 1 pela barra vertical grande que representa os 47,8% dos tempos de chegada cujo tempo de início de atendimento é 35. Para os demais 52.2% dos valores, o tempo de início de atendimento é o mesmo do tempo de chegada, resultando nas demais barras vistas no histograma do cliente 1.

Para os demais clientes o raciocínio é análogo, exceto o fato de que a distribuição de probabilidade do tempo de chegada não seja normal. Os clientes 3 e 7 possuem probabilidade de espera de 87,5% e de 96,7% respectivamente. Ou seja, quase sempre há espera, isto é visto no histograma pela barra vertical, onde nem é possível visualizar as barras correspondentes aos valores em que não há espera.

As figuras 6.4 e 6.5, são os histogramas do tempo de chegada para 2 rotas geradas a partir de instâncias de Solomon (1987) também com algumas mudanças nos valores das janelas de tempo para ilustrar qualitativamente o comportamento das distribuições de probabilidade ao longo de uma dada rota.

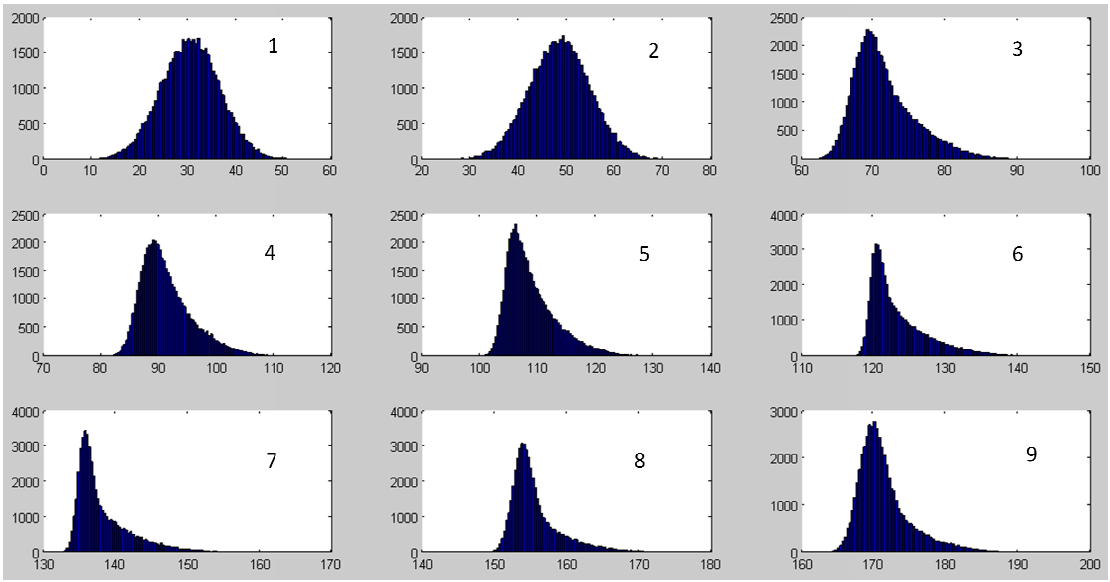


Figura 6.4: histograma do tempo chegada do problema R104

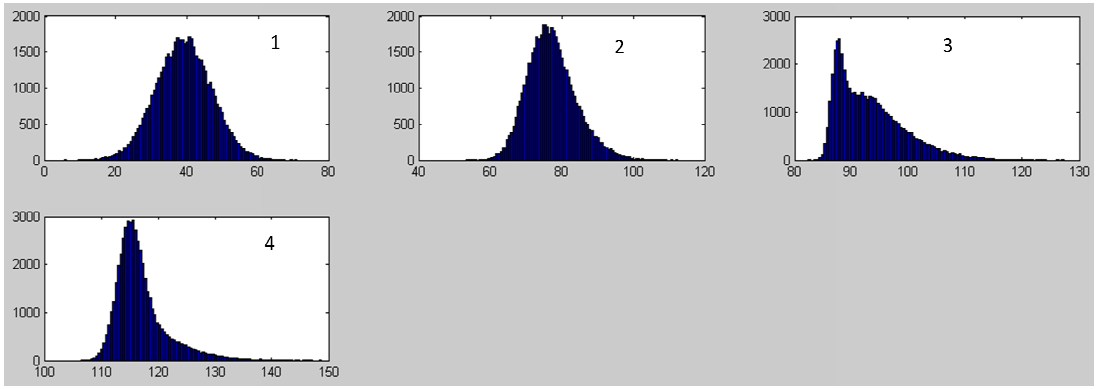


Figura 6.5: histograma do tempo chegada do problema RC101

Observa-se claramente que em diversas situações a distribuição não se assemelha a uma normal. Além disto, a forma da distribuição muda ao longo dos

clientes de uma rota. Qualquer rota, em que pode haver tempo de espera do veículo, está sujeita ao fenômeno investigado. Encontrar uma melhor maneira de lidar com este problema é o principal objetivo deste capítulo.

6.2 – ANÁLISE DO TEMPO DE CHEGADA

Ainda considerando o exemplo da tabela 6.1, vamos investigar a origem da distribuição de probabilidade do tempo de chegada do cliente 2. Ou seja, iremos investigar como acontece a mudança da distribuição do primeiro cliente para o segundo cliente da rota, vista na figura 6.6, a seguir.

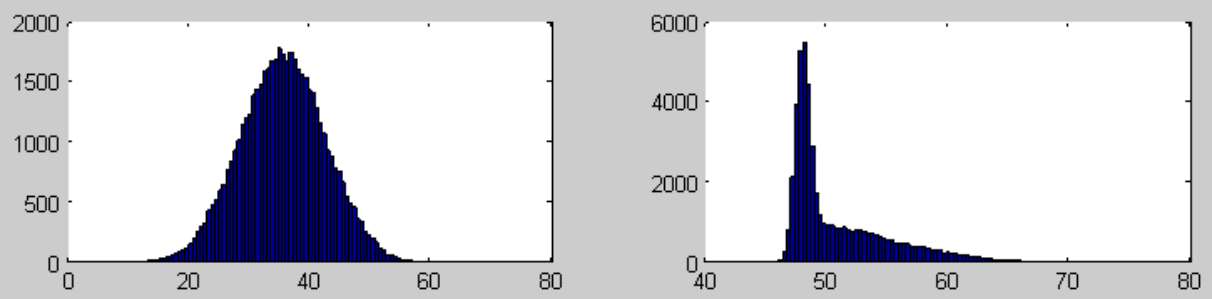
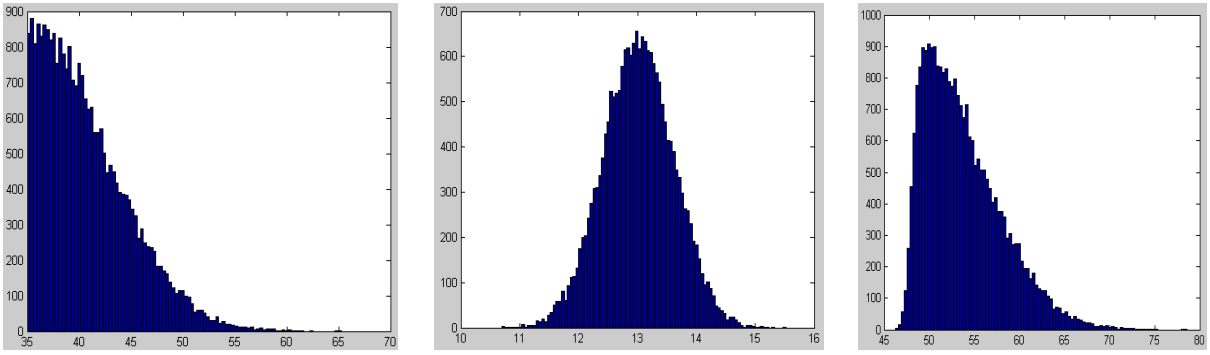


Figura 6.6: transição do primeiro para o segundo cliente

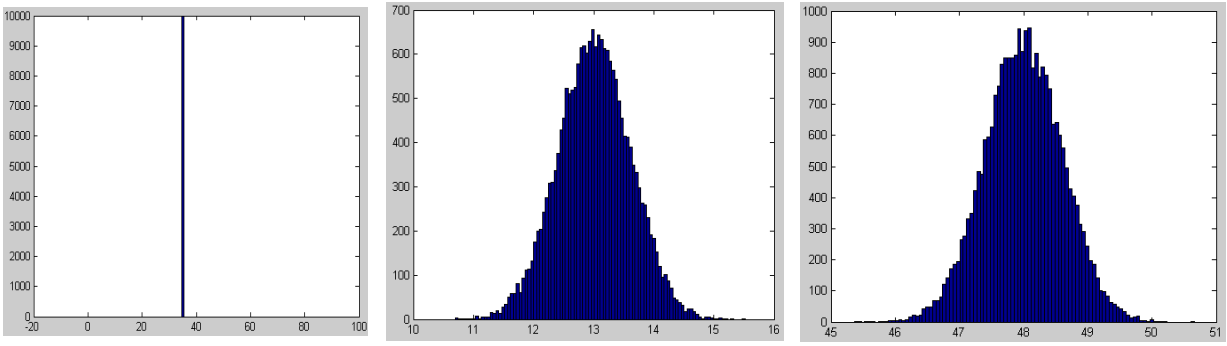
Na figura 6.7, TAT_1' é a distribuição de probabilidade dos valores do tempo de chegada após a abertura da janela de tempo do cliente 1 (minuto 35). No centro da figura, s_1+t_{12} representa a distribuição do tempo de viagem do cliente 1 ao 2 já somada ao valor do tempo de atendimento, ou seja, uma normal $N(\mu;\sigma^2) = N(3+10;0{,}6^2)$, onde a fonte dos dados é a tabela 6.1. À direita da figura, TCH_2' é o tempo de chegada no cliente 2, onde $TCH_2' = TAT_1' + (s_1 + t_{12})$.



TAT_1' s_1+t_{12} TCH_2'

Figura 6.7: mudança do tempo de chegada

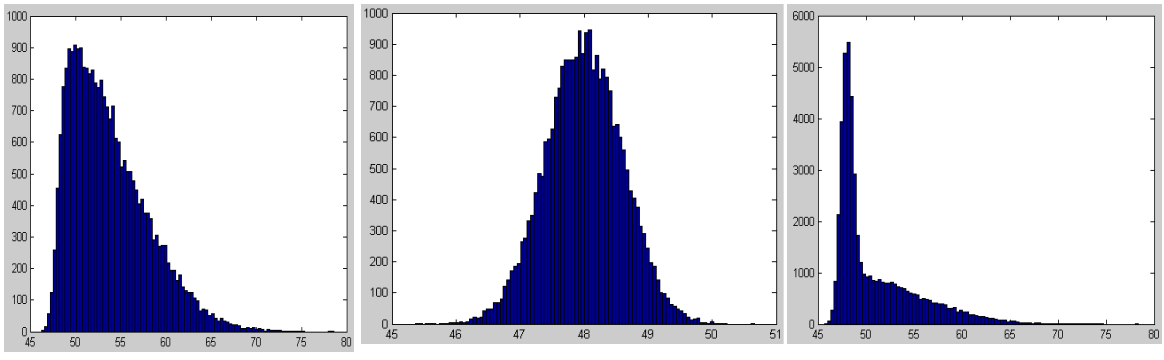
Na figura 6.8, TAT_1'' corresponde aos valores do tempo de chegada antes da abertura da janela de tempo do cliente 1 (minuto 35) que foram corrigidos para um valor constante e igual à abertura da janela (tempo de início de atendimento). No centro da figura, s_1+t_{12} representa a distribuição do tempo de viagem do cliente 1 ao 2 já somada ao valor do tempo de atendimento, ou seja, uma normal $N(\mu;\sigma^2) = N(3 + 10; 0{,}6^2)$. À direita da figura, TCH_2'' é o tempo de chegada no cliente 2, onde $TCH_2'' = TAT_1'' + (s_1 + t_{12})$.



TAT_1'' s_1+t_{12} TCH_2''

Figura 6.8: mudança do tempo de chegada

Observa-se claramente que TCH_2' não segue uma distribuição normal enquanto que TCH_2'' segue uma distribuição normal. Então, o tempo de chegada no cliente 2 é dado pela soma de uma componente normal com outra componente não normal, ou seja, $TCH_2 = TCH_2' + TCH_2''$. Isto é visto na figura 6.9.



TCH_2' TCH_2'' TCH_2

Figura 6.9: Componentes normal e não normal do tempo de chegada

6.3 – ESTIMATIVA DA MÉDIA, DESVIO-PADRÃO E PROBABILIDADE COM USO DA NORMAL TRUNCADA

No tópico anterior, na figura 6.7, observa-se que a variável aleatória TAT_1' segue uma distribuição de probabilidade normal truncada à esquerda, onde o ponto de truncamento é a abertura da janela de tempo. Isto motivou uma primeira estratégia para tentar melhorar os resultados obtidos para o cálculo da média, variância e probabilidade relacionados ao tempo de chegada do veículo em relação aos resultados obtidos pela simples suposição de normalidade do tempo de chegada.

A estratégia consiste em explorar o fato de que o tempo de chegada é composto da soma de uma componente normal com outra componente não normal. Ainda em relação ao exemplo da sessão anterior, não parece coerente tentar aproximar TCH_2 diretamente por uma normal. Podemos aproximar apenas a componente TCH_2' por uma normal, pois sabemos que a componente TCH_2'' é de fato normal. Então o erro do cálculo tem origem apenas na aproximação de uma das componentes.

A média e a variância do tempo de chegada são calculadas como:

$$\mu_{ch}^{j} = \mu_{at}^{i} + \mu_{ij} + s_i \tag{6.1}$$

$$var_{ch}^{j} = var_{at}^{i} + var_{ij} \tag{6.2}$$

Onde: μ_{ch}^{j} é a média do tempo de chegada no cliente *j*, μ_{at}^{i} é a média do tempo de atendimento no cliente *i*, μ_{ij} é a média do tempo de viagem do cliente *i* ao *j*, var_{ch}^{j} é a variância do tempo de chegada no cliente *j*, var_{at}^{i} é a variância do tempo de atendimento no cliente *i*, e var_{ij} é a variância do tempo de viagem do cliente *i* ao *j*.

Por sua vez, a média e a variância do tempo de atendimento são estimados como:

$$\mu_{at}^{i} = (p_1 * e_j) + (1 - p_1) * \mu_{tr}^{i} \tag{6.3}$$

$$var_{at}^{i} = f\left(p_1, var_{tr}^{i}\right) \tag{6.4}$$

Onde p_1 é a probabilidade de espera, ou seja, $P(TCH_i \leq e_i)$, em que TCH_i é uma variável aleatória que descreve o tempo de chegada no cliente *j*.

O termo μ_{tr}^{i} designa a média truncada do tempo de chegada no cliente *i* e var_{tr}^{i} é a variância do tempo de chegada truncado no cliente *i*.

A equação 6.3 faz uma ponderação entre o valor de abertura da janela e o valor da média truncada do tempo de chegada. A função da equação 6.4 será obtida via regressão linear multivariável em sessão posterior.

Para detalhes sobre a distribuição normal truncada citamos Johnson (2003) e Halperin (1952). A média truncada e o desvio padrão truncados podem ser calculados da seguinte forma:

$$\mu_{tr}^{i} = m * \mu_{ch}^{i} + e_i \tag{6.5}$$

$$var_{tr}^{i} = s * \sqrt{var_{ch}^{i}} \tag{6.6}$$

$$s = \sqrt{(1-m) * (m-k)} \tag{6.7}$$

$$m = \frac{\varphi(k)}{[1-\phi(k)]} \tag{6.8}$$

$$k = \frac{e_i - \mu_{ch}^{i}}{\sqrt{var_{ch}^{i}}} \tag{6.9}$$

Onde *k* é o ponto de truncamento e_i convertido para normal padrão, φ é a função densidade de probabilidade e ϕ é a função cumulativa de probabilidade.

Finalmente, o cálculo da probabilidade de violação da janela separando-se a componente normal da não normal é dado por:

$$p_s^j = p_1^i * p_b^j + \left(1 - p_1^i\right) * p_a^j \tag{6.10}$$

$$p_a^j = P\left(TCH1_j > l_j\right) \tag{6.11}$$

$$p_b^j = P\left(TCH2_j > l_j\right) \tag{6.12}$$

Onde p_s^j é a probabilidade de violar-se a janela do cliente *j*, p_1^i é a probabilidade de espera no cliente anterior *i*, p_a^j é a probabilidade do tempo de atendimento da componente não normal violar a janela do cliente *j* e p_b^j é a probabilidade do tempo de atendimento da componente normal violar a janela do cliente *j*.

Na equação 6.11, $TCH1_j$ é uma variável aleatória que descreve o tempo de chegada no cliente *j* partindo de uma situação em que não houve espera. A média teórica é dada por: $E[TCH1_j] = (\mu_{tr}^i + \mu_{ij} + s_i)$ e a variância por: $V[TCH1_j] = (var_{tr}^i + var_{ij})$. E, na equação 6.12, $TCH2_j$ é uma variável aleatória de média $(e_i + \mu_{ij} + s_i)$ que descreve o tempo de chegada no cliente *j* partindo de uma situação em que houve espera, ou seja, o tempo de início de atendimento ocorreu exatamente na abertura da janela. A média teórica é dada por: $E[TCH2_j] = (e_i + \mu_{ij} + s_i)$ e a variância por: $Var[TCH2_j] = (var_{ij})$.

As probabilidades p_ae p_b são calculadas com a suposição de normalidade e são obtidas utilizando-se aproximações numéricas segundo Abramowitz & Stegun (1964).

Analogamente, o mesmo método pode ser aplicado para calcular a probabilidade de espera (p_1) onde $p_{1_s}^j = p_{1_s}^i * p_{1_b}^j + (1 - p_1^i) * p_{1_a}^j$. Os demais termos são calculados como nas equações 6.11 e 6.12, mas em relação à e_j e não mais l_j. No primeiro cliente da rota, p_1^1 é calculado pelo método convencional sem separação, pois a distribuição do tempo de chegada é normal.

Convém notar que o método sugerido nesta sessão precisa calcular a probabilidade de haver tempo de espera (p_1), o que é tão difícil quanto calcular a probabilidade de violação. Esta é outra fonte de erro na estimativa, pois calcula-se a distribuição normal truncada para uma distribuição truncada que muitas vezes não é normal.

6.4 – APROXIMAÇÕES POR REGRESSÃO LINEAR MULTIVARIADA

A probabilidade de violação da janela de tempo no cliente *j* é dada por: $p^j = P(TCH_j > l_j)$, onde TCH_j é uma variável aleatória que descreve o tempo de chegada no cliente *j*. A média é dada por: $E[TCH_j] = (\mu_{tat}^i + \mu_{ij} + s_i)$ e a variância por: $V[TCH_j] = (var_{at}^i + var_{ij})$.

Para calcular a probabilidade de violação da janela de tempo para cada um dos clientes ao longo de uma rota existem algumas variáveis que precisam ser calculadas: a

média e a variância do tempo de chegada (μ_{ch} e var_{ch}), a média e a variância do tempo de atendimento (μ_{at} e var_{at}), a média e a variância truncadas do tempo de chegada (μ_{tr} e var_{tr}) e a probabilidade de espera (p_1).

As equações 6.3 e 6.4 são utilizadas para calcular a média e a variância do tempo de chegada (μ_{ch} e var_{ch}), assumindo-se que o tempo de viagem dos arcos sejam independentes. Porém, este cálculo é afetado pelas demais variáveis: μ_{at}, var_{at}, μ_{tr}, var_{tr} e p_1. Estas variáveis possuem erros nas suas estimativas pelo fato da distribuição do tempo de chegada não ser normal. Então nesta sessão, serão estudadas regressões lineares multivariadas que têm como objetivo reduzir os erros em cada uma das variáveis de interesse com o intuito de melhorar as estimativas da probabilidade de violação da janela de tempo. Convém notar que $p = f(\mu_{ch}, var_{ch}, a)$, onde *a* é o tipo da distribuição que descreve a variável aleatória do tempo de chegada. Então, mesmo quando a μ_{ch} e a var_{ch} são calculados corretamente, o parâmetro *a* continuará a ser uma fonte importante de erro para o cálculo da probabilidade de violação p.

Para estudar estas regressões foi criado um banco com 16 rotas geradas a partir de instâncias de Solomon (1987). Foram selecionadas rotas com resultados ruins quando simplesmente assumido normalidade. Isto foi feito para o banco de testes incluir situações de interesse ao estudo. Algumas instâncias também tiveram as janelas de tempo alteradas para forçar situações de interesse. O banco possui um total de 172 clientes, sendo que cada cliente fornece um dado a ser coletado para as regressões. Sendo os dados de natureza contínua, considerou-se amostragem suficiente para os estudos de regressão. Os resultados e figuras dos tópicos 6.4.1 a 6.4.7 foram obtidos a partir deste banco de rotas.

Antes de estudar as regressões que irão calcular fatores de correção para as variáveis de interesse, convém investigar quando um fator de correção não é necessário.

