UNIVERSIDADE FEDERAL DE PERNAMBUCO

DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO

CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

ROTEIRIZAÇÃO DE FROTA DE VEÍCULOS COM RESTRIÇÃO DE CONFIABILIDADE VIA ALGORITMOS GENÉTICOS

Trabalho de Conclusão de Curso

por

Andrea Pontual de Oliveira

Orientador: Enrique Andrés López Droguett, PhD

Recife, Novembro / 2009

UNIVERSIDADE FEDERAL DE PERNAMBUCO

DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

ROTEIRIZAÇÃO DE FROTA DE VEÍCULOS COM RESTRIÇÃO DE CONFIABILIDADE VIA ALGORITMOS GENÉTICOS

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia de Produção da Universidade Federal de Pernambuco - UFPE - como requisito parcial para obtenção de Grau em Engenharia de Produção.

Recife, Novembro / 2009

O48r **Oliveira, Andrea Pontual de.**
Roteirização de frota de veículos com restrição de confiabiblidade via algoritmos genéticos / Andrea Pontual de Oliveira. - Recife: O Autor, 2009.
vii, 45 folhas, il.: figs., tabs.

TCC (Graduação) – Universidade Federal de Pernambuco. CTG. Curso de Engenharia da Produção, 2009.

Inclui bibliografia.

1. Engenharia de Produção. 2. Problema de Roteirização de Veículos (PRV). 3. Algoritmos Genéticos (AG). 4. Confiabilidade. I. Título.

UFPE

658.5 **CDD (22. ed.)** **BCTG/2009-248**

AGRADECIMENTOS

Várias pessoas contribuíram direta ou indiretamente para a elaboração deste Trabalho de Conclusão de Curso. Gostaria de agradecer especialmente:

- a minha família, principalmente aos meus pais Newson de Oliveira e Rita de Cássia que me ensinaram a ser determinada e esforçada, sempre me incentivando e apoiando;
- ao meu orientador Enrique López Droguett por todos conhecimentos e experiências passadas;
- a Paulo Renato, Isis Diddier e Márcio Moura pela contribuição neste trabalho e pela amizade;
- a todos meus colegas de turma que tornaram essa jornada mais divertida e alegre.

RESUMO

Neste trabalho, propõe-se um modelo de roteirização que leve em consideração a confiabilidade dos veículos, ou seja, sua probabilidade de falha ao longo da rota. Assim, além da restrição de capacidade dos veículos, inclui-se uma restrição de confiabilidade mínima permitida ao final da rota. Assume-se uma frota heterogênea de veículos, cada um com características próprias de capacidade, confiabilidade e custo, sendo o objetivo encontrar os roteiros que minimizem o custo total da empresa. Diante da complexidade do problema, desenvolve-se uma metodologia de otimização baseada em Algoritmos Genéticos (AG). Para adaptar os AG ao modelo desenvolvido, foram criados novos operadores de *crossover* e mutação, além de um novo método de inicialização. A metodologia de otimização é testada através de dois problemas de validação e, em seguida, aplicada a dois estudos de caso.

Palavras-chave: Problema de Roteirização de Veículos (PRV); Algoritmos Genéticos (AG); Confiabilidade.

SUMÁRIO

1 INTRODUÇÃO 1

1.1 Objetivo Geral 2
1.2 Objetivos Específicos 3
1.3 Justificativa e Relevância 3
1.4 Metodologia 4
1.5 Organização do trabalho 4

2 EMBASAMENTO TEÓRICO 6

2.1 Problema de Roteirização de Veículos (PRV) 6
2.1.1 O Problema do Caixeiro Viajante (PCV) 6
2.2 Algoritmos Genéticos (AG) 8
2.2.1 AG e a Teoria de Evolução Natural 9
2.2.2 Etapas do processo de otimização via AG 10
2.2.3 Codificação por permutação em AG 11
2.3 Confiabilidade 14
2.3.1 Função de Confiabilidade 15
2.3.2 Função de Distribuição Acumulada (CDF) 15
2.3.3 Função densidade de Probabilidade (PDF) 16
2.3.4 Taxa de falha 16
2.3.5 Distribuições contínuas de Probabilidade 17

3 MODELO PROPOSTO 20

3.3 Algoritmos Genéticos para o modelo proposto 22
3.1.1 Cromossomo 22
3.1.2 Inicialização da População 23
3.1.3 Operador de *crossover* 25
3.1.4 Operador de Mutação 26
3.1.5 Seleção e Substituição 27
3.1.6 Dados de entrada 27

4 EXEMPLOS DE APLICAÇÃO 29

4.1 Problema de Validação I 29
4.2 Problema de Validação II 31
4.3 Exemplo de Aplicação I 33
4.3 Exemplo de Aplicação 2 36

5 CONSIDERAÇÕES FINAIS 40

5.1 Limitações 40
5.2 Sugestões para trabalhos futuros 41

LISTA DE FIGURAS

Figura 1.1 – Cinco objetivos de desempenho da produção 2
Figura 2.1 – Genótipo e Fenótipo 9
Figura 2.2 – Codificação por permutação 11
Figura 2.3 – Inviabilidade de Crossover de codificação binária ou real 11
Figura 2.4 – Crossover PBX 12
Figura 2.5 – Crossover OBX 13
Figura 2.6 – Cycle Crossover 13
Figura 2.7 – Swap Mutation 14
Figura 2.8 - Position-Based-Mutation 14
Figura 3.1 – Depósito deve abastecer demandas de clientes 20
Figura 3.2 – Codificação por permutação para o PCV 22
Figura 3.3 – Codificação com dois cromossomos 23
Figura 3.4 – Codificação com único cromossomo 23
Figura 3.5 – Codificação com cromossomo de duas partes 23
Figura 3.6 – Distâncias de clientes ao depósito central 24
Figura 3.7 – Crossover OBX Adaptado 25
Figura 3.8 – Crossover PBX Adaptado 26
Figura 3.9 – Mutação PBM Adaptada 26
Figura 3.10 – Mutação OBM Adaptada 26
Figura 3.11 – Cálculo de distância vetorial 27
Figura 4.1 – Problema de validação I 29
Figura 4.2 – Problema de validação II 31
Figura 4.3 – Exemplo de Aplicação I 33
Figura 4.4 – Solução do exemplo de aplicação I 35
Figura 4.5 – Exemplo de aplicação II 36
Figura 4.6 – Solução do exemplo de aplicação II 38

LISTA DE TABELAS

Tabela 3.1 – Resultados de simulação com diferentes métodos de Inicialização 25
Tabela 4.1 – Matriz de distâncias do problema de validação I 30
Tabela 4.2 – Demanda de carga de cada cidade do problema de validação I 30
Tabela 4.3 – Parâmetros de simulação para o problema de validação I 31
Tabela 4.4 – Resultados de simulação para o problema de validação I 31
Tabela 4.5 – Matriz de distâncias do o problema de validação I 32
Tabela 4.6 – Resultados de simulação do o problema de validação II 33
Tabela 4.7 – Dados dos veículos do exemplo de aplicação I 34
Tabela 4.8 – Demanda de carga das cidades do exemplo de aplicação I 34
Tabela 4.9 – Resultados de simulação do exemplo de aplicação I 35
Tabela 4.10 – Parâmetros de simulação do exemplo de aplicação I 35
Tabela 4.11 – Demanda de carga das cidades do exemplo de aplicação II 37
Tabela 4.12 – Dados dos veículos do exemplo de aplicação II 37
Tabela 4.13 – Parâmetros de simulação do exemplo de aplicação II 38

1 INTRODUÇÃO

Segundo Bowersox *et al.* (2002), o desempenho da cadeia de abastecimento é um fator crítico e determinante da eficiência e competitividade das empresas. Perante o mercado globalizado e competitivo, as empresas se deparam com exigências cada vez mais fortes em relação à qualidade e credibilidade do sistema de abastecimento, além da pressão por baixos custos. Nesse contexto, a eficiência do sistema de transporte torna-se fundamental para assegurar a sobrevivência e o sucesso no mercado.

De fato, para a maioria das empresas, o transporte absorve a maior parte dos custos logísticos, podendo representar de um a dois terços (BALLOU, 1998). Por ser uma atividade que não agrega valor ao produto final, a redução desses custos torna-se de extrema importância, trazendo benefícios tanto para as empresas produtoras como aos consumidores que poderão adquirir produtos com um custo mais razoável. Na tentativa de reduzir os custos de transporte, as empresas se deparam inevitavelmente com o problema de roteirização e programação de veículos.

Segundo Neto e Lima (2005), roteirizar significa determinar um ou mais roteiros a serem cumpridos por veículos de transporte de uma frota, com o objetivo de visitar um conjunto de pontos que estão geograficamente dispersos em locais pré-determinados e que necessitam de atendimento. O problema de roteirização de veículos (PRV) consiste, então, em definir as rotas que minimizem o custo total. Apesar da simples definição, ele representa um dos maiores desafios da Pesquisa Operacional, existindo um rico acervo literário com diversas abordagens para solução. Eksioglu *et. al.* (2009) realizaram, inclusive, um amplo estudo da literatura do PRV, mostrando estatísticas de evolução dos trabalhos publicados, além de classificar as várias versões do PRV já criadas.

Além da busca por baixos custos do sistema de transporte, é essencial que a empresa tenha credibilidade, sob pena de perder competitividade (SLACK *et. al.*, 2002). A credibilidade é alcançada quando a empresa mantém os compromissos assumidos com seus clientes, sendo um dos cincos objetivos de desempenho da produção.

No contexto do PRV, pode-se afirmar que a credibilidade ganha uma ênfase ainda maior, exigindo-se que todas as entregas programadas sejam cumpridas no tempo certo. Porém, por mais que as empresas se organizem para atuar rapidamente e respeitar os prazos estabelecidos, vários fatores podem contribuir para o desvio em relação ao que foi planejado. Por exemplo, condições ambientais adversas, desequilíbrios fisiológicos e/ou emocionais do condutor, acidentes de percurso, protestos de movimentos sociais, etc. Dentre tantos fatores, este trabalho destaca a probabilidade de falha ao longo da rota, ditada pela função confiabilidade dos veículos.

Figura 1.1 – Cinco objetivos de desempenho da produção
Fonte: Adaptado de Slack (2002)

A falha de veículos ao longo da rota representa uma grande preocupação do setor logístico, provocando uma série de conseqüências negativas para a credibilidade e imagem da empresa, já que dificilmente a mesma conseguirá manter as entregas previstas. Dependendo do contexto, a proteção contra as incertezas de desempenho da frota pode se tornar até mesmo mais importante do que o próprio problema de roteirização.

Entretanto, observa-se que, de uma forma geral, os modelos de roteirização existentes na literatura não incluem a incerteza no desempenho dos veículos, considerando que a probabilidade de sucesso na entrega do produto é sempre 100%. Este trabalho possui, então, o objetivo de construir um modelo capaz de incluir essa incerteza, ajudando as empresas a se protegeram contra as falhas de veículos da frota.

Frente à complexidade do modelo desenvolvido, propõe-se a utilização de Algoritmos Genéticos (AG) para otimizá-lo. AG são algoritmos heurísticos inspirados nos mecanismos de evolução natural e recombinação genética utilizados para resolver problemas de otimização (EIBEN; SMITH, 2003). Devido às características particulares do modelo, porém, torna-se necessário realizar algumas modificações na codificação e operadores utilizados. A partir dessas adaptações, será possível otimizar de forma eficiente problemas reais de roteirização.

1.1 Objetivo Geral

O presente trabalho objetiva definir uma nova abordagem para o problema de roteirização que leve em consideração a confiabilidade dos veículos da frota. Paralelamente, pretende-se desenvolver um modelo de otimização baseado em Algoritmos Genéticos (AG) capaz de solucionar esse problema de forma eficiente.

1.2 Objetivos Específicos

Com o intuito de alcançar o objetivo geral, divide-se o trabalho nos seguintes objetivos específicos:

- Revisão da literatura sobre o Problema de Roteirização de Veículos e principais técnicas de solução;
- Revisão da literatura de confiabilidade e risco;
- Revisão da literatura sobre métodos heurísticos, principalmente Algoritmos Genéticos;
- Proposição de um problema de roteirização baseado em distâncias determinísticas e que leve em consideração a confiabilidade e capacidade dos veículos;
- Adaptação da codificação de operadores e de métodos dos Algoritmos Genéticos ao modelo proposto;
- Validação do modelo;
- Aplicação do modelo proposto a estudos de caso e avaliação dos resultados.

1.3 Justificativa e Relevância

O transporte é reconhecido como uma das áreas mais importantes da logística, uma vez que consome a maior porcentagem dos custos logísticos (BOWERSOX *et al.*, 2002). Nesse contexto, existe uma série de dilemas enfrentados pelos gestores, como problemas de localização de centros de distribuição, seleção do modal de transporte, programação da consolidação do embarque, problemas de dimensionamento da capacidade, entre outros.

Este trabalho destaca o problema de roteirização de veículos, onde se tem uma grande diversidade de mercadorias ou serviços a serem entregues a diferentes mercados consumidores que estão espalhados geograficamente. De uma forma geral, observa-se que as organizações não atribuem a devida importância a tal decisão, sendo comum apenas a utilização da experiência e "sentimento" para defini-lo. Ressalta-se também que qualquer falha na rota programada acarretará prejuízos à imagem da empresa perante seus clientes.

Sendo assim, pode-se dizer que a roteirização, influencia três pontos determinantes de resultados das organizações, quais sejam custo, qualidade e credibilidade. A escolha de um roteiro pode representar o apogeu ou, até mesmo, o fim da empresa.

