

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Nathalia Souza Carvalho Garcia

Victor Hugo Prates Magalhães

**Desenvolvimento de um Sistema de Roteirização de Veículos Baseado em uma Meta-
Heurística de Algoritmo Genético: Um caso Envolvendo Frotas Heterogêneas e Janelas de
Tempo**

ITUIUTABA

2021

Nathalia Souza Carvalho Garcia

Victor Hugo Prates Magalhães

Desenvolvimento de um Sistema de Roteirização de Veículos Baseado em uma Meta-Heurística de Algoritmo Genético: Um caso Envolvendo Frotas Heterogêneas e Janelas de Tempo

Trabalho de Conclusão de Curso apresentado a Universidade Federal de Uberlândia, como parte das exigências para obtenção de título de Bacharel em Engenharia de Produção.

Orientador: Jorge von Atzingen dos Reis

ITUIUTABA

2021

Desenvolvimento de um Sistema de Roteirização de Veículos Baseado em uma Meta-Heurística de Algoritmo Genético: Um caso Envolvendo Frotas Heterogêneas e Janelas de Tempo

Trabalho de Conclusão de Curso aprovado para conclusão da graduação em Engenharia de Produção na Universidade Federal de Uberlândia, pela banca examinadora formada por:

Ituiutaba, 20 de julho de 2021.

Banca Examinadora:

Prof. Dr. Eng. Jorge von Atzingen dos Reis (orientador), Universidade Federal de Uberlândia

Prof. Dr. Eng. Eugênio Pacceli Costa, Universidade Federal de Uberlândia

Prof. Dr. Eng. Hebert Roberto da Silva, Universidade Federal de Uberlândia

“Dedico este trabalho primeiramente a Deus, por me fazer confiar em qual direção devo seguir na minha vida, aos meus pais Manoel Garcia da Costa Júnior e Andréa Souza C. Garcia que me apoiaram, asseguraram e acreditaram em mim durante toda a jornada na Universidade, a minha irmã gêmea Daniela Garcia que transmitiu toda energia e força para seguir em frente e aos meus irmãos Amanda Garcia, Mariana Garcia, Mateus Garcia que sempre estiveram ao meu lado me incentivando.

“Dedico também aos meus amigos que se fizeram presente nesta jornada em especial minha amiga Elaine Patussi. Ao meu namorado João Paulo Teixeira pelo seu incentivo, apoio e admiração. Ao corpo docente da Engenharia de Produção que me orientou e proporcionou um ensino de excelência, de maneira profissional.”

“Gostaria de agradecer ao meu colega Victor Hugo P. Magalhães que contribuiu fortemente a realização desse estudo, pela sua parceria e confiança, no qual se fez presente em todos instantes, se dedicando com muita responsabilidade e perseverança.”

Autor: Nathalia Souza Carvalho Garcia

“Dedico este trabalho primeiramente a Deus, aos meus pais Alexander e Herlene e a minha irmã Walleska que sempre me apoiaram e acreditaram em mim. A minha namorada Amanda que me acompanhou nos momentos mais difíceis desta jornada. A D.^a Jucélia por ter me acolhido como um filho e por todos os seus sábios conselhos. Dedico a todos os meus amigos que fizeram desta trajetória um aprendizado muito especial.”

“Gostaria também de agradecer a minha colega Nathalia que desde o início acreditou no potencial deste projeto e buscou de todas as formas vencer os obstáculos que se apresentaram no nosso caminho. Meu muito obrigado pela dedicação e perseverança.”

Autor: Victor Hugo Prates Magalhães

AGRADECIMENTOS

Agradecemos a Universidade Federal de Uberlândia, ao Prof. Dr. Eng. Jorge Von Atzingen dos Reis, que do início ao fim nos orientou com muita confiança e disposição. Somo muito gratos também à empresa 3 Maria's que nos proporcionou as condições necessárias para os teste e execução do estudo proposto. Agradecemos ao corpo docente da Universidade Federal de Uberlândia, Campus Pontal, e em especial aos Profs. Drs. Eugênio Pacceli Costa e Hebert Roberto da Silva que farão parte da banca examinadora.

RESUMO

A otimização das operações logísticas das organizações é de grande importância para manter o nível de serviço e os custos equilibrados. Diante disto, os sistemas de roteirização de veículos se tornam ferramentas extremamente eficientes para se alcançar tais níveis. Isto significa possuir um planejamento ágil que minimize os custos dentro das condicionantes operacionais ao mesmo tempo em que se atenda às exigências dos clientes. Este trabalho apresenta um problema de roteirização complexo, com frota heterogênea e janelas de tempo. Assim sendo, propôs-se um modelo de otimização baseado em um Algoritmo Genético. Este algoritmo caracteriza-se por avaliar grupos populacionais de soluções seguindo um processo estocástico iterativo evolutivo. Tal algoritmo pode apresentar comportamentos de otimização tanto de ótimos locais quanto globais. Os dados das localizações geográficas e dos percursos entre clientes foram obtidos por meio do uso de *Application Programming Interface (API's)* gratuitas disponibilizadas pelo *Google Maps*®. A empresa estudada é uma produtora e fornecedora de produtos alimentícios congelados que atende mais de 200 clientes na região do triângulo mineiro e sul de Goiás. Dentro da proposta do estudo, as otimizações conseguiram uma redução de 1.900 km mensais, o que corresponde a uma redução de distância percorrida de aproximadamente 13,50%. Estes valores representam cerca de R\$ 18.300,00 de custo anuais. Portanto, as principais vantagens para a empresa foram a redução de custo com combustíveis e menor tempo de mão de obra dos motoristas.

Palavras-Chave: Algoritmo Genético; Problema de Roteamento de Veículos com Janelas de Tempo; Meta-heurística.

ABSTRACT

Optimizing the logistical operations of organizations is of great importance to keep the service level and costs balanced. Therefore, vehicle routing systems become extremely efficient tools to reach such levels. This means having an agile planning that minimizes costs within the operational conditions while meeting customer requirements. This work presents a complex routing problem, with heterogeneous fleet and time windows. Therefore, an optimization model based on a Genetic Algorithm was proposed. This algorithm is characterized by evaluating population groups of solutions following an evolutionary iterative stochastic process. Such an algorithm can show both local and global optimal optimization behaviors. Data on geographic locations and routes between customers were obtained through the use of free APIs provided by Google Maps©. The company studied is a producer and supplier of frozen food products that serves more than 200 customers in the region of the Minas Gerais and southern Goiás triangle. Within the study proposal, the optimizations achieved a reduction of 1,900 km per month, which corresponds to a reduction of mileage driven of approximately 13.50%. These values represent approximately R\$ 18,300.00 in annual costs. Therefore, the main advantages for the company were the reduction in fuel costs and shorter labor time for drivers.

Keywords: Genetic Algorithm; Vehicle Routing Problem with Time Windows; Metaheuristics.

LISTA DE FIGURAS

Figura 3.1 – Procedimento algoritmo genético	17
Figura 3.2 – Fluxograma genérico do Algoritmo Genético.....	18
Figura 3.3 – Representação dos indivíduos	19
Figura 3.4 – Procedimento criação aleatória	21
Figura 3.5 – Procedimento <i>ERX</i>	26
Figura 3.6 – Início do procedimento <i>ERX</i>	27
Figura 3.7 – 1ª iteração do procedimento <i>ERX</i>	28
Figura 3.8 – 2ª iteração do procedimento <i>ERX</i>	29
Figura 3.9 – Cromossomo final	30
Figura 3.10 – Procedimento <i>CRX01</i>	31
Figura 3.11 – <i>VC</i> e <i>VR</i> de pai 01 e pai 02.....	32
Figura 3.12 – <i>VR</i> <i>CRX01</i> resultante.....	32
Figura 3.13 – Procedimento <i>CRX02</i>	33
Figura 3.14 – <i>LV</i> e <i>LR</i> de pai 01 e pai 02	34
Figura 3.15 – <i>VR</i> <i>CRX02</i> resultante.....	34
Figura 3.16 – Procedimento <i>MXVC</i>	36
Figura 3.17 – Mutação do <i>VC</i>	38
Figura 3.18 – Mutação <i>MXVR01</i>	39
Figura 3.19 – Mutação <i>MXVR02</i>	39
Figura 3.20 – Mutação <i>MXVR03</i>	39
Figura 3.21 – Mutação <i>MXVR04</i>	40
Figura 3.22 Matriz distância de clientes	43
Figura 4.1 – Mapa de execução da instância 23x1	46
Figura 4.2 - Mapa da instância 23x1 otimizada.....	46

LISTA DE GRÁFICOS

Gráfico 1.1 - Consumo de alimentos processados.....	2
Gráfico 4.1 – Comparação das instâncias otimizadas	45
Gráfico 4.2 – Desempenhos do algoritmo	48
Gráfico 4.3 – Consumo de memória e tempo de processamento do algoritmo.....	49

LISTA DE QUADROS

Quadro 2.1 – Característica do PRVJT estudado	8
Quadro 2.2 – Etapas de um Algoritmo Genético.....	14
Quadro 3.1 – Premissas do PRVJT abordado	22

LISTA DE TABELAS

Tabela 3.1 – Endereços de clientes.....	42
Tabela 3.2 – Coordenadas e endereços de clientes.....	42
Tabela 3.3 – Dados de distâncias e tempos de viagens	43
Tabela 4.1 – Utilização dos veículos	47
Tabela 4.2 – Variabilidade do algoritmo	50
Tabela 4.3 – Horários de atendimento dos clientes	50

LISTA DE SIGLAS E ABREVIATURAS

AG	Algoritmo Genético
API	<i>Application Programming Interface</i>
CM	Coordenada de Mutações
PO	Pesquisa Operacional
PL	Programação Linear
FO	Função Objetivo
VR	Vetor Rotas
LR	Lista de Rotas
VC	Vetor Clientes
LC	Lista de Clientes
VRCRX01	Vetor Rotas do operador CRX01
VRCRX02	Vetor Rotas do operador CRX02
QTM	Problema do Caixeiro Viajante
QTG	Quantidade Total de Mutações
QTC	Quantidade Total de Genes do Vetor Clientes
PRV	Quantidade Total de <i>Crossover</i>
PRV	Problema de Roteirização de Veículos

SUMÁRIO

1	INTRODUÇÃO.....	1
1.1	OBJETIVO GERAL.....	3
1.2	PROCEDIMENTOS METODOLÓGICOS	3
1.3	DELIMITAÇÃO DO TRABALHO.....	4
1.4	ESTRUTURA DO TRABALHO	4
2	FUNDAMENTAÇÃO TEÓRICA	6
2.1	PROBLEMAS DE ROTEAMENTO DE VEÍCULOS COM JANELAS DE TEMPO.....	7
2.2	PROGRAMAÇÃO LINEAR	9
2.3	MÉTODOS HEURÍSTICOS.....	12
2.3.1	META-HEURÍSTICA BASEADA EM ALGORITMO GENÉTICO.....	13
3	MÉTODOS DE PESQUISA	16
3.1	IMPLEMENTAÇÃO DA META-HEURÍSTICA	17
3.2	GERAÇÃO DA POPULAÇÃO INICIAL	20
3.3	FUNÇÃO OBJETIVO	21
3.4	BUSCA LOCAL	25
3.5	OPERADOR CROSSOVER.....	25
3.5.1	CROSSOVER DO VETOR CLIENTES	25
3.5.2	CROSSOVER DO VETOR ROTAS	31
3.6	OPERADOR MUTAÇÃO	35
3.6.1	MUTAÇÃO DO VETOR CLIENTES.....	36
3.6.2	MUTAÇÃO DO VETOR ROTAS	38
3.7	GERAÇÃO DA NOVA POPULAÇÃO E CONDIÇÕES DE PARADA	40
3.8	LEVANTAMENTO DOS DADOS	41

3.9	OTIMIZAÇÃO DAS ROTAS.....	44
4	RESULTADOS	45
5	CONCLUSÃO.....	51
	REFERÊNCIAS	53
	APÊNDICE	56
	APÊNDICE A - PROGRAMAÇÃO EM PYTHON DA API GOOGLE MAPS: COORDENADAS GEOGRÁFICAS.	56
	APÊNDICE B - PROGRAMAÇÃO EM PYTHON DA API GOOGLE MAPS DISTANCE MATRIX E TEMPO.	56

1 INTRODUÇÃO

O presente trabalho aborda a otimização de rotas e processos de decisão das operações logísticas, por meio de uma ferramenta computacional baseada no método meta-heurístico Algoritmo Genético, que envolve um Problema de Roteirização com Janelas de Tempos e Frota Heterogênea (PRVJT). O estudo é aplicado em uma empresa que produz alimentos congelados, que atende dezoito cidades da região centro-oeste brasileira, além da existência de um centro de armazenamento para distribuição em cidades nas proximidades locais.

O estudo se faz necessário para o aproveitamento de janelas de tempo, devido a uma pequena parcela dos clientes restringirem seus horários de entrega com precisão, além de decisões de utilização da frota, e otimização das rotas.