6.4.1 – Quando não é necessário corrigir

A primeira situação em que não é necessário corrigir o valor obtido de qualquer uma das variáveis de interesse é, naturalmente, os dados referentes ao primeiro cliente da rota. Como o tempo de partida do depósito é suposto determinístico e como o tempo de viagem do depósito ao cliente segue uma distribuição normal, então a

distribuição do tempo de chegada no primeiro cliente sempre é normal, logo correções não são necessárias.

No que diz respeito ao cálculo da probabilidade de espera p_1, uma situação de interesse é dada pela relação entre o fator de correção e o ponto de truncamento. Seja o parâmetro $KL_i = (e_i - \mu_{ch}^i) / \sigma_{ch}^i$ o ponto de truncamento da distribuição no cliente *i* convertido para normal padrão em relação ao início da janela de tempo. O fator de correção é calculado da seguinte forma: $Fcor = Preal - Pcal$, onde *Preal* é a probabilidade obtida via simulação e *Pcal* é a probabilidade calculada pelo método descrito na sessão 6.3. Ambos são dados em valores absolutos, portanto *Fcor*=0.01 significa uma correção de +1% no valor calculado. A figura 6.10 mostra um gráfico onde o fator de correção *Fcor* para o cálculo de p_1 é relacionado com *KL*.

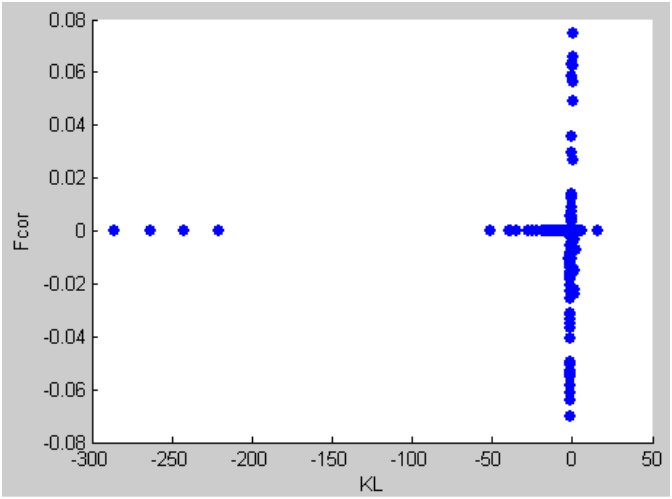


Figura 6.10: ponto de truncamento (*KL)* x fator de correção (*Fcor)*

Observa-se no gráfico da figura 6.10 que existiram correções variando aproximadamente de -7% a +8%. Observa-se que os valores de *Fcor* foram diferentes de zero apenas para valores na vizinhança de *KL*=0. Os resultados mostram que os erros são muito pequenos (inferior a 1%) para $KL \leq -2$. Este resultado faz sentido, pois quando $\mu_{ch}^i \leq (e_i - 2 * \sigma_{ch}^i)$ a probabilidade $(1 - p_1) = P(TCH_i > e_i)$ tende a ser muito baixa mesmo quando a distribuição não é normal.

No que diz respeito ao cálculo da probabilidade de violação p, há relação semelhante entre p e o ponto de truncamento em relação ao final da janela: $KL2_i = (l_i - \mu_{ch}^i) / \sigma_{ch}^i$. Os resultados estão no gráfico 6.11.

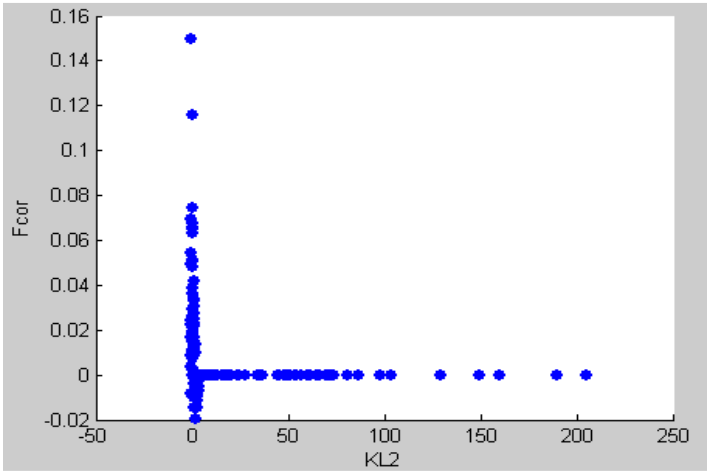


Figura 6.11: ponto de truncamento (*KL2)* x fator de correção (*Fcor)*

Nos testes observou-se que para $KL2 \geq 3$ não há necessidade de correção. Ou seja, quando a média do tempo de chegada do veículo ocorrer mais de 3 desvios-padrão após o final da janela, pode-se considerar que a probabilidade de violação é de 100%. Também é interessante observar que os erros para baixo variaram até 2% enquanto os erros para cima variaram até 15%.

Os casos anteriores tratam situações mais triviais, mas são úteis visto que o número de pontos que caem nesta situação é significativo.

Outro caso é mostrado na figura 6.12. Seja o quociente $q1_j = \sigma_{ij}/\sigma_{at}^j$. A figura mostra *q1* x *Fcor* quando *Fcor* aplicado para corrigir a probabilidade de violação p e também *q1* x *Fcor*, quando *Fcor* aplicado para corrigir a probabilidade de espera p_1.

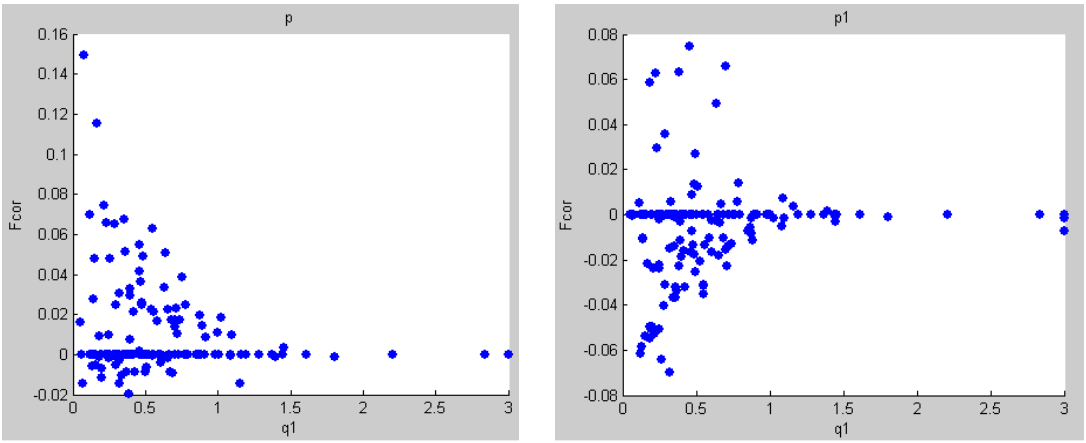


Figura 6.12: *q1* x *Fcor* para p e p_1

Os gráficos mostram que para $q1 > 1{,}25$ o fator de correção é zero ou muito próximo a zero, então se trata de uma situação em que não é necessário fazer correções, ou seja, *Fcor=0*. É interessante observar que nestes casos, a função distribuição de probabilidade do tempo de chegada é normal, ou muito próximo a isto. Cerca de 10% dos pontos caíram nesta região, o que é um número significativo de pontos em que sabemos não ser necessárias correções no cálculo.

6.4.2 – Parâmetros utilizados na regressão

A técnica utilizada para corrigir os resultados obtidos pela suposição de normalidade do tempo de chegada foi a Regressão Linear Multivariada. A técnica foi aplicada com o uso do Matlab, através do comando "*regress*". O comando "*corrcoef*" foi utilizado para calcular os coeficientes de regressão e também para obter o valor "*P-value*" que quando pequeno (digamos inferior a 0.05) significa que a correlação é significativa.

A variável de resposta é $Fcor = Real - Cal$ em algumas situações e $Fcor = Real/Cal$ para outras situações. Isto será especificado conforme o caso. *Real* é o valor obtido via simulação e *Cal* é o valor calculado. Para obter *Fcor* foi necessário estudar alguns parâmetros que mostrassem correlação com a variável de interesse.

Estes parâmetros foram:

$$q1_j = \sigma_{ij}/\sigma_{at}^j \tag{6.13}$$

$$q2_j = \sigma_{ij}/\sigma_{ch}^j \tag{6.14}$$

$$q3_j = \sigma_{ch}^j/\sigma_{ch}^i \tag{6.15}$$

$$KL_j = (e_j - \mu_{ch}^j)/\sigma_{ch}^j \tag{6.16}$$

$$KL2_j = (l_j - \mu_{ch}^j)/\sigma_{ch}^j \tag{6.17}$$

O quociente *q1* relaciona o desvio-padrão do arco com o desvio-padrão do tempo de início de atendimento. O quociente *q2* relaciona o desvio-padrão do arco com o desvio padrão do tempo de chegada. O quociente *q3* relaciona o desvio-padrão do tempo de chegada no cliente anterior com o desvio-padrão do tempo de chegada no cliente corrente. *KL* e *KL2* foram descritos na sessão 6.4.1 e estão relacionados à

distância do tempo de chegada em relação ao início e final da janela de tempo, respectivamente.

6.4.3 – Correções para a média truncada do tempo de chegada

Neste caso, temos $Fcor = Real/Cal$. O valor da média truncada foi calculado conforme explicado na sessão 6.3 com o uso das equações 6.5 a 6.9.

Fcor foi estimado a partir de $q1$ e KL:

$$Fcor = f(q1, KL) = c_0 + c_1 * q1 + c_2 * KL + c_3 * q1 * KL \quad (6.18)$$

Os valores encontrados para os coeficientes c_0 a c_3 respectivamente foram: 1.0194, -0.0155, 0.0311 e -0.0249.

A tabela 6.2 mostra os coeficientes de correlação entre *Fcor* e cada um dos parâmetros de entrada, juntamente com o respectivo nível de significância. Também é mostrado na figura 6.13 um gráfico com os resultados e regressão.

Parâmetros	**q1**	**KL**	**q1*KL**
Coef. Correlação	0.34	0.73	0.62
P-value	0.06	0.00	0.00

Tabela 6.2: correlação para média truncada

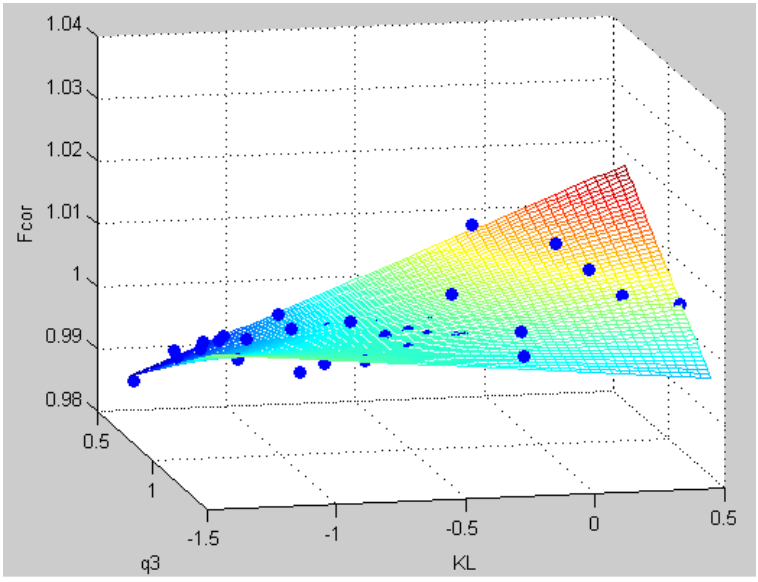


Figura 6.13: regressão para correção da média truncada

Comparando os resultados da aplicação da regressão com os resultados obtidos através da equação 6.18, testes com o banco de rotas mostraram que o erro médio caiu de 0.25 para 0.14 e que o erro máximo caiu de 1.59 para 0.86 em minutos.

6.4.4 – Correções para o desvio truncado do tempo de chegada

Neste caso, utilizou-se $\mathrm{Fcor} = \mathrm{Real}/\mathrm{Cal}$. O valor do desvio truncado foi calculado conforme explicado na sessão 6.3 com o uso das equações 6.5 a 6.9.

Fcor foi estimado a partir de *q1, q3* e *KL*:

$$Fcor = f(q1, q3, KL) = c_0 + c_1 * q1 + c_2 * q3 + c_3 * KL + c_4 * q1 * q3$$
$$+ c_5 * q1 * KL + c_6 * q3 * KL + c_7 * q1 * q3 * KL \quad (6.19)$$

Os valores encontrados para os coeficientes c_0 a c_7 respectivamente foram: 1.6828, 0.1330, -0.4105, 0.3865, -0.0944, 0.1212, -0.3789 e 0.1186.

A tabela 6.3 mostra os coeficientes de correlação entre *Fcor* e cada um dos parâmetros de entrada, juntamente com o respectivo nível de significância.

Parâmetros	**q1**	**q3**	**KL**	**q1*q3**	**q1*KL**	**q3*KL**	**q1*q3*KL**
Coef. Correlação	-0.38	-0.39	0.37	-0.4	0.51	0.52	0.56
P-value	0.02	0.06	0.02	0.01	0.00	0.00	0.00

Tabela 6.3: correlação para o desvio truncado

Comparando os resultados da aplicação da regressão com os resultados obtidos através da equação 6.19, testes com o banco de rotas mostraram que o erro médio caiu de 0.48 para 0.18 e que o erro máximo caiu de 1.73 para 1.20 em minutos.

6.4.5 – Regressão para cálculo do desvio do tempo de início de atendimento

Diferentemente das demais variáveis, o cálculo do desvio-padrão do tempo de início de atendimento (σ_{at}) não se utilizou de um fator de correção. O valor de σ_{at} foi estimado diretamente pelo uso de regressões conforme citado na sessão 6.3, equação 6.4.

σ_{at} foi estimado em função de σ_{tr} e p_1 da seguinte forma:

$$\sigma_{at} = f(\sigma_{tr}, p_1) = c_0 + c_1 * \sigma_{tr} + c_2 * p_1 + c_3 * \sigma_{tr} * p_1 \quad (6.20)$$

Os valores encontrados para os coeficientes c_0 a c_3 respectivamente foram: -0.1648, 1.1253, 0.0942 e -0.3637.

A tabela 6.4 mostra os coeficientes de correlação entre σ_{at} e cada um dos parâmetros de entrada, juntamente com o respectivo nível de significância. Também é mostrado na figura 6.14 um gráfico com os resultados e regressão.

Parâmetros	**q1**	**KL**	**q1*KL**
Coef. Correlação	0.96	-0.25	0.12
P-value	0.00	0.11	0.05

Tabela 6.4: correlação para desvio de atendimento

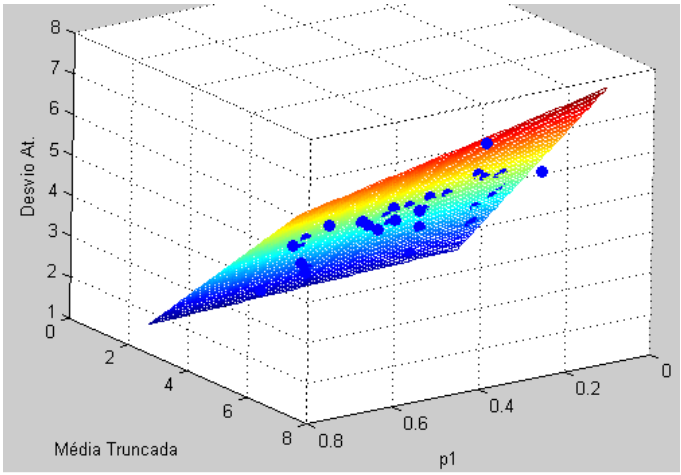


Figura 6.14: regressão para desvio de atendimento

Comparando os resultados da aplicação da regressão com os resultados obtidos via simulação, erro percentual médio foi de 4% e o erro máximo de 16%. Em valores absolutos, o erro médio foi de 0,1 e o erro máximo foi de 0,5. Nos testes, este erro não se mostrou um problema, uma vez que o cálculo da probabilidade mostrou-se bem mais sensível a erros no cálculo da média do que a erros no cálculo do desvio-padrão.

6.4.6 – Correções para a probabilidade de espera

Neste caso, utilizou-se $Fcor = Real - Cal$. O valor da probabilidade de espera (p_1) foi calculado conforme descrito na sessão 7.3.

Fcor foi estimado a partir de *q2, q3* e *KL*:

$$Fcor = f(q2, q3, KL) = c_0 + c_1 * q2 + c_2 * q3 + c_3 * KL + c_4 * q2 * q3 \quad (6.21)$$
$$+ c_5 * q2 * KL + c_6 * q3 * KL + c_7 * q2 * q3 * KL$$

Os valores encontrados para os coeficientes c_0 a c_7 respectivamente foram: 0.0169, -0.0141, -0.0246, 0.0241, 0.0294, -0.0752, 0.0079 e 0.0390.

A tabela 6.5 mostra os coeficientes de correlação entre *Fcor* e cada um dos parâmetros de entrada, juntamente com o respectivo nível de significância.

Parâmetros	**q2**	**q3**	**KL**	**q2*q3**	**q2*KL**	**q3*KL**	**q2*q3*KL**
Coef. Correlação	0.41	0.29	0.56	0.38	0.33	0.55	0.30
P-value	0.00	0.06	0.00	0.00	0.03	0.00	0.05

Tabela 6.5: correlação para o tempo de espera

A tabela 6.6 mostra os erros obtidos para diferentes métodos de cálculo. O método identificado como ProbJ consiste em calcular p_1 simplesmente assumindo normalidade, sendo o método adotado em Chang *et al.* (2009). Neste caso, $p_1 = P(TCH_j \leq e_j)$ onde TCH_j é uma variável aleatória que descreve o tempo de chegada no cliente *i*. A média é dada por: $E[TCH_j] = (\mu_{tat}^i + \mu_{ij} + s_i)$ e a variância por: $V[TCH_j] = (var_{at}^i + var_{ij})$.

O método ProbS (analogia a "separado") que tenta separar a componente normal da não normal, é o método descrito na sessão 6.3, da equação 6.10 a 6.12. O terceiro método é chamado de ProbR (analogia a "regressão") sendo o método que corrige o resultado de p_1 obtido pelo método ProbS através da regressão obtida nesta sessão.

Método	Erro médio	Erro máximo
ProbJ	3,05%	14,9%
ProbS	2,53%	11,5%
ProbR	1,99%	7,07%

Tabela 6.6: erros para cálculo da probabilidade de espera

Observa-de na tabela 6.6 que o método ProbS teve resultados melhores que ProbJ evidenciando que a idéia de separação é útil. O método ProbR obteve erros expressivamente menores tanto para a média quanto para o erro máximo em relação aos demais 2 métodos, mostrando que o uso de regressões de fato é capaz de reduzir o erro obtido pela suposição de normalidade.

6.4.7 – Correções para a probabilidade de violação da janela

Neste caso, utilizou-se $Fcor = Real - Cal$. O valor da probabilidade de violação (p) foi calculado conforme descrito na sessão 6.3 com o uso das equações 6.5 a 6.9.

Fcor foi estimado a partir de *q1, q3* e *KL2*:

$$Fcor = f(q1, q3, KL2) = c_0 + c_1 * q1 + c_2 * q3 + c_3 * KL2 + c_4 * q1 * q3 \quad (6.22)$$
$$+ c_5 * q1 * KL2 + c_6 * q3 * KL2 + c_7 * q1 * q3 * KL2$$

Os valores encontrados para os coeficientes c_0 a c_7 respectivamente foram: 0.2040, -0.2778, -0.1146, -0.1811, 0.1925, 0.2726, 0.1197 e -0.2034.

A tabela 6.7 mostra os coeficientes de correlação entre *Fcor* e cada um dos parâmetros de entrada, juntamente com o respectivo nível de significância.

Parâmetros	q1	q3	KL2	q1*q3	q1*KL2	q3*KL2	q1*q3*KL2
Coef. Correlação	-0.50	-0.23	-0.72	-0.47	-0.64	-0.74	-0.59
P-value	0.01	0.10	0.00	0.02	0.00	0.00	0.00

Tabela 6.7: correlação para a probabilidade de violação da janela de tempo

A tabela 6.8 mostra os erros obtidos para diferentes métodos de cálculo: ProbJ, ProbS e ProbR.

Método	Erro médio	Erro máximo
ProbJ	3,8%	15%
ProbS	3,2%	12,6%
ProbR	2,5%	7,4%

Tabela 6.8: erros para cálculo da probabilidade de violação

Observa-se que o método ProbS obteve um desempenho melhor que ProbJ e que o método ProbR foi superior ao ProbS.

Os parâmetros de entrada utilizados nas regressões descritas neste trabalho são aqueles cujas regressões mostraram melhores resultados. Naturalmente, muitos outros testes envolvendo outros parâmetros foram realizados. Até este momento, os estudos se concentraram no banco de 16 rotas descrito na sessão 6.4. No capítulo 8, experimentos serão realizados num banco de rotas maior com o objetivo de avaliar os métodos estudados.