Este trabalho pretende dar o real destaque ao problema de roteirização, na medida em que desenvolve um modelo que represente de forma mais fidedigna o problema de roteirização das empresas, incluindo restrições de confiabilidade que permitam a empresa ter uma maior segurança

da credibilidade do serviço oferecido aos clientes. Além disso, desenvolve modelos de otimização baseados em Algoritmos Genéticos para garantir a eficiência e eficácia das decisões logísticas. Com isso, pretende-se proporcionar uma redução dos custos, melhoria do nível de serviço, aumento da capacidade de cumprir prazos e elevação da competitividade e credibilidade para as organizações.

1.4 Metodologia

Segundo Cervo *et al.* (2007) a pesquisa é "um procedimento formal, com método de pensamento reflexivo, que requer um tratamento cientifico e se constitui no caminho para se conhecer a realidade ou para descobrir verdades parciais". O trabalho em questão se iniciou, portanto, com uma pesquisa bibliográfica sobre o tema abordado com o intuito de adquirir um vasto conhecimento sobre o que foi publicado em relação ao assunto.

Sendo um trabalho da área de pesquisa operacional, o próximo passo foi a definição do problema. Nesta etapa, foram definidos objetivos de estudo, identificando as possíveis alternativas de solução, limitações, restrições e exigências do sistema (ANDRADE, 2004). Em seguida, passa-se à construção do modelo feito com base na definição do problema. Esta foi a etapa mais longa, incluindo desde a definição do problema de programação matemática até a adaptação e implementação dos AG.

Uma vez implementado, partiu-se para a validação do modelo. Segundo Andrade (2004), a validação do modelo refere-se à capacidade de fornecer previsões aceitáveis sobre o comportamento de sistemas e respostas que contribuem para a qualidade da decisão a ser realizada. Neste trabalho, tal validação foi realizada através da avaliação do desempenho do algoritmo frente a problemas em que a resposta exata já é conhecida (como é o caso dos primeiros dois exemplos do capítulo 6). Após essa fase, o modelo pôde ser aplicado a casos mais realísticos, concluindo-se o trabalho através de uma análise final dos resultados do estudo.

1.5 Organização do trabalho

Além deste capítulo de introdução, esse trabalho possui mais cinco capítulos que estão organizados da seguinte maneira:

- Capítulo 2: Introdução aos principais conceitos utilizados no trabalho:
    - Definição e apresentação das diferentes versões do problema de roteirização e do Problema do Caixeiro Viajante (PCV);
    - Introdução à otimização via AG, dando uma ênfase especial à codificação de permutação;
    - Apresentação de conceitos básicos de confiabilidade de sistemas.

- Capítulo 3: Apresentação do modelo de roteirização proposto e das adaptações realizadas aos operadores dos AG;
- Capítulo 4: Resolução de testes de validação e exemplos de aplicação;
- Capítulo 5: Considerações finais.

2 EMBASAMENTO TEÓRICO

O presente capítulo discute os principais temas relacionados a este trabalho de conclusão de curso, quais sejam problema de roteirização de veículos, algoritmos genéticos e confiabilidade.

2.1 Problema de Roteirização de Veículos (PRV)

Desde que foi primeiramente proposto por Dantzig e Ramser (1959), o PRV já foi abordado em centenas de trabalhos, se tornando um dos problemas de otimização combinatória mais estudados. Segundo Laporte *et al.* (2000), o problema de roteirização de veículos consiste em definir rotas para uma frota de veículos de forma a minimizar o custo total. Todos os veículos devem partir de um depósito e entregar mercadorias a um conjunto de clientes dispersos geograficamente (cada um com demanda específica de mercadorias) e retornar ao depósito de origem de forma que cada cliente seja visitado uma única vez. O problema ainda inclui restrições de capacidade dos veículos, podendo também possuir restrições relativas à distância percorrida.

O PRV é, na verdade, uma extensão do reconhecido Problema do Caixeiro Viajante (PCV). De acordo com (GOLDBARG; LUNA, 2005), o PCV é um problema de otimização combinatória cujo objetivo é encontrar, para um dado conjunto de cidades e para a matriz de distâncias entre cada par destas cidades, o caminho mais curto que visite todas as cidades e retorne ao ponto de partida. Logo, pode-se constatar que o PRV seria uma versão avançada do PCV, considerando caixeiros múltiplos (como, de fato, descreve o Problema do Caixeiro Viajente Múltiplo - PCVM) com restrições de capacidade.

Torna-se importante, então, apresentar alguns conceitos e técnicas do PCV uma vez que os mesmos podem ser facilmente adaptados ao PRV. As seções seguintes discutirão mais profundamente o PCV.

2.1.1 O Problema do Caixeiro Viajante (PCV)

Dado um número finito de localidades e o custo de viagem entre elas, o PCV consiste em encontrar a forma mais econômica de visitar todas elas e retornar a origem. É importante ressaltar que nenhuma cidade pode ser visitada mais de uma vez e que a matriz de custos deve ser simétrica.

Na literatura, existem diversas formulações para o PCV, como as formulações de Miller-Tucker-Zemlin, Fox-Garvish-Graves e Claus (GOLDBARG; LUNA, 2005). Devido à importância e entendimento simplificado, expõe-se, aqui, a formulação matemática de Dantzig-Fulkerson-Johnson

(DFJ) que trata o PCV como um problema de programação binária sobre um grafo G(N,A), onde o objetivo é minimizar o somatório dos custos para ir de uma cidade "i" para outra "j", dado na equação:

$$Z = \sum_{j=1}^{n}\sum_{i=1}^{n} c_{ij}x_{ij} \tag{2.1}$$

Sujeito à,

$$\sum_{i=1}^{n} x_{ij} = 1 \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \tag{2.3}$$

$$\sum_{i=1}^{n} x_{ij} \trianglelefteq |S| - 1 \tag{2.4}$$

$$x_{ij} \in \{0,1\} \tag{2.5}$$

onde a variável x_{ij} assume o valor 1 se o arco (i,j) ∈ A for escolhido para integrar a solução, e zero, caso contrário. S é um subgrafo de *G,* em que |*S*| representa o número de vértices deste subgrafo. Nesta formulação está implícito que não existe x_{ii} e têm-se *n(n-1)* variáveis binárias. Um dos aspectos relevantes é a evidência da natureza combinatória do problema.

O PCV é considerado um problema computacionalmente difícil de ser resolvido. Esta dificuldade se deve, em parte, ao elevado número de possíveis soluções. Para um conjunto de *n* cidades, o número de soluções para uma matriz de custo simétrica é dado por *(n-1)!/2*. Para *n=30*, por exemplo, têm-se 4.42×10^{30} rotas diferentes. Assim, um computador que avalia uma rota em cerca de 10^{-10} segundos, levaria mais de 1.14×10^{13} *anos* para examinar todas as rotas. De fato, de acordo com a teoria da complexidade que estuda o quão longa é a solução de determinados problemas, o PCV é classificado como um problema *NP-difícil,* não existindo um algoritmo que o resolva em tempo polinomial (GOLDBARG; LUNA, 2005).

Dessa forma, o PCV representa, até hoje, um dos maiores desafios da Pesquisa Operacional, existindo um rico acervo literário com diversas abordagens para solução, tanto do ponto de vista determinístico quanto probabilístico. Como trabalhos envolvendo soluções exatas do PCV clássico, citam-se o de Laporte e Nobert (1973), Christofides e Eilon (1972) e de Miller e Pekny (1991). Recentemente, inclusive, Marcotte *et al.* (2004) mostraram que o PCV pode ser polinomialmente reduzido a uma ferramenta de otimização binível, se baseando em programação binível para demonstração. Trabalhos envolvendo algoritmos heurísticos, por outro lado, também merecem ser

citados. Cerqueira e Cravo (2006), por exemplo, resolvem o PCV através de colônia de formiga. Outros trabalhos que utilizam colônias de formiga para o PCV são Dorigo e Gambardella (1997) e Li e Gong (2003). Já a utilização de AG para resolução do PCV clássico é feita em Goldberg e Lingle (1985), Potvin (1996) e Chatterjee *et al.* (1996*).* Xu e Tsai (1991) e Potvin (1993), por outro lado, utilizam redes neurais.

Todas essas técnicas podem ser adaptadas para solucionar o PRV. Este trabalho pretende utilizar, por exemplo, AG para otimizar o modelo aqui desenvolvido. Sendo assim, a próxima seção apresenta alguns conceitos de AG necessários ao bom entendimento do trabalho.

2.2 Algoritmos Genéticos (AG)

Algoritmos Genéticos (AG) são algoritmos evolucionários inspirados nos mecanismos de evolução natural e recombinação genética utilizados para resolver problemas de otimização (MICHALEWICZ, 1996). AG fazem parte de uma classe de métodos de otimização estocásticos que simulam computacionalmente o processo de evolução natural denominada computação evolucionária (KENNEDY; EBERHART, 2001). Segundo Eiben e Smith (2003), eles se baseiam em uma população de indivíduos (soluções potenciais) que, em um dado ambiente, lutam pela sobrevivência e reprodução.

Os AG (assim como os demais métodos compreendidos pela computação evolucionária) são usados, principalmente, quando o número de variáveis é muito grande ou quando as funções-objetivo apresentam certas características não toleráveis pelos métodos clássicos de Programação Matemática, como, por exemplo, problemas que possuem funções multimodais e/ou espaços de busca de natureza discreta onde não é possível garantir as condições de diferenciabilidade e continuidade. Além disso, eles consideram uma população de soluções potenciais (indivíduos) simultaneamente ao invés de um único ponto, permitindo uma busca mais rápida e ampla.

Os primeiros trabalhos relacionados à computação evolucionária surgiram antes mesmo da revolução computacional. Data de 1948 a proposta de Turning sobre busca genética ou evolucionária. Entretanto, o desenvolvimento efetivo das técnicas de computação evolucionária só iniciou-se nos anos de 1970. Atualmente os algoritmos evolucionários são amplamente utilizados na literatura, sendo os AG um dos mais investigados devido a sua eficiência e variedade de problemas aos quais podem ser aplicados. Por exemplo, Martorell *et al.* (2000), utilizam AG para otimizar intervalos de teste em sistemas de energia nuclear. Azadivar & Shu (1998) desenvolvem uma metodologia que utiliza AG para otimizar políticas de manutenção considerando o estoque de processo (*Work in Process* – WIP) que diminui consideravelmente o efeito da falha dos equipamentos em uma linha de produção. Marseguerra *et al.* (2006), por outro lado, utilizam AG

multiobjetivo para otimização na área de confiabilidade, disponibilidade, manutenabilidade e segurança (RAMS). Lapa *et al*. (2000) maximizam a disponibilidade de sistemas nucleares em *standby* através da otimização de manutenção preventiva usando AG. Lapa *et al.* (2006) desenvolve uma nova metodologia para avaliação de políticas de manutenção preventiva baseada em custo e confiabilidade e utiliza AG na busca da política ótima de manutenção considerando a probabilidade e custo de manutenções corretivas, custo de manutenções preventivas e o impacto das mesmas na confiabilidade do sistema, além da probabilidade de reparo imperfeito.

2.2.1 AG e a Teoria de Evolução Natural

Através da implementação de operadores genéticos, os AGs tentam simular o processo de evolução natural. O termo "genético" vem justamente do fato de que foi nessa ciência que AG se inspirou, fornecendo os detalhes que permitiram traduzir o mecanismo em um modelo computacional. De fato, foi baseado no princípio Darwiniano de reprodução e sobrevivência dos mais aptos que o algoritmo foi desenvolvido.

Da mesma forma que acontece com ocorre na natureza, em um AG temos um grupo de indivíduos que competem entre si para garantirem sua sobrevivência e para assegurar que suas características sejam passadas adiante através de seus filhos. Fazendo a analogia, cada indivíduo é representado por seu cromossomo e está associado a uma solução potencial do problema de otimização. O meio ambiente é representado pelo algoritmo através de uma função avaliadora dos indivíduos denominada de *fitness* que dita o quão apto é o indivíduo, *i.e.*, a sua chance de sobrevivência para próxima geração. Os melhores indivíduos, ou seja, aqueles com um maior valor de *fitness* possuem uma maior chance de passar para próximas gerações e de transmitir suas características a descendentes. Assim, espera-se que as soluções evoluam até atingir o valor ótimo.

Em sua versão tradicional, os indivíduos têm um **fenótipo** associado, que é o vetor de valores assumidos pelas variáveis de decisão, e um **genótipo**, que é o fenótipo codificado. Na realidade, essa nomenclatura só é utilizada na codificação binária onde as variáveis de uma determinada natureza (real, categórica, etc.), são transformadas em uma base binária, sendo todo processo de otimização realizado com os números binários, só ocorrendo a decodificação para valores reais no momento de avaliação do *fitnes* (MICHALEWICZ, 1996). Quando não se trabalha com variáveis "binarizáveis", o genótipo é igual ao fenótipo, não se utilizando, portanto, esta nomenclatura.