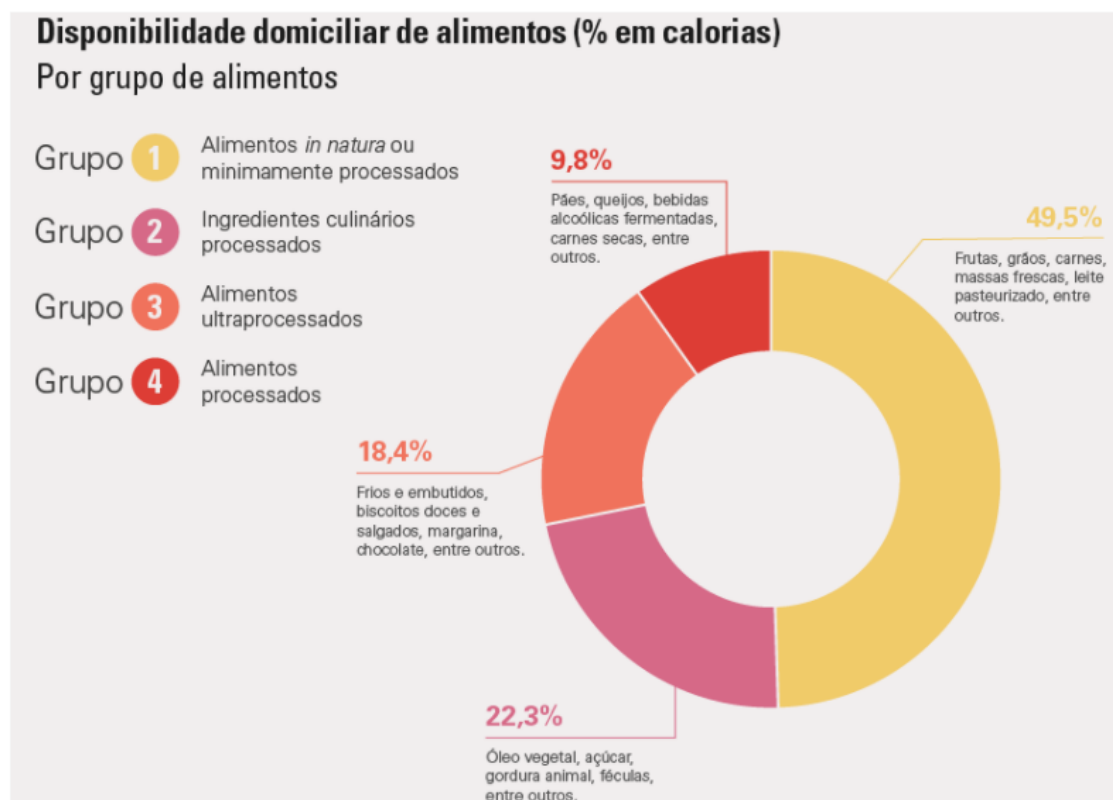
Segundo o autor Ballou 2006, os sistemas logísticos interligam locais de produção e consumo distantes, historicamente os povos antigos consumiam produtos próximos aos seus locais de origem, ou armazenavam o possível para consumo, contudo restringem o consumo de produtos oriundos de regiões específicas, assim se faz necessário estratégias e sistemas de distribuição avançados.

Além do desenvolvimento de sistemas logísticos, o surgimento de produtos congelados prontos para consumo e sua maneira de conservação foi idealizado para facilitar o dia a dia das pessoas. Ao final do século XX, com a grande preocupação com a qualidade de vida, houve uma considerável vantagem para o mercado de produtos congelados e resfriados, em vez de alimentos com altos índices de conservantes. Alimentos ultra processados ganham espaço e somam 18,4% das calorias adquiridas em casa, segundo a Agência IBGE notícias, quando se compara a evolução da aquisição de alimentos das pesquisas realizadas pelo Instituto Brasileiro de Geografia e Estatística, observa-se que em 2002/2003, os alimentos ultra processados subiram de 12,6% para 16% das calorias totais em 2008/2009, e no de 2020 chegaram a 18,4% na última pesquisa, mostrando um aumento de cerca de seis pontos percentuais nesse período (MARQUES, CÍCERO FERNANDES, 2008).

O Gráfico 1.1 demonstra a disponibilidade domiciliar de alimentos. Observa-se que a empresa em estudo se insere em dois dos grupos mencionados, o grupo 3 de alimentos ultra processados, sendo representado pelos alimentos frios e embutidos, biscoitos doces e salgados, chocolate, entre outros. Esses alimentos são produzidos com adição de sal e açúcar, podendo

conter gorduras, proteínas, extrato e substâncias sintetizadas em laboratório. O grupo 4, de alimentos processados, trata-se de alimentos onde há adição de sal, açúcar e óleo. Os mais comuns são pães, queijos, bebidas fermentadas, carnes secas, etc.

Gráfico 1.1 - Consumo de alimentos processados



Fonte: Pesquisa de Orçamentos Familiares - 2017/2018 da Agência IBGE notícias.

As decisões operacionais são aquelas que lidam com o dia a dia da empresa, procurando soluções para os seus problemas diários, no entanto essas decisões operacionais devem estar orientadas e alinhadas com as estratégias que a organização persegue a médio e longo prazo, relacionada aos requerimentos dos consumidores, clientes e fornecedores (BERTAGLIA, 2016).

Para Batistus (2009), o Problema de Roteirização com Janelas de Tempos (PRVJT) busca uma abordagem incremental utilizando restrições temporais às rotas. O termo “Janela de Tempo” indica a necessidade de se considerar horários específicos para a realização de certos atendimentos por parte dos clientes.

Rotas e cronogramas de transporte são altamente relevantes no que diz respeito à tomada de decisões operacionais, uma vez que estas determinam o tipo de veículo e a frequência de atendimento aos clientes (CHOPRA & MEINDL, 2003).

1.1 OBJETIVO GERAL

O objetivo do presente trabalho é reduzir os custos logísticos de uma empresa em torno de 3% da média mensal. Tais reduções serão alcançadas ao se utilizar um Algoritmo Genético para otimizar as distâncias das rotas de entrega da empresa.

Além disso, o trabalho envolve o desenvolvimento e testes de um algoritmo heurístico que busque solucionar o clássico modelo de Roteirização de Veículos adaptado à realidade da empresa envolvida. Portanto, para se avaliar o algoritmo os seguintes parâmetros deverão ser atendidos: não ultrapassar 30 minutos de execução; não ultrapassar 512MB em utilização de memória RAM; rotas devem conter os dados estimados de tempo e distâncias com detalhes; rotas devem ter algum grau de priorização sobre os horários dos clientes; rotas não podem ultrapassar a capacidade dos veículos disponíveis; as rotas geradas devem ser flexíveis para pequenas mudanças de trajeto ao longo do dia; O tempo em que os motoristas ficarão em trabalho deve ser estimado em cada rota; o tempo que os motoristas passarão trabalhando deve ser limitado de acordo com as normas internas da empresa.

1.2 PROCEDIMENTOS METODOLÓGICOS

O presente trabalho é um estudo de caso que desenvolveu e analisou os impactos de um algoritmo de otimização sobre os custos operacionais logísticos em uma empresa do ramo alimentício. Inicialmente desenvolveu-se um algoritmo de acordo com as necessidades da empresa, onde chegou-se à decisão que um Algoritmo Genético seria a melhor forma para abordar o problema. Depois iniciou-se a coleta e montagem do banco de dados dos clientes. Tais informações foram coletadas através de uma API do *Google Maps*® para determinar tanto as coordenadas geográficas de toda a carteira de clientes da empresa como também para montar uma Matriz de Distâncias e Tempo de Percurso entre todos os clientes. Paralelamente a isso, foram levantados os históricos de venda da empresa para entender suas sazonalidades, os seus principais clientes e suas principais rotas.

Após a coleta dos dados necessários iniciou-se a etapa de montagem das rotas a partir do algoritmo de otimização para que estas fossem comparadas as rotas executadas pela empresa. Tanto as rotas geradas pelo algoritmo como as praticadas pela empresa foram calculadas tomando como base a mesma Matriz de Distâncias. Por fim, foi possível estimar as distâncias, tempo e custos envolvidos nos dois cenários.

1.3 DELIMITAÇÃO DO TRABALHO

O estudo em questão focará nos aspectos logísticos da companhia sob a ótica de um Problema de Roteirização de Veículos, a fim de proporcionar algum grau de planejamento e consequentemente, redução de custos.

O foco será apenas nas operações logística da empresa em questão. Atuando na rotina dos colaboradores, obtendo dados de campo, registros de vendas, localização geográfica de clientes e seus respectivos históricos de compras.

1.4 ESTRUTURA DO TRABALHO

No presente capítulo é apresentada a introdução do trabalho, assim como a justificativa do estudo abordado, seguido dos objetivos gerais e específicos, os procedimentos metodológicos e a delimitação do trabalho. No capítulo 2 será abordado em que o estudo se fundamenta, realçando revisões da Logística, tema de importância para a sustentação e planejamento no que diz respeito à distribuição de produtos, seguido de Roteirização de Veículos, PRV, PRVJT e suas variantes, que se baseia em ferramentas computacionais estudadas com o objetivo de otimizar os sistemas de distribuição, reduzindo assim os custos e tempos de rotas e atendendo às restrições de tempos de entrega.

No capítulo seguinte é apresentada a revisão da literatura sobre Pesquisa Operacional e suas correlações inerentes ao presente estudo, sendo elas, Programação Linear, Heurística, Meta-Heurística, Algoritmo Genético e a justificativa da sua escolha para solucionar o problema deste estudo.

Posteriormente são apresentados os detalhes sobre a implementação do algoritmo com as principais etapas que compõem um Algoritmo Genético. No mesmo capítulo é explicado

como foi realizada a coleta de dados, quais foram as ferramentas utilizadas e o porquê da sua escolha.

2 FUNDAMENTAÇÃO TEÓRICA

Nas épocas mais antigas da história da humanidade os povos se instalavam próximos aos locais de produção, onde tinham acesso a mercadorias, e alimentação, porém, havia uma escassez de produtos no qual a região não produzia. Com os modelos logísticos mais avançados e a capacidade de importar e exportar mercadorias, a logística tornou-se essencial ao comércio e a qualidade de vida de maneira geral, portanto o valor da logística é manifestado primeiramente em termos de tempo e lugar (BALLOU, 2006).

Logística Empresarial tem como objetivo prover um melhor nível de serviço ao cliente, providenciando bens ou serviços corretos, no lugar certo, no tempo exato, nas condições desejadas e ao menor custo, assim, promovendo um sistema logístico satisfatório para a população. O grande desafio dessa área é conseguir equilibrar duas importantes variáveis, custos e nível de serviço (BALLOU, 2006).

As atividades primárias são importantes para garantir os custos e nível de serviço logístico equilibrados, sendo elas, a logística interna que se refere ao manejo de materiais, armazenamento, controle de inventários e pedidos a fornecedores, as operações nas atividades relacionadas com a transformação dos produtos, manutenção das máquinas e instalações, e a logística externa relacionadas com o controle dos produtos acabados, programação dos pedidos e entregas aos clientes, além do setor de marketing e vendas que estuda a publicidade, promoção, vendas, distribuição geográfica e por último os serviços instalação, pós-venda, orientações técnicas e assistência técnica. (PLATT, 2013).

A logística é um processo que integra o planejamento, a implementação e o controle do fluxo de mercadorias e produtos de forma eficiente e eficaz, desde o fornecedor até o consumidor, possuindo como objetivo atender às exigências dos clientes (BALLOU, 2006).

As organizações necessitam monitorar continuamente os seus custos e desenvolver alternativas para a sua redução, uma vantagem competitiva em custos é o entendimento dos geradores de custos, um deles é a alternativa de uma criação de canais de distribuição alternativos (BERTAGLIA, 2016).

Para Bertaglia (2016), as empresas possuidoras de frotas de veículos destinados à prestação de serviços de transporte ou utilização interna necessitam seguir certas práticas de negócio para se manterem competitivas no mercado. São elas:

- Analisar a atual situação da organização, com a finalidade de identificar um potencial redução de custos e eventuais oportunidades de melhoria na prestação de serviços;
- Identificar custos não visíveis, denominados “custos escondidos”;
- Controlar efetivamente os custos de manutenção;
- Ter indicadores que meçam a eficiência da frota;
- Utilizar aplicações em computadores para auxiliar nas diversas tarefas da área de transporte e distribuição;
- Determinar a frota ótima dentro da conjuntura de trabalho e serviços prestados, assim como reduzir seus custos logísticos, uma vez que eles influenciam diretamente o preço final dos produtos.

Por outro lado, para se manter uma vantagem competitiva, deve-se proporcionar aos clientes um serviço superior por meio de aperfeiçoamento contínuo. Empresas líderes possuem sistemas de informação capazes de monitorar seu desempenho logístico em tempo real, o que lhes possibilita identificar possíveis falhas operacionais e adotar providências corretivas antes da ocorrência de falhas nos serviços junto aos clientes (BOWERSOX et al., 2007).

2.1 PROBLEMAS DE ROTEAMENTO DE VEÍCULOS COM JANELAS DE TEMPO

Entende-se por transporte a movimentação de produtos de um local para outro, incluindo desde o início da cadeia de suprimentos até o consumidor final. Como raramente os produtos são consumidos no mesmo local da produção, o transporte torna-se uma questão fundamental para o bom desempenho de uma empresa (CHOPRA & MEINDL, 2003).

Os Sistemas de Roteirização de Veículos surgiram como ferramentas computacionais que possibilitam a criação de roteiros ou itinerários otimizados, considerando diferentes tipos de restrições e fatores condicionantes que objetivam excelência nos processos de distribuição física, assim como reduções de tempo e custos (MELO e FILHO, 2001; GOLDBARD e LUNA, 2005).

Dentro dos aspectos operacionais da área de transportes, abordaremos o Problema de Roteamento de Veículos (PRV) que envolve o planejamento de rotas e cronogramas de entregas e coletas a custo mínimo, iniciando-se de um ponto de origem para vários destinos, sujeito a

diversas restrições, envolvendo questões como capacidade e disponibilidade dos veículos, janelas de oportunidade entre outros, seguindo alguns procedimentos e princípios lógicos, como, o sequenciamento de paradas em um roteiro deve estar em forma de pétala (ARENALES et al., 2015; BALLOU, 2006). O objetivo desse problema é determinar a programação de expedição que minimize o custo total de transporte e, ao mesmo tempo, satisfazer os limites de fornecimento e demanda, bem como as exigências dos clientes (TAHA, 2008).

O PRV possui muitas variações, das quais abordaremos mais detalhadamente o Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT), onde o objetivo é determinar a melhor rota que um veículo irá seguir, com base em uma rede de distribuição, podendo esta ser rodovias, ferrovias e meios aéreos. Tais problemas envolvem questões complexas como horários específicos para atendimento (janelas de tempo). Ao longo da programação da rota, existirão janelas de oportunidade onde o veículo deverá aproveitá-las da melhor maneira possível atendendo os clientes que exigirem tal condicionante (CUNHA, 2000; BALLOU, 2006).