6.5 - ALGORITMO PARA O PRVJT COM TEMPO DE VIAGEM ESTOCÁSTICO

Implementou-se uma heurística baseada na HJT (utilizada para resolver o PRV com janela de tempo) chamada aqui de HJTE (analogia à heurística com janela de tempo estocástica). Algumas alterações foram necessárias para tornar a heurística capaz de resolver o problema de roteamento tratando os tempos de viagem de modo estocástico e não mais determinístico.

A principal mudança diz respeito ao modo de checar a viabilidade de uma rota quanto ao respeito à janela de tempo. Agora, além da janela propriamente dita, há também um parâmetro de entrada que é o nível de serviço desejado para as rotas. O tempo de viagem entre os arcos não é mais uma constante e sim uma variável aleatória de distribuição normal com parâmetros μ_{ij} e σ_{ij} (média e desvio-padrão, respectivamente).

A principal diferença entre o HJT e HJTE é que no segundo, todos os procedimentos de verificação de rota determinísticos do primeiro foram substituídos pelo procedimento de verificação estocástico descrito a seguir no algoritmo 6.1. Trata-se de um pseudocódigo que tem como objetivo a clareza do método e não uma programação enxuta.

No pseudocódigo, *r* é um vetor com a rota a ser verificada; *num* é uma constante com o número de veículos da rota; *e* trata-se de um vetor com a abertura da janela de tempo dos clientes; *l* é um vetor com o fechamento da janela de tempo dos clientes; *s* é um vetor com o tempo de atendimento (serviço) em cada cliente; *dis* é uma matriz com a distância entre os clientes (neste caso, também o tempo, considerando-se velocidade unitária); *multip* é uma constante multiplicadora para determinar o desvio-padrão do arco, ou seja, $\sigma_{ij} = \mu_{ij} * multip$. Finalmente, *NS* é uma constante que significa o nível de serviço desejado. A função *VerificarViabilidadeRota* retorna um valor binário *NSok*, onde $NSok = 1$ significa que a rota é viável.

A função $CalProbN(\mu, \sigma, \delta)$ retorna a probabilidade: $P(V \leq \delta)$, onde *V* é uma variável aleatória supostamente normal, dada por $V \cong N(\mu, \sigma^2)$. *O* cálculo é realizado segundo Abramowitz & Stegun (1964). O termo *equacao* refere-se às equações obtidas pelas regressões lineares realizadas nas sessões 6.4.3 a 6.4.7. E o termo *parametros* refere-se aos parâmetros utilizados nas equações obtidas via regressão, descritos na sessão 6.4.2.

Algoritmo 6.1: *[NSok]* = **VerificarViabilidadeRota**(*r, num, e, l, s, dis, multip, NS*)

1: *NSok=1;*
2: *MedArco = dis(dep, r(1));*
3: *DevArco = multip* MedArco;*
4: *tch = MedArco;*
5: *dev = DevArco;*
6: *J = l(r(1)); J1 = e(r(1)); S = s(r(1));*
7: *prob = CalProbN(tch, dev, J);*
8: *[medtrunc, devtrunc]=CalTrunc(tch,dev,J1);*
9: *prob1 = CalProbN(tch, dev, J1);*
10: *tat =((prob1)*J1 +(1-prob1)*medtrunc;*
11: *tat1= medtrunc; tat2 = J1;*
12: *devat = RegressãoDevAt(equacao, parametros);*
13: *var =* [$devat^2$]*; var1 =* [$devtrunc^2$]*;*
14: **para** $(z = 1, 2, \dots, num - 1)$ **faça**
15: *MedArco = dis(r(z), r(z+1));*
16: *DevArco = multip* MedArco;*
17; *J = l(r(z)); J1 = e(r(z)); S = s(r(z));*
18: *tch = tat+MedArco+S;*
19: *tch1 = tat1 + MedArco + S;*
20: *tch2 = tat2 + MedArco + S;*
21: $dev = [var + DevArco^2]^{1/2}$;
22: $dev1 = [var1 + DevArco^2]^{1/2}$;
23: *parametros = CalcularParametros(dados);*
24: *normal = AssumirNormalidade(parametros);*
25: *proba = CalProbN(tch1, dev1, J);*
26: *probb = CalProbN(tch2, dev2, J);*
27: *probs=(prob1*probb)+((1-prob1)*proba);*
28: *Fcor = RegressaoProb(equacoes, parametros, normal);*
29: *prob = probs + Fcor;*
30: **se** *(prob > 1 - NS)*
31: *NSok = 0;*
32: *retornar NSok;*
33: **fim**;
34: *[medtrunc, devtrunc]=CalTrunc(tch, dev, J1);*
35: *Fcor = RegressaoMedTrunc(equacoes, parametros, normal);*
36: *medtrunc = medtrunc*Fcor;*
37: *Fcor = RegressaoDevTrunc(equacoes, parametros, normal);*
38: *devtrunc = devtrunc *Fcor;*
38: *proba = CalProbN(tch1, dev1, J1);*
39: *probb = CalProbN(tch2, dev2, J1);*
40: *prob1s=(prob1*probb)+((1-prob1)*proba);*
41: *Fcor = RegressaoProb1(equacoes, parametros, normal);*
42: *prob1 = prob1s + Fcor;*
43: *tat =((prob1)*J1 +(1-prob1)*medtrunc;*
44: *tat1= medtrunc; tat2 = J1;*
45: *devat = RegressãoDevAt(equacoes, parametros);*
46: *var =* [$devat^2$]*; var1 =* [$devtrunc^2$]*;*
47: **fim**;

Convém notar que transformar o problema determinístico em estocástico não é a tarefa mais difícil. A maior dificuldade é fazer o modelo com tempo de viagem estocástico funcionar adequadamente, no sentido de calcular a probabilidade de violação da janela de cada cliente em cada rota com o menor erro possível. Portanto, todos os estudos realizados concentraram-se em criar um método de cálculo que fornecesse erros no cálculo da probabilidade menores que demais métodos descritos na literatura.

O HJTE implementado é um CCP (*Chance Constrained Programming Model*) e resolve o modelo matemático descrito na sessão 4.1.4, com duas diferenças: o tempo de atendimento num dado cliente é determinístico e não há nível de serviço especificado para o depósito. Ambas as diferenças não implicam em perda de generalidade do método proposto. A implementação do modelo CCP permite que um modelo SPR seja implementado com facilidade, apenas atribuindo um termo de penalidade na função objetivo associado à probabilidade de violação da janela de tempo mesmo quando o nível de serviço é atendido.

Reforça-se que o mais difícil é calcular a probabilidade de violação da janela com o menor erro possível devido às dificuldades conseqüentes do desconhecimento da distribuição de probabilidade do tempo de chegada do veículo nos clientes, de modo que o modelo implementado no trabalho atende integralmente este propósito.

6.6 – DISCUSSÃO FINAL

Neste capítulo, tratou-se do PRVJT com Tempo de Viagem Estocástico. Inicialmente mostramos que a suposição de normalidade para o tempo de chegada do veículo é frágil, pois em muitas situações a distribuição não é normal.

Também mostramos que o tempo de chegada num dado cliente tem origem na soma de dois componentes, sendo um normal e o outro não normal. Esta observação motivou o desenvolvimento de uma estratégia (identificada como ProbS) que calcula a probabilidade de violação (e também de espera) separando estas duas componentes. Isto faz sentido, pois a fonte do erro fica limitada apenas à componente não normal referente aos valores de chegada em que não há espera.

Regressões lineares multivariadas foram estudas para reduzir os erros das variáveis que precisam ser estimadas para o cálculo das probabilidades envolvidas.

Observou-se que para o cálculos das probabilidades de violação e de espera, os resultados mostraram uma potencial vantagem dos métodos ProbS e ProbR sobre o método ProbJ (que apenas assume normalidade). O desempenho deste método será melhor avaliado nos experimentos descritos no capítulo 8.

Finalmente foi descrito o funcionamento da heurística que resolve o problema de roteamento estocástico, especificamente o procedimento que realiza a verificação de violação das janelas de tempo de uma dada rota.

O fenômeno investigado diz respeito ao efeito das janelas de tempo no tempo de chegada dos veículos, caracterizado por uma distribuição de probabilidade cuja forma varia ao longo dos clientes de uma dada rota. A investigação conduzida neste capítulo permitiu desenvolver uma metodologia inédita na literatura com o propósito de estimar os parâmetros da média e do desvio-padrão do tempo de chegada e também para calcular a probabilidade de violação.

CAPÍTULO 7

RESULTADOS DOS MODELOS DETERMINÍSTICOS

Neste capítulo são realizados experimentos para validar os modelos determinísticos desenvolvidos: capacitado, capacitado com distância máxima e capacitado com janela de tempo. Os experimentos foram realizados em um laptop com 1GB RAM, processador Dual Core de 1.8 GHz, sistema operacional Windows XP. Os algoritmos foram implementados em Matlab, versão 7 (R14).

7.1 – RESULTADOS PARA O PRV CAPACITADO

As heurísticas H1, H2, H3 e H4 são aplicáveis a problemas de roteamento de veículos capacitados, numa abordagem estática e determinística. Elas foram testadas em 7 instâncias de Christofides *et al.* (1979).

Relembrando, H1 utiliza as estratégias *ILS* e *VND*. H2 é igual a H1, porém com a adição da Lista Tabu. Já H3 é igual a H2, mas com a adição do "Operador Adaptativo". Finalmente, H4 é igual a H3 com a adição do *GLS* na fase de perturbação.

Os resultados são mostrados nas tabelas a seguir, onde *N* é o número de clientes, *NV* é o número de veículos e *D* é a distância total percorrida pelos veículos.

Em todos os experimentos e algoritmos, o critério de parada utilizado foi de 7 iterações consecutivas sem melhoria no melhor resultado encontrado.

7.1.1 – Resultados para a Heurística H1

A heurística H1 é a versão mais simples das heurísticas desenvolvidas neste trabalho. As buscas "Realocação" e "Intercâmbio" trabalharam com *NR*=4. A busca "Inter Rotas Probabilístico" utilizou 10**N* como número máximo de movimentos a serem testados em cada execução, onde *N* é o número de clientes do problema. Os melhores resultados em 10 execuções são descritos na tabela 7.1.

O *Gap Médio (%)* trata-se da média dos *gaps* de cada instância. O *gap* de cada instância é calculado da seguinte forma: $gap\%_i = (s_i - s_i^*)/s_i^*$. Onde $gap\%_i$ é o

gap% obtido para a instância *i*; s_i é a melhor avaliação da função objetivo conseguida pela heurística; e s_i^* é a melhor solução conhecida para a instância *i*. Este é o modo como Mester (2007) relata os resultados de sua pesquisa, sendo o mesmo utilizado neste trabalho.

		Melhor Conhecido		**Resultado H1**		
Problema	N	NV	D	NV	D	Gap (%)
vrpnc1	50	5	524.6	5	524.6	0.00
vrpnc2	75	10	835.26	10	844.08	1.04
vrpnc3	100	8	826.14	8	827.39	0.15
vrpnc4	150	12	1028.42	12	1046.8	1.76
vrpnc5	199	17	1298.8	17	1335.1	2.72
vrpnc11	120	7	1042.11	7	1052.4	0.98
vrpnc12	100	10	819.56	10	819.96	0.05
					Gap Médio (%)	0.96

Tabela 7.1: Resultados para Christofides *et al.* (1979)

As instâncias 1,3, e 12 apresentaram bons resultados, sendo que instância 1 teve o melhor resultado da literatura igualado (que neste caso é de fato o valor ótimo). As instâncias 4 e 5 apresentaram resultados piores. Estas instâncias ficam presas em mínimos locais muito cedo e mesmo alterando-se o critério de parada de forma a realizar-se mais tentativas, melhores resultados não são conseguidos. Espera-se que a implementação da busca tabu na heurística H2 ajude o algoritmo a lidar melhor com estes ótimos locais.

7.1.2 – Resultados para heurística H2

A heurística H2 trata-se de uma versão com a implementação da Busca Tabu. A busca denominada "Busca Tabu" utilizou *NR*=4. A Lista Tabu teve tamanho 5, ou seja, a inversão de qualquer dos últimos 5 movimentos que levaram ao melhor resultado corrente do algoritmo era proibida. Estes valores foram os mesmos para H3 e H4. Os melhores resultados em 10 execuções são mostrados na tabela 7.2, juntamente com os resultados de H1 para facilitar a análise.

		Melhor Conhecido		Resultado H1			Resultado H2		
Problema	N	NV	D	NV	D	Gap (%)	NV	D	Gap(%)
1	50	5	524.6	5	524.6	0.00	5	524.6	0.00
2	75	10	835.26	10	844.08	1.04	10	843.6	0.99
3	100	8	826.14	8	827.39	0.15	8	827.39	0.15
4	150	12	1028.42	12	1046.8	1.76	12	1045.4	1.62
5	199	17	1298.8	17	1335.1	2.72	17	1330.9	2.41
11	120	7	1042.11	7	1052.4	0.98	7	1049.8	0.73
12	100	10	819.56	10	819.96	0.05	10	819.60	0.00
				Gap Médio (%)		0.96	*Gap* Médio (%)		0.84

Tabela 7.2: Resultados para Christofides *et al*. (1979)

Exceto as instâncias 1 e 3 em que H1 e H2 tiveram os mesmos resultados, em todas as demais instâncias, H2 melhorou. Para a instância 12, H2 igualou o melhor resultado da literatura. Isto indica que a Lista Tabu ajudou o algoritmo a lidar melhor com os ótimos locais.

7.1.3 – Resultados para a heurística H3

A heurística H3 é uma versão que usa a Memória Adaptativa além da Busca Tabu. O procedimento "Operador Adaptativo" utilizou n=0.1; B= N/(2*nv), onde nv é o número de veículos da primeira solução obtida pelo método de *Gillet* e *Miller*, e N é o número de clientes do problema. O valor de μ foi calculado como 0.5* (S/A), onde S é o valor da função objetivo para a melhor solução corrente e A é o valor do arco de maior comprimento da melhor solução corrente. O número de filhos foi M=3. Os melhores resultados em 10 execuções são mostrados na tabela 7.3.

		Melhor Conhecido		Resultado H3		
Problema	N	NV	D	NV	D	Gap (%)
vrpnc1	50	5	524.6	5	524.6	0.00
vrpnc2	75	10	835.26	10	837.58	0.28
vrpnc3	100	8	826.14	8	827.39	0.15
vrpnc4	150	12	1028.42	12	1035.4	0.67
vrpnc5	199	17	1298.8	17	1327.2	2.14
vrpnc11	120	7	1042.11	7	1048.1	0.57
vrpnc12	100	10	819.56	10	819.56	0.00
				Gap Médio (%)		0.54

Tabela 7.3: Resultados para Christofides *et al*. (1979)

Os resultados de H3 mostram uma melhora relevante do algoritmo em relação às heurísticas H1 e H2, mostrando que a inclusão da Memória Adaptativa no algoritmo de fato trouxe benefícios. As instâncias 2,4,5e 11 tiveram melhores resultados que H2. O *gap* médio de H3 em relação a H2 foi reduzido em 35%, e de H3 em relação a H1, teve uma redução de 44%.

7.1.4 – Resultados para a heurística H4

A heurística H4 é uma versão que usa, além da Memória Adaptativa e da Busca Tabu, um critério de seleção baseado no *GLS* dentro do mecanismo de perturbação. O valor do parâmetro μ utilizado no cálculo da função pseudo-objetivo foi calculado da mesma forma que em H3. Os melhores resultados em 10 execuções são mostrados na tabela 7.4.

		Melhor Conhecido		Resultado H4		
Problema	N	NV	D	NV	D	Gap (%)
vrpnc1	50	5	524.61	5	524.61	0.00
vrpnc2	75	10	835.26	10	835.26	0.00
vrpnc3	100	8	826.14	8	827.39	0.15
vrpnc4	150	12	1028.42	12	1033.4	0.48
vrpnc5	199	17	1298.80	17	1325.20	1.99
vrpnc11	120	7	1042.11	7	1048.10	0.57
vrpnc12	100	10	819.56	10	819.56	0.00
				Gap Médio (%)		0.46

Tabela 7.4: Resultados para Christofides *et al*. (1979)

O algoritmo H4 melhorou os resultados obtidos por H3 nas instâncias 2,4 e 5. No caso da instância 2, H4 conseguiu igualar mais um resultado ao melhor encontrado na literatura. O fato de H4 ter encontrado melhores resultados para três instâncias mostra que o critério de seleção baseado no *GLS* foi útil.

7.1.5 – Comparação das 4 heurísticas

Na tabela 7.5 é descrito a média e o desvio-padrão tanto da avaliação da função objetivo como também para o tempo de execução de cada uma das 4 heurísticas para as instâncias testadas. Foram executadas 10 rodadas para cada uma das instâncias numa seqüência aleatoriamente permutada. A instância de 50 clientes não foi descrita aqui, pois todas as heurísticas encontraram o valor ótimo desta instância, de modo que ela não contribui para o objetivo desta sessão que é comparar o desempenho das 4 heurísticas.

Avaliação da Função Objetivo									
		H1		**H2**		**H3**		**H4**	
Probl.	N	Média	Desvio	Média	Desvio	Média	Desvio	Média	Desvio
2	75	847,02	1,94	845,35	2,28	843,25	4,27	842,87	4,49
3	100	834,00	4,79	832,81	4,63	831,37	3,31	828,90	1,67
4	150	1049,77	3,23	1049,80	3,75	1041,23	3,39	1039,30	4,91
5	199	1343,96	5,22	1341,48	5,56	1337,87	5,85	1335,94	5,83
11	120	1058,30	5,06	1054,68	5,57	1052,22	4,59	1051,92	5,07
12	100	823,78	3,25	823,57	3,71	822,00	3,14	822,03	3,12

Tabela 7.5: Resultados para Christofides *et al*. (1979)

Os dados utilizados para obter a tabela anterior foram utilizados para a aplicação de testes estatísticos não paramétricos que pudessem comparar a média e a variância dos resultados obtidos e permitir afirmar se alguma heurística realmente tem melhor avaliação da função objetivo que as demais.

Testes 2 a 2 foram realizados entre as 4 heurísticas, formando 6 cenários de teste. Para cada cenário, foi aplicado inicialmente o Teste de Levene para saber se havia diferença significativa entre as variâncias. Em caso de diferença significativa, aplicou-se a seguir o Teste T para 2 amostras assumindo variâncias diferentes com o objetivo de comparar as médias. Em caso de diferença não significativa, aplicou-se o Teste de Mann-Whitney. A tabela 7.6 mostra o nível de significância obtido em cada um dos cenários para cada uma das instâncias para a variância σ^2 e para a média μ. O nível de confiança desejado foi de 90%, de forma que um nível de significância menor ou igual a 0.10 indica que há diferenças entre os 2 grupos testados. Os testes foram realizados com o uso do Minitab 15.

Testes Estatísticos – Nível de significância												
	Prob. 2		**Prob. 3**		**Prob. 4**		**Prob. 5**		**Prob. 11**		**Prob. 12**	
Cenários	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ
H1 x H2	0.85	0.16	0.77	0.47	0.40	0.85	0.83	0.25	0.90	0.09	0.55	0.70
H1 x H3	0.02	0.03	0.33	0.22	0.89	0.00	0.84	0.03	0.45	0.01	0.56	0.10
H1 x H4	0.05	0.02	0.06	0.01	0.17	0.00	0.81	0.01	0.46	0.01	0.53	0.10
H2 x H3	0.03	0.19	0.05	0.43	0.34	0.00	0.71	0.12	0.66	0.22	0.34	0.27
H2 x H4	0.05	0.03	0.00	0.03	0.44	0.00	0.98	0.04	0.61	0.18	0.32	0.34
H3 x H4	0.95	0.82	0.09	0.06	0.15	0.24	0.71	0.40	0.88	1.00	0.97	0.90

Tabela 7.6: Nível de Significância dos Testes de Hipótese

Para tomada de decisão utilizou-se o seguinte critério: quando foi possível afirmar que a heurística H_i apresentou menor média que a heurística H_j para uma dada instância, então 1 ponto foi computado para H_i. Por exemplo, no cenário H1 x H2 para a instância 11, foi possível afirmar que H2 apresentou menor média que H1, implicando em 1 ponto para H2. Isto gerou a tabela 7.7 a seguir:

Heurística	Pontos
H1	0
H2	1
H3	6
H4	10

Tabela 7.7: Ranqueamento das Heurísticas quanto à função objetivo

A tabela 7.7 permite afirmar que, dentro do escopo dos experimentos realizados, a heurística H4 tem o melhor desempenho quanto à função objetivo.

A tabela 7.8 possui os dados referentes ao custo computacional das heurísticas medido em função do número de operações realizadas. A operação escolhida foi o número de verificações da viabilidade de uma rota quanto ao respeito da capacidade do veículo. Esta operação foi escolhida, pois sempre que um movimento inter-rotas é realizado é necessário que esta verificação seja feita, de modo que isto tem relação direta com o tempo de execução do algoritmo.