Genótipo (10001011101101010001 11)
Fenótipo 0.637197

Figura 2.1 – Genótipo e Fenótipo

2.2.2 Etapas do processo de otimização via AG

O processo de otimização por meio de AG possui as seguintes etapas:

- **Geração da população inicial**: os indivíduos que formam a população inicial são gerados de maneira aleatória ou de acordo com um método específico. Consiste em criar valores iniciais para as variáveis de decisão.
- **Avaliação**: essa é a etapa em que o *fitness* (ou mais de um, no caso multiobjetivo) dos indivíduos é avaliado. Isto é, os valores das variáveis de decisão são substituídos na função de *fitness* para que o valor da mesma seja encontrado.
- **Penalização (para problemas com restrições)**: se um indivíduo não satisfizer as restrições do problema, ele é penalizado. Ou seja, se o problema for de maximização, o valor de *fitness* a ele associado é reduzido de uma parcela. No caso de minimização ocorre o inverso: aumenta-se o valor de *fitness* associado ao indivíduo inviável.
- **Seleção:** essa é a fase em que os indivíduos são selecionados para continuar nas próximas gerações (iterações do AG). Em geral, os indivíduos que apresentam os melhores valores de *fitness* tendem a ser selecionados. A seleção é o operador que imita o meio ambiente ao selecionar os indivíduos que transmitirão sua carga genética a populações futuras. O intuito da seleção é restringir a busca a determinadas regiões do espaço de busca e melhorar a qualidade média da população (ZITZLER, 1999).
- ***Crossover* (ou recombinação genética):** indivíduos da população são escolhidos de acordo com uma probabilidade de *crossover*. Nessa etapa, os indivíduos selecionados (pais) trocam partes dos seus genótipos a fim de gerar novos indivíduos (filhos) que se espera que tenham informações genéticas de melhor qualidade (que possam resultar em melhores valores de *fitness*).
- **Mutação:** de acordo com uma probabilidade de mutação, uma determinada parte do genótipo dos indivíduos pode ser modificada. A mutação é de grande importância para garantir a diversidade na população e a exploração do espaço de busca.
- **Substituição:** como a população é de tamanho constante, nem todos os indivíduos podem seguir em frente, uma vez que agora existem os filhos, (indivíduos resultantes do *crossover*). Assim, alguns indivíduos devem ser substituídos a fim de que o tamanho da população permaneça constante. Após a substituição, tem-se uma nova população e se termina uma iteração do algoritmo. Se o número de gerações não foi alcançado, segue-se para a etapa de seleção. Caso contrário, é o fim do AG.

2.2.3 Codificação por permutação em AG

Por se tratar de um problema que depende da ordenação e adjacência das variáveis, é recomendável utilizar uma representação especial para o PRV. Alguns exemplos de codificação desenvolvidos especialmente para esses tipos de problemas são a representação por adjacência, a representação ordinal e a codificação por permutação (também chamada de representação por caminhos) (SILVA; OLIVEIRA, 2006). Por ser mais natural e intuitiva, optou-se por utilizar a codificação por permutação neste trabalho.

Na codificação por permutação, cada indivíduo (possível solução para o problema) é representado por um cromossomo formado por uma seqüência de números inteiros. No caso do problema de roteirização, cada número refere-se a uma cidade a ser visitada pelo veículo e a ordem dos números indica a ordem das visitas. Por exemplo, o indivíduo da Figura 2.2 deve passar primeiro pela cidade 2, seguir para a três e assim sucessivamente até atingir a cidade 5.

Figura 2.2 – Codificação por permutação

Uma das restrições do problema de roteirização é o fato de que os veículos devem passar por uma cidade uma única vez, o que significa a impossibilidade de números repetidos nos cromossomos. Os operadores de *crossover* e mutação utilizados nas codificações binária e real, entretanto, não garantem a inexistência de números iguais em um mesmo cromossomo, conforme mostra a Figura 2.3.

Figura 2.3 – Inviabilidade de Crossover de codificação binária ou real

Para resolver este problema, são necessárias, então, adaptações aos operadores de recombinação genética e mutação para que os mesmos se tornem compatíveis com o problema combinatório. Assim, surgiram diversos tipos de *crossover* e mutação, específicos para a codificação por permutação. As próximas seções descrevem alguns deles.

2.2.3.1 Operadores de *Crossover*

- ***Partially Mapped Crossover* (PMX)** - Este modelo de *crossover* foi primeiramente proposto por Goldberg e Lingle (1985). O PMX segue os seguintes passos:

1. Selecionar, aleatoriamente, dois pontos de corte e copiar o segmento entre eles do *pai 1* para o primeiro filho;
2. Começando do primeiro ponto de corte, buscar elementos nesse segmento do *pai 2* que não foram copiados;
3. Denominando o(s) elemento(s) encontrado(s) no passo 2 de *i*, para cada um deles, verificar no filho qual elemento *j* do *pai 1* que foi copiado em seu lugar;
4. Posicionar, então, o elemento *i* na posição onde se encontra o *j* no *pai 2,* uma vez que se sabe que não se vai colocar *j* nesta posição, pois *j* já se encontra presente no filho;
5. Caso o lócus ocupado por *j* no *pai 2* já esteja preenchido no filho por um elemento *k*, colocar *i* na posição ocupada por *k* no *pai 2*.
6. Uma vez feito isso para todos os elementos encontrados no passo 2, o restante do filho é preenchido com os genes do *pai 2*. O segundo filho é criado, então, através do mesmo procedimento.

- ***Position Based Crossover* (PBX) –** Este operador enfatiza a preservação de uma posição no cromossomo. Primeiramente, posições do cromossomo dos pais são selecionadas aleatoriamente e os genes dessas posições são passados para cada um dos filhos. Em seguida, os genes restantes são passados para o filho na ordem em que eles aparecem no pai alternativo. A Figura 2.4 exemplifica o *crossover* PBX.

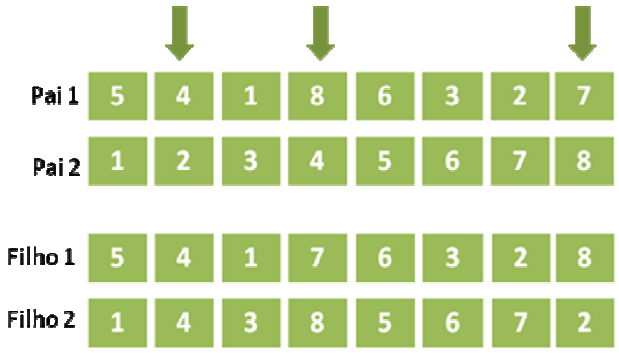


Figura 2.4 – Crossover PBX

- ***Order Based Crossover* (OBX) -** Para este *crossover*, os pontos do cromossomo são selecionados aleatoriamente e impõe-se uma ordem para os elementos selecionados do *pai 1* que seja igual à ordem dos respectivos no *pai 2*. Dessa forma, os filhos recebem os mesmos genes do pai, porém os elementos que foram selecionados são dispostos na ordem do pai alternativo, conforme a Figura 2.5.

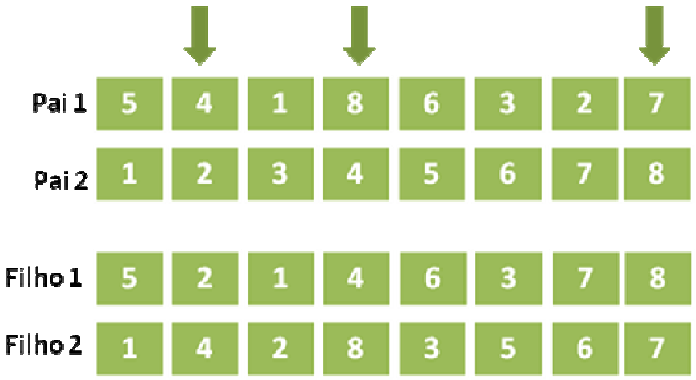


Figura 2.5 – Crossover OBX

- ***Order Crossover*** – Assim como no PMX, inicialmente este modelo seleciona aleatoriamente um seguimento do cromossomo do pai, transmitindo-o para o filho. Entretanto, a fase seguinte efetua um procedimento diferente do PMX devido à necessidade de transmitir ao filho a informação referente à ordem relativa dos elementos restantes, obtidos do segundo pai. Seu funcionamento é descrito a seguir:
    1. Escolhem-se dois pontos de cruzamento e efetua-se a cópia do segmento localizado entre eles, do primeiro *pai 1* para o primeiro filho.
    2. A partir do segundo ponto de cruzamento, copiam-se os elementos restantes (ainda não copiados) do segundo *pai 2* para o primeiro filho, obedecendo à ordem em que eles aparecem no *pai 2*.
    3. Cria-se o segundo filho de maneira análoga, trocando o *pai 1* pelo *pai 2* nas regras acima.

- ***Cycle Crossover –*** Neste *crossover*, cada cidade do cromossomo dos filhos deve vir de um dos pais (a cidade na mesma posição). A Figura 2.6 ilustra o cycle *crossover*. Para maiores informações ver Michalewicz (1996).

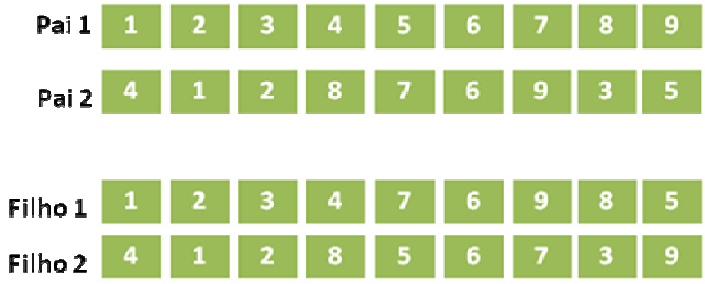


Figura 2.6 – Cycle Crossover

2.2.3.2 Operadores de Mutação

Entre os principais operadores de mutação (responsáveis por introduzir variabilidade genética na população) específicos para codificação por permutação, pode-se citar:

- ***Order-Based Mutation* (ou *Swap Mutation*) -** Esse operador consiste em selecionar aleatoriamente duas posições do cromossomo e trocar seus valores. A Figura 2.7 exemplifica este tipo de mutação.

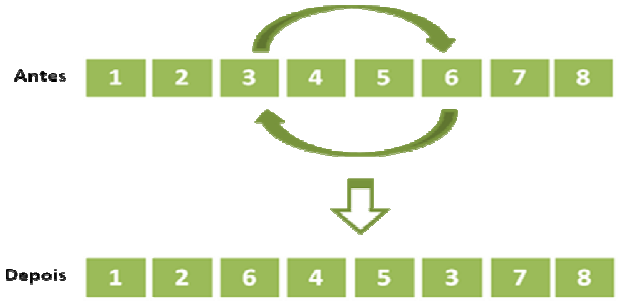


Figura 2.7 – Swap Mutation

- ***Position-Based Mutation (PBM)* –** Neste tipo de mutação, seleciona-se primeiramente uma posição *i* do cromossomo, de forma aleatória. Em seguida, uma segunda posição *j* é selecionada, também aleatoriamente, e retira-se o elemento da posição *i*, posicionando-o na posição *j*. A Figura 2.8 ilustra o PBM.

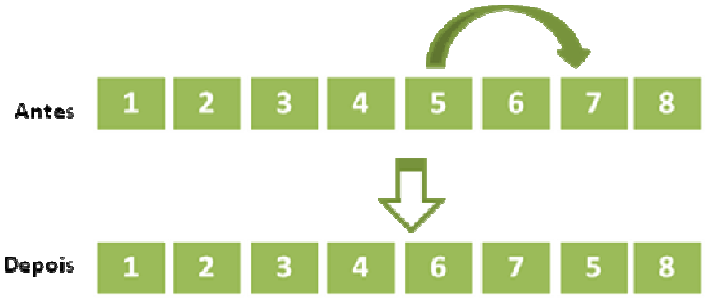


Figura 2.8 - Position-Based-Mutation

- ***Insert Mutation* -** Consiste em selecionar aleatoriamente duas posições do cromossomo e mover uma delas de forma que fique junto à outra.

2.3 Confiabilidade

Esta seção possui o objetivo de introduzir conceitos de confiabilidade que serão posteriormente utilizados neste trabalho. É importante ressaltar que mesmo sendo sistemas reparáveis, os veículos da frota são tratados aqui como sistemas não-reparáveis. Isso acontece, pois estamos interessados apenas em sua primeira falha, já que sabemos que o tempo de reparo dos veículos é, em geral,

elevado, inviabilizando a continuação da rota prevista. Logo, todos os conceitos aqui expostos se aplicam a sistemas não-reparáveis.

2.3.1 Função de Confiabilidade

A função de Confiabilidade *R(t)* é definida como a probabilidade que um sistema irá funcionar sob condições operacionais específicas durante algum período de tempo *t* sem falhar (MODARRES; KAMINSKIY, 1999). Ela pode ser expressa através da seguinte equação:

$$R(t) = P(T \geq t)\,,\ t \geq 0 \tag{2.6}$$

Onde:

- t - instante correspondente ao final do período de observação do sistema
- *T* - variável aleatória contínua que expressa o tempo de falha do sistema ($T \geq t$)

Dessa forma, a função confiabilidade pode ser interpretada como a probabilidade do sistema falhar pela primeira vez ou após o instante *t*. Por outro lado, em se tratando de um conjunto de componentes, podemos dizer que *R(t)* seria a fração esperada de componentes que está operacional em um determinado tempo *t*. De qualquer forma, *R(t)* deve ser uma função monotônica decrescente que satisfaz as seguintes condições:

$$R(0) = 1 \tag{2.7}$$

$$\lim_{t \to \infty} R(t) = 0. \tag{2.8}$$

O gráfico abaixo representa o comportamento da função confiabilidade, onde podemos comprovar que as condições descritas acima são respeitadas.