As extensões ao modelo matemático clássico PRV podem incluir variações como múltiplos locais de origem e destino, veículos com capacidades diferentes (frotas heterogêneas) e tempo máximo de mão de obra por parte dos motoristas. (BALLOU, 2006; BATISTUS, 2009). Com o intuito de fornecer ao leitor uma visão geral do PRVJT abordado, segue o resumo de acordo com o Quadro 2.1.

Quadro 2.1 – Característica do PRVJT estudado

Característica	Possibilidades	PRVJT Presente
Tamanho da Frota	• Um veículo • Múltiplos Veículos	Múltiplos Veículos
Tipo de Frota	• Homogênea • Heterogênea	Heterogênea
Pontos de Origem	• Origem única • Múltiplas origens	Origem única
Natureza da Demanda	• Determinística • Estocástica	Determinística

Tempo Máximo da Rota	• Imposto (Mesmo tempo para todas as rotas) • Impostos (Específico de cada rota) • Não imposto	Impostos (Específico de cada rota)
Operação	• Somente Coleta • Somente Entrega • Mista • Entrega Fracionada	Somente Entrega
Objetivos	• Minimizar custo total • Minimizar número de veículos necessários • Minimizar distância total percorrida • Maximizar função utilidade baseada nas prioridades dos clientes	Minimizar distância total percorrida

Fonte: Ballou (2006). Adaptado.

Tendo em vista que o PRV é de natureza combinatório e tipo *NP-hard* (não polinomial difícil), ou seja, o tempo para a resolução cresce de forma não polinomial com a quantidade de pontos a serem analisados no problema, para solucionar tais modelos exige-se muito esforço computacional e tempo para analisar por completo todas as soluções do problema (CUNHA, 2000; REIS, 2013).

Dito isto, ao se analisar o PRVJT proposto neste trabalho, percebe-se que estamos lidando não apenas com a dimensão espacial das variáveis (como no PRV), mas também temporal, e isto implica diretamente em um nível de complexidade ainda maior (BATISTUS, 2009).

Assim sendo, este capítulo se dedicou a explicar tanto as características do problema abordado no trabalho como seu nível de complexidade técnica e computacional. Os métodos computacionais desenvolvidos e suas aplicações são abordados de forma mais detalhada na metodologia deste trabalho.

2.2 PROGRAMAÇÃO LINEAR

A Pesquisa Operacional (PO) surge inicialmente no contexto da Segunda Guerra Mundial onde os exércitos dos países envolvidos tinham de enfrentar diversos problemas

operacionais como: distribuição de suprimentos, peças de reposição de veículos, munições e tropas. Países como a Inglaterra e Estados Unidos reuniram equipes de cientistas para que realizassem pesquisas e desenvolvessem métodos mais eficientes para o gerenciamento das suas operações militares (HILLIER & LIEBERMAN, 2006).

Após este período o mundo iniciou um acelerado processo de globalização que obrigou as empresas a se tornarem competitivas e inovadoras. Sendo assim, as organizações tiveram que oferecer produtos inéditos, lidar com a concorrência e ao mesmo tempo manter seus custos operacionais sob controle. Diante destes acontecimentos, a PO se estabeleceu como um importante meio para o gerenciamento de recursos limitados (ARENALES et al., 2015).

A PO utiliza o método científico para o estudo de qualquer sistema complexo que envolva a produção de algum bem ou serviço, com o objetivo claro de maximizar a performance deste sistema sob certas condições específicas. Para tal, ferramentas computacionais aliadas a conceitos matemáticos são amplamente estudadas e utilizadas neste meio. Suas aplicações se estendem por todos os processos de produção, sistemas financeiros, redes de comunicação e cadeias de suprimentos (REEVES, 2003; ARENALES et al., 2015).

A abordagem de um problema utilizando pesquisa operacional se inicia com a definição das variáveis de decisão, que representa a escolha a ser feita em um problema, por exemplo, a rota a ser seguida por um caminhão. Logo em seguida, deve-se compreender as restrições do problema, que nada mais é que suas limitações, recursos disponíveis, modo de funcionamento e capacidades de cargas dos veículos. Por último, deve-se definir a maximização ou minimização da Função Objetivo (FO), que é a meta de desempenho do sistema, por exemplo minimizar a distância total percorrida pela frota de caminhões. Geralmente a FO é uma expressão de custo ou lucro, mas há casos onde está é representada por um fluxo de informações/produtos, *lead time* de projetos, sobra de insumos, risco operacional etc. (HILLIER & LIEBERMAN, 2006).

A maior parte dos problemas abordados pela pesquisa operacional são os modelos de Programação Linear. Desde o seu surgimento, em meados dos anos 1950, a Programação Linear (PL) se destacou como uma das principais descobertas científicas do Século XX. Milhares de dólares que certamente seriam centralizados em operações pouco eficientes foram poupados em inúmeras empresas. Recursos financeiros estes que puderam ser melhor investidos nos negócios e na geração de riqueza para a sociedade (HILLIER & LIEBERMAN, 2006).

Em linhas gerais, a PL atua em problemas onde se busca a melhor forma de alocar recursos escassos em atividades distintas que influenciam no objetivo final. Estas atividades devem ser interpretadas através de equações e inequações lineares, de tal forma que, todas as características do sistema em questão estejam bem definidas. Para tal, percorre-se todo o espaço de soluções viáveis, isto é, dado um conjunto de restrições, os métodos de resolução exatos – como simplex – irão testar todas as soluções factíveis ao problema até que encontrem a solução matemática ótima.

$$FO = \sum_i^m C_i \cdot x_i \quad (1.1)$$

$$\sum_i^m a_{ij} \cdot x_i \leq b_j \quad \forall j \leq n \quad (1.2)$$

$$\sum_i^m c_{ki} \cdot x_i \leq d_k \quad \forall k \leq o \quad (1.3)$$

$$x \in \mathbb{R}$$

$$m, n, o \in \mathbb{N}$$

$$i, j, k \in \mathbb{N}$$

$$C, a, b, c, d \in \mathbb{R}$$

A equação 1.1 representa uma Função Objetivo genérica que está ligada diretamente com as variáveis do problema e seus pesos. Já as equações 1.2 e 1.3 indicam as restrições do modelo genérico. Estas podem ser tanto equações quanto inequações.

Existem três hipóteses que um modelo linear deve seguir: aditividade, proporcionalidade e fracionamento. A primeira diz respeito que toda função em um modelo de PL é igual a soma das partes. Já a segunda indica que a influência de cada variável ao valor da FO e das restrições é proporcional ao nível da atividade atribuída aquela variável. E por último, o fracionamento significa que as variáveis podem atingir valores fracionados, isto é, que pertencem ao conjunto dos números reais. Mas também é importante lembrar que podem assumir valores pertencentes ao conjunto dos números inteiros ou binários (HILLIER e LIEBERMAN, 2006; ARENALES *et al.*, 2015; TAHA, 2008)

A PL possui um campo de aplicação muito abrangente, podendo ir desde problemas de corte, roteamento, mix de produção, nivelamento de produção e estoque até ao planejamento urbano. Contudo, possui suas limitações práticas de implementação, uma vez que se trata de um método exato, e dependendo do problema abordado, o número de variáveis e restrições pode inviabilizar a solução do modelo (ARENALES *et al.*, 2015; TAHA, 2008).

Como o PRVJT é extremamente complexo de ser resolvido, uma vez que seu espaço de soluções é muito grande para ser percorrido, uma abordagem exata como um modelo de PL exigiria uma elevada quantidade de recursos e tempo computacional para solucioná-lo.

Reis *et al.* (2019) destacou a limitação de se implementar um modelo linear para a resolução de um PRV com frotas homogêneas sem considerar janelas de tempo. No estudo, quando houve picos de demanda – cerca de 35 clientes – o modelo enfrentou um tempo de resolução maior que 3 horas. Para que o Método Simplex utilizasse as rotas ótimas, foi necessário dividir o problema por zonas para tornar o modelo factível de resolução, uma vez que instâncias maiores que 20 clientes passavam a exigir tempos de resolução muito elevados.

Vecchi (2016) apresentou uma abordagem exata para o *PRV* com percursos em arcos para coleta de resíduos conhecido na literatura como Problema de Roteamento em Arcos Capacitados (CARP). Para contornar a natureza *NP-hard* e evitar soluções inviáveis, uma segunda etapa teve de ser implementada. Adaptações ao modelo original foram necessárias bem como a utilização do algoritmo de Hierholzer. Apesar do trabalho apresentar resultados promissores, houve um limitante de 71 pontos de coleta devido à natureza do problema.

Vale ressaltar que o Método Simplex, apesar de ser uma das principais descobertas da área e por se tratar de uma ferramenta extremamente eficiente para a resolução de modelos PL, não será abordado neste trabalho por não ser o foco do mesmo.

2.3 MÉTODOS HEURÍSTICOS

Os métodos heurísticos são procedimentos que visam encontrar uma solução de boa qualidade para um problema que seja muito complexo. As Meta-Heurísticas são procedimentos heurísticos que realizam buscas próximas às soluções obtidas com um procedimento mais elaborado para o refinamento da solução. Apoiadas em um conjunto de restrições, um modelo de heurística segue uma sequência de passos específicos para cada problema iniciando com

combinações aleatórias ou com um histórico de eventos, naturalmente providos de mecanismos para tentar escapar de ótimos locais ainda distantes dos ótimos globais (SOUZA, 2009; REIS, 2013).

Ao se tratar de buscas locais, o esforço de encontrar comprovadas soluções ótimas em problemas de otimização combinatória, através de técnicas de ligamentos (*branch and bound*), planos de corte, algoritmos aproximados, não necessariamente trazem resultados otimizados, deste modo muitas heurísticas são baseadas nas diretrizes de meta-heurística como, Busca Tabu (BT), *Simulated Annealing* (SA), Pesquisa Local Iterada (*Iterated Local Search*), Otimização de Colônias, procedimentos de busca gulosa, aleatória e adaptativa (GRASP), Algoritmos Genéticos (AG) entre outros métodos (RESENDE M.G.C. & SILVA R.M.A, 2013).

As Meta-Heurísticas podem se subdividir em dois tipos de busca, explorando a Busca Local ou Busca Populacional. A Busca Local procura um espaço de soluções por movimentos, os quais são aplicados a cada passo sobre a solução corrente, gerando uma evolução favorável em sua vizinhança, esse pode ser citado os métodos de Busca Tabu, *Simulated Annealing*, Busca em Vizinhança Variável (*Variable Neighborhood Search*) e *Iterated Local Search*. Por outro lado, a Busca Populacional mantém um conjunto de boas soluções e combina essas, de forma a tentar produzir soluções ainda melhores, evidenciados nos métodos clássicos de Algoritmos Genéticos, os Algoritmos Meméticos e o Algoritmo Colônia de Formigas (SOUZA, 2009).

2.3.1 Meta-heurística baseada em Algoritmo Genético

Segundo Holland (1975), os Algoritmos Genéticos (AG) são inspirados nos conceitos da Teoria da Evolução das Espécies, proposto por Charles Darwin em 1850. Uma meta-heurística baseada em Algoritmos Genéticos possui mecanismos de busca que têm a capacidade de varrer espaços complexos de soluções de forma eficiente. Estas soluções podem não necessariamente estar dentro de todas as restrições do problema, mas como o algoritmo possui natureza estocástica e iterativa, a tendência é que se crie soluções otimizadas e factíveis ao longo das próximas gerações (GOLDBERG, 1989; TORQUATO, 2017).

Estes algoritmos simulam os mecanismos da Seleção Natural das Espécies criando uma população de indivíduos (soluções) e os avaliando através de uma função de avaliação que representa a Função Objetivo (FO) de cada indivíduo, e assim pode-se determinar o quão aptos estão ao ambiente (restrições do problema). Logo em seguida, os indivíduos mais aptos são

submetidos a mutações e cruzamentos com outros indivíduos, resultando em uma nova população. Sendo assim, uma nova geração contendo características superiores dos indivíduos mais evoluídos é criada. Deste modo o algoritmo consegue realizar buscas por diversas regiões do espaço de soluções, podendo chegar em ótimos locais bem como sair destes para encontrar outras oportunidades de melhoria (GOLDBERG, 1989; REEVES, 2003; TORQUATO, 2017; BEVILAQUA, 2019).

De forma mais específica, em um AG cada indivíduo possui um cromossomo que é representado por uma cadeia de caracteres (*strings*) de tamanho bem definido. Estas *strings* podem assumir valores inteiros, binários, reais etc. Como cada indivíduo é um candidato a solucionar o problema, é no seu cromossomo que estão as informações de como esta solução será executada. Portanto, é através da avaliação destas informações que é possível mensurar o quão evoluídos estão os indivíduos (BEVILAQUA, 2019).

Para Golberg (1989), cada AG é único e particular pois existem muitas maneiras de desenvolvê-los, entretanto, todos seguem a mesma estrutura e sequência apresentada no Quadro 2.2.

Quadro 2.2 – Etapas de um Algoritmo Genético

Etapas	Funcionamento
1. Inicialização	Uma população de <i>indivíduos</i> é criada de forma aleatória
2. Avaliação	Os <i>indivíduos</i> então passam por um processo de avaliação onde cada um terá sua respectiva FO
3. Seleção	Os <i>indivíduos</i> são então escolhidos de acordo com suas respectivas FO's para serem utilizados na próxima etapa
4. <i>Crossover</i>	Os <i>indivíduos</i> selecionados serão submetidos ao cruzamento para que gerem uma nova população
5. Mutação	Dos novos <i>indivíduos</i> gerados, existe possibilidade de alguns sofrerem uma mutação aleatória com o intuito de promover maior diversidade
6. Substituição	Aqui os <i>indivíduos</i> recém criados irão substituir a população da geração anterior
7. Parada	Enquanto uma condição de parada não for atendida, as iterações 2 a 6 irão continuar. Caso esta condição seja alcançada, então o algoritmo é encerrado

Fonte: Goldberg (1989). Adaptado.