Avaliação do Custo Computacional									
		H1		H2		H3		H4	
Probl.	N	Média	Desvio	Média	Desvio	Média	Desvio	Média	Desvio
2	75	44041	16168	37786	7827	66265	22202	74966	16004
3	100	98128	51677	116323	63724	148247	54862	159828	70799
4	150	220226	53162	224131	45123	257977	72459	250202	58354
5	199	689179	225027	595345	189654	749235	308404	605790	160838
11	120	103604	10212	99954	7596	105570	3798	105289	4212
12	100	50258	4240	55593	9714	65841	6251	66121	6118
Média		200906		188189		232189		211866	

Tabela 7.8: Número de operações realizadas

Pela tabela 7.8, mesmo que a heurística H4 apresente um custo computacional diferente das demais, não considerou-se esta possível diferença significativa o suficiente para penalizar a heurística, de modo que consideramos a heurística H4 a melhor dentre as 4 heurísticas desenvolvidas. Isto evidencia que o uso dos conceitos de Busca Tabu, Memória Adaptativa e *Guided Local Search* foram inseridos com sucesso no contexto da heurística *ILS* com *VND*.

Segundo Hoos & Stutzle (2004), pode ser interessante visualizar as diferenças entre os algoritmos graficamente. Como exemplo, a figura 7.1 mostra a convergência dos algoritmos para a instância 4 com 150 clientes. As curvas foram geradas através de uma regressão logarítmica obtida a partir da média dos dados de 10 execuções de cada algoritmo. O valor da estatística R^2 é dado nos gráficos para indicar a qualidade da regressão. A figura 7.2 mostra o gráfico para os mesmos dados com o eixo horizontal em escala logarítmica.

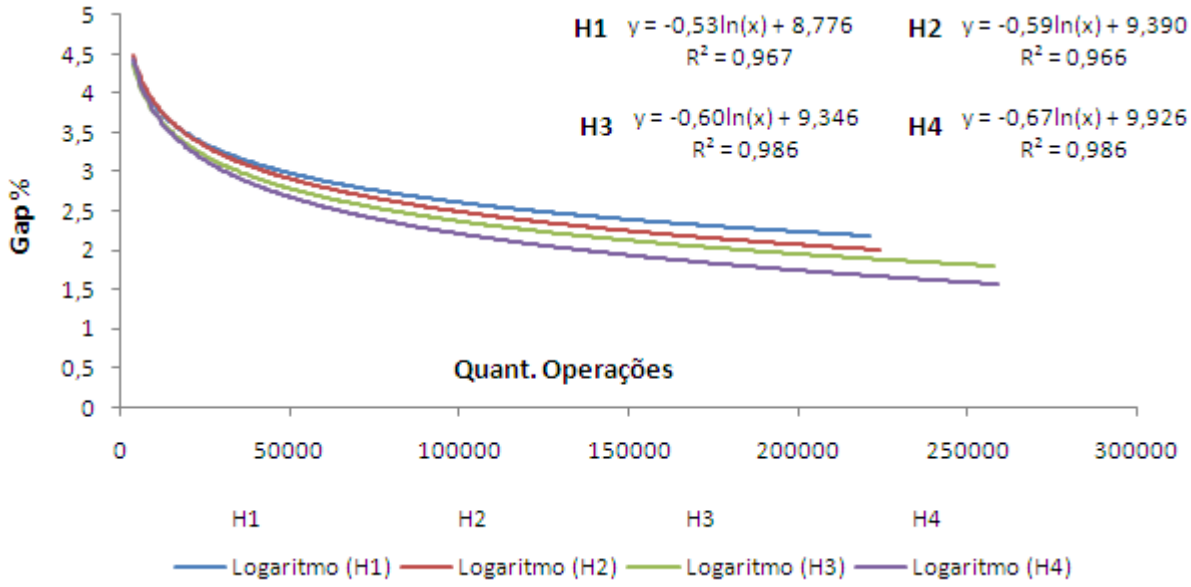


Figura 7.1: Gap x Quantidade de Operações (instância 4: 150 clientes)

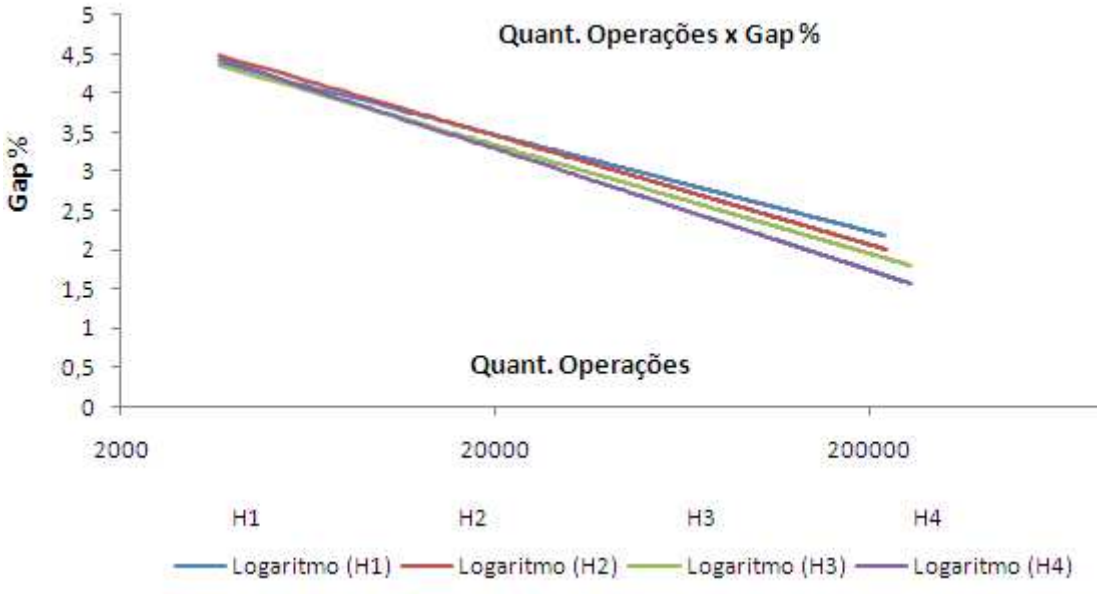


Figura 7.2: *Gap* x Operações (problema 4). Escala Logarítmica para eixo x.

Também pode ser interessante conhecer a função densidade de probabilidade de um algoritmo. Assim, para os algoritmos H1 e H4, e para a instância 4, foram realizadas 100 execuções limitadas a 4 minutos. Desta forma foi possível fazer o histograma e conhecer a distribuição de probabilidade destas heurísticas. Para ambas, houve adesão no teste de normalidade. Resultados nas figuras 7.3 e 7.4.

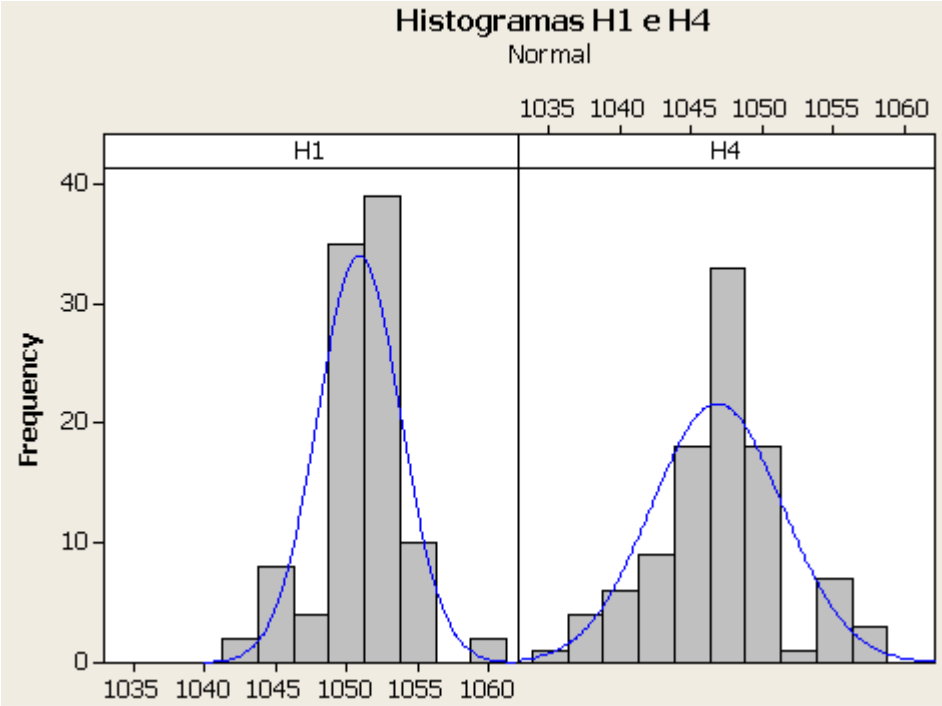


Figura 7.3: histograma para H1 e H4 separados

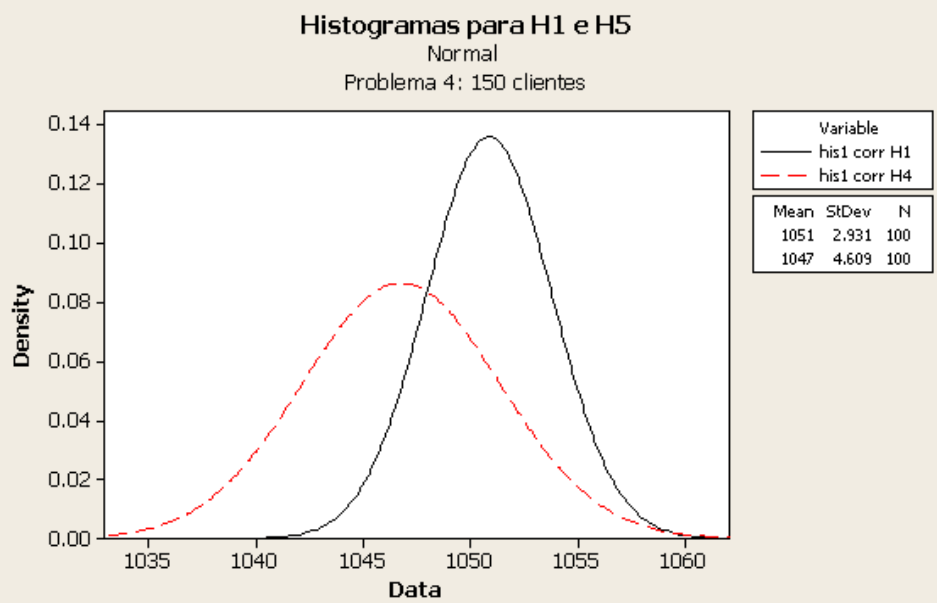


Figura 7.4: histograma para H1 e H4 juntos

Nos testes realizados com base em dados obtidos em 10 execuções não houve adesão à distribuição normal, mas para 100 execuções houve esta adesão.

De qualquer forma, os resultados das figuras 7.3 e 7.4 continuam a confirmar a superioridade da heurística H4 sobre as demais.

7.2 – RESULTADOS PARA O PRV CAPACITADO COM DISTÂNCIA MÁXIMA

A heurística H4 foi alterada para resolver também os problemas com distância máxima das rotas, ou seja, consideram a existência de uma janela de tempo apenas para os veículos. Esta heurística foi escolhida por ter encontrado os melhores resultados dentre as 4 heurísticas: H1, H2, H3 e H4. A heurística foi validada com 7 instâncias de Christofides *et al*. (1979). Estas 7 instâncias juntamente com as 7 utilizadas no tópico anterior completam todas as 14 instâncias de Christofides. Os melhores resultados dentre 10 execuções para cada uma das instâncias são apresentados na tabela 7.9. Em todos os experimentos, o critério de parada utilizado foi de 7 iterações consecutivas sem melhoria no melhor resultado encontrado.

		Melhor Conhecido		**Resultado H4**		
Problema	N	NV	D	NV	D	Gap (%)
vrpnc6	50	6	555.43	6	555.43	0.00
vrpnc7	75	11	909.68	11	912.9	0.35
vrpnc8	100	9	865.94	9	865.94	0.00
vrpnc9	150	14	1162.55	14	1182.4	1.68
vrpnc10	199	18	1395.85	18	1424.4	2.00
vrpnc13	120	11	1541.14	11	1553.2	0.80
vrpnc14	100	11	866.37	11	866.37	0.00
					Gap Médio (%)	0.70 %

Tabela 7.9: Resultados para Christofides *et al*. (1979).

Observa-se que o algoritmo foi capaz de igualar o melhor resultado conhecido em 3 instâncias, atestando sua eficiência.

Foi implementado ainda um algoritmo que será utilizado num ambiente real de uma empresa de distribuição que realiza em média duas mil entregas por dia (detalhes em Pessoa & Miranda, 2011). Logo, tratando-se de instâncias classificadas como *Very Large Scale*, o custo computacional passa a ser importante para a viabilidade prática do algoritmo. Para mais informações sobre problemas de roteamento de larga escala, fazemos referencia a Kytöjoki *et al*., 2007.

Desta forma, o algoritmo implementado trata-se de uma versão simplificada da heurística H1 adaptada para resolver o PRV com distância máxima. Diz-se tratar-se de uma versão simplificada pois a fase de perturbação que originalmente era composta de três operadores genéticos, nesta versão é composta por apenas 1 operador. Além disto, diferentemente de todos os demais algoritmos deste trabalho que foram implementados em Matlab, este foi implementado em Delphi.

A tabela 7.10 mostra os resultados desta heurística. O tempo computacional é dado em relação ao tempo de execução da mesma versão em Matlab, ou seja: [(tempo em Delphi)/ (tempo em Matlab)]*100. O *gap* relatado é o melhor de 10 execuções e o tempo foi calculado a partir da média de 10 execuções.

		Melhor Conhecido		**Heurística Delphi**			
Problema	N	NV	D	NV	D	Gap (%)	Tempo (%)
vrpnc6	50	6	555.43	6	560.88	0.97	5.0
vrpnc7	75	11	909.68	11	920.5	1.18	3.3
vrpnc8	100	9	865.94	9	871.98	0.69	10.4
vrpnc9	150	14	1162.55	14	1185.5	1.94	7.71
vrpnc10	199	18	1395.85	18	1431.7	2.50	9.24
vrpnc13	120	11	1541.14	11	1554.8	0.88	14.86
vrpnc14	100	11	866.37	11	869.06	0.31	4.38
				Gap Médio (%)		1.21 %	

Tabela 7.10: Resultados para Christofides *et a*l. (1979)

A tabela 7.10 mostra que o algoritmo implementado em Delphi conseguiu reduzir expressivamente o tempo computacional ainda com uma boa qualidade da solução.

7.3 – *BENCHMARKS* PARA VRP CAPACITADO

A tabela 7.11 mostra os resultados obtidos pela heurística H4 (implementada em Matlab) e pela heurística H1 (implementada em Delphi). A qualidade das soluções e o tempo computacional são comparados com outros trabalhos reconhecidos na literatura. Os resultados referem-se a todas as 14 instâncias de Christofides *et al*. (1979), sendo as 7 primeiras puramente capacitadas e as demais com restrição de distância

máxima para as rotas. Cada coluna fornece a melhor solução encontrada por cada uma das heurísticas. As últimas 4 linhas da tabela relatam respectivamente: o gap médio percentual em relação às melhores soluções conhecidas, o tipo e a velocidade em MHz do computador utilizado, e o tempo médio de execução por problema em minutos. As informações foram obtidas em Mester (2007). O *gap* de H1 e H4 refere-se à melhor solução encontrada em 10 execuções. O tempo de H1 e H4 é dado pela média destas mesmas 10 execuções.

A linha com o *gap* % trata-se da média dos gaps de cada instância. O gap de cada instância é calculado da seguinte forma: $gap\%_i^h = \left(s_i^h - s_i^*\right)/s_i^*$. Onde $gap\%_i^h$ é o *gap%* da instância *i* obtida pela heurística *h*; s_i^h é a melhor avaliação da função objetivo conseguida pela heurística *h* na instância *i*; e s_i^* é a melhor solução conhecida para a instância *i*.

Problema	**H4 (Matlab)**	**H1 (Delphi)**	**Mester (2007)**	**Tarantilis (2005)**	**Reimann et. al. (2004)**	**Toth & Vigo (2003)**
1 (50)	524.61	524.61	524.61	524.61	524.61	524.61
2 (75)	835.26	845.71	835.26	835.26	840.61	838.60
3 (100)	827.39	828.51	826.14	826.14	828.21	828.56
4 (150)	1033.40	1048.60	1028.42	1028.42	1037.57	1033.21
5 (199)	1325.20	1339.35	1291.29	1311.48	1306.91	1318.25
6 (50)	555.43	560.88	555.43	555.43	553.43	555.43
7 (75)	912.90	920.50	909.68	909.68	917.50	920.72
8 (100)	865.94	871.98	865.94	865.94	865.94	869.48
9 (150)	1182.40	1185.50	1162.55	1162.55	1173.94	1173.12
10 (199)	1424.40	1431.70	1401.12	1407.21	1415.53	1435.74
11 (120)	1048.10	1055.60	1042.11	1042.11	1043.46	1042.87
12 (100)	819.56	819.96	819.56	819.56	819.56	819.56
13 (120)	1553.2	1554.8	1541.14	1544.01	1546.84	1545.51
14 (100)	866.37	869.06	866.37	866.37	866.37	866.37
Gap %	0.62	1.23	0.03	0.18	0.48	0.64
Tipo PC	AMD	AMD	Pentium IV	Pentium	Pentium	Pentium
MHz PC	1800	1800	2800	400	900	200
Minutos	6.1	0.91	2.8	6.6	3.8	3.8

Tabela 7.11: *Gaps* para *benchmarks* de Christofides *et. al* (1979)

O foco do trabalho está na qualidade das soluções obtidas, indicada pelos *gaps* obtidos, desejando-se aproximar os resultados obtidos dos melhores alcançados na literatura. Embora este seja o principal objetivo, e embora estejamos usando uma linguagem de programação muito mais lenta que linguagens como C, C++, Java, Delphi e outras, temos consciência de que não podemos ter tempos de execução muito mais altos do que os tempos reportados na literatura, ou mesmo, tempos proibitivos.

Sabemos ser difícil comparar tempos de execução de algoritmos implementados em linguagens diferentes, computadores diferentes e em anos diferentes, mas pela tabela 7.11, pode ser visto que os tempos computacionais obtidos são razoáveis, dentro do que esperamos para este trabalho. Observa-se que a heurística H1 implementada teve um tempo de execução 85% menor que a heurística H4. Embora H1 seja mais simples que H4, atribui-se a maior parte desta diferença à simples mudança na linguagem de programação, já que os testes foram executados na mesma máquina.

Quanto à qualidade das soluções, observa-se que H4 teve um *gap%* médio ligeiramente melhor que Toth & Vigo (2003), embora tenha sido superado pelas demais heurísticas (exceto H1). Consideramos estes *gaps* pequenos e que os resultados foram satisfatórios.

7.4 – RESULTADOS PARA O PRV COM JANELA DE TEMPO

A heurística HJT descrita no item 5.6 foi validada em 12 instâncias de Solomon de 100 clientes. Escolheu-se 2 instâncias de cada uma das 6 classes de problemas (C1, C2, R1, R2, RC1, RC2) para ser possível testar a heurística em problemas com diferentes características. Um fator diferencial que motivou a escolha da segunda instância para cada uma das classes foi o fato destas instâncias não terem ótimo conhecido (Alvarenga, 2005). Isto é um indicador da dificuldade destes problemas. A tabela 7.12 resume as características principais das instâncias escolhidas.

Problema	Distribuição Geográfica	Largura da Janela	Horizonte de Tempo	Clientes c/ Janela (%)
C101	*Cluster*	60	1236	100
C104	*Cluster*	60	1236	25
C201	*Cluster*	160	3390	100
C204	*Cluster*	160	3390	25
R101	Uniforme	10	230	100
R104	Uniforme	10	230	25
R201	Uniforme	27 a 212	1000	100
R204	Uniforme	27 a 212	1000	25
RC101	Misto	30	240	100
RC104	Misto	30	240	25
RC201	Misto	120	960	100
RC204	Misto	120	960	25

Tabela 7.12: características das instâncias de teste

O critério de parada utilizado foi de 7 iterações consecutivas sem melhoria no melhor resultado encontrado. Os resultados são mostrados na tabela 7.13 e confrontados com os melhores resultados conhecidos segundo Repoussis *et al*. (2009). Os resultados também são comparados com a heurística HGC_NV de Alvarenga (2005).

	Melhor		Resultado HJT				Resultado HGC_NV		
Problema	NV	D	NV	D	*Gap*	Minutos	NV	D	*Gap*
C101	10	824.94	10	828.93	0.48	4.2	10	828.94	0.48
C104	10	824.78	10	838.45	1.63	9.6	10	824.78	0.00
C201	3	591.56	3	591.56	0.00	1.5	3	591.56	0.00
C204	3	590.6	3	590,6	0.00	1.4	3	590.6	0.00
R101	19	1645.79	19	1652.8	0.42	33.6	19	1650.8	0.30
R104	9	1007.24	9	1015.6	0.82	30.9	9	1012.59	0.53
R201	4	1252.34	4	1275.2	1.79	43.4	4	1263.8	0.91
R204	2	825.52	2	871.76	5.3	31.8	2	887.56	6.99
RC101	14	1696.94	14	1701.82	0.29	10.8	14	1698.82	0.11
RC104	10	1135.48	10	1158.3	1.97	12.5	10	1163.31	2.39
RC201	4	1406.91	4	1469.8	4.28	23.2	4	1462.21	3.78
RC204	3	798.41	3	818.23	2.42	19.5	3	815.62	2.11
Total	91	12601	91	12813	-	222	91	12791	-
Gap %	-	-	0	1.66	-	-	0	1.49	-

Tabela 7.13: Resultados para Solomon (1987).