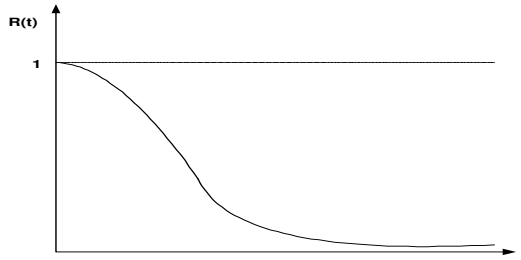


Figura 2.9 - Gráfico da Função de Confiabilidade

2.3.2 Função de Distribuição Acumulada (CDF)

A função de Distribuição Acumulada *(Cumulative Distribution Function – CDF)* de uma variável aleatória T é definida pela seguinte função (RIGDON; BASU, 2000):

$$F(t) = P(T \leq t)\text{ , t}\geq 0 \tag{2.9}$$

Dessa forma, a CDF representa o complementar da função confiabilidade e vice-versa. Logo, *F(t)* é uma função monotônica crescente.

$$R(t) = P(T \geq t) = 1 - P(T \leq t) = 1 - F(t) \tag{2.10}$$

2.3.3 Função densidade de Probabilidade (PDF)

A função densidade de probabilidade *(Probability Density Function – PDF)* é definida como a derivada da *CDF* (dado que a derivada existe realmente) (RIGDON; BASU, 2000).

$$f(t) = \frac{d}{dt}F(t) = -\frac{d}{dx}R(t) \tag{2.11}$$

Outra forma de representar a PDF seria através do limite:

$$f(t) = \lim_{\Delta x \to 0} \frac{F(t+\Delta t) - F(t)}{\Delta t} = \lim_{\Delta x \to 0} \frac{P(t < T < t+\Delta t)}{\Delta t} \tag{2.12}$$

A PDF descreve a forma da função da distribuição do tempo de falha e sua integral deve ter o valor de um:

$$\int_0^\infty f(t)dt = 1 \tag{2.13}$$

Além disso, através das equações expostas comprovamos que:

$$F(t) = \int_0^t f(t)dt \tag{2.14}$$

$$R(t) = \int_t^\infty f(t)dt \tag{2.15}$$

2.3.4 Taxa de falha

A taxa de falha *h(t)* pode ser definida como a intensidade com que um sistema passa de um estado para outro (MODARRES; KAMINSKIY, 1999). Ela é expressa pela seguinte equação:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t < T \leq t + \Delta t \mid T > t)}{\Delta t} \tag{2.16}$$

Como a equação mostra, *h(t)* é o limite da probabilidade de que um sistema falhe (pela primeira e única vez) em um pequeno intervalo de tempo, dado que ele funcionou até o início do intervalo.

A função de taxa de falha indica o desgaste ou melhoria do sistema. Por exemplo, uma taxa de falha crescente (ou seja, o limite da probabilidade de falha em um intervalo de tempo pequeno dividida pelo tamanho do intervalo cresce com o tempo) significa que o sistema está sujeito a um processo de desgaste e possui uma maior probabilidade de falha à medida que o tempo operacional aumenta. Por outro lado, se a taxa de falha é decrescente, o sistema possui uma menor probabilidade de falha com o passar do tempo operacional (também se pode dizer que o sistema está em *burn-in*).

Também é comum representar *h(t)* como a razão entre a *PDF e R(t).*

$$h(t) = \frac{f(t)}{R(t)} \tag{2.17}$$

2.3.5 Distribuições contínuas de Probabilidade

Nesta seção, serão apresentadas duas distribuições contínuas de probabilidade que serão posteriormente utilizadas no modelo desenvolvido: a distribuição exponencial e a distribuição *Weibull*.

2.3.5.1 Distribuição Exponencial

A distribuição exponencial é uma das distribuições contínuas de probabilidade mais simples e bastante utilizada. Ela deve ser utilizada em sistemas onde a taxa de falha é (ou pode ser considerada) constante. Ou seja, para sistemas que não apresentem maior ou menor probabilidade de falha com o acúmulo de tempo operacional.

Ela é bastante simples, pois possui apenas um parâmetro: a taxa de falha λ. Assim, a PDF da distribuição exponencial é definida como conforme equação (MODARRES; KAMINSKIY, 1999):

$$f(t) = -\lambda \exp(-\lambda t) \tag{2.18}$$

com média igual a $\frac{1}{\lambda}$, e variância igual a $\frac{1}{\lambda^2}$.

Sua função de confiabilidade e CDF são respectivamente:

$$R(t) = e^{-\lambda t} \tag{2.19}$$

$$F(t) = 1 - e^{-\lambda t} \tag{2.20}$$

Uma característica importante da distribuição exponencial é o fato de que o tempo de falha depende somente do tamanho do intervalo de tempo de operação (*t*) e não do tempo operacional acumulado do equipamento. Assim, é comum utilizar a expressão de que a distribuição exponencial não possui memória. O gráfico abaixo mostra a distribuição exponencial para diferentes parâmtros λ.

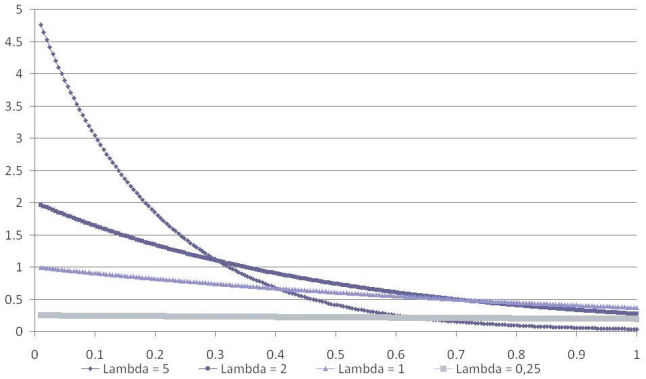


Figura 2.10 - Distribuição exponencial variando o λ

2.3.5.2 Distribuição *Weibull*

A distribuição de *Weibull* é dada pela seguinte equação:

$$f(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} \exp(-(t/\alpha)^{\beta}) \tag{2.21}$$

$\alpha, \beta > 0$

$t > 0$

Sua taxa de falha e confiabilidade são então descritas pelas seguintes equações, respectivamente:

$$h(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} \tag{2.22}$$

$$R(t) = e^{-(t/\alpha)^{\beta}} \tag{2.23}$$

Como se pode ver, a *weibull* possui dois parâmetros:

- β – parâmetro de forma (define a forma da distribuição)
- α – parâmetro de escala (define a escala da distribuição)

A distribuição *Weibull* é bastante utilizada devido a sua flexibilidade já que permite descrever taxas de falha constantes, crescentes e decrescentes. Por exemplo, quando temos β>1 a taxa de falha é crescente e o sistema está sujeito a desgaste. Já se β<1, a taxa de falha será decrescente e o sistema está passando por um processo de burn-in. Por fim, se β=1 a taxa de falha é constante e

igual a 1/α. Ou seja, a distribuição exponencial seria um caso especial ta distribuição *weibull* (quando β=1). Podemos observar este efeito claramente através do gráfico abaixo.

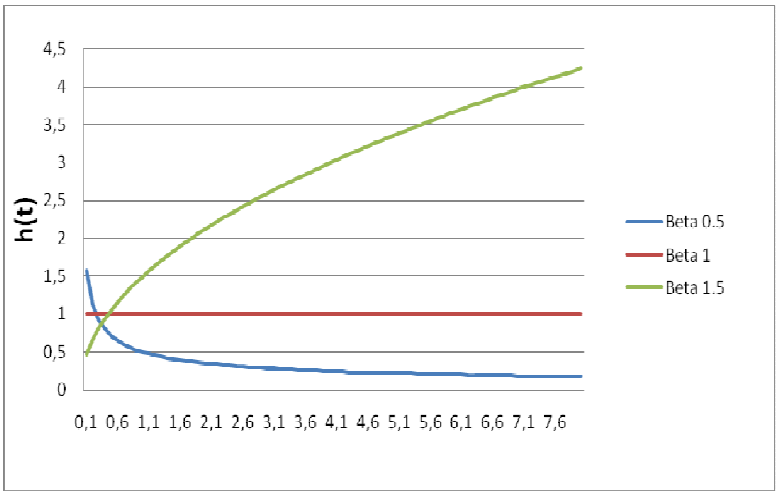


Figura 2.11 - Taxa de falha para diferentes valores da constante beta

3 MODELO PROPOSTO

Considere uma empresa que possui um único depósito central com um conjunto de mercadorias que devem ser entregues a clientes geograficamente dispersos. Cada cliente, por sua vez, possui demandas em quantidades diferentes e deve ser visitado uma única vez.

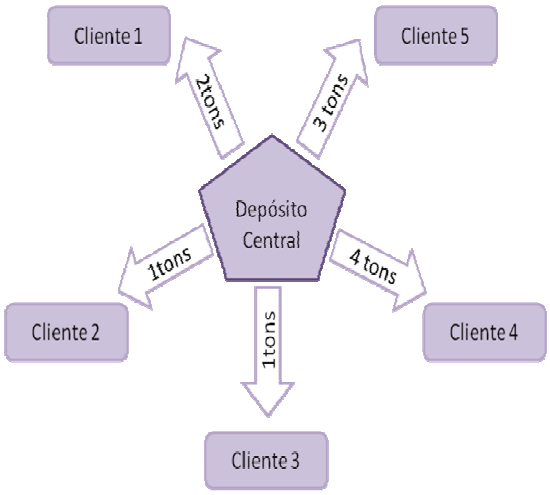


Figura 3.1 – Depósito deve abastecer demandas de clientes

A empresa dispõe de uma frota heterogênea de veículos, cada um com características próprias de capacidade e confiabilidade. Todos os veículos devem, necessariamente, partir do depósito central e retornar ao mesmo, após visitar as cidades previstas no roteiro. Ao utilizar um veículo, incorre-se em um custo fixo (referente ao aluguel do veículo e pagamento do motorista) e um custo que varia conforme a quantidade de km percorridos (referente ao consumo de combustível e à depreciação do veículo). Diante dessa situação, a empresa deseja encontrar a roteirização ótima de forma a minimizar seu custo total, porém, sem se submeter a riscos extremos de falha dos veículos ao longo da rota e conseqüente perda de credibilidade perante seus clientes.

Com o intuito de incluir a preocupação da empresa com a falha dos veículos, propõe-se uma nova abordagem do PRV que leve em consideração a confiabilidade dos veículos. Assim, além da restrição de capacidade dos veículos, adiciona-se uma restrição de confiabilidade mínima permitida ao final da rota de cada veículo. O problema de programação é escrito da seguinte forma:

Minimizar:

$$Custo - \sum_{i=0}^{k}\left[C1_k.\left(\sum_{j=0}^{n} x_{k0j}\right) + C2_k\left(.\sum_{i=0}^{n}\sum_{j=0}^{n} D_{ij}.x_{kij}\right)\right] \tag{3.1}$$

Sendo cada veículo *k* sujeito às seguintes restrições:

$$R_k\left(\sum_{i=0}^{n}\sum_{j=0}^{n} D_{ij}.x_{kij}\right) \geq R_0 \tag{3.2}$$

$$\sum_{i=0}^{n}\left[w_i.\left(\sum_{j=0}^{n} x_{kij}\right)\right] < W_k \tag{3.3}$$

Onde:

k - Número total de veículos disponíveis na frota
n - Número de cidades a serem visitadas
$C1_k$ - Custo de utilizar o veículo k
x_{kij} - Variável binária que indica se o veículo k percorre o trecho da cidade i até a cidade j
$C2_k$ - Custo/km do veículo k
D_{ij} - Distância entre as cidades i e j
$R_k(d)$ - Função de confiabilidade do veículo k (varia em função da distência d)
R_0 - Nível mínimo de confiabilidade permitida ao final da rota
w_i - Carga de mercadorias a serem entregues à cidade i
W_k - Capacidade máxima de carga do veículo k

Tem-se, assim, um modelo com $2k$ restrições e $(k.n!)/2.(n-2)!$ variáveis binárias (ver fórmula abaixo). Essas variáveis referem-se aos roteiros de cada veículo, indicando o número de veículos utilizados. Ou seja, o modelo deve otimizar não só os roteiros mas também o número de veículos utilizados.

$$\textit{Número de variáveis} = k\binom{n}{2} = \frac{k.n!}{2.(n-2)!} \tag{3.4}$$

O valor mínimo de confiabilidade deve ser definido pela empresa de acordo com sua política de gestão, uma vez que quanto maior for o limite de confiabilidade adotado (ou seja, quão mais protegida a empresa estiver em relação à falha de veículos), maior será o custo com a rota devido à necessidade de utilizar veículos mais novos ou de aumentar a frota de veículos. Por outro lado, o modelo também permite que a empresa modifique o limiar a depender da importância do cliente.

Vale ressaltar que neste trabalho assume-se que a função de confiabilidade é dada em função da distância percorrida. Isso significa que ao longo da rota tem-se o aumento da probabilidade de falha. A função de confiabilidade pode ser modelada, por exemplo, através de uma densidade paramétrica, tal como a Exponencial ou a *Weibull*, ou até mesmo por um modelo causal, tal como uma árvore de falhas (MODARRES; KAMINSKIY, 1999) ou redes Bayesianas (KORB; NICHOLSON, 2003).

O modelo aqui desenvolvido pode ser aplicado a uma grande variedade de situações reais, tornando-se importante principalmente para empresas que transportam produtos perecíveis ou que, de certa forma, são afetadas profundamente com a falha do veículo ao longo da rota. Um exemplo seria uma empresa que fabrica sorvetes e pretende vendê-los em diferentes mercados consumidores dispersos geograficamente.