Visto que o PRV e suas variantes são do tipo combinatório e *NP-hard*, o seu espaço de soluções é complexo, demasiado grande e com muitos picos de ótimos locais. Isto torna o Algoritmo Genético uma excelente ferramenta para abordar este problema. Por possuir elevada habilidade de sair de ótimos locais devido suas características de mutação e por abordar grupos populacionais de soluções, os AG's conseguem varrer grandes espaços de soluções de forma extremamente eficaz (BARBOZA, 2005; KONAK, et al.,2006).

Reina (2012) abordou o PRVJT através de um Algoritmo Genético. O autor utilizou o procedimento de Prins (2004) para realizar a divisão ótima dos cromossomos. A população inicial foi criada em parte por cromossomos aleatórios, mas contou também com o uso da heurística de inserção I1 de Solomon (1987). Para a recombinação utilizou-se de cinco técnicas distintas: uniforme, 2 pontos, heurístico, PMX (*Partially Mapped Crossover*) e mutação gulosa. Posteriormente os resultados foram comparados com os grupos de problemas de Solomon (1987). O AG proposto apresentou ótimo resultado para problemas em que os clientes estão mais agrupados, porém teve maior dificuldade para instâncias onde os clientes estavam mais distantes entre si. O estudo conclui que o AG é uma ferramenta eficaz para auxiliar no planejamento de rotas.

3 MÉTODOS DE PESQUISA

O estudo de caso abordado neste trabalho desenvolverá um sistema de otimização que obterá o menor custo por quilômetro da frota, assim como determinará os tipos de veículos mais adequados em conformidade com a política de funcionamento do processo de pedidos de vendas e disponibilidade dos motoristas. É importante salientar que este estudo envolve frota heterogênea, desta forma os custos de quilometragem são diferentes para cada tipo de veículo utilizado pela empresa. A empresa também necessita atender exigências de horários de atendimento de uma parcela dos clientes e outra parcela de clientes com horários flexíveis. Esta situação proporciona uma oportunidade de janela de tempo e melhores condições de otimização das rotas.

A empresa estudada é situada no estado de Minas Gerais e tem um centro de distribuição no estado de Goiás. A transferência de mercadorias é feita de acordo com as previsões de vendas e a disponibilidade do espaço de armazenagem no centro de distribuição. Vale lembrar que o estudo não abordará as estratégias de rotas para o centro de distribuição, assim como os locais nas proximidades.

Em razão da distância entre os lugares de atuação, algumas rotas levam mais de um dia de viagem e, portanto, certos veículos não se encontram disponíveis em alguns dias da semana. Isso reduz o número de veículos disponíveis diariamente, além disso as decisões discernem dentre condições mecânicas do veículo e disponibilidade dos motoristas.

Por se tratar de um estudo com frota heterogênea, é importante ressaltar que no momento do estudo a empresa dispunha de seis veículos com capacidades distintas que variavam de 650 a 6.200 quilos.

Diante do alto nível de decisões a serem tomadas estrategicamente, houve a necessidade de inserir ao código fonte do algoritmo genético a liberdade de incluir ou excluir a disponibilidade dos veículos, assim como dos motoristas, além de delimitar os limites inferiores e superiores dos horários de atendimento.

Neste capítulo abordaremos como o algoritmo de otimização foi desenvolvido e posteriormente como os dados da empresa foram coletados e interpretados.

3.1 IMPLEMENTAÇÃO DA META-HEURÍSTICA

Como descrito anteriormente, um AG segue uma sequência lógica bem definida de etapas, sendo elas respectivamente: (1) a criação de uma população aleatória; (2) a avaliação dos indivíduos da população; (3) seleção dos indivíduos mais aptos; (4) execução do crossover entre os indivíduos selecionados e a população geral; (5) mutação dos indivíduos recém criados; (6) avaliação dos indivíduos criados; (7) substituição dos indivíduos da população anterior para criar a nova população. A Figura 3.1 mostra o procedimento do AG criado.

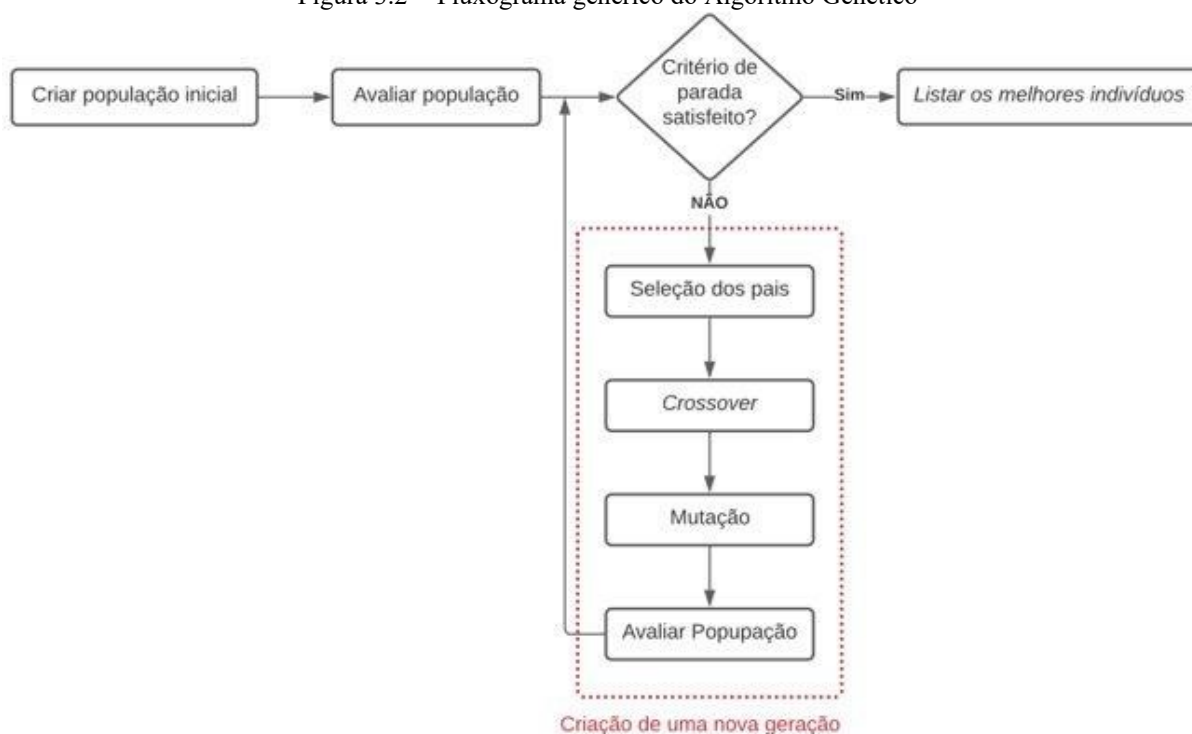
Figura 3.1 – Procedimento algoritmo genético

- 1 Criar população de indivíduos aleatórios**
 - 2 Para (cada indivíduo): Realizar Punição**
 - 2.1 Calcular a distancia total das rotas do indivíduo
 - 2.2 Calcular punição em função da carga excedente de cada veículo
 - 2.3 Calcular punição em função da duração excedente das rotas de cada veículo
 - 2.4 Calcular punição em função de cada atendimento fora do horário exigido
 - 2.5 Calcular punição em função do quantidade excedente de clientes por rota
 - 2.6 Calcular punição em função da quantidade de rotas excedentes por veículo
 - Fazer:
 - 3 Seleção de indivíduos**
 - 3.1 Escolher aleatoriamente uma quantidade N de indivíduos da população
 - 3.2 Realizar um ranqueamento dos indivíduos escolhidos na etapa 3.1
 - 3.3 Selecionar os K melhores ranqueados da etapa 3.2
 - 4 Para (cada indivíduo): Realizar Crossover**
 - 4.1 Escolher aleatoriamente um dos indivíduos selecionados na etapa 3
 - 4.2 Realizar a operação *crossover* entre os indivíduos
 - 4.3 Armazenar o indivíduo gerado da etapa 4.2
 - 5 Para (cada indivíduo gerado na etapa 4): Realizar Mutação**
 - 5.1 Escolher aleatoriamente um dos indivíduos criados na etapa 4
 - 5.2 Realizar trocas aleatórias das posições do cromossomo
 - 5.3 Armazenar o indivíduo gerado da etapa 5.2
 - 6 Para (cada indivíduo gerado nas etapas 4 e 5): Realizar Punição**
 - 6.1 Chamar a função da etapa 2
 - 7 Substituir a população anterior pelos indivíduos gerados nas etapas 4 e 5**
- Enquanto: A nova população da etapa 7 for melhor que a antecessora
Fim

Fonte: Autoria própria.

Observa-se que inicialmente o código parte de um espaço de soluções pseudo aleatório, uma vez que não precisa de soluções de referência para começar as operações. As etapas seguintes seguem tanto a estrutura que foi citada de forma lógica no parágrafo anterior como os conceitos revistos no Quadro 2.2. Nos tópicos subsequentes as implementações das etapas 1 a 5 serão abordadas com maiores detalhes e bem como a representação computacional de como é a estrutura de um indivíduo para o AG.

Figura 3.2 – Fluxograma genérico do Algoritmo Genético



Fonte: Autoria própria.

Uma das etapas mais críticas do processo de desenvolvimento do AG é a representação computacional de um indivíduo de acordo com o problema real enfrentado. A representação deve ser precisa e factível de mudanças para que o algoritmo possa gerar movimentos de trocas ao longo das iterações.

O trabalho de Prins (2004) seguiu como referência para o desenvolvimento da representação do indivíduo. Isto é, o cromossomo dos indivíduos é particionado em rotas. Contudo, muitas adaptações foram necessárias para atender aos requisitos do problema abordado.

No presente trabalho, cada indivíduo representa uma forma diferente de como a carteira de clientes será atendida. Para tal, este indivíduo deve carregar consigo as informações

referentes à relação de clientes, a ordem de atendimento, a sequência das rotas, quais são os veículos e suas capacidades.

Para conseguir essa riqueza de detalhes da forma mais simples possível foi definido que cada indivíduo teria dois vetores de números inteiros. O primeiro vetor representa a sequência de clientes que será atendida. Já o segundo possui três funções: a primeira é de quebrar o primeiro vetor em rotas; a segunda é de determinar quais veículos irão atender quais rotas e em terceiro, quantos clientes cada rota possui. A Figura 3.3 busca mostrar visualmente a representação explicada.

Figura 3.3 – Representação dos indivíduos

Indivíduo 1UYA44845A2100	
Vetor Clientes	3 39 11 10 99 64 314 12 921 44 21 63 65 9 999
Vetor Rotas	5 2 1 5 1 6 5 2

Rotas Veículo 5	
1º	3 39
2º	9 999

Rotas Veículo 1	
1º	11 10 99 64 314
2º	12 921 44 21 63 65

Fonte: Autoria própria.

Para exemplificar da melhor maneira possível um indivíduo fictício foi criado. Os dois vetores chamados de Vetor Clientes (*VC*) e Vetor Rotas (*VR*) representam respectivamente o primeiro e segundo vetores apresentados na Figura 3.3.

O vetor Clientes é composto pelos códigos individuais de cada cliente da carteira de pedidos. A sua sequência interfere diretamente no modo com que as rotas são geradas. Já o Vetor Rotas possui uma lógica sequencial sempre em pares. O primeiro número representa o código do veículo, o segundo número indica quantos clientes serão atendidos por este veículo. Portanto, cruzando as informações entre os dois vetores conclui-se que uma rota distinguida pela cor verde será composta pelos clientes 3 e 39 e será executada pelo veículo 5.

Dando prosseguimento ao raciocínio, o próximo número do *VR* é 1, o número 5 é o subsequente e informa quantos clientes serão atendidos. Sendo assim, a rota amarela será executada pelo veículo 1 e é composta pelos clientes 11-10-99-64-314 em sequência. O mesmo vale para o restante da análise.

Na Figura 3.3 vemos que tanto o veículo 5 como o veículo 1 possuem duas rotas cada. Observe que a ordem das rotas é determinada e deve ser obedecida. Seguindo o exemplo da Figura 3.3, o roteiro do veículo 1 é atender os clientes 3 e 39, retornar ao depósito, carregar os pedidos da segunda rota para só então atender aos clientes 9 e 999.

Como o problema tratado neste trabalho possui uma abordagem temporal, a ordem com que as rotas são seguidas influencia diretamente na aptidão do indivíduo. Isto é, caso o cliente 999 deseje receber seu pedido no final do dia, então o código tenderá a inseri-lo nas últimas posições das últimas rotas do veículo que irá atendê-lo.

3.2 GERAÇÃO DA POPULAÇÃO INICIAL

Como mencionado na etapa 1 do procedimento do Algoritmo Genético na Figura 3.1, o algoritmo inicia sua busca no espaço de soluções de forma totalmente aleatória. Esta é uma das grandes vantagens do AG em comparação com outras meta-heurísticas, visto que a inserção de soluções iniciais pré estabelecidas pode enviesar o algoritmo a realizar apenas buscas em ótimos locais.

Nesta etapa, o algoritmo realizará o preenchimento do *VC* e *VR* de forma aleatória. Isto ocorre mediante uma lista contendo os pedidos dos clientes e outra com os veículos disponíveis. Cada posição de ambos os vetores escolhe de forma randômica qual será o cliente, veículo e tamanho da rota. Ao final das iterações tem-se uma população constituída por diversos indivíduos espalhados pelo espaço de soluções. A Figura 3.4 mostra todas as etapas da construção aleatória de soluções.