Os resultados da tabela 7.13 podem ser considerados satisfatórios, pois mostram que a heurística desenvolvida aproximou-se dos melhores resultados conhecidos. Os melhores resultados foram obtidos para os problemas C201 e C204 em que a solução ótima foi encontrada. Os piores resultados foram obtidos para as instâncias R204 e RC201 com *gaps* acima de 4%. A heurística HGC_NV também teve os piores resultados exatamente para estas 2 instâncias, indicando que estes problemas são particularmente difíceis. Quanto ao número de veículos, foi possível encontrar o valor mínimo em todas as instâncias.

Comparando o HJT com o HGC_NV, observa-se que o HJT igualou 3 instâncias (C101, C201 e C204), superou 2 instâncias (R204 e RC104) e foi superado nas demais. Os resultados podem ser considerados competitivos uma vez que o *gap* total foi bastante próximo, sendo 1.66% no HJT e 1.49% no HGC_NV.

O tempo computacional foi dado apenas como referência. No HGC_NV, em todos os resultados obtidos, o critério de parada foi o mesmo, sendo limitado por um tempo de 75 minutos. O tempo total no HJT foi de 222 minutos e no HGC_NV de 900 minutos. Sabemos da dificuldade em comparar tempos de execução de algoritmos

implementados em linguagens diferentes, computadores diferentes e em anos diferentes, mas pode ser visto que os tempos computacionais obtidos são razoáveis, dentro do que esperamos para este trabalho.

A discussão final dos resultados obtidos neste capítulo será vista no capítulo 9.

CAPÍTULO 8

RESULTADOS PARA O VRP COM TEMPO DE VIAGEM ESTOCÁSTICO

Este capítulo tem o propósito de realizar experimentos mais abrangentes que possam testar a metodologia desenvolvida no capítulo 6. A partir de um banco de rotas maior que o utilizado no capítulo 6, experimentos são conduzidos para testar as estimativas da média e do desvio padrão do tempo de chegada, além da probabilidade de violação. Comparações com outros métodos da literatura são realizadas. O tamanho do erro de cada método é dado por sua diferença em relação aos resultados obtidos por simulação estocástica. Um pseudo-agoritmo de como realizar esta simulação é dado em Li *et al.* (2010). No final do capítulo, experimentos são conduzidos para resolver algumas instâncias do problema de roteamento com tempo de viagem estocástico a partir de cenários com diferentes níveis de serviço, diferentes variâncias para o tempo de viagem entre os clientes e diferentes larguras para a janela de tempo.

8.1 – BANCO DE ROTAS TESTE

Para a avaliação dos métodos implementados neste trabalho, gerou-se um banco composto por 180 rotas. Estas rotas foram geradas a partir da execução da heurística I1 de Solomon (1987) utilizando-se diferentes parâmetros de entrada. A heurística foi aplicada a 9 instâncias de Solomon, sendo originadas 20 diferentes rotas a partir de cada uma das 9 instâncias.

A tabela 8.1 dá informações sobre o número de clientes das rotas geradas para cada uma das instâncias.

	Quantidade de Clientes nas Rotas			
Instância	Média	Desvio	Menor	Maior
C101	7.6	3.3	3	13
C104	8.8	3.2	2	12
C201	17.9	14.7	3	35
R101	4.6	1.8	2	8
R104	7.9	3.3	2	13
R201	16.2	11.6	3	38
RC101	5.1	2.2	2	11
RC104	7.7	3	2	13
RC201	14.9	10.8	3	34

Tabela 8.1: instâncias de origem do banco de rotas

Observa-se que foram geradas rotas curtas (2 clientes) e também rotas longas com muitos clientes (até 38). Isto para instâncias com diferentes características como já descrito no capítulo 3, sobre os problemas determinísticos com janela de tempo.

8.2 – ESTIMATIVA DE MÉDIA E DESVIO-PADRÃO

O objetivo aqui é avaliar a qualidade da estimativa da média e do desvio-padrão do tempo de chegada do veículo ao longo dos clientes de uma dada rota. A avaliação é feita comparando três métodos. Dois são os descritos nas sessões 4.1.5 e 4.1.6, de Jula *et al*. (2006) e Chang *et al*. (2009) respectivamente. O outro método é o proposto por este trabalho, com o uso de regressões lineares, explicado na sessão 6.4. O desempenho destes 3 métodos é comparado com o resultado obtido via simulação estocástica. Utilizou-se o banco de 180 rotas descrito na seção 8.1. Conforme explicado anteriormente, uma boa estimativa da média e do desvio-padrão faz-se fundamental para o cálculo do nível de serviço das rotas.

Os resultados são descritos na tabela 8.2. Nela, é dado o erro percentual médio e o desvio percentual médio tanto da média como do desvio-padrão quando comparados aos resultados obtidos pela simulação das rotas. Também fornece o pior caso, ou seja, o maior erro percentual encontrado dentre todos os clientes de todas as rotas testadas. O nível de serviço desejado foi de 95% e considerou-se o desvio-padrão dos arcos como 20% do seu comprimento. A simulação foi executada com 1 corrida de 10000 réplicas.

	Erros da Média (%)			Erros do Desvio-Padrão (%)		
Método	Média	Desvio	Pior Caso	Média	Desvio	Pior Caso
1)Regressão	0.028	0.023	0.374	1.10	1.00	15.3
2)Chang	0.030	0.026	0.404	1.56	1.18	23.94
3)Jula	0.443	0.417	5.156	11.30	8.76	45.01

Tabela 8.2: qualidade das estimativas de média e desvio.

Na tabela 8.2, o método 1 utiliza a mesma forma de cálculo do método 2, porém no método 1, o resultado alcançado é corrigido por um fator de correção obtido por regressão linear, conforme descrito no item 6.4. Nota-se que embora os resultados de 1 e de 2 sejam semelhantes, pode-se dizer que o método 1 (proposto por este trabalho) foi superior, obtendo-se vantagem expressiva na melhoria do pior resultado obtido. Isto evidencia que o uso da técnica de regressão linear multivariada foi útil.

Claramente o método 3 (de Jula) obteve erros expressivamente maiores que os demais métodos. Isto pode ser explicado pela simplicidade do método que não trata adequadamente os efeitos do tempo de espera como uma variável aleatória que afeta a média e a variância do tempo de chegada nos clientes.

8.3 – ESTIMATIVA DO NÍVEL DE SERVIÇO DAS ROTAS

O objetivo aqui é avaliar a qualidade da estimativa da probabilidade do tempo de chegada do veículo em um cliente ocorrer antes do final da janela de tempo do respectivo cliente, ou seja, avaliar a qualidade do cálculo do nível de serviço (NS) para cada cliente de uma dada rota. A avaliação é feita comparando os 6 métodos descritos a seguir com os resultados obtidos via simulação estocástica. Utilizou-se o banco de 180 rotas descrito na seção 8.1.

O método identificado como ProbChang assume normalidade no cálculo da probabilidade de violação; e estima a média e o desvio do tempo de chegada conforme descrito em 4.1.6 (Chang *et al*. 2009). O método ProbJ também assume normalidade no cálculo da probabilidade de violação; porém a estimativa da média e do desvio do tempo de chegada é feito com o uso das regressões, conforme sessão 6.4. Os métodos identificados como Chernoff e Chebyshev utilizam-se de suas respectivas desigualdades conforme sessão 4.1.5 e usado em Jula *et al*. (2006). O método ProbS faz uso da estratégia de separação da componente normal e da não normal do tempo de chegada,

descrito em 6.3. Finalmente, o método ProbR faz uso das regressões para corrigir a estimativa feita com suposição de normalidade.

Os resultados são descritos na tabela 8.3. Nela, é dado o erro percentual médio e o desvio percentual médio do NS quando comparados aos resultados obtidos pela simulação das rotas. Também fornece o pior caso, ou seja, o maior erro percentual encontrado dentre todos os clientes de todas as rotas testadas. O nível de serviço desejado foi de 95% e considerou-se o desvio-padrão dos arcos como 20% do seu comprimento. A simulação foi executada com 1 corrida de 10000 réplicas.

	Erros para o NS (%)		
Método	Média	Desvio	Pior Caso
ProbR	0.116	0.238	6.504
ProbS	0.156	0.265	7.504
ProbJ	0.171	0.291	11.095
ProbChang	0.319	0.528	11.095
Chernoff	3.313	7.661	58.812
Chebyshev	6.121	11.215	81.700

Tabela 8.3: qualidade das estimativas do nível de serviço (NS).

Observa-se pela tabela 8.3 que o método que utiliza a Desigualdade de Chebyshev para calcular a probabilidade do veículo violar a janela de tempo do cliente apresentou os maiores erros. Isto ocorre pelo fato desta desigualdade dar apenas um limite inferior para o NS calculado, sendo muito conservadora.

O método que utiliza a Desigualdade de Chernoff forneceu um limite inferior mais apertado que Chebyshev, porém os resultados continuaram ruins.

O método identificado como ProbChang faz a suposição de normalidade no cálculo do NS e utiliza estimativas de média e desvio descritas no tópico 4.1.6. Observa-se que os resultados são muito melhores que os fornecidos pelas Desigualdades de Chebyshev e Chernoff, porém ainda significativamente piores que os demais 3 métodos. Isto mostra que supor normalidade é melhor que assumir total desconhecimento da forma da função distribuição de probabilidade.

O método ProbJ assim como ProbChang assume normalidade. O resultado melhor de ProbJ pode ser explicado pela melhor estimativa da média e desvio-padrão do tempo de chegada do veículo no cliente.

Os resultados para ProbS apontam para uma pequena melhora no cálculo do NS, particularmente na redução do pior caso em relação ao ProbJ.

ProbR forneceu os melhores resultados com redução significativa da média e do pior caso do erro. Isto reforça mais uma vez a utilidade das regressões utilizadas para corrigir a probabilidade calculada do cliente violar sua janela de atendimento.

Nestes testes, assumiu-se que a variância é 20% do valor do arco. Será que o desempenho comparativo entre os diferentes métodos muda se forem utilizados outros valores percentuais para o parâmetro variância do arco? Para responder a esta questão, os mesmos experimentos cujos resultados são descritos na tabela 8.3 foram repetidos para variâncias de 10% e 40% do arco. Os resultados são descritos na tabela 8.4.

	10%			**40%**		
Método	Média	Desvio	Pior Caso	Média	Desvio	Pior Caso
ProbR	0.132	0.211	5.112	0.157	0.301	7.107
ProbS	0.169	0.252	6.912	0.179	0.391	9.115
ProbJ	0.186	0.311	12.162	0.185	0.421	13.161
ProbChang	0.378	0.579	13.281	0.399	0.556	15.124
Chernoff	3.461	7.972	54.753	3.313	7.661	61.921
Chebyshev	6.369	11.831	77.835	6.121	11.215	84.631

Tabela 8.4: erro do nível de serviço (NS) para diferentes parâmetros de variância

A tabela 8.4 mostra que existiram mudanças muito pequenas na média e desvio do erro. Houve algumas diferenças mais significativas no pior caso. Observa-se que o desempenho relativo entre os diferentes métodos não mudou, de forma que o método ProbR continuou a ser o método com o melhor resultado.

As figuras 8.1 a 8.3 ajudam a entender o motivo dos erros terem tido uma pequena mudança para diferentes parâmetros da variância. Para a mesma rota descrita na sessão 6.1, gerou-se via simulação os histogramas do tempo de chegada do veículo em cada cliente da rota.

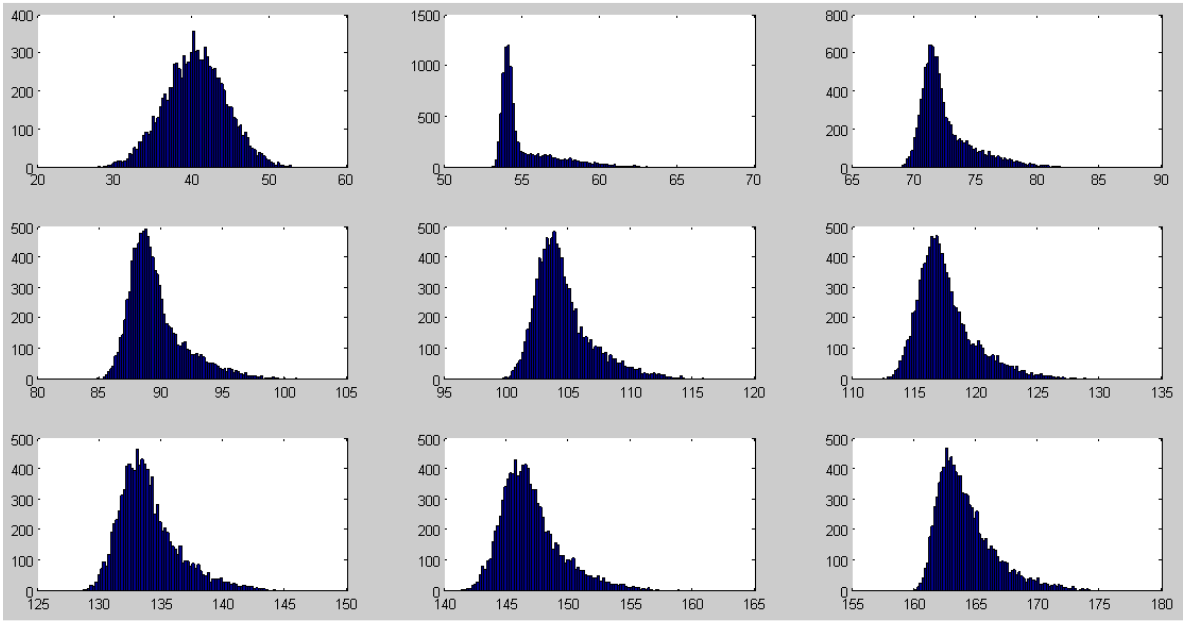


Figura 8.1. Distribuição do tempo de chegada dos clientes para variância de 10% do arco

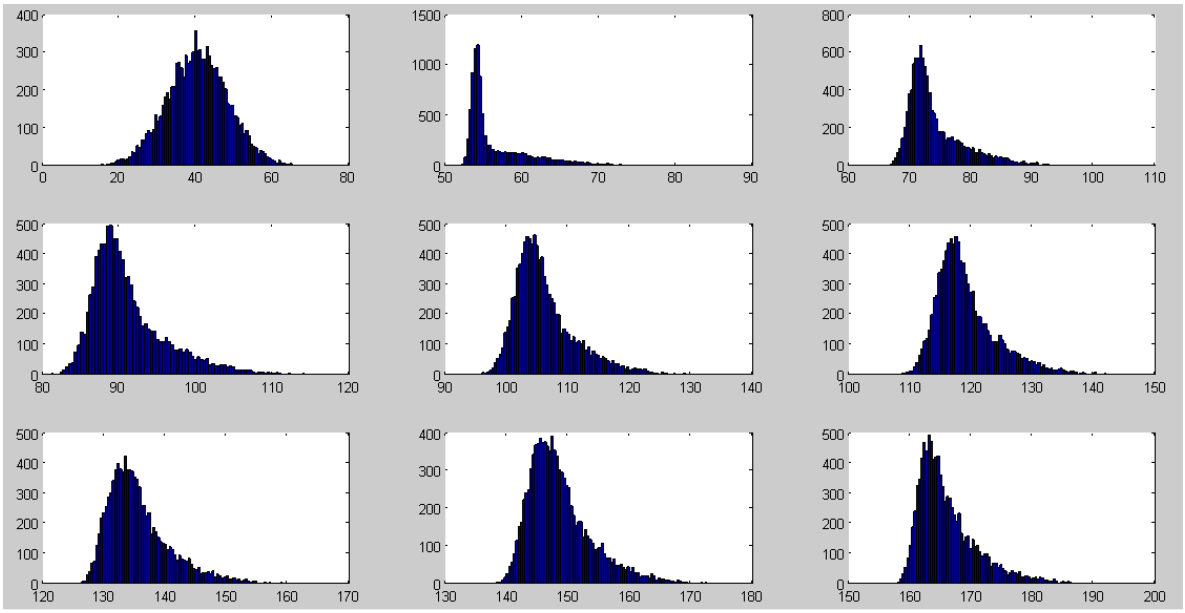


Figura 8.2. Distribuição do tempo de chegada dos clientes para variância de 20% do arco

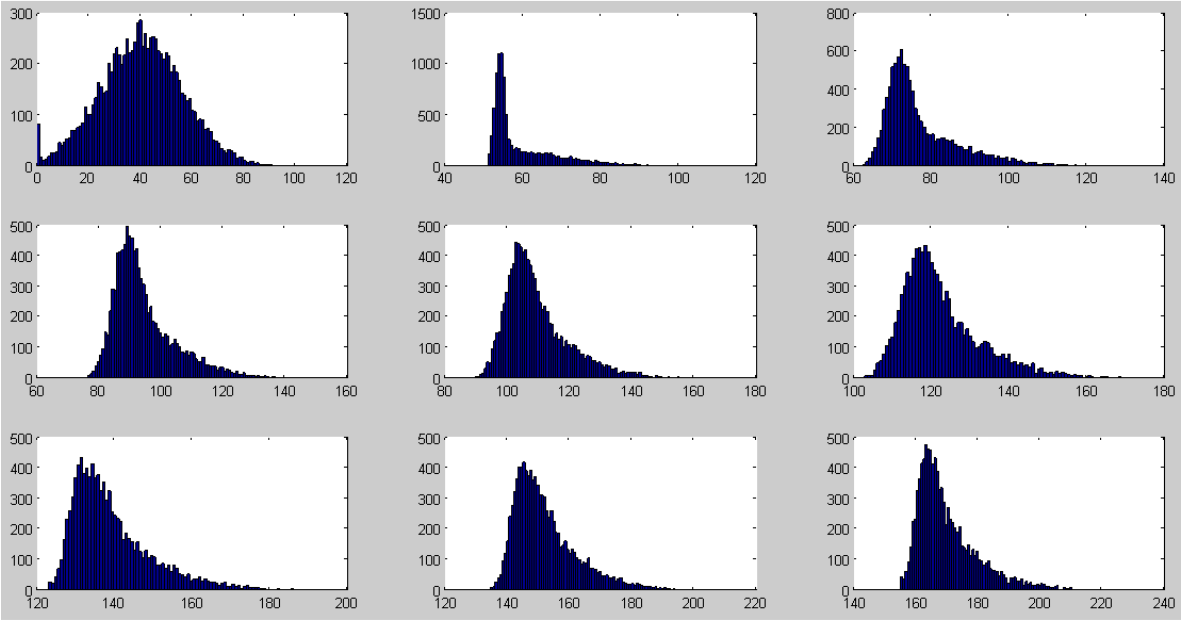


Figura 8.3. Distribuição do tempo de chegada dos clientes para variância de 40% do arco

Observa-se nas figuras 8.1 a 8.3 que, de modo geral, não houve mudanças na forma da distribuição para diferentes modos de gerar o valor das variâncias. De fato, observa-se um aumento da variância e também do tempo médio de chegada, mas não se observa mudanças expressivas na forma da distribuição. A média aumenta (deslocamento para direita) quando o desvio aumenta, pelo fato da distribuição ser truncada à esquerda. O fato de não haver mudanças expressivas na forma da distribuição apóia os resultados similares obtidos nas tabelas 8.3 e 8.4.

8.4 – VERIFICAÇÃO DA VIABILIDADE DE UMA ROTA

No contexto de uma heurística de roteamento com tempo de viagem estocástico, há a presença de uma restrição exigindo que o NS desejado de uma rota seja atendido. Assim, faz-se importante testar se o método que faz esta verificação é capaz de aprovar e rejeitar adequadamente uma dada rota, medindo-se a quantidade de falsas aprovações e falsas rejeições do método. Este é o objetivo deste tópico.

8.4.1 – Banco de Rotas Estendido

Testes com o banco de 180 rotas descrito em 8.1 mostraram uma quantidade muito baixa de falsas rejeições e falsas aprovações. Então gerou-se um banco maior com 3693 rotas especificamente para a execução dos experimentos desta seção. A necessidade de uma amostra maior pode ser explicada pelo fato da métrica de interesse ser de natureza discreta e ainda ter baixa freqüência de ocorrência. Esta baixa ocorrência pode ser explicada por alguns motivos. Caso $\mu_{ch}^{i} \leq l_i - 3\sigma_{ch}^{i}$ então a probabilidade de chegar antes do final da janela é muito alta, então a rota será sempre aprovada. Lembramos que $\mu_{ch}^{i}, \sigma_{ch}^{i}, l_i$ são respectivamente: média do tempo de chegada no cliente *i*, desvio-padrão do tempo de chegada no cliente *i* e final da janela.