3.3 Algoritmos Genéticos para o modelo proposto

Conforme exposto na seção 3.3, os problemas que dependem de ordenação e adjacência como é o caso do problema de roteirização, exigem uma codificação especial. Neste trabalho, utiliza-se a codificação de permutação (ou "*path representation*" (MICHALEWICZ, 1996) por ser mais intuitiva e difundida. Porém, para o modelo proposto anteriormente, é necessário realizar algumas adaptações à codificação por permutação, bem como aos operadores e métodos utilizados. Esta seção possui, então, o objetivo de explicar detalhadamente as modificações realizadas e os métodos implementados.

3.1.1 Cromossomo

O cromossomo típico do Problema do Caixeiro Viajante segundo a codificação por permutação pode ser visualizado na figura abaixo.

Figura 3.2 – Codificação por permutação para o PCV

Como se pode observar, este cromossomo indica a rota para apenas um veículo. O modelo desenvolvido, entretanto, consiste em roteirizar uma frota de veículos, sendo necessário realizar algumas modificações à codificação usual. Algumas propostas para tal seriam:

- **Codificação com dois cromossomos** – Nesta alternativa, cada indivíduo possui dois cromossomos, um que indica a ordem das cidades e outro que informa qual o veículo responsável pela entrega na cidade respectiva. Por exemplo, o indivíduo abaixo indica que o veículo 1 visita as cidade 4 – 9 – 5, ao passo que o veículo 2 visita 8 – 1 – 6 e finalmente o veículo 3 visita as cidades 2 – 3 – 7 (necessariamente nesta ordem). (MALMBORG, 1996)

Cromossomo de Cidades:

Figura 3.3 – Codificação com dois cromossomos

- **Codificação com único cromossomo –** Esta codificação apresenta números negativos para indicar quais cidades são visitadas por quais veículos, conforme se pode visualizar na figura abaixo. (TANG; RONG; YANG, 2000)

Figura 3.4 – Codificação com único cromossomo

- **Codificação com cromossomo de duas partes** – Similar à codificação anterior, esta alternativa possui apenas um cromossomo, porém este é dividido em duas partes. Na primeira parte têm-se as cidades ordenadas e na segunda parte o número de cidades a serem visitadas por cada veículo. Por exemplo, a figura abaixo significa que o primeiro veículo irá visitar as cidades 1 e 2, o segundo veículo irá para as cidades 3 – 4 – 5, e o terceiro veículo visitará as cidades 6 – 7 – 8.

Figura 3.5 – Codificação com cromossomo de duas partes

Segundo o trabalho de (CARTER; RAGSDALE, 2002) que comparou os três tipos de codificações, a codificação com cromossomos de duas partes é mais vantajosa, atingindo melhores resultados. Por conta disso, adotou-se esta codificação neste trabalho.

3.1.2 Inicialização da População

Neste trabalho foram implementados dois tipos de inicialização: inicialização aleatória e a inicialização o caminho mais curto. As seções seguintes irão descrever cada uma delas.

3.1.2.1 Inicialização aleatória

É a inicialização mais comum e trivial. Consiste em gerar aleatoriamente cada um dos bits do cromossomo. É importante ressaltar que, para o modelo aqui discutido, deve-se estar atento ao fato de que não é possível repetir números para a primeira parte do cromossomo. Além disso, o último bit da segunda parte do cromossomo não necessita ser sorteado uma vez que deve ser o complementar da soma de cidade já visitada pelos veículos anteriores.

3.1.2.2 Inicialização do caminho mais curto

Esta inicialização foi desenvolvida especialmente para o modelo proposto. Como o próprio nome diz, ela leva em consideração as distâncias entre as cidades na geração dos indivíduos. Ela segue, basicamente, os seguintes passos:

1 – Cálculo da matriz de vizinhança: A partir da matriz de distâncias, é construída uma matriz que informa para cada cidade quais são seus vizinhos mais próximos, em ordem decrescente.

2 – Geração da segunda parte do cromossomo: consiste na geração aleatória da segunda parte do cromossomo, que diz respeito ao número de cidades visitadas por cada veículo. Dessa forma, gera-se *n-1* números, sendo *n* o número de veículos (o ultimo veículo irá necessariamente visitar o complementar da soma de cidades visitadas pelos demais veículos da frota).

3 – Geração da primeira parte do cromossomo: com base na matriz de vizinhança e do número de cidades visitadas por cada veículo gera-se a primeira parte do cromossomo. Sabendo que todos os veículos partem necessariamente do depósito central, assume-se que as cidades mais próximas terão uma probabilidade maior de serem selecionadas, de acordo com uma função de densidade acumulada. Por exemplo, considerando a situação visualizada na Figura 3.6, a cidade um teria uma probabilidade de 5/12 de ser sorteada, ao passo que a cidade 2 teria ¼ e a cidade 3 teria uma chance de 1/3.

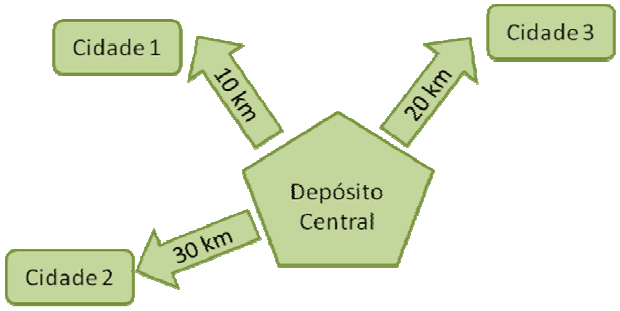


Figura 3.6 – Distâncias de clientes ao depósito central

3.1.2.3 Comparação dos métodos de inicialização

Com o intuito de comparar a eficiência dos dois métodos de inicialização apresentados, foram realizadas 50 simulações de um problema de teste. Os resultados estão expostos na Tabela 3.1.

Tabela 3.1 – Resultados de simulação com diferentes métodos de Inicialização

	Inicialização do caminho mais curto	**Inicialização Aleatória**
Tempo Médio	0.159s	0.149s
***Fitness* Médio**	144	149.6
Melhor *Fitness*	140	140
Pior *Fitness*	180	180
Desvio Padrão	12.122	16.777

Como pode-se observar, o método de inicialização do caminho mais curto possui um tempo de simulação um pouco mais elevado, porém encontra resultados melhores e menos dispersos. Assim, a decisão de qual método utilizar irá depender do problema e do contexto.

3.1.3 Operador de *crossover*

A partir da codificação com cromossomo de duas partes, foi necessário adaptar também os operadores de *crossover* e mutação. Inspirado nos modelos de *crossover* existentes na literatura utilizou-se uma abordagem mista de *crossover*, i.e., um *crossover* de codificação de permutação para a primeira parte e um *crossover* de codificação real para a segunda parte do cromossomo. Para a parte do cromossomo que indica a ordem das cidades, implementou-se o *Order Based Crossover* (OBX) e o *Position Based Crossover* (PBX), já explicados na seção 3.3. Já para a segunda parte do cromossomo que indica a quantidade de cidades a serem visitadas por cada caixeiro, utilizou-se o *crossover* BLX-α (ESHELMAN; SCHAFFER, 1993).

As figuras abaixo resumem então os dois tipos de *crossover* criados, a primeira baseada no OBX e a segunda no PBX.

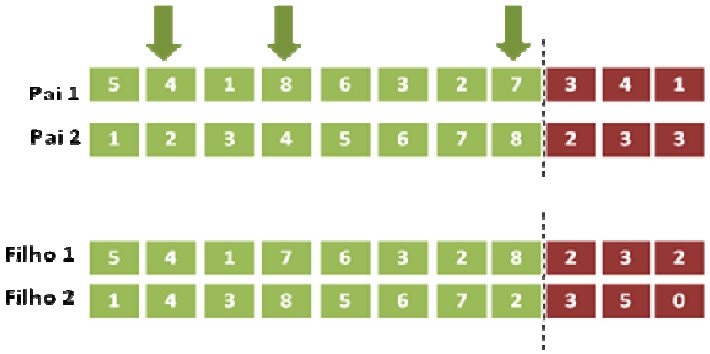


Figura 3.7 – Crossover OBX Adaptado

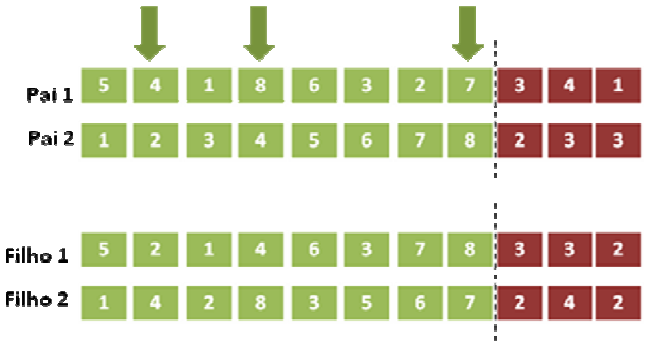


Figura 3.8 – Crossover PBX Adaptado

3.1.4 Operador de Mutação

Para o operador de mutação, também foi adotada uma estratégia mista. A primeira parte do cromossomo é mutada conforme a *Order Based Mutation* (OBM) ou a *Position Based Mutation* (PBM). Já para a segunda parte do cromossomo, implementou-se a mutação usada na codificação real, porém adicionando a restrição de que o número de total de cidades visitadas por todos os veículos não pode ultrapassar o número total de cidades disponíveis.

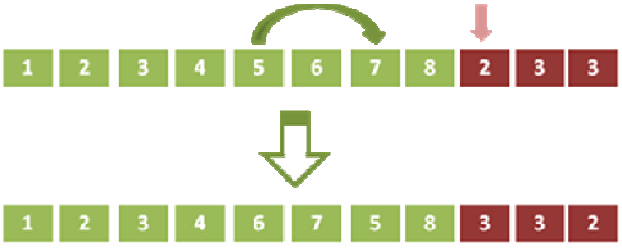


Figura 3.9 – Mutação PBM Adaptada

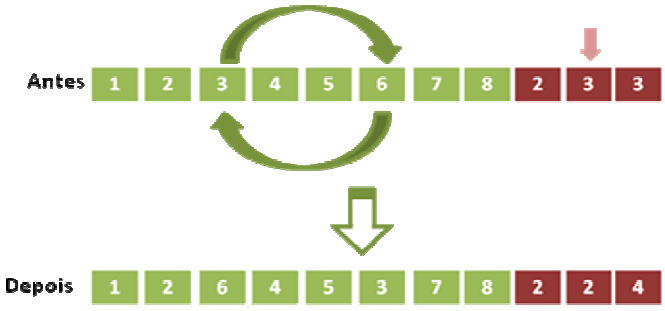


Figura 3.10 – Mutação OBM Adaptada

Apesar da mutação do AG tradicional modificar apenas um bit de forma independente, pode-se observar que devido à impossibilidade de números repetidos na codificação por permutação, a mutação de um bit exige a reordenação de todo o cromossomo. Com base nesse

argumento, criou-se neste trabalho uma mutação alternativa: a mutação de inicialização. Esta mutação consiste em simplesmente reinicializar o indivíduo novamente, de acordo com o método de inicialização adotado.

3.1.5 Seleção e Substituição

Neste trabalho, adotou-se o método de Torneio para a seleção dos indivíduos que irão sobreviver para as próximas gerações. Este método é bastante simples e consiste em selecionar aleatoriamente *k* indivíduos da população e comparar seus valores de *fitness*. Aquele que tiver o maior valor sobreviverá para próxima geração. Nesse trabalho, adotou-se *k=2*.

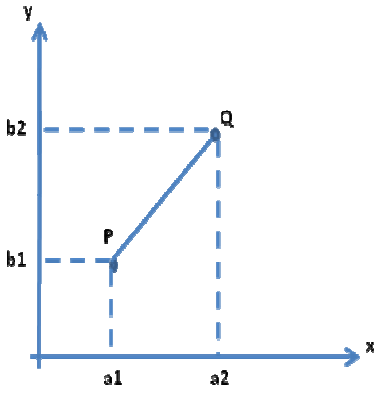
Quanto à estratégia de substituição, escolheu-se o método em que os pais cedem lugar aos filhos (*Children replace parents*).

3.1.6 Dados de entrada

De forma geral, os dados de entrada necessários são:

- **Matriz de distâncias**

Grande parte dos trabalhos envolvendo o PRV recebe como dado de entrada as coordenadas de cada cidade, calculando a distância entre elas através da equação de distância vetorial entre dois pontos.



$$D_{PQ} = \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2}$$

Figura 3.11 – Cálculo de distância vetorial

Para aplicações reais, entretanto, esta representação não é adequada uma vez que a real distância a ser percorrida entre duas cidades depende do sistema rodoviário existente. Assim, com o intuito de representar o problema de forma mais realística, optou-se por utilizar uma matriz de distâncias pré-definidas que correspondem a real distância a ser percorrida pelos veículos.