Figura 3.4 – Procedimento criação aleatória

Fazer:Fazer:

- 1 Selecionar um cliente da lista de pedidos aleatoriamente
- 2 Inserir o cliente selecionado no "Vetor Clientes"
- 3 Remover da lista de pedidos o cliente recém selecionado

Enquanto: Todos os clientes não forem inseridosFazer:

- 4 Selecionar um veículo da lista de veículos disponíveis
- 5 Inserir veículo selecionado no "Vetor Veículos"
- 6 Gerar aleatoriamente quantos clientes o veículo recém selecionado irá atender
- 7 Inserir valor de quantos clientes serão atendidos no "Vetor Veículos"

Enquanto: Quantidade de rotas não atender todos os clientes do pedido

- 8 Criar indivíduo com base em "Vetor Clientes" e "Vetor Veículos"
- 9 Limpar "Vetor Clientes" e "Vetor Veículos"

Enquanto: Limite de criações de indivíduos não for atingido

Fim

Fonte: Autoria própria.

3.3 FUNÇÃO OBJETIVO

Esta é uma etapa essencial pois determina qual direção do espaço de soluções o código estará seguindo. Se a punição não for precisa e inconsistente com a realidade do problema estudado, soluções inviáveis ou pouco otimizadas serão geradas.

Cada indivíduo gerado possui um valor de Função Objetiva - FO - que irá depender da distância total das suas rotas. A partir desse valor é então iniciado o processo de avaliação que tende a aumentá-lo como punição caso uma ou várias restrições não sejam atendidas.

Os requisitos avaliados para a punição estão todos contidos na Etapa 2 da Figura 3.1, sendo estes: i) distância total das rotas; ii) excesso de cargas nos veículos; iii) duração excedente do turno de trabalho para cada veículo; iv) excesso de clientes atendidos por veículo por rota; v) limite de rotas por veículo e vi) aderência dos atendimentos às janelas de tempo dos clientes. Para o dimensionamento das distâncias percorridas pelos veículos durante as rotas adotam-se as seguintes regras básicas de acordo com o Quadro 3.1.

Quadro 3.1 – Premissas do PRVJT abordado

Regras	Premissas
1 ^a	Todo veículo deve iniciar e finalizar a rota saindo e retornando do mesmo depósito.
2 ^a	Cada cliente será atendido apenas uma única vez por um único veículo.
3 ^a	Caso o veículo tenha mais de uma rota programada, deve finalizar a primeira, retornar ao depósito, carregar a mercadoria para iniciar a próxima.

Fonte: Autoria própria.

Como esta etapa do Algoritmo Genético exige muito cuidado e precisa ser desenvolvida pensando nas restrições do problema abordado, cada punição tem uma função matemática específica. As expressões mostradas abaixo são aplicadas para cada indivíduo do AG.

- Seja i o i -ésimo veículo do indivíduo; j a j -ésima rota do indivíduo; x o x -ésimo cliente do indivíduo;
- K indica a soma de todas distâncias das rotas do indivíduo;
- EC_{ij} indica o excesso de cargas que o veículo i teve de transportar na rota j ;
- CAP_i indica a capacidade que o veículo i pode transportar;
- EH_i indica a quantidade de horas excedentes trabalhadas pelo veículo i ;
- MO_i indica a disponibilidade máxima de horas que o veículo i pode trabalhar;
- ECL_{ij} indica o excesso de clientes atendidos pelo veículo i durante a rota j
- LCL_i limite de clientes que o veículo i pode atender em uma única rota;
- ER_i quantidade de rotas que excedem o limite estabelecido para o veículo i ;
- LR_i limite de rotas que o veículo i pode realizar;
- A_x momento em que o cliente x é atendido;
- LAF_x limite final exigido pelo cliente para o recebimento do pedido;
- LAI_x limite inicial exigido pelo cliente x para o recebimento do pedido
- $\Delta JT_x = LAF_x - LAI_x$, janela de tempo do cliente x
- C_y constante que indica o peso para cada função punição fp_y .

A função (3.1), $fp_1(EC_{ij}, CAP_i, K)$, descreve a punição do indivíduo em função do excesso de cargas por veículo por rota EC_{ij} e pela distância total das rotas K .

$$fp_1(EC_{ij}, CAP_i, K) = \left(1 + \sum \frac{EC_{ij}}{CAP_i}\right) * K, \forall i \quad (3.1)$$

A função (3.2), $fp_2(EH_i, MO_i, K)$, descreve a punição do indivíduo em função do excesso de horas trabalhadas pelo veículo EH_i e pela distância total das rotas K .

$$fp_2(EH_i, MO_i, K) = \left(1 + \sum \frac{EH_i}{MO_i}\right) * K \quad (3.2)$$

A função (3.3), $fp_3(ECL_{ij}, LCL_i, K)$, descreve a punição do indivíduo em função do excesso de clientes atendidos por cada veículo em cada rota ECL_{ij} e pela distância total das rotas K ;

$$fp_3(ECL_{ij}, LCL_i, K) = \left(1 + \sum \frac{ECL_{ij}}{LCL_i}\right) * K, \forall i \quad (3.3)$$

A função (3.4), $fp_4(ER_i, LR_i, K)$, descreve a punição do indivíduo em função do excesso de rotas realizadas por cada veículo ER_i e pela distância total das rotas K ;

$$fp_4(ER_i, LR_i, K) = \left(1 + \sum \frac{ER_i}{LR_i}\right) * K \quad (3.4)$$

A função (3.5); $fp_5(A_x, LAF_x, LAI_x, \Delta JT_x)$, descreve a punição do indivíduo em função da aderência de horários de atendimento e pela distância total das rotas K .

$$fp_5(A_x, LAF_x, LAI_x, \Delta JT_x, K) = \left[1 + \begin{cases} 0, se \ LAI_x \leq A_x \leq LAF_x \\ \sum \frac{(LAI_x - A_x) * 0,9}{\Delta JT_x}, se \ A_x < LAI_x \\ \sum \frac{(A_x - LAF_x) * 1,3}{\Delta JT_x}, se \ A_x > LAF_x \end{cases} \right] * K \quad (3.5)$$

E finalmente, a função (3.6), $FO(fp_1, fp_2, fp_3, fp_4, fp_5, K)$, descreve a Função Objetivo de cada indivíduo no AG. É através da minimização desta função que se espera a otimização do problema abordado.

$$FO(fp_1, fp_2, fp_3, fp_4, fp_5, K) = \frac{KC_6 + \sum_{y=1}^5 fp_y C_y}{100} \mid \sum_{y=1}^6 C_y = 100 \quad (3.6)$$

Percebe-se inicialmente que as equações (3.1), (3.2) e (3.5) serão as mais críticas para a geração de soluções otimizadas. Para que isso seja visível ao AG, as constantes C1, C2 e C5 terão maiores pesos na equação (3.6). Geralmente os valores variam em torno de 15, 35 e 30 respectivamente. Lembrando que a somatória das constantes deve ser igual a 100.

As demais equações – (3.3) e (3.4) – têm o objetivo de auxiliar o AG a gerar soluções mais factíveis à realidade da empresa. Como por exemplo limitar quantas rotas um veículo deve fazer ou limitar quantos clientes uma determinada rota deve atender. Estas funções, contudo, podem ser facilmente anuladas da etapa de punição atribuindo suas respectivas constantes C a zero caso necessário.

Uma observação importante que deve ser destacada é a presença da variável K em todas as equações. Como este é um problema logístico, é de interesse de todos que a distância percorrida pelos veículos seja a menor possível. Pensando nisso, ao se calcular a FO de um dado indivíduo, a distância total percorrida por este é inserida em todas as etapas do cálculo.

Desta forma é razoável concluir que os indivíduos que proporcionarem rotas mais curtas estarão mais bem colocados em comparação com os demais. Dessa maneira, a otimização das distâncias é um objetivo intrínseco à evolução das gerações do AG.

3.4 BUSCA LOCAL

Esta etapa do algoritmo consiste em selecionar quais indivíduos irão transferir suas características para a próxima geração. O método escolhido para este trabalho foi a Seleção por Torneio em Duas Fases.

A primeira etapa desta estratégia visa selecionar um certo número N de indivíduos aleatoriamente da população. Logo em seguida, dentre estes N selecionados, realiza-se um ranqueamento em ordem crescente em função das respectivas FO's para a escolha dos P melhores.

Uma das grandes vantagens deste método é a sua característica de preservar a diversidade de indivíduos uma vez que se escolhe aleatoriamente quem irá participar do torneio. Ou seja, evita-se que indivíduos não tão evoluídos sejam extintos precocemente da população (LINDEN, 2006; TELES E GOMES, 2010). O segundo fator positivo que pesou na sua escolha foi a simplicidade de implementação e eficiência.

3.5 OPERADOR CROSSOVER

Neste tópico abordaremos a implementação dos operadores crossover e suas respectivas funções dentro do algoritmo. Duas etapas foram criadas: a primeira é responsável pelos cruzamentos dos *vetores clientes* (VC) de ambos os pais selecionados, já a segunda é composta por dois operadores crossover distintos responsáveis pelos *vetores rota* (VR).

3.5.1 Crossover do Vetor Clientes

A estratégia utilizada para realizar o *crossover* do vetor clientes foi o operador *ERX* (*Edge Recombination*) proposto por Whitley, Starkweather e Fuquay (1989). Este operador realiza a montagem do cromossomo filho a partir das arestas dos genes dos cromossomos pais. Aqui nos referimos a arestas como sendo o par de elementos vizinhos de uma posição específica de um vetor.

Usualmente o *ERX* foi concebido para o PCV – Problema do Caixeiro Viajante – mas foi adaptado para o presente trabalho. Esta estratégia se torna a mais confiável pois em média

95% das arestas podem ser herdadas sem que haja necessidade de inserções aleatórias forçadas (WHITLEY, STARKWEATHER, FUQUAY, 1989; WHITLEY, STARKWEATHER, SHANER, 2002). Abaixo na Figura 3.5 segue o procedimento do operador *ERX* e logo em seguida sua representação visual passo a passo.

Figura 3.5 – Procedimento *ERX*

- 1 Selecionar os dois Indivíduos pais para o crossover
 - 2 Criar uma Lista de Arestas para cada cliente de acordo com os *vetores clientes* dos pais
 - 3 Selecionar o primeiro cliente de um dos pais que tenha o menor número de arestas
 - 3.1 Se: Ambos possuírem o mesmo número de arestas. Então:
 - 3.1.1 Escolher um dos dois aleatoriamente
 - 4 Adicionar o cliente selecionado no cromossomo filho
 - 5 Retirar da Lista de Arestas o ultimo cliente selecionado
- Fazer:
- 6.1 Se: Houverem arestas para o ultimo cliente do cromossomo filho. Então:
 - 6.1.1 Para: Cada cliente que faz parte da aresta do ultimo cliente do cromossomo filho. Fazer:
 - 6.1.1.1 Determinar quantas arestas cada um possui
 - 6.1.2 Selecionar aquele cliente que possuir o menor quantidade de arestas
 - 6.1.3 Se: Houver empate no número de arestas. Então:
 - 6.1.3.1 Selecionar aleatoriamente dentre os que possuem menor quantidade de arestas
 - 6.1.4 Chamar etapa 4
 - 6.1.5 Chamar etapa 5
 - 6.2 Se não:
 - 6.2.1 Selecionar aleatoriamente um cliente da Lista de Arestas
 - 6.2.2 Chamar etapa 4
 - 6.2.3 Chamar etapa 5
- Enquanto: Todos os clientes não estiverem inseridos no cromossomo filho

Fonte: Autoria própria.

Dado o procedimento *ERX*, prosseguiremos com um exemplo partindo de dois indivíduos fictícios. Cada etapa da explicação irá referenciar a sua correspondente do procedimento. Inicialmente temos o indivíduo pai *01* composto pelo seguinte *VC*: 3-39-11-10-99-64-314-12-921-44-21-63-65-9-999, e o indivíduo pai *02* com 9-39-44-99-10-999-12-11-63-314-21-921-3-65-64. A lista de arestas é dinâmica e mudará em cada iteração do operador *ERX*.

Figura 3.6 – Início do procedimento ERX

		Lista de Arestas					
		Cliente	Arestas				Qte Arestas
Pai 01		3	39	999	65	921	4
Vetor Clientes	3 39 11 10 99 64 314 12 921 44 21 63 65 9 999	39	3	11	9	44	4
		11	39	10	12	63	4
		10	11	99	999		3
		99	10	64	44	10	4
		64	99	314	65	9	4
		314	64	12	63	21	4
		12	314	921	999	12	4
		921	12	44	21	3	4
		44	921	21	39	99	4
		21	44	63	39	99	4
		63	21	65	11	314	4
		65	63	9	3	64	4
		9	39	64	65	999	4
		999	9	3	10	12	4
Pai 02							
Vetor Clientes	9 39 44 99 10 999 12 11 63 314 21 921 3 65 64						

Fonte: Autoria própria.

De acordo com procedimento *ERX*, as etapas 1 e 2 já foram executadas visto que já temos os dois pais e as suas respectivas listas de arestas. Para a etapa 3 teremos que escolher entre os clientes da primeira posição de cada pai. Então os candidatos são 3 do *pai 01* e 9 do *pai 02*. Dentre os dois candidatos iremos selecionar aquele que possuir a menor quantidade de arestas. Contudo ambos possuem 4 arestas: 39-64-65-999 para o cliente 9, e 39-999-65-921 para o cliente 3 como pode ser visto na lista de arestas.

Então como descrito na etapa 3.1 e 3.1.1, se ambos possuírem a mesma quantidade de arestas, realiza-se a escolha aleatoriamente. Aqui escolhemos então o cliente 9 para compor o primeiro gene do cromossomo filho e concluir a etapa 4.

A etapa 5 informa que o cliente 9 deverá ser removido da lista de arestas, mas para fins didáticos apenas deixaremos os elementos que foram removidos em destaque. Abaixo na Figura 3.7 podemos visualizar como está o procedimento após a 1ª iteração.