Um segundo motivo pode ser exemplificado da seguinte maneira. Seja um método qualquer de cálculo da probabilidade de atendimento da janela de tempo do cliente. Suponha que o método forneça um erro ε uniformemente distribuído entre -5% e +5%, ou seja, $\varepsilon \cong U(-5\%, +5\%)$. Seja o *NS* (nível de serviço desejado) de 90%. Suponha ainda que para um determinado valor de μ_{ch}^{i} e σ_{ch}^{i} temos que a probabilidade real de atendimento (p_{real}) seja de 88%. Então, a rota deve ser rejeitada, pois o trecho da rota no cliente *i* não atende *NS*. Seja p_{cal} a probabilidade calculada pelo método, então teremos $p_{cal} \cong U(83\%, 93\%)$. Desta forma, a probabilidade do método aprovar incorretamente este trecho é a probabilidade de calcular um valor superior a 90%, que neste caso é de apenas 30%. Ou seja, para que o erro do método tenha maior probabilidade de rejeitar ou aprovar incorretamente certo trecho da rota, é preciso que p_{real} seja razoavelmente próximo a *NS*, caso contrário, a diferença entre p_{real} e *NS* servirá como uma margem de segurança para o método.

Para a geração das rotas do banco estendido, utilizou-se o mesmo procedimento utilizado para gerar o banco de 180 rotas, a partir das mesmas instâncias, alterando-se os parâmetros de entrada da heurística I1. Naturalmente, garantiu-se a inexistência de duplicidades, sendo este o principal fator de ter sido gerado uma quantidade diferentes de rotas para cada uma das instâncias de origem, conforme tabela 8.5.

Instância	Quantidade
C101	220
C104	375
C201	74
R101	717
R104	526
R201	303
RC101	652
RC104	489
RC201	337

Tabela 8.5: Banco de Rotas

8.4.2 – Resultados para a verificação de viabilidade

Devido a necessidade de simular-se todas as rotas para coletar dados, estes experimentos demandam maior tempo de execução. Então não serão testados aqui todos os métodos. Considerando-se os resultados obtidos nos tópicos anteriores, decidiu-se testar 2 métodos nesta seção: ProbJ e ProbR. O maior interesse aqui é buscar descobrir possíveis diferenças entre a simples suposição de normalidade e a utilização de regressões multivariadas para corrigir o cálculo fornecido pela suposição de normalidade.

A simulação de cada rota foi executada com 1 rodada de 10000 réplicas para coletar os dados com os quais os métodos ProbJ e ProbR serão comparados. Utilizou-se 95% como nível de serviço desejado e um desvio-padrão de 20% do comprimento do arco.

Métrica	ProbJ	ProbR
Quantidade de Rejeições	2057	2061
Quant. Falsas Rejeições	6	3
NS para o Pior Caso de Falsa Rejeição	96.5 %	95.4%
Quantidade de Aprovações	1636	1632
Quant. De Falsas Aprovações	18	11
NS para o Pior Caso de Falsa Aprov.	92.8%	93.7%

Tabela 8.6: verificação de viabilidade. NS desejado = 95%

Os resultados da tabela 8.6 mostram que o método ProbR teve um menor número de falsas rejeições e de falsas aprovações, apoiando a utilidade do uso das regressões no aumento da eficácia da aprovação ou rejeição de uma rota para um dado nível de serviço.

8.5 – EXPERIMENTOS COM A HEURÍSTICA DE ROTEAMENTO COM TEMPO DE VIAGEM ESTOCÁSTICO

O objetivo aqui é testar o comportamento de uma heurística com a restrição para nível mínimo de serviço desejado.

8.5.1 – Experimentos utilizando diferentes valores para nível de serviço e desvio-padrão

O objetivo é avaliar possíveis diferenças na solução do PRV usando-se diferentes métodos de verificação do nível de serviço desejado. Além disto, deseja-se investigar se estas possíveis diferenças sofrem alterações em cenários com diferentes valores para o nível de serviço e desvio-padrão.

Os experimentos foram realizados para as instâncias R101, R104 e RC101 de Solomon com 25 clientes. Selecionou-se estas 3 instâncias por terem janelas mais apertadas, sendo teoricamente instâncias mais difíceis. Utilizou-se nível de serviço (NS) de 90%, 95% e 99%. Também foram testados cenários com desvio-padrão de 20% e de 40% do comprimento do arco.

Testes preliminares mostraram que diferentes rodadas realizadas com diferentes sementes de geração de números aleatórios não tiveram influencia na solução final do VRP. Isto ocorre provavelmente devido ao espaço de busca reduzido da instância. Vale lembrar que os procedimentos que verificam a violação do nível de serviço de uma rota, embora calculem a probabilidade de violação da janela, têm natureza determinística (não probabilística).

As tabelas 8.7 a 8.12 mostram os resultados para a instância R101. Como referência, esta instância tem como solução ótima na versão determinística: função objetivo de 617,1 e 8 veículos.

A coluna “NScal” possui o nível de serviço da rota com menor nível de serviço da solução do PRV. A coluna “NSreal” tem o valor do nível de serviço da mesma rota porém obtido via simulação.

R101: NS=90% e Desvio=20%				
Método	NV	FO	NScal (%)	NSreal (%)
ProbJ	9	675.10	91.37	91.43
ProbS	9	675.10	91.37	91.43
ProbR	9	675.10	91.37	91.43
ProbChang	9	675.10	91.37	91.43
Chebyshev	12	747.27	58.87	100.0
Chernoff	12	745.80	70.35	99.0

Tabela 8.7: Experimentos com R101, 25 clientes, NS=90%

R101: NS=95% e Desvio=20%				
Método	NV	FO	NScal (%)	NSreal (%)
ProbJ	10	728.24	95.05	95.01
ProbS	10	728.24	95.05	95.01
ProbR	10	728.24	95.05	95.01
ProbChang	10	728.24	95.05	95.01
Chebyshev	12	747.27	58.87	100.0
Chernoff	12	745.80	70.35	99.0

Tabela 8.8: Experimentos com R101, 25 clientes, NS=95%

R101: NS=99% e Desvio=20%				
Método	NV	FO	NScal (%)	NSreal (%)
ProbJ	12	745.80	99.05	99.00
ProbS	12	745.80	99.05	99.00
ProbR	12	745.80	99.05	99.00
ProbChang	12	745.80	99.05	99.00
Chebyshev	15	888.62	100.00	100.00
Chernoff	12	747.27	99.99	100.00

Tabela 8.9: Experimentos com R101, 25 clientes, NS=99%

R101: NS=90% e Desvio=40%				
Método	NV	FO	NScal (%)	NSreal (%)
ProbJ	12	747.27	99.7	99.5
ProbS	12	747.27	99.6	99.5
ProbR	12	747.27	99.5	99.5
ProbChang	12	747.27	99.7	99.5
Chebyshev	13	796.41	90.5	100.0
Chernoff	12	747.27	93.6	99.5

Tabela 8.10: Experimentos com R101, 25 clientes, desvio=40%

R101: NS=95% e Desvio=40%				
Método	NV	FO	NScal (%)	NSreal (%)
ProbJ	12	747.27	99.7	99.5
ProbS	12	747.27	99.6	99.5
ProbR	12	747.27	99.5	99.5
ProbChang	12	747.27	99.7	99.5
Chebyshev	15	888.62	97.6	100.0
Chernoff	13	783.55	98.1	99.7

Tabela 8.11: Experimentos com R101, 25 clientes, NS=95%

R101: NS=99% e Desvio=40%				
Método	NV	FO	NScal (%)	NSreal (%)
ProbJ	12	747.27	99.7	99.5
ProbS	12	747.27	99.6	99.5
ProbR	12	747.27	99.5	99.5
ProbChang	12	747.27	99.7	99.5
Chebyshev	18	993.78	99.1	100.0
Chernoff	13	796.41	99.5	100.0

Tabela 8.12: Experimentos com R101, 25 clientes, NS=99%

Sobre os resultados das tabelas 8.7 a 8.12, a solução obtida para o PRV foi a mesma em todos os cenários testados para os métodos ProbJ, ProbS, ProbR e ProbChang. Estes métodos tendem a aprovar e rejeitar as mesmas rotas pelos motivos comentados na sessão 8.4.1, onde é visto que os erros nas rejeições ou aprovações das

rotas só aparecem quando o nível de serviço e a probabilidade de violação são próximos. Além disto, o nível de serviço calculado em todos estes métodos foi o mesmo pelo fato do tempo de chegada no cliente relatado ter sido normal.

Mudanças importantes foram obtidas para os métodos Chebyshev e Chernoff que obtiveram um maior número de veículos e maior avaliação da função objetivo. Para Chebyshev isto é muito mais evidente.

Também foi possível observar pelos resultados que o número de veículos e a distância total percorrida aumentaram quando o nível de serviço aumentou para os cenários com desvio-padrão de 20%. Mas o mesmo não foi observado para os cenários com desvio-padrão de 40%. Nestes cenários, observa-se que o número de veículos aumentou bastante logo para um NS de 90%, obtendo soluções com NS bastante superior ao desejado. Esta "folga" no nível de serviço obtido provavelmente foi o motivo do número de veículos e função objetivo não terem aumentado quando o NS desejado aumentou, pelo menos para os métodos ProbJ, ProbS, ProbR e ProbChang.

Os mesmos experimentos foram realizados com as instâncias R104 e RC101 com resultados similares aos da instância R101. Como nenhuma nova informação foi obtida, estes resultados não foram inseridos neste trabalho.

8.5.2 – Experimentos utilizando diferentes larguras para a janela de tempo

No tópico anterior, o NS e desvio-padrão foram variados e a largura da janela de tempo mantida a mesma da instância original. Aqui, variamos a largura da janela para um cenário com o mesmo NS e desvio-padrão. Utilizou-se NS de 90% e desvio-padrão de 20% do comprimento do arco. A instância escolhida foi a R101 (mesma do tópico anterior). Originalmente esta instância possui a janela do depósito variando de 0 a 230 unidades de tempo. E a largura de todas as janelas é de 10 unidades. Esta largura foi alterada conforme mostrado pela tabela 8.13.

Nesta tabela, a largura da janela de tempo dos clientes é calculada em função da largura da janela do depósito, variando de 100% a 1%. Também foi incluso um cenário em que os tempos de abertura da janela foram zerados para incluir um cenário

particular onde não há tempos de espera, ou seja, mesmo havendo uma janela de tempo, a suposição de normalidade é teoricamente válida.

Cenário	e	l	%
Sem Janela	0	230	100%
Sem Tempo de Espera	0	original	variável
Cenário 50	l(i)-0.50*ld	original	50%
Cenário 25	l(i)-0.25*ld	original	25%
Cenário 10	l(i)-0.10*ld	original	10%
Cenário 5	l(i)-0.05*ld	original	5%
Cenário 3	l(i)-0.03*ld	original	3%
Cenário 1	l(i)-0.01*ld	original	1%

ld: final da janela do depósito

e(i): início da janela de tempo

l(i): final da janela de tempo

Tabela 8.13: variação da largura da janela de tempo

A tabela 8.14 tem os resultados para a instância R101 com 25 clientes, utilizando o método ProbR.

Cenário	NV	FO	NScal (%)	NSreal (%)
Sem Janela	3	350.4	100.0	100.0
Sem Tempo de Espera	6	474.8	98.1	98.0
Cenário 50	6	477.8	99.9	99.9
Cenário 25	6	483.7	97.6	97.8
Cenário 10	7	601.5	95.0	95.1
Cenário 5	8	618.3	91.1	91.2
Cenário 3	12	745.8	96.8	96.6
Cenário 1	12	747.3	95.4	100.0

Tabela 8.14: resultados para variação da largura

Pela tabela 8.14, observa-se que o crescimento do número de veículos e também da função objetivo é rápido à medida que a largura da janela diminui. No Cenário 1 houve um erro significativo no valor do nível de serviço calculado, mas a solução de fato é viável. O mesmo teste realizado com os métodos ProbJ, ProbS e

ProbR obtiveram os mesmos resultados, exceto por pequenas diferenças no cálculo do NS que não afetaram o resultado da solução do PRV.

A tabela 8.15 tem dados referentes à execução dos mesmos experimentos com diferentes larguras para a janela com o objetivo de comparar o modelo determinístico com o modelo estocástico. Os resultados para o modelo estocástico foram obtidos pela execução da heurística para o PRV usando simulação estocástica como método de verificação do nível de serviço das rotas. A simulação de cada rota foi executada 2000 vezes. Adotou-se mais uma vez NS de 90% e desvio-padrão de 20% do arco.

	Determinístico		**Estocástico**	
Método	NV	FO	NV	FO
Sem Janela	3	350.4	3	350.4
Sem Tempo de Espera	4	457.8	6	475.8
Cenário 50	4	457.8	6	477.8
Cenário 25	5	486.7	6	483.7
Cenário 10	6	589.6	7	601.5
Cenário 5	8	618.3	8	618.3
Cenário 3	10	728.2	12	745.8
Cenário 1	10	735.1	12	747.3

Tabela 8.15: Experimentos com R101, 25 clientes, NS=90%

Os resultados da tabela 8.15 mostram que os resultados para o cenário "Sem Janela" são os mesmos. Isto ocorre, pois o modelo "Sem Janela" assume que cada cliente tem janela igual à do depósito, implicando em probabilidades de violação muito baixas. Na prática, é como se a restrição probabilística fosse desativada.

Também observa-se que o número de veículos e a função objetivo no modelo estocástico crescem mais rápido que no modelo determinístico. Também convém observar que reduzir a largura da janela tem efeito até um certo ponto, pois janelas muito estreitas farão com que a probabilidade de espera em todos os clientes das rotas seja sempre de 100%. Vale lembrar que o tempo de fechamento das janelas permaneceu sempre o mesmo (igual ao do problema original), sendo o tempo de abertura da janela que se aproximou do tempo de fechamento. Assim, à medida que o início da janela é deslocado para direita (aumentado), maior é a probabilidade que a chegada do veículo ocorra antes da abertura da janela, ou seja, maior a probabilidade de espera.

Para o cenário 5, os modelos tiveram os mesmos resultados. Curiosamente, este cenário é próximo da largura original das janelas.

8.5.3 – Experimentos para instâncias com 100 clientes

No tópico 8.5.2 os experimentos realizados em instâncias com 25 clientes não mostraram diferenças entre os métodos ProbJ, ProbS, ProbR e ProbChang. Aqui, iremos realizar testes em instâncias maiores, com 100 clientes para tentar identificar possíveis diferenças entre os métodos. Serão testados aqui apenas os métodos ProbJ e ProbR para comparar o método que assume normalidade com o método que utiliza regressões. O método ProbChang não foi selecionado, pois embora também assuma normalidade, os testes da sessão 8.3 mostraram que a estimativa de média e variância do tempo de chegada do método ProbJ é melhor.

Foram escolhidas 3 instâncias que nos testes da sessão 8.4.2 obtiveram um maior número de violações. Como explicado na sessão 8.4.1 e visto na sessão 8.4.2, não é simples encontrar situações em que os erros dos métodos apareçam. Realizou-se 5 execuções para cada uma das 3 instâncias. A execução 1 da instância 1 foi inicializada com a mesma semente da execução 1 das demais instâncias e assim sucessivamente. Os resultados são mostrados nas tabelas 8.16 a 8.18, onde *NV* é o número de veículos, *D* é a distância total percorrida, *NScal* é o nível de serviço calculado pelo método para a rota com o menor nível de serviço, *NSreal* é o valor do nível de serviço da mesma rota obtida via simulação. Vale lembrar que *NScal* e *NSreal* são calculados para cada cliente de cada uma das rotas da solução. Nas tabelas seguintes é fornecido o resultado para o cliente que obteve o menor *NS* entre todos os demais.

RC101; N=100; Desvio=20%; NS=90%

	ProbJ					ProbR				
rod	**D**	**NV**	**Nscal**	**Nsreal**	**abs(Erro)**	**D**	**NV**	**Nscal**	**Nsreal**	**abs(Erro)**
1	1800,6	19	92,5	88,4	4,1	1802,3	19	94,3	94,2	0,1
2	1776,2	18	91,1	90,9	0,2	1848,3	19	93,2	91,9	1,3
3	1749,6	18	93,9	86,8	7,1	1776,2	18	91	90,9	0,1
4	1848,3	19	92	91,9	0,1	1793,8	19	94,5	94,1	0,4
5	1749,6	18	93,9	86,8	7,1	1800,6	19	91,2	88,4	2,8
Média	1784,86	18,4	92,68	88,96	3,72	1804,24	18,8	92,84	91,9	0,94
Violações	3					1				

Tabela 8.16: Experimentos para instância RC101

C101; N=100; Desvio=20%; NS=90%

rod	ProbJ					ProbR				
	NV	D	Nscal	Nsreal	Erro	NV	D	Nscal	Nsreal	Erro
1	23	1846	90,2	89,8	0,4	23	1868	92,2	91,8	0,4
2	24	1889	93,8	93,9	0,1	24	1887	95,1	95,2	0,1
3	23	1846	90,2	89,8	0,4	23	1846	90,3	89,8	0,5
4	23	1868	92	91,8	0,2	23	1849	91,7	91,1	0,6
5	24	1882	99,9	94,3	5,6	23	1846	90,3	89,8	0,5
Média	23,4	1866	93,22	91,92	1,34	23,2	1859	91,92	91,54	0,42
Violações	2					2				

Tabela 8.17: Experimentos para instância C101

C104; N=100; Desvio=20%; NS=90%

rod	ProbJ					ProbR				
	NV	D	Nscal	Nsreal	Erro	NV	D	Nscal	Nsreal	Erro
1	11	1095	93,7	88,6	5,1	12	1097	90,7	91,7	1
2	11	1099	99,4	90,3	9,1	12	1109	95,2	93,8	1,4
3	11	1095	93,7	88,6	5,1	11	1099	90,4	90,3	0,1
4	12	1122	94,3	94,3	0	11	1102	92,5	92,7	0,2
5	12	1118	93,7	93,9	0,2	12	1130	97,9	96,7	1,2
Média	11,4	1106	94,96	91,14	3,9	11,6	1108	93,34	93,04	0,78
Violações	2					0				

Tabela 8.18: Experimentos para instância C104

Observa-se, que no total das 3 instâncias, existiram 7 violações para ProbJ e apenas 3 violações para ProbR que também obteve erros menores no cálculo das probabilidades. Os resultados são uma evidência de que a correção do cálculo da probabilidade de violação pelas regressões reduz o erro e que isto tem influência na solução do problema de roteamento.

Ou seja, em geral, quando o problema de roteamento é resolvido usando-se o método ProbR tende-se a obter soluções com pior avaliação da função objetivo, porém soluções viáveis. Quando o problema é resolvido com o método ProbJ, tende-se a obter soluções com melhor avaliação da função objetivo, porém não viáveis.

A discussão final dos resultados obtidos neste capítulo será vista no capítulo 9.

CAPÍTULO 9

CONCLUSÕES

Este capítulo apresenta as conclusões finais, resume as principais contribuições do trabalho e identifica trabalhos futuros a serem realizados.

9.1 – CONCLUSÕES E TRABALHOS FUTUROS SOBRE O PROBLEMA DETERMINÍSTICO

Os estudos realizados no capítulo 7 foram importantes para mostrar o desempenho das heurísticas desenvolvidas para os modelos determinísticos, tanto para o PRV Capacitado quanto para o PRV com Janela de Tempo.

Vários experimentos foram realizados para comparar o desempenho das 4 heurísticas desenvolvidas para resolver o PRV puramente capacitado. Os resultados mostraram que H4 obteve o melhor desempenho quanto à qualidade da solução e teve custo computacional competitivo com as demais heurísticas. Isto é uma evidência de que a inclusão da Busca Tabu, Memória Adaptativa e *GLS* foi satisfatória. Foi possível observar não apenas que H4 foi melhor, mas também que H3 superou H2 e H2 superou H1. Isto mostra que a inclusão apenas da Busca Tabu permitiu melhorar os resultados. A seguir, a inclusão da Memória Adaptativa em H3 permitiu alcançar resultados melhores, e por fim a inclusão do *GLS* em H4.

Foi feito uma comparação de H4 (implementado em Matlab) e H1 (implementado em Delphi) com outros reconhecidos trabalhos na literatura, utilizando-se todas as 14 instâncias de Christofides *et al.* (1979). Os resultados obtidos por H4 podem ser considerados satisfatórios tanto no que diz respeito à qualidade da solução, como também quanto ao tempo computacional. A melhoria que H4 pode sofrer quando implementado numa linguagem mais rápida pode ser estimada pela redução no tempo conseguido por H1 (quando implementado em Delphi).

A integração de todas estas técnicas dentro de uma estrutura *ILS* e *VND* é inédita na literatura. O mesmo acontece quanto à utilização de um conjunto auxiliar de

boas soluções que permite ao algoritmo sempre trabalhar com soluções de qualidade, ajudando na convergência dos resultados e potencializando o funcionamento da fase de perturbação. Os resultados obtidos são promissores e encorajam a realização de trabalhos futuros para refinar o funcionamento da metaheurística e testá-la em instâncias mais complexas, obtendo-se mais dados para realizar comparações com as melhores metaheurísticas de outros trabalhos.