A matriz de distância consiste em uma matriz *(n+1)x(nx1)*, sendo *n* o número de cidades a serem visitadas. Vale ressaltar que para todos os problemas aplicados, utilizaram-se matrizes simétricas, porém isso não é uma obrigatoriedade do modelo.

$$Matriz\ de\ Distâncias = \begin{bmatrix} D_{01} & \cdots & D_{0n} \\ \vdots & \ddots & \vdots \\ D_{n0} & \cdots & D_{nn} \end{bmatrix} \quad (3.5)$$

- **Vetor de Cargas**

Corresponde a um vetor de tamanho *(n+1)* que indica, para cada cidade, a quantidade de carga que deve ser entregue. Como a cidade zero representa o depósito central, o primeiro valor do vetor é sempre zero.

$$Vetor\ de\ Cargas = [w_0 = 0 \quad w_1 \quad w_2 \quad \ldots \quad w_n] \quad (3.6)$$

- **Confiabilidade mínima**

Refere-se à confiabilidade mínima permitida ao final da rota. Mais do que um nível de confiabilidade, representa a sensibilidade da empresa ao risco.

- **Dados da frota**

Deve-se informar o número de veículos disponíveis para roteirização, a capacidade e função de confiabilidade de cada veículo.

- **Dados de Custo**

Para cada veículo, deve-se informar o custo de disponibilizá-lo para roteirização (constante *C1* do modelo) e o valor que ele gasta por *km* rodado (constante *C2* do modelo). *C1* representa um custo fixo que a empresa incorre ao utilizar o veículo, incluindo, por exemplo, os gastos com o pagamento do motorista, custos administrativos, etc. Já a segunda constante, representa um custo variável com a distância percorrida, associado ao gasto de combustível ao longo do trajeto.

4 EXEMPLOS DE APLICAÇÃO

Esta seção expõe alguns exemplos de aplicação. Os dois primeiros problemas foram aplicados apenas com o objetivo de validar o algoritmo. Logo, trata-se de exemplos fictícios, para os quais a solução ótima já é conhecida. Já os últimos dois exemplos de aplicação, são mais realísticos e contextualizados,

Todas as simulações foram rodadas em um computador com processador Intel Pentium Dual e 2.00 GB de memória RAM.

4.1 Problema de Validação I

Considere que uma empresa prestadora de serviços em logística deve entregar mercadorias em dez localidades dentro de um espaço geográfico finito. Cada uma dessas localidades tem sua posição bem definida, conforme a Figura 4.1. As únicas rodovias existentes são aquelas que aparecem como ligações na figura. Ou seja, para ir, por exemplo, da cidade do cliente 6 para a do cliente 8 tem-se que passar necessariamente pelo cliente 7.

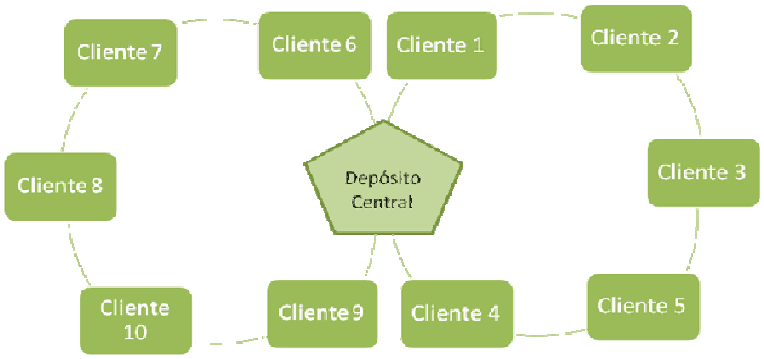


Figura 4.1 – Problema de validação I

Cada ligação é assumida ter uma distância de 1 km, como pode ser comprovado na matriz de distâncias da Tabela 4.1. A Tabela 4.2 mostra a quantidade de toneladas a serem entregues em cada cidade.

Considera-se que a empresa possui uma frota *homogênea* de dois veículos, cada um com capacidade para transportar até 10 toneladas e uma função de confiabilidade exponencial com taxa de falha de 0.09. A empresa definiu que, de acordo com sua política de gestão, a confiabilidade mínima permitida ao final do percurso deveria ser de 40%. Além disso, sabe-se que a cada *km* rodado incorre-se em um custo de *R$ 10,00* (constante *C2* do modelo) e o custo de utilizar um dos veículos para roteirização também é *R$ 10,00* (constante *C1* do modelo).

Tabela 4.1 – Matriz de distâncias do problema de validação I

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

Tabela 4.2 – Demanda de carga de cada cidade do problema de validação I

Cidade	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Carga (tons)	1	2	2	3	1	5	1	1	1	1

Solução ótima:

Ao analisar rapidamente o problema, percebe-se que a solução ótima seria utilizar dois veículos, um para visitar as cidades 1-2-3-5-4 e o outro para visitar as cidades 6-7-8-10-9, necessariamente nessa ordem. O custo mínimo seria, então, de *R$ 140,00*. Apesar da opção de utilizar um único veículo ser mais atrativa do ponto de vista financeiro, ela violaria a restrição de confiabilidade mínima, já que a confiabilidade atingiria um valor abaixo de 40%.

Apesar de simples, vale ressaltar, que esse problema possui 39 916 800 possíveis combinações, o que aliado ao fato de ser um problema *NP-difícil*, justifica a utilização de técnicas heurísticas para resolvê-lo.

$$\textit{Número de combinações} = 10! \times 11 = 39\,916\,800$$

Resultados:

- **Método Exaustivo**

Para comparar a eficiência do algoritmo, também foi implementado um método exaustivo que consiste em comparar todas as 39 916 800 possibilidades, indicando aquela com menor custo. O método exaustivo levou cerca de 1 hora e 13 minutos para encontrar a solução ótima, o que indica a importância da utilização do AG para resolver o modelo.

- **Algoritmos Genéticos**

Os parâmetros utilizados estão descritos na Tabela 4.3. Foi adotada a inicialização do caminho mais curto, método de *crossover* OBX e a mutação de inicialização.

Tabela 4.3 – Parâmetros de simulação para o problema de validação I

Tamanho da População	**400**
Número de Gerações	200
Probabilidade de *Crossover*	75%
Pontos de Corte do *Crossover*	1
Probabilidade de Mutação	5%

O algoritmo foi simulado 20 vezes, com o intuito de contabilizar o tempo médio, *fitness* médio e desvio padrão. A Tabela 4.4 mostra os resultados e, como se pode observar, a metodologia de otimização desenvolvida mostrou-se ser bastante eficaz, atingindo sempre a solução ótima. Além disso, a mesma é encontrada em apenas 2 segundos, ou seja, 0.046% do tempo de simulação do algoritmo exato.

Tabela 4.4 – Resultados de simulação para o problema de validação I

Tempo médio	2.093s
***Fitness* médio**	140
Desvio Padrão	0

4.2 Problema de Validação II

Este problema é apenas uma extensão do anterior, onde se aumentou o número de clientes para 16, como pode ser observado na Figura 4.2. A Tabela 4.5 expõe a matriz de distâncias.

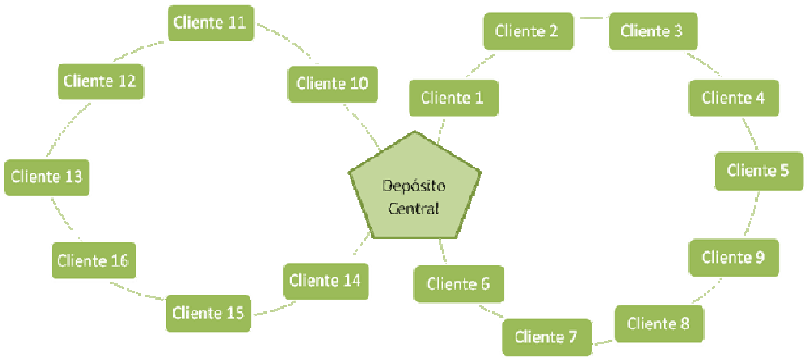


Figura 4.2 – Problema de validação II

Como se trata de um problema *NP-difícil*, a inclusão de 6 cidades aumenta muito a complexidade do problema, encontrando-se $3.56.10^{-14}$ possíveis combinações. Estima-se que o método exaustivo levaria mais de 1220 anos para avaliar todas essas possibilidades!

Número de combinações = 16! x 17 = 3.56 10 14

Tabela 4.5 – Matriz de distâncias do o problema de validação I

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	1	2	3	4	5	1	2	3	4	1	2	3	4	1	2	3
1	1	0	1	2	3	4	2	3	4	5	2	3	4	5	2	3	4
2	2	1	0	1	2	3	3	4	5	4	3	4	5	6	3	4	5
3	3	2	1	0	1	2	4	5	4	3	4	5	6	7	4	5	6
4	4	3	2	1	0	1	5	4	3	2	5	6	7	8	5	6	7
5	5	4	3	2	1	0	4	3	2	1	6	7	8	9	6	7	8
6	1	2	3	4	5	4	0	1	2	3	2	3	4	5	2	3	4
7	2	3	4	5	4	3	1	0	1	2	3	4	5	6	3	4	5
8	3	4	5	4	3	2	2	1	0	1	4	5	6	7	4	5	6
9	4	5	4	3	2	1	3	2	1	0	5	6	7	8	5	6	7
10	1	2	3	4	5	6	2	3	4	5	0	1	2	3	2	3	4
11	2	3	4	5	6	7	3	4	5	6	1	0	1	2	3	4	3
12	3	4	5	6	7	8	4	5	6	7	2	1	0	1	4	3	2
13	4	5	6	7	8	9	5	6	7	8	3	2	1	0	3	2	1
14	1	2	3	4	5	6	2	3	4	5	2	3	4	3	0	1	2
15	2	3	4	5	6	7	3	4	5	6	3	4	3	2	1	0	1
16	3	4	5	6	7	8	4	5	6	7	4	3	2	1	2	1	0

Assume-se que cada cidade tem uma demanda de uma tonelada e que a empresa possui uma frota *heterogênea* de dois veículos: um com capacidade de oito toneladas e outro com capacidade de 10 toneladas. Ambos possuem uma função de confiabilidade dada por uma *Weibull* com parâmetro de escala *70* e parâmetro de forma de *1.2*. As constantes *C1* e *C2* do modelo são, respectivamente, *R$ 400,00* e *R$ 18,00*.

Solução ótima:

A solução ótima seria, então, que o veículo de capacidade de *oito toneladas* visitasse as cidades 10-11-12-13-16-15-14 (ou 14-15-16-13-12-11-10) e que o veículo de capacidade de 10 toneladas visitasse as cidades 1-2-3-4-5-9-8-7-6 (ou 6-7-8-9-5-4-3-2-1). O custo ótimo é de *R$ 1124,00*.

Resultados:

- **Método Exaustivo**

Não foi possível resolver este problema através do método exaustivo, uma vez que ele levaria cerca de 1220 anos para comparar todas as possíveis soluções.

- **Algoritmos Genéticos**

Para este problema, utilizou-se os mesmos métodos e parâmetros do anterior, porém o tamanho da população foi de 5000 e foram realizadas 500 gerações. Em todas as 22 simulações do algoritmo foi encontrado o valor ótimo. A Tabela 4.6 expõe os resultados.

Tabela 4.6 – Resultados de simulação do o problema de validação II

Tempo médio	67.15s
***Fitness* médio**	1 124,00
Desvio Padrão	0

Sendo assim, comprova-se que, mesmo com o aumento do número de cidades, o algoritmo ainda se mostra eficiente, o que leva a utilizá-lo em outros exemplos de aplicação, conforme mostra as próximas seções.

4.3 Exemplo de Aplicação I

Considere que uma empresa de prestação de serviços em logística localizada em São Paulo deve entregar mercadorias em nove cidades: Guarulhos, Campinas, Sorocaba, Santos, São José dos Campos, Osasco, Santo André, Ribeirão Preto e São Bernardo do campo. A Figura 4.3 mostra a localização dessas cidades no estado de São Paulo.

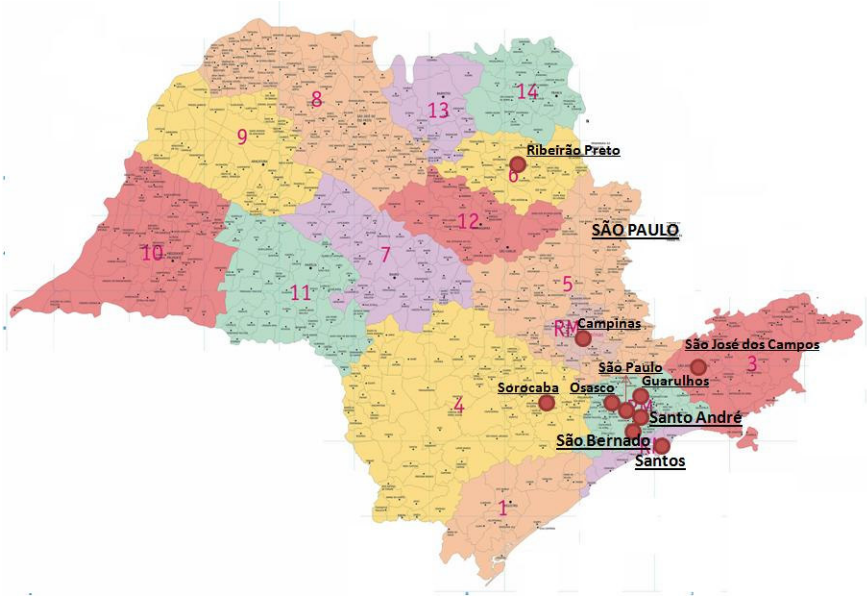


Figura 4.3 – Exemplo de Aplicação I

A empresa possui uma frota *heterogênea* de três veículos cujas funções de confiabilidade seguem uma *Weibull*. A Tabela 4.7 mostra os parâmetros de cada veículo da frota, enquanto que a Tabela 4.8 informa a carga que deve ser entregue em cada cidade.