Figura 3.7 – 1ª iteração do procedimento *ERX*

1ª Iteração		Lista de Arestas					
Pai 01		Cliente	Arestas				Qte Arestas
		3	39	999	65	921	4
		39	3	11	9	44	3
		11	39	10	12	63	4
		10	11	99	999		3
		99	10	64	44		3
		64	99	314	65	9	3
		314	64	12	63	21	4
		12	314	921	999	11	4
		921	12	44	21	3	4
		44	921	21	39	99	4
		21	44	63	39	99	4
		63	21	65	11	314	4
		65	63	9	3	64	3
		9	39	64	65	999	-
		999	9	3	10	12	3
Pai 02							
Vetor Clientes	3 39 11 10 99 64 314 12 921 44 21 63 65 9 999						
Pai 02							
Vetor Clientes	9 39 44 99 10 999 12 11 63 314 21 921 3 65 64						
Cromossomo filho							
Vetor Clientes	9						

Fonte: Autoria própria.

Após os primeiros procedimentos vemos que houve uma considerável mudança na lista de arestas. Inicialmente o cliente 39 possuía 4 arestas, 3-11-9-44, mas como o cliente 9 foi escolhido na última operação, então a nova relação de arestas passa a ser 3-99-44. O mesmo ocorre para os clientes 64-65-999.

O próximo passo é a etapa 6.1 onde é verificado se o último cliente do cromossomo filho possui arestas. No nosso exemplo o cliente 9 possui ainda as áreas 39-64-65-99. Dentre estes quatro candidatos devemos escolher aquele que possuir a menor quantidade de arestas segundo a etapa 6.1.2.

Contudo, de acordo com 6.1.3 e 6.1.3.1, como todos possuem 3 arestas, realiza-se uma escolha aleatória, logo, o próximo será o cliente 999. Depois da seleção deve-se executar novamente as etapas 4 e 5.

Figura 3.8 – 2ª iteração do procedimento ERX

2ª Iteração														Lista de Arestas								
Pai 01														Cliente	Arestas				Qte Arestas			
Vetor Clientes		3	39	11	10	99	64	314	12	921	44	21	63	65	9	999	3	39	999	65	921	3
														39	3	11	9	44		3		
														11	39	10	12	63		4		
														10	11	99	999		2			
														99	10	64	44		3			
														64	99	314	65	9	3			
														314	64	12	63	21	4			
														12	314	921	999	11	3			
														921	12	44	21	3	4			
														44	921	21	39	99	4			
														21	44	63	39	99	4			
														63	21	65	11	314	4			
														65	63	9	3	64	3			
														9	39	64	65	999	-			
														999	9	3	10	12	-			
Pai 02																						
Vetor Clientes		9	39	44	99	10	999	12	11	63	314	21	921	3	65	64						
Cromossomo filho																						
Vetor Clientes		9	999																			

Fonte: Autoria própria.

Prosseguindo, como o novo cliente do cromossomo filho é o 999, então temos que suas arestas são 9-3-10-12. Dentre estes o que possui menor número de arestas é o 10, com os clientes 11 e 99. Portanto, o terceiro gene do cromossomo filho será o cliente 10. Esta sequência deve prosseguir até que todos os clientes estejam inseridos no cromossomo filho.

Agora suponhamos que, durante a iteração destes indivíduos por exemplo, acontecesse do cliente 999 já ter os seus candidatos seguintes – 9-3-10-12 – inseridos no cromossomo filho. Então a sua lista de arestas estaria vazia e a etapa 6.2 seria executada. Desta forma, uma inserção forçada seria necessária através da escolha de um cliente ainda não selecionado anteriormente para compor o cromossomo filho.

Esta situação configura uma mutação indesejada no operador *ERX* pois o cliente inserido de forma forçada não é mais uma aresta do seu antecessor - aqui exemplificado pelo cliente 999. Contudo, este cenário é evitado ao se priorizar sempre os clientes de menor quantidade de arestas, e estas ocorrências acabam se tornando raras no operador em questão.

Figura 3.9 – Cromossomo final

Pai 01	
Vetor Clientes	3 39 11 10 99 64 314 12 921 44 21 63 65 9 999
Pai 02	
Vetor Clientes	9 39 44 99 10 999 12 11 63 314 21 921 3 65 64
Cromossomo filho	
Vetor Clientes	9 999 10 99 64 65 3 39 11 63 21 44 921 12 314

Fonte: Autoria própria.

Na Figura 3.9 podemos ver um possível resultado após a 15ª iteração para o cromossomo filho. Para fins didáticos, alguns genes foram coloridos a fim de mostrar a origem das suas características. Os genes marcados de azul, cinza, e vermelho no cromossomo filho indicam que foram herdados das arestas do pai 01. Já os genes marcados de verde e laranja indicam que a herança veio do pai 02. A mesma regra se estende para os demais genes do filho.

Vale lembrar que, quando no crossover ocorre a etapa 6.2, esta herança não é observada em algum momento da cadeia de genes do cromossomo filho, e, portanto, a regra das arestas acaba não sendo aplicada para 100% dos genes.

Outro ponto importante a ser observado é que este é um dos resultados possíveis do crossover exemplificado. Isto acontece pois em cada momento que ocorre um empate na quantidade de arestas, uma escolha aleatória é necessária. Por exemplo, se na 1ª iteração do nosso exemplo tivéssemos escolhido ao invés do cliente 9, o cliente 3, um resultado totalmente diferente seria observado. Ou seja, cada empate no momento da escolha do próximo cliente cria no mínimo duas ramificações possíveis para o cromossomo que será gerado.

Esta é uma característica positiva que pesou a favor na escolha do *ERX* pois, ao se executar a mesma instância do código para os mesmos pais, via de regra, movimentos diferentes são gerados, o que proporciona maior variabilidade ao algoritmo.

3.5.2 Crossover do Vetor Rotas

A segunda etapa do crossover é responsável por gerar cromossomos a partir dos vetores rotas (*VR*) dos pais. Esta etapa foi totalmente desenvolvida para o trabalho em questão pois é uma característica muito peculiar do problema abordado aqui.

Basicamente, para que um operador crossover seja eficaz, duas premissas devem ser atendidas: capacidade de transferir características dos pais para os filhos e capacidade de gerar movimentos para todas as direções possíveis. Pensando desta forma, dois operadores diferentes foram desenvolvidos, aqui chamados de *CRX01* e *CRX02*.

Além disso, existe uma certa chance de algum dos operadores não ser executado em um dado cruzamento. Ou seja, durante a geração de um indivíduo, pode acontecer de apenas o *CRX02* ser executado ou vice versa.

Como aqui busca-se proporcionar ao AG a possibilidade de executar movimentos em todas as direções possíveis do espaço de soluções, a não execução de um dos operadores é uma opção que deve estar presente no leque de movimentos possíveis do *crossover*.

O operador *CRX01* buscará manter a configuração das rotas contidas nos indivíduos pais, mas apenas mudando a sua ordem. Isto é, os veículos farão a mesma quantidade de entregas, porém para clientes diferentes. Abaixo temos o procedimento do operador *CRX01*.

Figura 3.10 – Procedimento *CRX01*

Vetor rotas *CRX01* - *VR_{CRX01}*

Fazer:

- 1 Selecionar uma rota aleatoriamente de *VR* do Pai 01
- 2 Selecionar uma rota aleatoriamente de *VR* do Pai 02
- 3 Inserir uma das rotas selecionadas em 1 e 2 em *VR_{CRX01}*

Enquanto: *VR_{CRX01}* não atender todos os clientes

Fonte: Autoria própria.

Tomando como exemplo os mesmos indivíduos, suponhamos que pai 01 e pai 02 respectivamente possuem *VR* de 5-2-1-5-1-6-5-2 e 3-3-5-7-1-5 como mostra a Figura 3.11. Nos baseando no procedimento para *CRX01* ilustrado na Figura 3.11, as etapas 1 e 2 escolhem uma rota de cada *VR* pai. Aqui seguiremos com as rotas 5-2 (azul) e 5-7 (amarelo) dos pais 01 e 02 respectivamente de acordo com a Figura 3.11. A etapa 3 realiza a inserção de uma das duas

rotas em *VR**CRX01*, portanto seguiremos com a escolha aleatória da rota 5-7. Na iteração seguinte, o procedimento segue escolhendo as rotas 1-6 (cinza) e 3-3 (verde). Agora a escolha de inserção será 1-6.

Figura 3.11 – *VC* e *VR* de pai 01 e pai 02

Pai 01														
VC	3	39	11	10	99	64	314	12	921	44	21	63	65	9 999
VR	5	2	1	5	1	6	5	2						

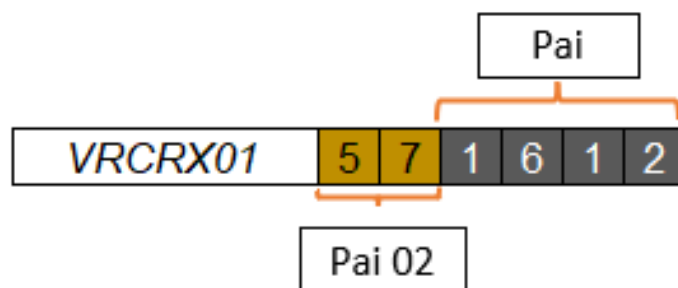
Pai 02														
VC	9	39	44	99	10	999	12	11	63	314	21	921	3	65 64
VR	3	3	5	7	1	5								

Fonte: Autoria própria.

Ao final da segunda etapa do operador *CRX01*, o vetor *VR**CRX01* já terá duas rotas com a seguinte configuração: 5-7-1-6. O procedimento seguirá até que *VR**CRX01* atenda todos os clientes disponíveis.

Uma possível configuração após o fim desse crossover seria *VR**CRX01* igual a 5-7-1-6-1-6. Neste caso vemos que a rota 1-6 do *VR* pai 01 se repetiu duas vezes e que a rota 5-7 pertence ao *VR* pai 02. Logo tivemos a contribuição de ambos os pais para a geração do próximo indivíduo.

Figura 3.12 – *VR**CRX01* resultante



Fonte: Autoria própria.

Vale lembrar que a soma do total de clientes atendidos por *VR**CRX01* é de 19. Como apenas há 15 clientes, uma pequena modificação é realizada. Sendo assim, a nova configuração de *VR**CRX01* é de 5-7-1-6-1-2 como mostra a Figura 3.12.

Uma segunda observação que deve ser mencionada em relação a este operador é que como ele busca conservar a configuração das rotas dos pais, limitando-se apenas a fazer um mix entre elas, poucos movimentos são gerados. Reduzindo assim as possibilidades de se percorrer determinados pontos do espaço de soluções.

Em relação ao segundo operador, o *CRX02* foi desenvolvido para auxiliar o *CRX01* nos movimentos de crossover. Sua característica é de tentar forçar novas configurações de rotas, podendo mudar os padrões presentes nas rotas dos pais. Na Figura 3.13 temos o procedimento do *CRX02*.

Figura 3.13 – Procedimento *CRX02*

Lista de veículos - *LV*
 Lista de rotas - *LR*
 Vetor rotas *CRX02* - *VR**CRX02*

- 1 Para cada: Rota de *VR* do Pai 01
 - 1.1 Salvar o veículo da rota em *LV*
 - 1.2 Salvar o tamanho da rota em *LR*
- 2 Para cada: Rota de *VR* do Pai 02
 - 2.1 Salvar o veículo da rota em *LV*
 - 2.2 Salvar o tamanho da rota em *LR*

Fazer:

- 3 Selecionar um elemento aleatoriamente de *LV*
- 4 Inserir elemento selecionado de *LV* em *VR**CRX02*
- 5 Selecionar um elemento aleatoriamente de *LR*
- 6 Inserir elemento selecionado de *LR* em *VR**CRX02*

Enquanto: *VR**CRX02* não atender todos os clientes

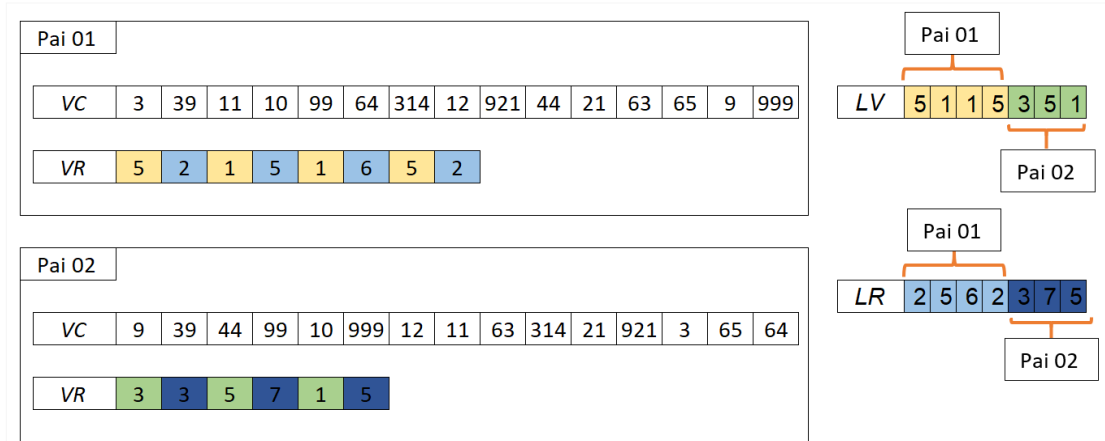
Fonte: Autoria própria.

Ainda com os mesmos indivíduos do tópico anterior, temos que pai 01 e pai 02 respectivamente possuem o *VR*: 5-2-1-5-1-6-5-2 e 3-3-5-7-1-5 como mostra a Figura 3.14.

Na etapa 1 e 2 do procedimento *CRX02* vemos que inicialmente ambos os vetores são extraídos para outros dois a partir do tipo de informação. Isto é, dois novos vetores são criados: Lista de Veículos (*LV*) e Lista de Rotas (*LR*). O primeiro contém apenas o código que identifica os veículos das rotas de *VR* dos pais, já o segundo é composto apenas pelos números que

representam o tamanho das rotas. Na Figura 3.14 podemos ver essa separação de forma mais intuitiva.