Também foram realizados experimentos para o modelo com janela de tempo, utilizando-se 12 instâncias de Solomon (1987). A metaheurística conseguiu minimizar o número de veículos em todos os problemas demonstrando a eficiência do mecanismo de eliminação de rotas. Convém que melhores estudos sejam feitos com este procedimento de eliminação de rotas para comparar seu desempenho com outros procedimentos da literatura.

No que diz respeito à qualidade das soluções, foi medido a diferença das respostas obtidas com as melhores soluções já alcançadas e também com um *benchmark* específico. As respostas foram consideradas satisfatórias, mas também demonstraram que há espaço para melhorias. Estas devem ser realizadas em trabalhos futuros, incluindo testes não só em outras instâncias de Solomon (1987), mas também em instâncias feitas por outros autores.

9.2 – CONCLUSÕES E TRABALHOS FUTUROS SOBRE O PROBLEMA ESTOCÁSTICO

Os estudos realizados no capítulo 8 foram importantes para deixar claro que a existência de janelas de tempo em problemas em que o tempo de viagem entre os clientes é assumido normal faz com que a distribuição dos tempos de chegada não siga uma distribuição normal.

Métodos para estimar a média e a variância do tempo de chegada nas rotas foram desenvolvidos. Também foi elaborado um método para calcular a probabilidade de violação e de espera dos veículos. Experimentos compararam o método desenvolvido com outros métodos descritos na literatura e evidenciaram a superioridade do método

proposto. Ele conseguiu obter erros significativamente menores no que diz respeito à principal variável de interesse que é o cálculo da probabilidade de violação das janelas.

O método elaborado também mostrou-se superior aos demais nos experimentos que avaliaram a capacidade dos métodos de rejeitarem a aprovarem corretamente as rotas quanto ao nível de serviço desejado. Ele permitiu uma redução na quantidade de falsas rejeições e aprovações.

Experimentos foram realizados na heurística de roteamento para algumas instâncias de 25 clientes. Observou-se claramente que os métodos Chebyshev e Chernoff conduzem a soluções com maior número de veículos e maior distância percorrida. Embora tenha ficado claro que a suposição de normalidade pode implicar em erros grandes no cálculo da probabilidade de violação, diferenças entre os demais métodos não foram percebidas quanto à solução final obtida pelo modelo de roteamento. Por este motivo, novos experimentos foram realizados em instâncias de 100 clientes. Eles mostraram que os erros no cálculo da probabilidade obtidos pela suposição de normalidade podem sim fazer com que a solução obtida pelo problema de roteamento seja inviável. Os resultados também mostraram que o método proposto reduz o número de soluções violadas obtidas no problema de roteamento.

Nos experimentos observou-se que janelas de tempo menores fazem com que o número de veículos e a distância total percorrida aumentem. Este efeito é ainda maior quando é adotado um nível de serviço para o atendimento aos clientes.

Verificamos que existem poucos trabalhos que abordam diretamente o efeito da janela de tempo dos clientes no tempo de chegada dos veículos, sendo esta uma questão ainda não respondida na literatura. É preciso uma discussão mais profunda do assunto, pois como foi visto, apenas assumir normalidade pode levar a erros no cálculo do nível de serviço superiores a 15%, ou seja, grande o suficiente para inviabilizar o uso prático desta abordagem.

Até onde conhecemos, o método desenvolvido neste trabalho que utiliza regressão linear multivariada para corrigir os erros da suposição de normalidade é inédito na literatura. O método foi capaz de dar uma resposta mais adequada para a seguinte pergunta: como calcular a probabilidade de violação da janela de tempo dos clientes sem o uso da simulação? Deste modo, acreditamos ter dado uma contribuição para as pesquisas que estudam o PRVJT com Tempo de Viagem Estocástico.

Faz-se também importante a realização de futuras pesquisas para estudar melhor as regressões utilizadas, tentar identificar outros parâmetros de entrada que possam ser utilizados nos modelos de regressão para buscar reduzir ainda mais os erros obtidos nos cálculos de probabilidade.

Também se faz conveniente estudar o comportamento do algoritmo de roteamento em instâncias mais complexas, cujas soluções possuam um maior número de clientes na mesma rota e com diferentes larguras de janela de tempo, com o objetivo de identificar situações em que seja possível encontrar diferenças mais significativas entre a utilização de diferentes métodos de cálculo.

Faz-se também interessante estudar o comportamento do modelo estocástico com inclusão do tempo de serviço como variável aleatória. A introdução de outros parâmetros estocásticos talvez possa dificultar o uso de regressões, algo que também merece ser estudado futuramente.

O problema foi abordado segundo o modelo *CCP* em que a restrição probabilística tem uma resposta discreta no sentido de aprovar ou rejeitar uma dada rota. Observou-se que muitas vezes os erros no cálculo de probabilidade não causam falsas aprovações e rejeições. Faz-se interessante avaliar o modelo como um *SPR* em que o cálculo da probabilidade é incluído na função objetivo. Assim, a resposta do calculo da probabilidade de violação é um dado contínuo, levando à hipótese de que provavelmente o modelo *SPR* seja mais sensível aos erros de probabilidade que o modelo *CCP*. Algo que convém ser investigado nos próximos trabalhos.

9.3 – COMENTÁRIOS FINAIS

Esta dissertação permitiu adquirir e amadurecer o conhecimento de importantes técnicas de soluções para algumas variantes do problema de roteamento.

Nos problemas determinísticos, importantes oportunidades se fazem presentes na solução de problemas de larga escala e no refinamento das técnicas utilizadas. Nos problemas estocásticos, existem oportunidades e desafios para lidar melhor com a variação da distribuição de probabilidade dos tempos de chegada ao longo dos clientes de uma dada rota.

Deseja-se, nos próximos trabalhos, não apenas continuar o desenvolvimento das técnicas estudadas no presente trabalho, mas também incluir características que as aproximem de outras situações vividas no mundo real, resolvendo-se problemas num contexto multiobjetivo, com múltiplos depósitos e com frota heterogênea de veículos; onde com certeza mais oportunidades serão deslumbradas.

REFERÊNCIAS BIBLIOGRÁFICAS

Abramowitz M. & Stegun I. A. (1964). *Handbook of Mathematical Functions With Formulas, Graphs and Mathematical Tools*. National Bureau of Standards Applied Mathematics Series.

Alvarenga G. B. & Mateus G. R. (2004). A Two-Phase Genetic and Set Partitioning Approach for the Vehicle Routing Problem with Time Windows. Fourth International Conference on Hybrid Intelligent Systems (HIS04), *IEEE* Computer Society Press.

Alvarenga G. B. (2005). Um Algoritmo Híbrido para o Problema de Roteamento de Veículos Estático e Dinâmico com Janela de Tempo. Tese (Doutorado), UFMG, Departamento de Ciência da computação.

Assis L. P. (2007). Algoritmos para o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas. Dissertação (Mestrado). UFMG. Instituto de Ciências Exatas.

Azi, N. & Gendreau, M. & Potvin, J. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operations Research*, Volume 202, 756-763

Baker, E.K. (1982). Vehicle routing with time windows constraints. *Logistic and Transportation Review*, 18(4), p.385-401.

Baker B. M. & Ayechew M.A. (2003). Agenetic algorithm for the vehicle routing problema. *Computers & Operations Research* 30 787–800

Bard, J.F.; Kontoravdis, G.; Yu, G. (2002). A Branch and Cut Procedure for the Vehicle Routing Problens with Time Windows. *Transportation Science*, 36(2), p.250-269.

Bent R.; Van Hentenryck P. (2004). A two stage hibrid local search for the vehicle rounting problema with time Windows. *Transportation Science* 38 (4): 515-530.

Bergeglia, G.; Cordeua, J. & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operations Research*, Volume 202, 8-15

Bertsimas D. (1992). A vehicle routing problem with stochastic demand. *Operations Research,* 40(3).

Bischoff, E.; Ratcliff, M. (1995). Loading multiple pallets. *Journal of the Operational Research Society*. v 46, p 1322-1336.

Bramel J. & David S. (1997). The Logic of logistics: theory, algorithms, and applications for logistics management. *Springer*.

Braysy O.; Hasle G. & Dullaert W. (2004). A multi-start local search algorithm for the Vehicle Routing Problens with Time Windows. *European Journal of Operations Research*, Volume 159.

Bullnheimer, B.; Hartl R.F. & Strauss C. (1999). An improved ant system for the vehicle routing problem. *Annals of Operations Research*, vol. 89.

Campos V. & Mota E. (2000). Heuristic Procedures for the Capacitated Vehicle Routing Problem. *Computational Optimization and Applications*, 16, 265–277.

Caprara A. & Fischetti M. (1997). Branch-and-cut algorithms. Bibliographies in Combinatorial Optimization, Wiley, New York, pp. 45-64.

Chang, T., Wan Y. & Tsang W. (2009). A stochastic dynamic traveling salesman problem with hard time Windows. *European Journal of Operational Research*, vol. 198.

Cheng, C. & Wang, K. (2009). Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm. *Expert Systems with Applications*, vol. 36.

Christofides, N.; Mingozzi, A. & Toth. P. (1979). The vehicle routing problem. Combinatorial Optimization. Wiley, Chichester.

Clarke G & Wright. J. V. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, vol. 12.

Cordeau J. F., Gendreau M., Laporte G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, vol. 30.

Cordeau, J.-F.; Laporte G.; Mercier A. (2000). A unified tabu search heuristic for vehicle routing problems with time windows. Technical Report CRT-00-03, Centre for Research on Transportation, Montreal, Canada.

Dantzig G.B.; Ramser. J.H. (1959). The truck dispatching problem. *Management Science*, vol. 6.

Dantzig, G. B. & Wolfe, P. (1960) The decomposition algorithm for linear programming. *Operations Research*, vol. 8.

Desrochers, M.; Lenstra, J.K. & Savelsbergh M.W.P. (1990). A classification scheme for vehicle routing and scheduling problems. *Journal of Operational Research Society*, vol. 46.

Desrochers, M.; Verhoog, T.W. (1991). A new heuristic for the fleet size and mix vehicle routing problem. *Computers & Operations Research*, vol.18.

Desrochers, M.; Desrosiers, J. & Solomon, M. (1992). A new optimization algorithms for Vehicle Routing Problem with Time Windows. *Operations Research*, vol. 40.

Dueck G. & Scheurer T. (1990). A general purpose optimization algorithm. *Journal of' Computational Physics*, vol. 90.

Dror M. & Trudeau P. (1986). Stochastic vehicle routing with modified savings algorithm. *European Journal of Operational Research*, vol. 23.

Dror, M., Laporte, G. and Trudeau, P. (1989), Vehicle Routing with Stochastic Demands: properties and solution frameworks, *Transportation Science*, vol. 23.

Dullaert, W. (2000). Impact of relative route length on the choice of time insertion criteria for insertion heuristics for the vehicle routing problem with time windows. *Knowledge Tools Transportation Logist*. Faculty of Engineering, University of Rome, Italy, 153–156.

Dullaert, W.; Janssens, G.K.; Sorensen, K. & Vernimmen, B. (2002). New heuristics for Fleet Size and Mix Vehicle Routing with Time Windows. *Journal of the Operational Reasearch Society*, vol. 53.

Dullaert, W. & O. Bräysy. (2003). New insertion criteria for Solomon's (1987) insertion heuristic for the vehicle routing problem with time windows, Internal Report STF42 A03002, SINTEF Applied Mathematics, Department of Optimisation, Oslo, Norway (2003).

Feo T. A. & Resende M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, vol. 6.

Fisher M.L. & Jaikumar R. (1981) A generalized assignment heuristic for the vehicle routing problem. *Networks*, vol. 11.

Fisher M. L. (1994). Optimal solution of vehicle routing problems using minimum fc-trees. *Operations Research*, vol. 42.

Fisher. M.L. (1995). Vehicle routing. *Handbooks in Operations Research and Management Science,* vol. 8, North-Holland, Amsterdam.

Fischetti, M.; Toth, P.; Vigo D. (1994). A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations Research*, vol. 42.

Fleischer. M. (1995) Simulated annealing: past, present, and future. Winter Simulation Conference.

Francis, P. M.; Smilowitz K. R. & Tzur M. (2008). The Period Vehicle Routing Problem and its Extensions. *Operations Research/Computer Science Interfaces Series*, vol. 43.

Fukasawa R.; Longo H.; Lysgaard J.; Aragão M. P.; Reis M.; Uchoa E.; & Werneck R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, vol. 106.

Garey M. R. & Jonhson D.S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman.

Gaskell T.J. (1967). Bases for vehicle fleet scheduling. *Operational Research Quarterly*, vol. 18.

Gendreau M.; Hertz A. & Laporte G. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, vol. 40.

Gendreau M.; Laporte G.& Seguin R. (1996). Stochastic vehicle routing. *European Journal of Operational Research*, vol. 88.

Gendreau M. (2010). Conferencia no 42º Simpósio Brasileiro de Pesquisa Operacional (SBPO). Disponível em: <http://w3.ufsm.br/42sbpo/index.php?canal=material>. Acessado em agosto de 2011.

Gillett B.E. & Miller L.R. (1974). A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, vol. 22.

Glover, F.; Laguna, M. (1993). Tabu Search. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell, Oxford.

Glover, F.; Laguna, M. (1997). Tabu Search. Kluwer, Boston.

Goetschalckx M. & Jacobs-Blecha C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research*, vol. 42.

Golden B.L.; Magnanti T.L.; & Nguyen H.Q. (1977). *Implementing vehicle routing algorithms*. *Networks*, vol. 7.

Golden, B.L.; Wasil, E.A.; Kelly, J.P. & Chao. I.M. (1998). Metaheuristics in vehicle routing. *Fleet Management and Logistics*, Kluwer, Boston, pp. 33-56.

Groer, C.; Golden, B. & Wasil, E. (2009). The Consistent Vehicle Routing Problem. *Manufacturing & Service Operations Management*, vol. 11.

Halperin M. (1952). Estimation in the Truncated Normal Distribution. *Journal of the American Statistical Association*, vol. 47.

Hansen P. & Mladenovic N. (2003). A tutorial on variable neighborhood search. Technical Report, Les Cahiers du GERAD, HEC Montreal and GERAD.

Hertz, A.; Taillard, EÂ. D.; Werra, D. (1997). Tabu search. *Local Search in Combinatorial Optimization.* Wiley, Chichester.

Hjorring, C. A. (1995). The vehicle routing problem and local search metaheuristics. PhD Thesis. The University of Aucland.

Holland J.H. (1975). Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI.

Homberger J. & Gehring H. (1999). Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR*, vol. 37.

Hoos H. H. & Stutzle T. (2004) Stochastic Local Search: Foundations and Applications. *Elsevier*.

Hopfield J.J. & Tank D.W. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics*, vol. 52.

Ichoua S.; Gendreau M. & Potvin J-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, vol. 144

Jaillet P. & Odoni A. (1988). Vehicle Routing: Methods and Studies, In: *The Probabilistic Vehicle Routing Problem*. North-Holland, Amsterdam.

Jepsen, M.; Petersen, B.; Spoorendonk, S. & Pisinger, D. (2006). A non-robust branch-and-cutand- price algorithm for the vehicle routing problem with time windows. Technical Report, Department of Computer Science, University of Copenhagen, Denmark.

Jie G. (2010). Model and Algorithm of Vehicle Routing Problem with Time Windows in Stochastic Traffic Network. *Logistics Systems and Intelligent Management*, vol. 3.

Johnson A. C. (2003). Characteristics and tables of the left-truncated normal distribution. Disponível em: <http://domin.dom.edu/faculty/ajohnson/truncnorm.htm> Última atualização em maio de 2003. Acessado em agosto de 2011.

Jtinger M. & Thienel.S. (1998). Introduction to ABACUS—A Branch-and-Cut System. *Operations Research Letters*, vol. 22.

Jula H., Dessouky M. & Iannou P. (2006). Truck route planning in nonstationary stochastic networks with time windows at customer locations. *IEEE Transactions on Intelligent Transportation Systems*, vol. 7.

Kant, G. & Jacks, M. & Aantjes, C. (2008). Coca-Cola Enterprises Optimizes Vehicle Routes for Efficient Product Delivery. *Interfaces*, vol. 38.

Kao E. (1978). A preference order dynamic program for a stochastic travelling salesman problem. *Operations Research*, vol. 26.

Kenion A. S. & Morton D. P. (2003). Stochastic Vehicle Routing with Random Travel Times. *Transportation Science*, vol. 37.

Kindervater G.A.P. & Savelsbergh M.W.P. (1997). Vehicle routing: Handling edge exchanges. In E.H.L. Aarts & J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, Wiley, Chichester, UK, pp. 337-360.

Kolen, A.W.J.; Rinnooy, A.H.G. & Trienekens, H. (1987). Vehicle Routing with Time Windows. *Operations Research*, vol. 35.

Kytöjoki J.; Nuortio T.; Bräysy O.; Gendreau M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, vol. 34.

Laporte G. & Nobert. Y. (1987) Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathematics*, vol. 31.

Laporte, G.; Louveaux, F. & Mercure, H. (1992). The vehicle routing problem with stochastic travel times. *Operations Research Society of America*, vol. 26.
Laporte G.; Gendreau M.; Potvinb J. Y. & Semetc F. (2000). *Classical and modern heuristics for the vehicle routing Problem*. *Intl. Trans. in Op. Research*. vol. 7

Laporte, G. (2009). Fifty Years of Vehicle Routing. *Transportation Science*, vol. 43.

Larsen A. (2000). The Dynamic Vehicle Routing, PhD. Thesis. Department of Mathematical Modelling (IMM) at the Techical University of Denmark (DTU).

Lenstra, J. & Rinnooy Kan, A. (1981). Complexity of vehicle routing and scheduling problems, *Networks*, vol.11.

Li X.; Tian P. & Leung S. C. H. (2010). Vehicle routing problems with time windows and stochastic travel and service times: models and algorithms. *Int. J. Production Economics*, vol. 125.

Lin S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, vol. 44.

Lin S. & Kernighan B.W. (1973). An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, vol. 21.

Liu, F.H. & Shen S.Y. (1999). The fleet size and mix vehicle routing problema with time windows. *Journal of the Operational Research Society*, vol. 50.

Lourenço, H. R.; Martin, O. C. & Stüzle T. (2003). *Handbook of Metaheuristics*, chapter Iterated Local Search, pages 321–353. Kluwer Academic Publishers.

Lund, K.; Madsen, O.B.G.; Rygaard, J. M. (1996). Vehicle routing problems with varying degrees of dynamism. Technical report, IMM, The Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

Mester D`.; Bräysy O. & Dullaert W. (2005). A multi-parametric evolution strategies algorithm for vehicle routing problems.Working paper, University of Haifa, Israel, 2005.

Mester D`. & Bräysy O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research,* vol. 34.

Miller D.L. (1995). A matching based exact algorithm for capacitated vehicle routing problems. *ORSA Journal on Computing*, vol. 7.

Min H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research,* vol. 23.

Miranda, D. M. & Conceição S. V. (2010). Um problema de roteamento dinâmico, com tempos de viagem estocásticos, múltiplos veículos e janelas de tempo. Anais do 42º Simpósio Brasileiro de Pesquisa Operacional (SBPO). Disponível em: <http://www.sobrapo.org.br/sbpo2010/xliisbpo_pdf/71954.pdf>. Acessado em agosto de 2011.

Mole R.H. & Jameson S.R. (1976). A sequential route-building algorithm employing a generalized savings criterion. *Operational Research Quarterly*, vol. 27.

Nagata, W. (2007). Efficient Evolutionary Algorithm for the vehicle routing problems with time windows: Edge Assembly Crossver for the VRPTW. *IEEE Congress on Evolutionary Computation*, pag. 1175 – 1182.

Nagata, W. (2007). Edge assembly crossover for the capacitated vehicle routing problem. *Lecture Notes in Computer Science*, vol. 4446.

Nicholson T. (1971). Optimization in industry, optimization techniques. Longman Group Limited, v.1.

Osman I.H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, vol. 41.

Or I. (1976). Traveling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking. PhD. dissertation, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Pereira D. L. (2010). Heurísticas e Algoritmo Exato para o problema de roteamento de veículos com coleta e entrega simultâneas. Dissertação (Mestrado), Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas (UFMG).

Pessoa C. M. S. & Miranda D. M. (2011). Uso da modelagem de processo de negócio para integração de solução de roteamento em uma empresa de distribuição. Monografia do Curso de Bacharelado em Sistemas de Informação. UFMG – ICEx – DCC.

Pisinger D. & Ropke S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, vol. 34.

Potvian, J. & Xu, Y. & Benyahia, I. (2006). Vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research*, vol. 33.

Potvin J. Y. & Robillard C. (1995). Clustering for vehicle routing with a competitive neural network. *Neurocomputing*, vol. 8.

Potvin, J.-Y., J.-M. Rousseau. (1995). An exchange heuristic for routing problems with time windows. *J. Oper. Res.Soc.* vol. 46.

Potvin, J.-Y (2007). Evolutionary Algorithms for Vehicle Routing. CIRRELT-2007-48.

Prins, C. (2004). A simple and efective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, vol. 31.