Tabela 4.7 – Dados dos veículos do exemplo de aplicação I

	Capacidade	**Parâmetro de escala**	**Parâmetro de forma**	**C1**	**C2**
Veículo A	20	1500	1.5	350	1.4
Veículo B	50	1700	1.3	460	1.9
Veículo C	30	1600	1.4	410	2

Tabela 4.8 – Demanda de carga das cidades do exemplo de aplicação I

Número da Cidade	**Cidade**	**Carga (Tons)**
1	Guarulhos	9,6
2	Campinas	7,9
3	Sorocaba	4,3
4	Santos	3,1
5	São José dos Campos	4,6
6	Osasco	5,3
7	Santo André	5,0
8	Ribeirão Preto	4,2
9	São Bernardo do Campo	6,0

Resultados:

- **Método Exaustivo**

O método exaustivo levou cerca de 40 minutos para analisar todas as 19 958 400 possíveis soluções do problema. A solução ótima encontrada foi utilizar apenas o segundo veículo com a seguinte seqüência de cidades a serem visitadas: Santo André – Santos – São Bernado do Campo – Guarulhos – São José dos Campos – Campinas - Ribeirão Preto – Sorocaba - Osasco (como a matriz de distâncias utilizada é simétrica, o roteiro inverso também é válido).

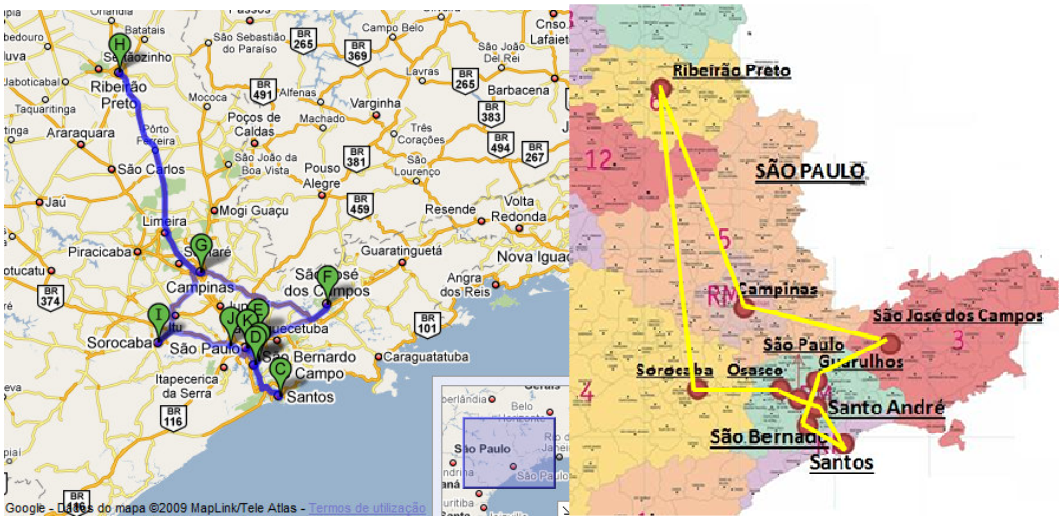


Figura 4.4 – Solução do exemplo de aplicação I

- **Algoritmos Genéticos**

Foram realizadas 10 simulações e os resultados estão dispostos na Tabela 4.9, enquanto que os parâmetros utilizados estão na Tabela 4.10.

Tabela 4.9 – Resultados de simulação do exemplo de aplicação I

Simulação	Melhor *Fitness* (R$)	Cromossomo	Tempo de Simulação
1	2 463.17	(1, 5, 8, 2, 3, 6, 9, 7, 4, 0, 9, 0)	26s
2	2 463.17	(1, 5, 8, 2, 3, 6, 9, 7, 4, 0, 9, 0)	27s
3	2 424.03	(7, 4, 9, 1, 5, 2, 8, 3, 6, 0, 9, 0)	26.5s
4	2 427.64	(1, 5, 2, 8, 3, 6, 9, 4, 7, 0, 9, 0)	29.5s
5	2 427.64	(1, 5, 2, 8, 3, 6, 9, 4, 7, 0, 9, 0)	30s
6	2 427.64	(7, 4, 9, 6, 3, 8, 2, 3, 1, 0, 9, 0)	28s
7	2 424.03	(7, 4, 9, 1, 5, 2, 8, 3, 6, 0, 9, 0)	29.5s
8	2 427.64	(7, 4, 9, 6, 3, 8, 2, 3, 1, 0, 9, 0)	28.5s
9	2 424.03	(7, 4, 9, 1, 5, 2, 8, 3, 6, 0, 9, 0)	30s
10	2 463.17	(1, 5, 8, 2, 3, 6, 9, 7, 4, 0, 9, 0)	31s

Tabela 4.10 – Parâmetros de simulação do exemplo de aplicação I

Tamanho da População	1000
Número de Gerações	1000
Probabilidade de *Crossover*	75%
Pontos de Corte do *Crossover*	1
Probabilidade de Mutação	5%

Como pode ser observado, o algoritmo atingiu a solução ótima em apenas três das 10 simulações. O *fitness* médio encontrado foi de *R$ 2 437.216* e o tempo médio de simulação foi de *28.6* segundos. Apesar de ser bem mais rápido que o método exaustivo, o método deixa a desejar em relação à eficácia dos resultados encontrados. Acredita-se que isso acontece **principalmente** devido ao aumento do número de veículos na solução. Essa conclusão foi constatada através da

comparação com primeiro exemplo de aplicação: tinha-se cerca de *40 milhões* de possíveis soluções com dois veículos na frota e o algoritmo se mostrou bastante eficiente. Para este exemplo de aplicação, entretanto, tem-se um número menor de combinações (menos de *20 milhões*), porém disponibilizam-se três veículos para roteirização. Ou seja, suspeita-se que o algoritmo genético desenvolvido encontra uma maior dificuldade em otimizar o número de veículos a serem utilizados (e suas respectivas alocações aos roteiros) que encontrar os roteiros mais curtos. A comprovação e melhor investigação dessa questão foram sugeridas como trabalho futuro.

Por outro lado, também seria interessante aumentar o tamanho da população e/ou o número de gerações para permitir que o algoritmo encontre melhores soluções. Entretanto, isso não foi possível devido à exigência de memória RAM adicional no computador. Apesar de saber que essa questão poderia ser resolvida através de um melhor gerenciamento da memória no código de programação, devido à limitação de tempo, esta medida não foi contemplada no presente trabalho, deixando a recomendação para trabalhos futuros (conforme discute o item 7.2).

4.3 Exemplo de Aplicação 2

Considere a mesma empresa do exemplo anterior, porém agora a empresa deve abastecer 16 cidades, conforme o mapa da Figura 4.5. A

Tabela 4.11 mostra para cada cidade a quantidade de carga que deve ser entregue.

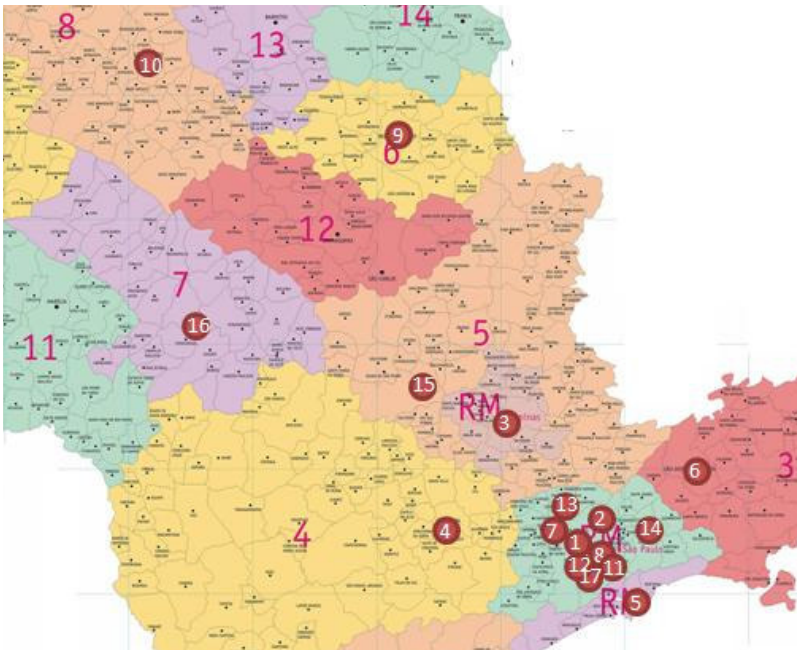


Figura 4.5 – Exemplo de aplicação II

Tabela 4.11 – Demanda de carga das cidades do exemplo de aplicação II

Número da Cidade	Cidade	Carga (Tons)
1	São Paulo (Origem)	-
2	Guarulhos	6,8
3	Campinas	5,6
4	Sorocaba	3,1
5	Santos	2,2
6	São José dos Campos	3,2
7	osasco	3,8
8	Santo André	3,6
9	Ribeirão Preto	3,0
10	São José do Rio Preto	2,2
11	Mauá	2,2
12	Diadema	2,1
13	Carapicuíba	2,1
14	Mogi das Cruzes	2,0
15	Piracicaba	1,9
16	Bauru	1,9
17	São Bernado do Campo	4,3

A empresa possui quatro veículos para realizar as entregas, cada um com função confiabilidade modela por uma *Weibull*. Os parâmetros de cada veículo são expostos na Tabela 4.12. De acordo com os as prioridades da empresa, o nível mínimo de confiabilidade permitido ao final da rota é de 55%.

Tabela 4.12 – Dados dos veículos do exemplo de aplicação II

	Capacidade	Parâmetro de escala	Parâmetro de forma	C1	C2
Veículo A	15	1500	1.5	350	1.4
Veículo B	30	1700	1.3	460	1.9
Veículo C	30	1600	1.4	410	2.0
Veículo D	15	1500	1.6	310	1.5

Resultados:

- **Método Exaustivo**

Não foi possível resolver esse exemplo de aplicação através do método exaustivo, já que ele levaria mais de 70 mil anos para comparar todas as possíveis soluções.

- **Algoritmos Genéticos**

Para este problema, utilizou-se a mutação de inicialização e o *crossover* OBX com um único ponto de corte. Os valores dos outros parâmetros utilizados estão expostos na Tabela 4.13.

Tabela 4.13 – Parâmetros de simulação do exemplo de aplicação II

Tamanho da População	1000
Número de Gerações	1000
Probabilidade de *Crossover*	75%
Pontos de Corte do *Crossover*	1
Probabilidade de Mutação	5%

O tempo médio de simulação foi de 71 segundos e o melhor *fitness* encontrado foi de R$ 3 915,72 com a seguinte combinação (ver Figura 4.6):

- Veículo A (rota verde): Santos – Mogi das Cruzes – São José dos Campos – Piracicaba – Sorocaba – Carapicuíba. Distância percorrida: *682.6 km*
- Veículo B (rota azul): Guarulhos – Mauá – Santo André – São Bermardo do Campo – Diadema – Osasco. Distância percorrida: *135.2 km*
- Veículo C: Não deve ser utilizado
- Veículo D (rota vermelha): Campinas – Ribeirão Preto – São José do Rio Preto – Bauru. Distância percorrida: *1082.1*

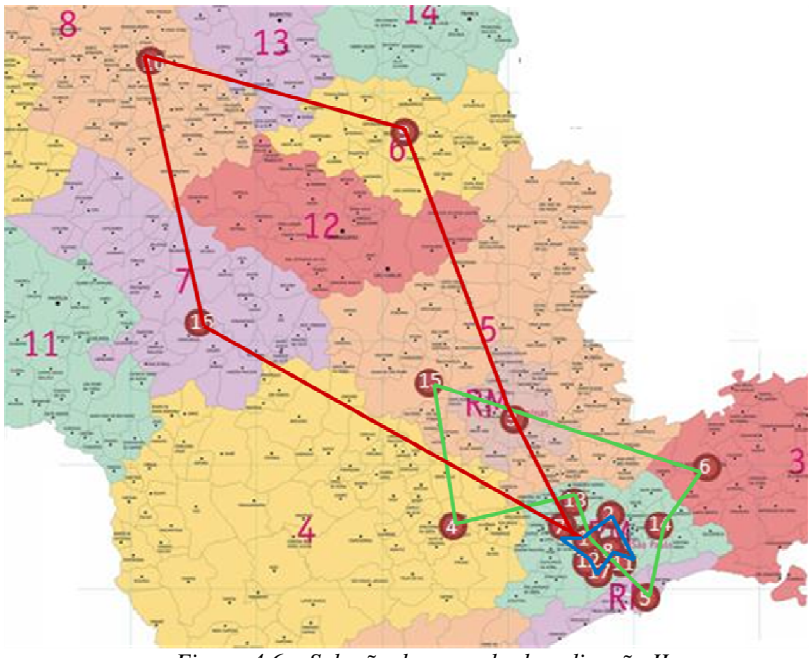


Figura 4.6 – Solução do exemplo de aplicação II

Apesar de não se poder garantir que esta é a melhor combinação possível, observa-se que, de fato, se trata de uma boa alternativa. Inicialmente, poder-se-ia argumentar que o veículo A deveria percorrer a maior distância, já que ele possui o menor custo por *km*. Porém, por outro lado, ele possui uma taxa de falha relativamente intensa. Logo, ao percorrer grandes distâncias incorre-se em uma alta probabilidade de falha desse veículo e provavelmente a restrição de confiabilidade do modelo não seria respeitada. Percebe-se também que o veículo C não foi utilizado e uma das razões é devido ao fato que este veículo possui o maior custo por *km* rodado. O veículo B também possui um custo elevado e por isso que na solução final ele visita apenas as cidades mais próximas à origem, percorrendo menos de *136 km*. Já o veículo D possui um custo intermediário e uma taxa de falha não tão intensa, o que justifica sua alocação às cidades mais distantes da origem. Assim, de uma forma geral, pode-se verificar que esta seria uma boa solução a ser adotada tendo em vista à complexidade do problema e a necessidade de rapidez nas decisões.