Figura 3.14 – LV e LR de pai 01 e pai 02

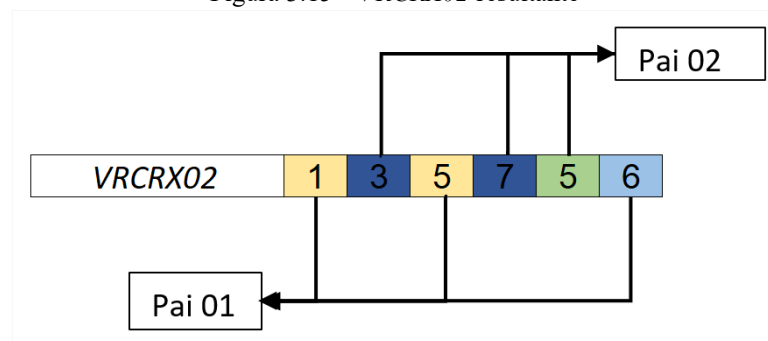


Fonte: Autoria própria.

As etapas 3 e 5 informam que se deve escolher um elemento aleatório de LV e LR. Vamos supor que se escolha o 3º elemento em LV e o 5º elemento em LR, isto é, os valores 1 e 3. Isto significa que a primeira rota gerada pelo operador *CRX02* será executada pelo veículo 1 que atenderá três clientes em sequência.

Então estes dois elementos irão compor o início de *VR**CRX02* de acordo com as etapas 4 e 6. Repare que, o resultado dessa operação estará contido em *VR**CRX02*, e o procedimento só se encerrará quando todos os clientes forem atendidos. Na Figura 3.15 podemos ver um possível resultado deste operador.

Figura 3.15 – *VR**CRX02* resultante



Fonte: Autoria própria.

Ao final da operação, três rotas foram criadas com tamanhos totalmente distintos e que carregam consigo as características dos seus pais. Percebe-se que como o veículo 5 era o mais

comum entre os pais (utilizado três vezes em LV mostrado na Figura 3.14), é de se esperar que o mesmo comportamento se repita para o cromossomo filho.

Este operador, contudo, possui algumas limitações. A primeira delas é que na maioria das vezes a última das rotas terá seu tamanho editado de forma forçada – semelhante ao operador $CRX01$. Isto é, se fizermos a soma de todos os clientes atendidos em VR_{CRX02} teríamos um total de 16, entretanto, o exemplo possui apenas 15 pedidos. Isto faz com que a segunda rota do veículo 5 passe a atender cinco ao invés de seis clientes. Portanto o resultado final para VR_{CRX02} é 1-3-5-7-5-5.

Uma segunda limitação diz respeito à conservação de boas rotas. Como o VR de ambos os pais é totalmente fragmentado durante o operador $CRX02$, uma boa configuração de rotas pode ser perdida durante as operações de *crossover*.

Como já mencionado no início deste tópico, $CRX01$ e $CRX02$ se complementam ao cobrirem as limitações um do outro. Sendo assim, o AG ganha mais possibilidades de caminhos distintos no espaço de soluções.

Vemos que dois filhos distintos foram gerados a partir do mesmo par de pais. Sendo que o filho 01 utilizou os operadores ERX e $CRX01$, enquanto o filho 02 foi o resultado de ERX e $CRX02$. O vetor VC de ambos os filhos é idêntico, contudo, são soluções totalmente distintas pelo fato de não terem o mesmo VR. Portanto, essas foram duas soluções possíveis de serem criadas a partir da estrutura dos operadores *crossover* deste AG.

3.6 OPERADOR MUTAÇÃO

Como mencionado no Quadro 2, a mutação visa proporcionar maior diversidade ao grupo de indivíduos recém gerados através de mudanças pontuais e aleatórias nos seus cromossomos. Ao contrário do *crossover* que tem um padrão bem definido, a mutação é caótica, não seguindo uma regra rígida, podendo levar o AG a espaços totalmente novos (REEVES, 2003).

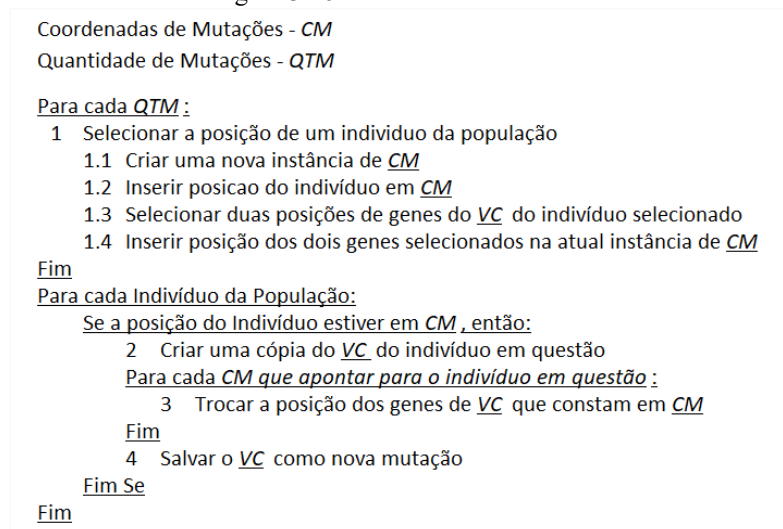
A sua principal característica é a de dar ao algoritmo alguma probabilidade de percorrer todos os locais do espaço de soluções. Isto é, a probabilidade do AG chegar em um dado ponto do espaço de soluções nunca é nula. Contudo esta característica da mutação faz com que o algoritmo leve mais tempo para convergir em soluções otimizadas (REINA, 2012).

Este tópico terá a mesma estrutura de explicação dos operadores crossover. Primeiramente será explicado como é feita a mutação do Vetor Clientes (VC) e logo em seguida o Vetor Rotas (VR).

3.6.1 Mutação do Vetor Clientes

Para realizar a mutação de VC, o AG aborda a troca dos genes que compõem os cromossomos avaliados. Para tal, cria-se uma lista contendo todos os cromossomos que serão mutados e os seus respectivos genes. Obviamente, a lista em questão seleciona aleatoriamente tanto os cromossomos quanto os seus genes. Ao final, é possível que um mesmo cromossomo sofra várias mutações. Na Figura 3.16 temos o procedimento para mutar o cromossomo VC dos indivíduos, aqui chamado de *MXVC*.

Figura 3.16 – Procedimento MXVC



Fonte: Autoria própria.

A Quantidade de Mutações *QTM* é determinada pelo parâmetro de mutação α . Este parâmetro é configurado manualmente pelo usuário e costuma variar de 0,02 a 0,15. A equação XX mostra como *QTM* é calculado e como α influência no valor final.

- *QTM* indica a quantidade total de mutações;
- α a taxa de mutação;
- *PTC* indica a população total de crossover;

- *QTG* indica a quantidade total de genes de VC.

$$QTM = \alpha * PTC * QTG \quad (3.7)$$

Isto é, *PTC* indica quantos indivíduos foram criados a partir dos operadores crossover da geração atual. Já *QTG* indica quantos genes um cromossomo VC possui. Esse valor é determinado pela quantidade de clientes que fizeram um pedido. Então, seguindo os mesmos exemplos do tópico anterior, a quantidade de clientes atendidos é 15, logo este será o valor de *QTG*.

Se adotarmos que: α é 0,05; então foram criados 150 indivíduos do *crossover*; e que 15 clientes serão atendidos, então *QTM* é igual a 112. Isto significa que dos 2.250 genes em VC dos indivíduos recém criados dos operadores crossover, 112 sofrerão alguma mudança.

Cada instância da variável *CM* (coordenada de mutações) possui três informações: posição do indivíduo na população *crossover*, posição do gene 01 e gene 02 do VC do respectivo indivíduo. Com estas três informações o *AG* consegue fazer a troca entre dois genes de um indivíduo específico. Portanto, cada instância de *CM* indica uma mutação.

Ao executar o procedimento *MXVC*, teríamos *QTM* igual a 112, logo, esta seria também a quantidade de instâncias *CM*. De todas as instâncias, apenas as seguintes são importantes no momento: $CM_{[5]}(10, 1, 3)$; $CM_{[71]}(10, 4, 5)$; $CM_{[80]}(10, 1, 14)$. Vamos supor que a posição 10 indicada nos *CM*'s acima represente o indivíduo Filho 02 que foi gerado a partir do operador *CRX02* no tópico anterior. Portanto o VC aqui abordado será o 9-999-10-99-64-65-3-39-11-63-21-44-921-12-314.

Teríamos então 3 mutações para o VC do Filho 02. A primeira, representada por $CM_{[5]}(10, 1, 3)$ nos informa que no indivíduo 10, as posições 1 e 3 do VC serão trocadas. Significa que os clientes 9 e 10 serão invertidos. Já o $CM_{[71]}(10, 4, 5)$ informa que ainda no indivíduo 10, os clientes 99 e 64 serão trocados (posições 4 e 5 respectivamente). Por último, $CM_{[80]}$ realiza a troca do cliente 10 por 12 nas posições 1 e 14 respectivamente. O resultado final do VC mutado é mostrado na Figura 3.17.

Figura 3.17 – Mutação do VC

Filho 02															
VC	9	999	10	99	64	65	3	39	11	63	21	44	921	12	314

Filho 02 Mutado															
VC	12	999	9	64	99	65	3	39	11	63	21	44	921	10	314

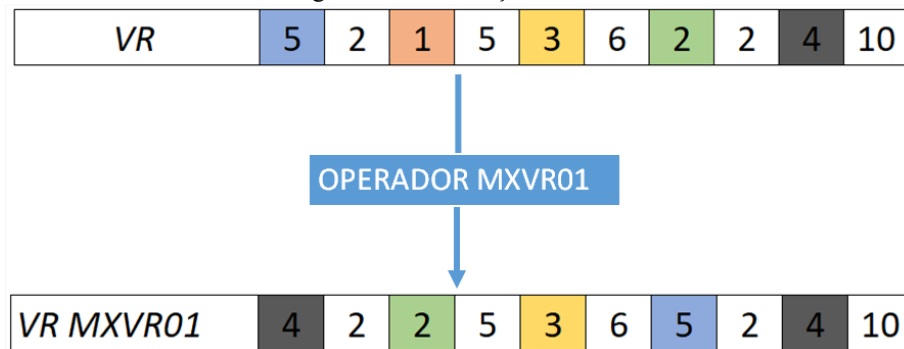
Fonte: Autoria própria.

Repare que no VC de ambos são muito parecidos, mas com pequenas mudanças provocadas pela mutação. Vale lembrar que o AG não irá substituir o VC do Filho 02, mas sim criar um segundo indivíduo com o VC recém mutado. Outro ponto importante é que o AG pode mutar o mesmo gene mais de uma vez. Isto aconteceu com a posição 01 onde tanto o $CM_{[5]}$ quanto o $CM_{[80]}$ fizeram a troca do mesmo gene.

3.6.2 Mutação do Vetor Rotas

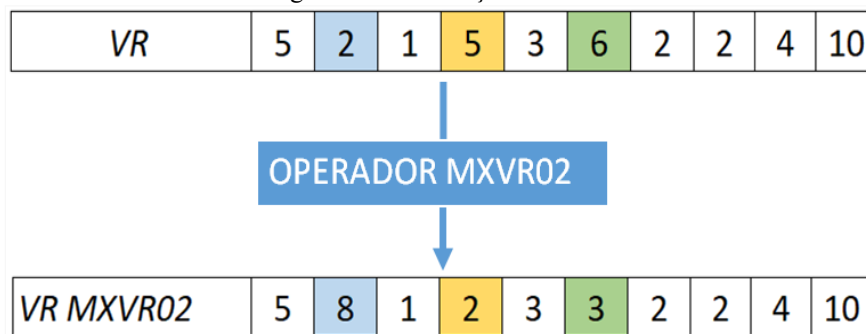
Esta etapa do AG é responsável por gerar a mutação no vetor rotas (VR). São utilizados quatro operadores com padrões distintos para essa etapa: *MXVR01*, *MXVR02*, *MXVR03* e *MXVR04*. Tantos operadores são necessários para que se mantenha o máximo possível de diversidade dos novos indivíduos criados.

O primeiro operador, *MXVR01*, realizará apenas a troca dos veículos em suas respectivas rotas em VR ao mesmo tempo que mantém as demais características como: sequência, paradas, lista de clientes e quantidade de clientes. A Figura 3.18 exemplifica melhor a implementação deste operador.

Figura 3.18 – Mutação *MXVR01*

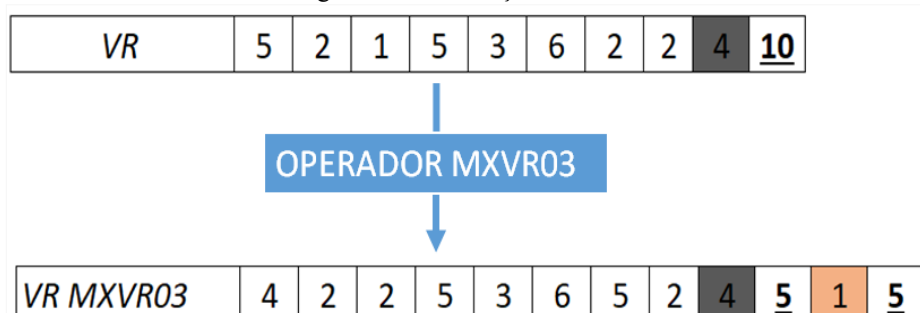
Fonte: Autoria própria.

O segundo operador, *MXVR02*, possui o mesmo princípio de *MXVR01*, contudo, ao invés de realizar trocas dos veículos, muda-se a quantidade de clientes atendidos em cada rota. As demais características da rota mutada são mantidas.

Figura 3.19 – Mutação *MXVR02*

Fonte: Autoria própria.