Psaraftis H. (1980) A dynamic programming solution to the single vehicle many-tomany immediate request dial-a-ride problem. *Transportation Science*, vol. 14.

Psaraftis H. (1995). Dynamic Vehicle Routing: Status and Prospects. *Annals of Operations Research*, vol. 61.

Pureza V. M. M. (1990). Problemas de Roteamento de Veículos Via Metaheurística Tabu. Dissertação (Mestrado). Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas.

Reeves C.R. (1993). Modern Heuristic Techniques for Combinatorial Problems. John Wiley & Sons. Inc. New York, NY.

Renaud J.; Laporte G. & Boctor F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, vol. 23.

Renaud, J. & Boctor, F. F. (2002). A sweep-based algorithm for the fleet size and mix vehicle routing problem, *European Journal of Operational Research*, vol. 140.

Repoussis P. P.; Tarantilis C. D.; Ioannou G. (2009). Arc-Guided Evolutionary Algorithm for the Vehicle Routing Problem with Time Windows. *IEEE Transactions on Evolutionary Computation*, vol. 13.

Ribeiro C. C. (2004). Curso "Metaheurísticas", XI Escola Brasileira de Computação. Disponível em: < http://www.ic.uff.br/~celso/grupo_de_pesquisa.htm> Acesso em julho/2011.

Rochat, Y.; Taillard, EÂ D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics,* vol. 1.

Russell, R. (1977). An effective heuristic for the M-tour traveling salesman problem with some side conditions. *Oper. Research.* vol. 25.

Savelsbergh, M. W. P.; Sol, M. (1995). The General Pickup and Delivery Problem. *Transportation Science,* vol. 29.

Semet, F. (1995). A two-phase algorithm for the partial accessibility constrained vehicle routing. *Ann. Oper. Research*, vol. 61.

SINTEF, (2004). Acesso em julho/2011. Disponível em: http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html

Solomon, M. M. (1986). On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks,* vol. 16.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Research*, vol. 35.

Solomon, M. M. & Desrosiers, J. (1988). Time window constrained routing and scheduling problems. *Transportation Science*, vol. 22.

Taillard E. D. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, vol. 23.

Taillard E. D.; Badeau P.; Gendreau M.; Guertin F. & Potvin J.Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, vol. 31.

Taillard E.D.; Gambardella L.M.; Gendreau M.; & Potvin J.Y. (1998). Adaptive memory programming: A unified view of metaheuristics. Research Report IDSIA/19-98, IDSIA, Lugano, Switzerland.

Tarantilis C.D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research*, vol. 32.

Thompson P.M. & Psaraftis H.N. (1993). Cyclic transfer algorithms for multi-vehicle routing and scheduling problems. *Operations Research*, vol. 41.

Tillman F. (1969). The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, vol. 3.

Toth P. & Vigo D. (1998). Exact algorithms for vehicle routing. In T. Crainic & G. Laporte, editors, *Fleet Management and Logistics*, Kluwer, Boston, MA, pp. 1-31.

Toth, P. & Vigo, D., (1998b). Granular tabu search. Working paper, DEIS, University of Bologna.

Toth, P. & Vigo, D., (2002). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics* vol. 123.

Toth, P. & Vigo, D., (2002b). The Vehicle Rounting Problem. Copyright © 2002 by Society for Industrial and Applied Mathematics.

Vasconcelos J. A. (2008). Notas de aula da disciplina "Otimização Multiobjetivo" – UFMG – Programa de Pós Gradução em Engenharia Elétrica. Acessado em agosto/2011.
< http://www.cpdee.ufmg.br/~joao/OtimMultiobjetivo/OMO_JOAO_A5_GA.pdf>

Vieira, H. P. (2009). Metaheuristica para a Solução de Problemas de Roteamento de Veículos com Janela de Tempo. Dissertação (Mestrado). Unicamp, Instituto de Matemática, Estatística e Computação Científica.

Vigo D. (1996). A heuristic algorithm for the asymmetric capacitated vehicle routing problem. *European Journal of Operational Research*, vol. 89.

Voudouris C. & Tsang E. P. K. (2003). Handbook of Metaheuristics, chapter Guided Local Search, pages 186–218. Kluwer Academic Publishers.

Yellow. P. (1970). A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly*, vol. 21.

Waters, C.D.J. (1989), Vehicle-scheduling problems with uncertainty and omitted customers, *Journal of the Operational Research Society* vol. 40.

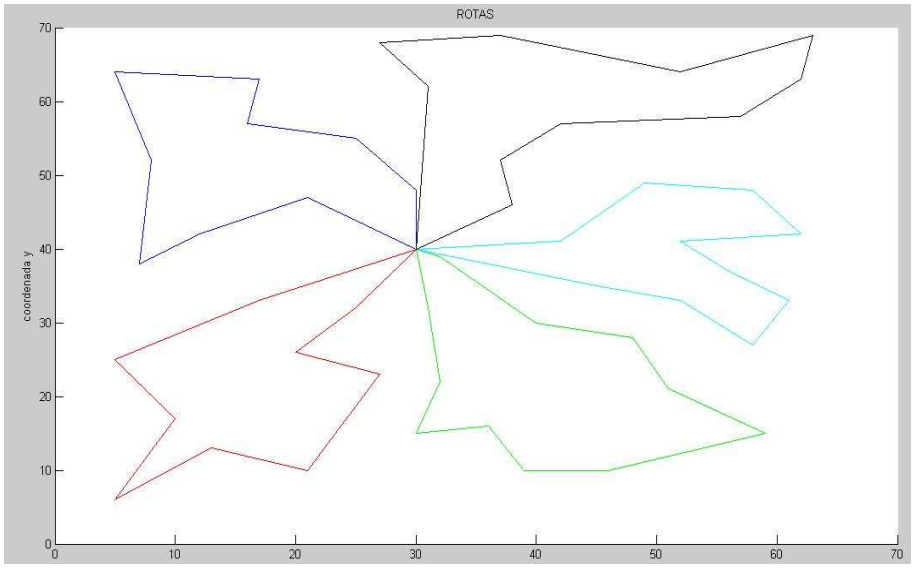
Wren, A. & Holliday. (1972). A Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly*, vol. 23.

Zeimpekis, V.; Tarantilis C. D.; Giaglis G. M. & Minis I. (2007). Dynamic Fleet Management. Concepts, Systems, Algorithms and Case Studies. *Springer*.

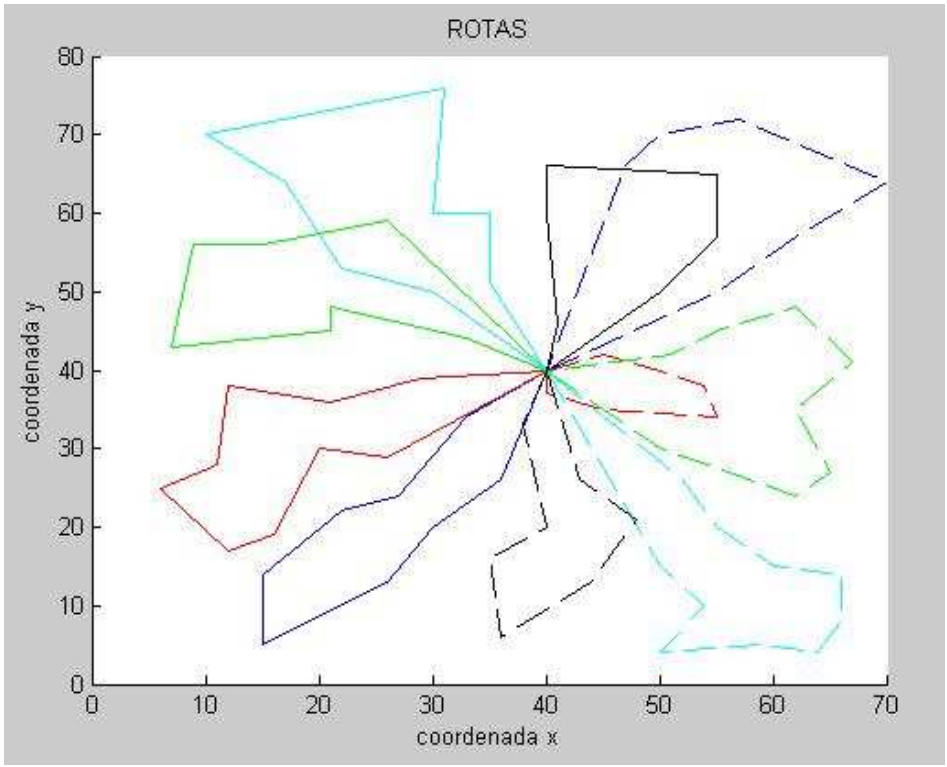
Ziviani, N. (2002). Projeto de Algoritmos. *Cengage Learning*.

APÊNDICE A

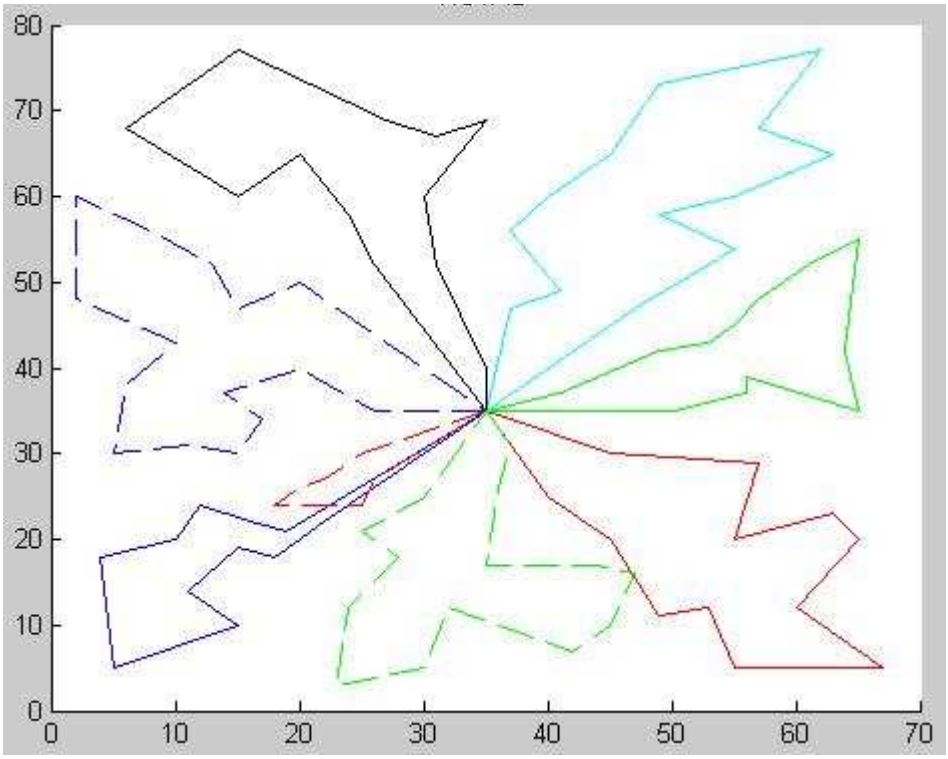
Algumas soluções encontradas pela heurística H1



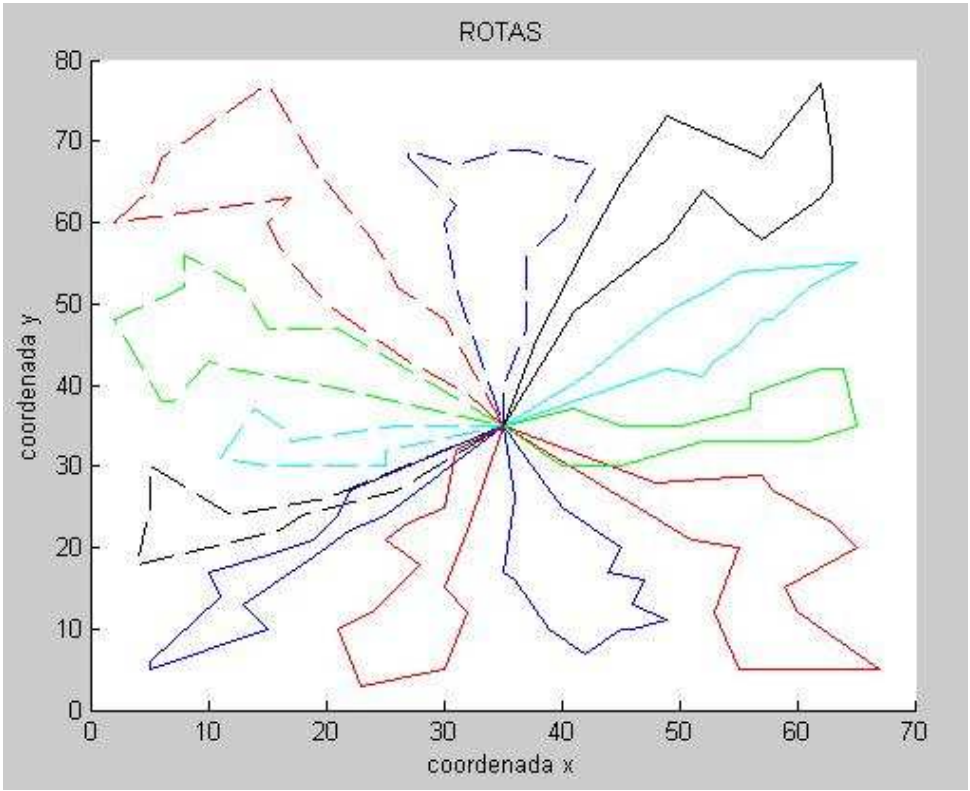
Instância vrpnc1, 50 clientes, distância: 524.6



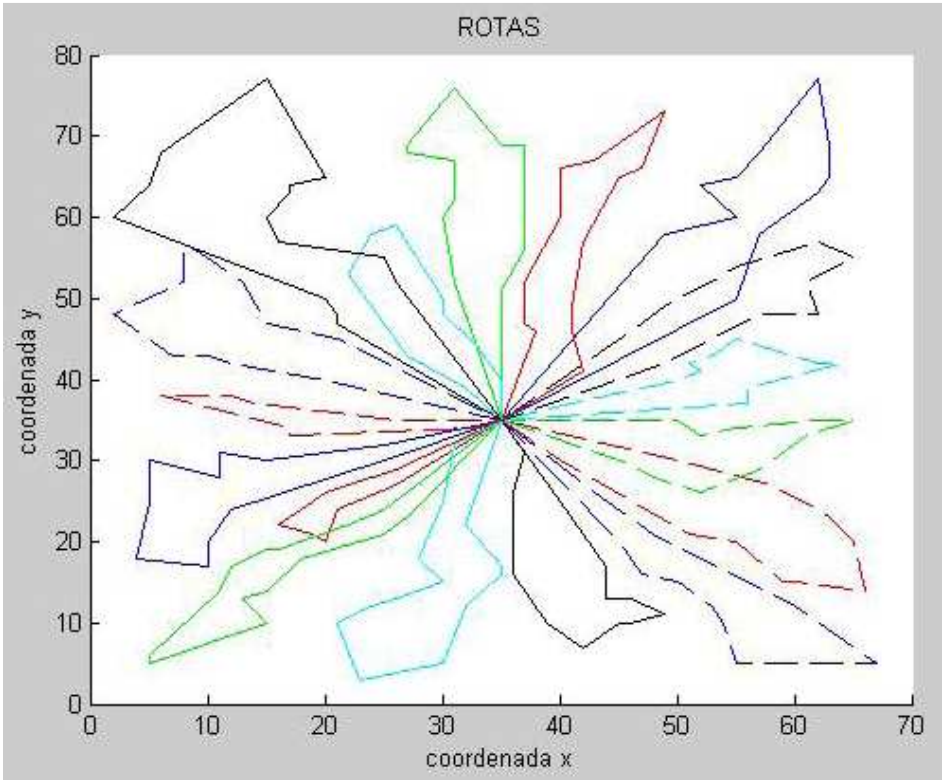
Instância vrpnc2, 75 clientes, distância: 849.9



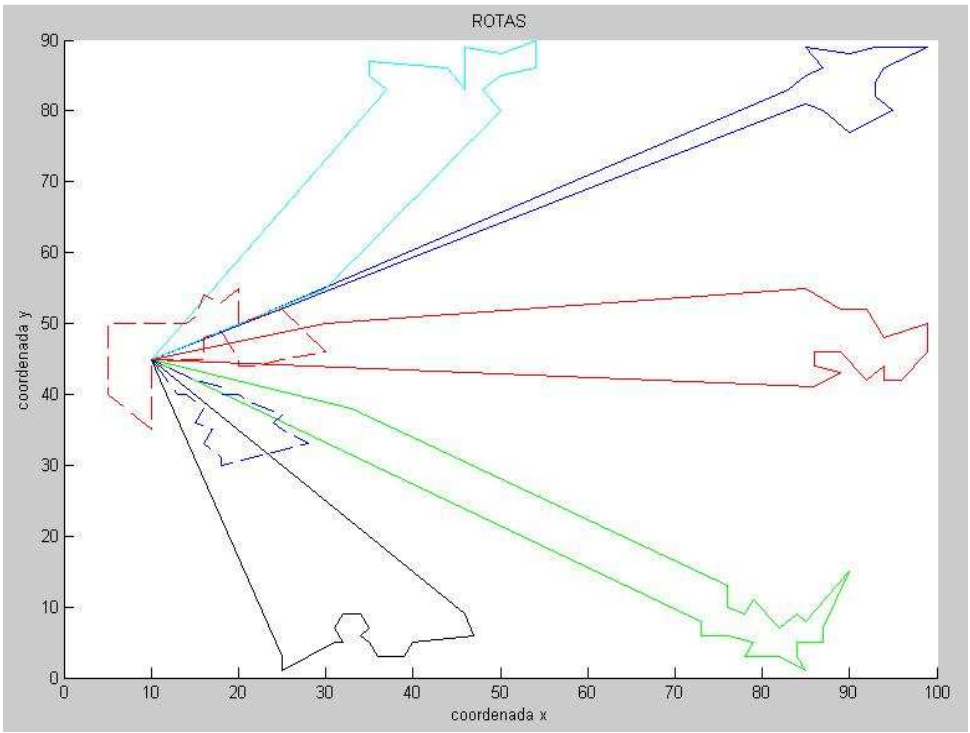
Instância vrpnc3, 100 clientes, distância: 832.8



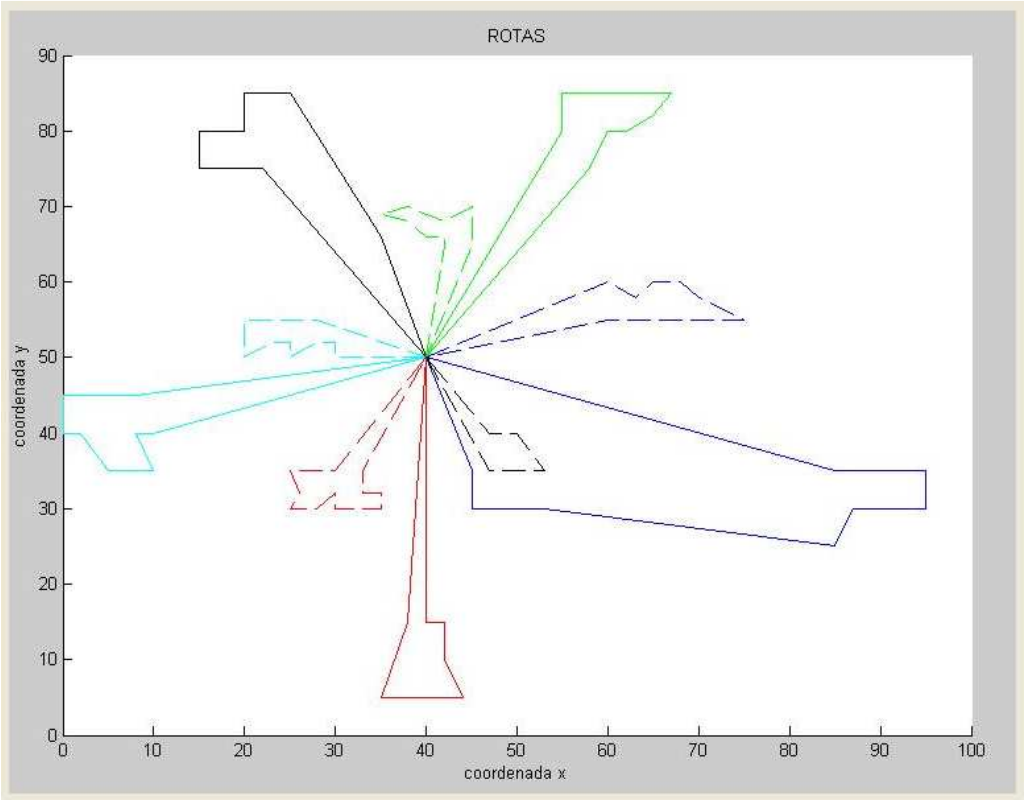
Instância vrpnc4, 150 clientes, distância: 1061.8



Instância vrpnc5, 199 clientes, distância: 1365.5



Instância vrpnc11, 120 clientes, distância: 1052.4



Instância vrpnc11, 100 clientes, distância: 819.96