5 CONSIDERAÇÕES FINAIS

Este trabalho buscou desenvolver um modelo de roteirização capaz de levar em consideração a confiabilidade dos veículos. A principal motivação do trabalho surgiu a partir da constatação que a falha dos veículos ao longo da rota acarreta sérias conseqüências à imagem e credibilidade das empresas. Ressalta-se ainda o fato da maioria dos trabalhos da literatura considerar que os veículos são 100% confiáveis, quando se sabe que todos são passíveis a falha, possuindo uma taxa de falha associada.

Para resolver o modelo proposto, foi desenvolvida uma metodologia de otimização baseada em AG. Utilizou-se uma codificação por permutação e foram realizadas algumas adaptações aos operadores e métodos. De forma geral, o trabalho contribuiu para a criação de um novo método de inicialização da população que leva em consideração a matriz de distâncias do problema. Os resultados demonstraram a importância desse método, principalmente para problemas onde o número de cidades é elevado. Além disso, para os operadores genéticos de mutação e *crossover*, foram criadas estratégias mistas de codificação real e codificação por permutação, baseados em conceitos e métodos reconhecidos na literatura.

Através dessa metodologia, foram resolvidos quatro exemplos de aplicação, sendo dois para validação do algoritmo e dois exemplos aplicados à realidade de empresas logísticas no estado de São Paulo. Os resultados mostraram a eficiência da técnica, bem como sua importância frente a problemas de alta complexidade.

Dessa forma, a elaboração desse trabalho possibilitou um conhecimento mais aprofundado dos temas abordados, quais sejam problema de roteirização de veículos, confiabilidade ou otimização via AG. Além disso, ressalta-se a conhecimento adquirido na liguagem de programação C++, utilizada para implementar o modelo de otimização.

5.1 Limitações

Apesar de incluir a confiabilidade dos veículos da frota, o modelo aqui desenvolvido não considera outros aspectos importantes da realidade dos problemas de roteirização, como, por exemplo, as restrições de janelas de tempo.

Quanto ao modelo de otimização, por se tratar de um método heurístico, ressalta-se a limitação de não garantia da solução ótima. Assim, por mais eficiente que seja a técnica, não se

pode ter certeza de que a solução encontrada seja, de fato, a solução ideal. Por outro lado, identificou-se o fato de que as estratégias de *crossover* e mutação criadas atuam de forma independente para a primeira e segunda parte do cromossomo, o que possivelmente prejudica a eficiência desses operadores genéticos.

Além disso, de um ponto de vista operacional, verificou-se certa limitação no gerenciamento de memória computacional do algoritmo implementado, impossibilitando a simulação com um número elevado de indivíduos e/ou gerações. Para problemas mais complexos, como é o caso do último exemplo de aplicação do capítulo 6, isso se torna um aspecto bastante negativo, já que dificilmente o algoritmo irá conseguir atingir bons resultados.

5.2 Sugestões para trabalhos futuros

Algumas sugestões para trabalhos futuros são:

- Incorporar restrições de janela de tempo ao modelo de roteirização proposto;
- Desenvolver um modelo de otimização híbrido entre Algoritmos Genéticos e métodos de busca local para resolver o problema de roteirização de forma mais eficiente;
- Solucionar o problema de gerenciamento de memória que impede a simulação do algoritmo com um número elevado de indivíduos e/ou gerações;
- Implementar a possibilidade de utilizar populações independentes em algoritmos genéticos;
- Intensificar os testes de validação do algoritmo, investigando qual seria a melhor combinação de valores para os parâmetros do modelo e métodos utilizados;
- Desenvolver novas técnicas de *crossover* e mutação que atuem no cromossomo como um todo, sem dividi-lo em primeira e segunda parte;
- Investigar o impacto do aumento do número de veículos da frota na eficiência do algoritmo genético e desenvolver soluções para tal;
- Incorporar idéias de otimização geográfica de outros algoritmos determinísticos existentes na literatura

- Utilizar técnicas como árvores de falha e/ou redes bayseanas para modelar a confiabilidade dos veículos
- Construir um modelo de otimização multiobjetivo, onde se deve minimizar o custo ao mesmo tempo em que se deseja maximizar a confiabilidade dos veículos ao final da rota.

REFERÊNCIAS

ANDRADE, E. L. **Introdução à Pesquisa Operacional – Métodos e modelos para análise de decisão**. 3 ed. [S.l.]: LTC Editora, 2004.

AZADIVAR, F.; SHU, J. V. Use of simulation in optimization of maintenance policies. **Winter Simulation Conference**, p.1061-1067, 1998.

AUSIELLO, G.; BONIFACI, V.; LAURA, L. The online prize-collecting traveling salesman problem. **Information Processing Letter**, doi: 10.1016/j.ipl.2008.03.002, 2008.

BALLOU, R. H. **Business Logistics Management**, 4 ed, New Jersey: Prentice Hall, 1998.

BAR-YEHUDA, E. G.; SHAHAR, S. On approximating a geometric prize-collecting traveling salesman problem with time windows. **Journal of Algorithms**, v. 55, 1, p. 76-92, 2005.

BELFIORE, P.; COSTA, O. L. V.; FÁVERO, L. P. L. Problema de Estoque e Roteirização com Demanda Determinística e Estocástica: Revisão da Literatura. *In*: XII SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO, 2005, Bauru. **Anais …**, 2005.

BEKTAS, T. The multiple traveling salesman problem: an overview of formulations and solution procedures. **Omega – The international Journal of Management Science**, v. 34, p. 209 – 219, 2006.

BIENSTOCK, D. *et. al.* A note on the prize collecting traveling salesman problem. **Mathematical Programming**, v. 59, p. 413-420, 1993.

BOWERSOX, D. J.; CLOSS, D. J.; COOPER, M. B. **Supply chain logistics management**. 1 ed. vol. 1. New York: McGraw-Hill, 2002.

CARTER, A. E.; RAGSDALE, C. T. Scheduling pre-printed newspaper advertising inserts using genetic algorithms. **Omega**, v. 30, p.415-421, 2002.

CERQUEIRA, F. R.; CRAVO, G. L. Metaheurística colônia de formigas aplicada ao problema do caixeiro viajante. **Educação e Tecnologia**, v. 2, 2006.

CERQUEIRA, F. R. S. Genetic algorithms applied to DNA physical mapping problem. *In*: IV ESCOLA REGIONAL DE INFORMÁTICA DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2004, Vitória-ES. **Anais ...**, 2004.

CERVO, A. L.; BERVIAN, P. A.; SILVA, R. **Metodologia Científica**. 6 ed. São Paulo: Pearson Prentice Hall, 2007.

CHATTERJEE, S.; CARRERA, C.; LYNCH, L. A. Genetic Algortithm and Traveling Salesman Problem. **European Journal of Operational Research**, v. 93, p. 490-510, 1996.

CHAVES, A. A. **Modelagens Exata e Heurística para Resolução do Problema do Caixeiro Viajante com Coleta de Prêmios**. 2003. Grau Monografia (Graduação em Ciência da Computação) – Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, Ouro Preto, 2003.

CHRISTOFIDES, N.; EILON, S. Algorithms for Large Scale Traveling Salesman Problem. **Operational Research Quarterly**, v. 23, p. 511-518, 1972.

CHRISTOFIDES, N. **Worst-case analysis of a new heuristic for the traveling salesman problem**. 1976. Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management Science**, v. 6, p. 80-91, 1959.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: A Cooperative Learning Approach to the Travelling Salesman Problem. **IEEE Transactions on Evolutionary Computation**, v. 1, p. 53-66, 1997.

EIBEN, A. E.; SMITH, J. E. **Introduction to evolutionary computing**. [S.l.]: Springer, Verlag Berlin Heidelberg, 2003.

EKSIOGLU, B.; VURAL, A. V.; REISMAN, A. The vehicle routing problem: A taxonomic review. **Computers & Industrial Engineering**, v.57, p. 1472-1483, 2009.

ESHELMAN, L. J.; SCHAFFER, J. D. Real-coded Genetics Algorithms and Interval-Schemata. **Foundations of Genetic Algorithms**, v.2, p.187-202, 1993.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização Combinatória e Programação Linear**. 2. ed. [S.l]: Elsevier Editora, 2005.

GOLDBERG, D. E.; LINGLE, R. Alleles, Loci, and Traveling Salesman Problem. **Proceedings of an International Conference on Genetic Algorithms and their Applications**, p. 154-159, 1985.

KENNEDY, J., ERBERHART, R. C. **Swarm intelligence**. San Francisco: Morgan Kaufmann, 2001.

KORB, K. B.; NICHOLSON, A. E. **Bayesian artificial intelligence**. Florida: Chapman & Hall/CRC, 2003

LAPA, C. M. F., PEREIRA, C. M. N. A.; MOL, A. C. A. Maximization of a nuclear system availability through maintenance scheduling optimization using a genetic algorithm. **Nuclear Engineering and Design**, v. 196, p.219–231, 2000.

LAPA, C. M. F., PEREIRA, C. M. N. A.; BARROS, M. P. A model for preventive maintenance planning by genetic algorithms based in cost and reliability. **Reliability Engineering and System Safety**, v.91, p.233–240, 2006.

LAPORTE, G.; NOBERT, Y. **Exact Algorithms for the Traveling Salesman Problem**. Université du Québec, 1973.

LI, Y.; GONG, S. Dynamic ant Colony Optimisation for TSP. **The International Journal of Advanced Manufacturing Technology**, v. 22, p. 528-533, 2003.

MALMBORG, C. A genetic algorithm for service level based vehicle scheduling. **Europeen Journal of Operations Research**, v. 93, p. 121-134, 1996.

MARCOTTE, P.; SAVARD, G.; SEMET, F. A Bilevel Programming Approach to the Traveling Salesman Problem. **Operation Research Letters,** v. 32, p. 240-248, 2004.

MARSEGUERRA, M. E., ZIO, E. & MARTORELL, S. Basics of genetic algorithms optimization for RAMS applications. **Reliability Engineering and System Safety**, v. 91, p. 977–991, 2006.

MARTORELL, S., CARLOS, S., SANCHEZ, A.; SERRADELL, V. Constrained optimization of test intervals using a steady-state genetic algorithm. **Reliability Engineering and System Safety**, v.67, p. 215–232, 2000.

MICHALEWICZ, Z. **Genetic algorithms + data structures = evolution programs**. Berlin: Springer, 1996.

MILLER, D.; PEKNY, J. Exact Solution of Large Asymmetric Traveling Salesman Problems. **Science**, v. 251, p. 754-761, 1991.

MITROVIC-MINIC, S.; KRISHNAMURTI, R. The multiple TSP with time windows: vehicle bounds based on precedence graphs. **Operations Research Letters**, v. 34, p. 111 – 120, 2006.

MODARRES, M. KAMINSKLY, M. **Reliability Engineering and Risk Analysis: A Practical Guide**. New York: Marcel Dekker, 1999.

NETO, R. F. T. **Planejamento de vistorias usando robôs móveis autônomos e otimização pelo algoritmo de colônia de formigas.** 2005. Programa de Pós-Graduação em Engenharia de Produção e Sistemas - Pontifícia Universidade Católica do Paraná, Curitiba, 2005.

NETO, A. F.; LIMA, R. S. Roteirização de veículos de uma rede atacadista com o auxílio de Sistemas de Informações Geográficas (SIG). In: XXV ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, Porto Alegre, RS, 2005. **Anais...** , 2005.

POTVIN, J. Y. Genetic Algorithms for the Traveling Salesman Problem. **Annals of Operations Research**, v. 63, p. 339–370, 1996.

POTVIN, J. V. The Traveling Salesman Problem: A Neural Network Perspective. **INFORMS Journal on Computing**, v. 5, p. 328-348, 1993.

RIGDON, S. E.; BASU, P. A. **Statistical Methods for the Reliability of Repairable Systems.** New York: John Wiley & Sons, 2000.

SILVA, A. F.; OLIVEIRA, A. C. Algoritmos genéticos: alguns experimentos com os operadores de cruzamento ("*Crossover*") para o problema do caixeiro viajante assimétrico. In: XXVI ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, Fortaleza, CE, 2006.

SLACK, N.: CHAMBERS, S.; JOHNSTON, R. **Administração da produção.** 2 ed. São Paulo: Editora Atlas, 2002.

TANG, L.; LIU, J.; RONG A.; YANG, Z. A multiple traveling salesman problem model for hot rolling schedule in Shanghai Baoshan Iron & Steel Complex. **Europeen Journal of Operations Research,** v.124, p. 267-282, 2000.

XU, X.; TSAI, W. T. Effective Neural Algorithms for the Traveling Salesman Problem. **Neural Networks**, v. 4, p. 193–205, 1991.

ZITZLER, E. **Evolutionary algorithms for multiobjective optimization: methods and applications**. Tese de Doutorado. Swiss Federal Intitute of Technology, Zurich, 1999.