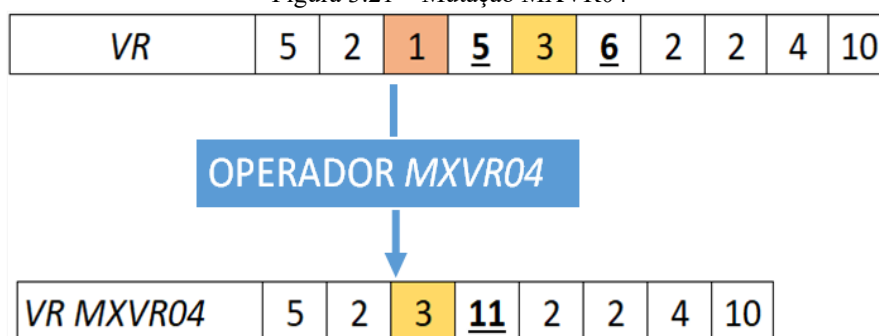
O terceiro operador, *MXVR03*, objetiva fragmentar uma determinada rota em duas. Isto é, se um veículo possui uma rota com 10 clientes, este operador irá criar uma segunda rota que inicie a partir do sexto cliente da rota do veículo e até o décimo. Sendo assim, a solução do indivíduo mutado poderá atender um maior volume de clientes em um intervalo de tempo menor. A Figura 3.20 nos mostra como funciona.

Figura 3.20 – Mutação *MXVR03*

Fonte: Autoria própria.

O último operador, *MXVR04*, é assimétrico ao *MXVR03* pois possui um comportamento totalmente oposto. Este operador buscará simplificar rotas, ao contrário do seu antecessor que as multiplica. Se dois veículos, *x* e *y*, possuírem rotas com 5 e 6 clientes respectivamente, este operador simplificará as duas rotas em apenas uma com 11 clientes. Somente um dos dois veículos será escolhido aleatoriamente para executar a nova rota unificada.

Figura 3.21 – Mutação *MXVR04*



Fonte: Autoria própria.

Naturalmente, todos os quatro operadores trabalham em conjunto para que o AG possa percorrer diferentes locais do espaço de soluções. Tanto o primeiro quanto o segundo operador são assimétricos, assim como o terceiro e quarto. Contudo, os quatro formam dois grupos de operadores que têm movimentos distintos, e cada operador atua em uma direção diferente, mas com o mesmo movimento do seu respectivo par.

Vale ressaltar que existe uma certa chance desses operadores serem executados durante o código. Essa chance é em função da taxa α de mutação que varia ao longo das gerações e cada operador tem sua probabilidade calculada independentemente dos demais.

3.7 GERAÇÃO DA NOVA POPULAÇÃO E CONDIÇÕES DE PARADA

Após o AG criar novos indivíduos a partir dos operadores, é realizada uma nova seleção com a mesma estratégia explicada no tópico 3.4. O torneio é composto apenas pelos novos indivíduos criados. Sendo assim, finaliza-se o ciclo de uma geração e o algoritmo repete todo o processo de crossover e mutação.

Este procedimento só será interrompido caso o AG chegue a um ponto de convergência onde a população não ofereça mais possibilidades de otimização. Para tal, se um determinado

número I de gerações forem criadas sem que haja evolução na FO do grupo, o algoritmo é finalizado. A FO é uma média simples das 20 melhores soluções da geração corrente.

O valor I de gerações é definido inicialmente pelo usuário e costuma variar entre 3 a 5. Além dessa restrição, é também inserida no código uma segunda condição de parada que é a quantidade de gerações criadas. Para o presente código o valor limite de novas foi de 120.

3.8 LEVANTAMENTO DOS DADOS

A empresa em questão, como dito anteriormente atua em dezoito cidades da região centro oeste brasileira, e possui uma carteira de clientes com em torno de 300 parceiros, dentre esses parceiros, um quarto deles contribuem com aproximadamente setenta por cento das vendas mensais, os demais fazem pedidos esporadicamente, ou seja, com um intervalo longo de tempo.

Faz-se necessário uma coleta de dados, esses dados compõem-se por parte da carteira de clientes da empresa, para que possamos analisar e comparar as distâncias percorridas das rotas de entrega que hoje a empresa efetua com os mesmos dados, porém inseridos no algoritmo genético.

Para ajustar os clientes e possuir uma base das informações de distância e tempo de todos eles, combinando-os um a um, foi utilizado uma *API* do *Google Maps* desenvolvida em linguagem *Python*.

A sigla *API* refere-se a Interface de Programação de Aplicativos, e é um conjunto de aplicações que permitem a construção de uma interface inteligente, codificando um meio pelo qual dois sistemas se comunicam. A *Google Maps API* tem uma plataforma dividida entre três produtos, o *Maps*, *Routes* e *Places*. Para facilitar a identificação dos endereços dos clientes foi utilizado uma das APIs disponíveis no *Places*, a *Geocoding* que permite converter endereços em coordenadas geográficas ou vice-versa. Foram consultados e salvos em um arquivo de entrada em formato *xlsx*, todos os endereços da carteira de clientes da empresa em questão. Em seguida, selecionou-se uma amostra de rotas realizadas pela empresa por um período de uma semana, contendo 11 rotas, nas quais foram atendidos 120 clientes, esses salvos em um arquivo de dados de entrada em formato *.xlsx*. A Tabela 3.1 é uma representação de como seria um arquivo de entrada de dados.

Tabela 3.1 – Endereços de clientes

ID	DESCRIÇÃO
01	Avenida '', Número '', Bairro '', Cidade '', Estado''.
02	Avenida '', Número '', Bairro '', Cidade '', Estado''.
03	Avenida '', Número '', Bairro '', Cidade '', Estado''.

Fonte: Autoria própria.

Com o banco de dados coletado, utilizou-se a programação da *API Geocoding* do *Google Maps* em linguagem de programação *Python*, a fim de converter os endereços em coordenadas geográficas. Os dados são gerados em um arquivo de saída em formato *.xlsx*, através da compilação do código descrito no Apêndice A. A Tabela 3.3 visa representar o arquivo em questão de forma genérica.

Tabela 3.2 – Coordenadas e endereços de clientes

ID	LOCAL	LAT	LNG	COORDENADA
0	Origem	-18,00001	-48,000001	-18,00001,-48,000001
01	Avenida '', Número '', Bairro '', Cidade '', Estado''.	-18,00002	-48,000002	-18,00002,-48,000002
02	Avenida '', Número '', Bairro '', Cidade '', Estado''.	-18,00003	-48,000003	-18,00003,-48,000003
03	Avenida '', Número '', Bairro '', Cidade '', Estado''.	-18,00004	-48,000004	-18,00004,-48,000004

Fonte: Autoria própria.

Após a criação de um banco de dados mestre contendo a coordenada de cada cliente da amostra, iniciou-se a segunda etapa da coleta de dados que foi o cruzamento das distâncias e seus respectivos tempos de rota entre todos os clientes com o intuito de se criar uma Matriz Distância e Tempo. Apesar de haver uma certa possibilidade das distâncias não representarem o real valor entre dois endereços, de forma geral a plataforma utilizada consegue oferecer aos usuários uma base de dados confiável e de forma totalmente acessível.

Para o cruzamento desses dados utilizou-se uma API disponível no *Places*, a *Google Maps Distance Matrix API*, essa interface estima a distância e tempo de percursos de vários destinos. Para os 120 endereços considerados na amostragem, obteve-se duas matrizes de 14.400 elementos, uma para distância e outra para tempo de viagem. A Tabela 3.3 a seguir demonstra um formato geral de como seria um arquivo com os dados de saída da matriz em formato *.xlsx*. Nele tem-se as distâncias e tempos de todos os clientes, gerado com a compilação do código descrito no Apêndice B.

Tabela 3.3 – Dados de distâncias e tempos de viagens

ID	ORIGEM	DESTINO	DISTÂNCIA	DURAÇÃO	COORDENADA_ORIGEM	COORDENADA_DESTINO
0	0	0	0	0	-18,00001,-48,000001	-18,00001,-48,000001
1	0	01	4240	460	-18,00001,-48,000001	-18,00002,-48,000002
2	0	02	14620	1483	-18,00001,-48,000001	-18,00003,-48,000003
3	0	03	3771	447	-18,00001,-48,000001	-18,00004,-48,000004
4	01	01	4450	604	-18,00002,-48,000002	-18,00002,-48,000002
5	01	02	14365	1448	-18,00002,-48,000002	-18,00003,-48,000003
6	01	03	7219	1039	-18,00002,-48,000002	-18,00004,-48,000004
7	02	02	11726	839	-18,00003,-48,000003	-18,00003,-48,000003
8	02	03	35710	2187	-18,00003,-48,000003	-18,00004,-48,000004
9	03	03	37337	2454	-18,00004,-48,000004	-18,00004,-48,000004

Fonte: Autoria própria.

Para a organização dos dados em uma estrutura matricial, utilizou-se o software *Excel*, com os respectivos códigos dos clientes transpostos em uma planilha, através de funções de busca, formou-se a base de dados de distâncias e tempos dos clientes.

Na Figura 3.22 vemos um trecho da Matriz Distância obtida no final da coleta. A coluna e linha destacadas de azul indicam os códigos dos clientes, já os elementos da matriz estão destacados de vermelho, e indicam as referidas distâncias em metros. Vale ressaltar que a matriz em questão possui dimensão 120 x 120. Além disso, tem-se uma segunda matriz com esta mesma estrutura, contudo, é composta apenas pelos tempos de viagem entre os clientes.

Figura 3.22 Matriz distância de clientes

	4	5	7	14	32	33	48	65	77
4	0	131861	1470	210302	300749	488235	218888	214915	460708
5	131861	0	133331	90545	187578	367026	105716	101537	347536
7	1470	133331	0	211772	302219	489705	220358	218890	462178
14	210302	90545	211772	0	121375	277328	25611	50058	281404
32	300749	187578	302219	121375	0	260757	96738	97122	159993
33	488235	367026	489705	277328	260757	0	288009	298319	204558
48	218888	105716	220358	25611	96738	288009	0	25421	256766
65	214915	101537	218890	50058	97122	298319	25421	0	257080
77	460708	347536	462178	281404	159993	204558	256766	257080	0

Fonte: Autoria própria.

3.9 OTIMIZAÇÃO DAS ROTAS

Com os dados geográficos de todos os clientes em mãos e com o algoritmo totalmente desenvolvido, iniciou-se a fase de otimização das rotas. Coletou-se cerca de 2 semanas de dados referentes aos roteiros de entregas da empresa.

Primeiro estimou-se a distância destas rotas que a empresa segue. Para tal, utilizou-se o banco de dados de distâncias explicado no tópico 3.8 desta seção. Quanto ao cálculo de tempo de atendimento utilizou-se uma média entre a quantidade de produtos entregues pela capacidade de descarregamento por funcionário. Outra medida importante foi o horário de trabalho dos motoristas que levou em conta o horário comercial de 07:00h às 18:00h.

Após a estimativa de distância e duração das rotas realizadas pela empresa, iniciou-se a etapa de otimização destas rotas no algoritmo desenvolvido. Para tal, o algoritmo tinha apenas os seguintes dados de entrada: a relação de clientes que foram atendidos naquele dia; os veículos disponíveis no dia; os dados de distâncias descritos no tópico 3.8. Com estes dados o algoritmo era então executado para gerar uma nova rota otimizada.

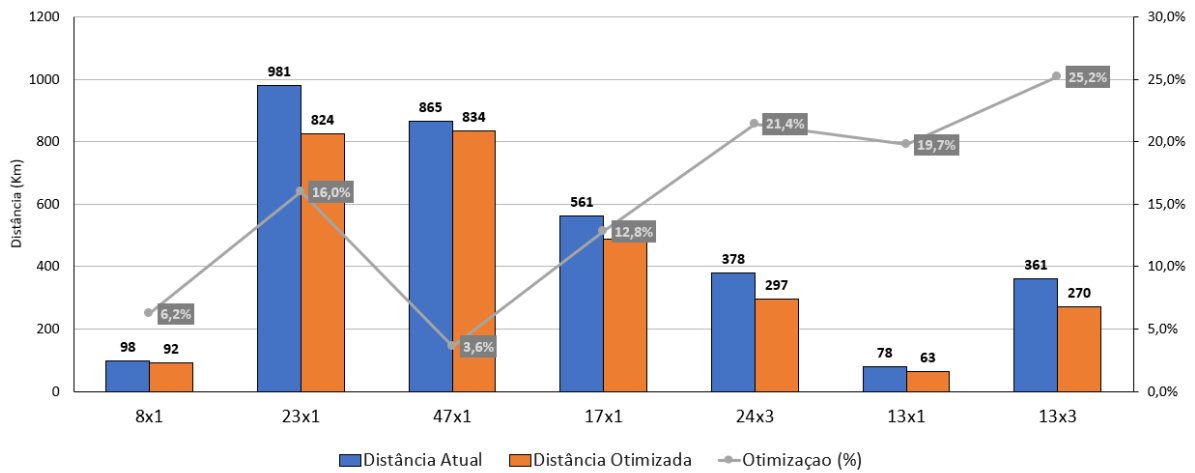
Sendo assim, teve-se duas rotas para comparação, uma realizada pela empresa e outra gerada pelo algoritmo. Portanto, foi através dessa comparação que pôde-se verificar o nível de otimização das distâncias.

Vale destacar que para cada rota analisada, realizou-se cerca de 5 simulações diferentes. A ideia é que se tenha uma boa noção da convergência do algoritmo. Por se tratar de um método estocástico, algumas pequenas variações já eram esperadas por parte do algoritmo.

4 RESULTADOS

Esta seção tem como objetivo detalhar os resultados obtidos nas instâncias coletadas. Vale lembrar que uma instância se refere ao subgrupo de clientes que serão atendidos por n viagens, e rota é a sequência de clientes a serem atendidos por um único veículo. Os valores de FO, como mencionado nos tópicos anteriores, é uma medida adimensional, contudo, tanto os seus valores quanto as distâncias serão consideradas nessa análise separadamente.

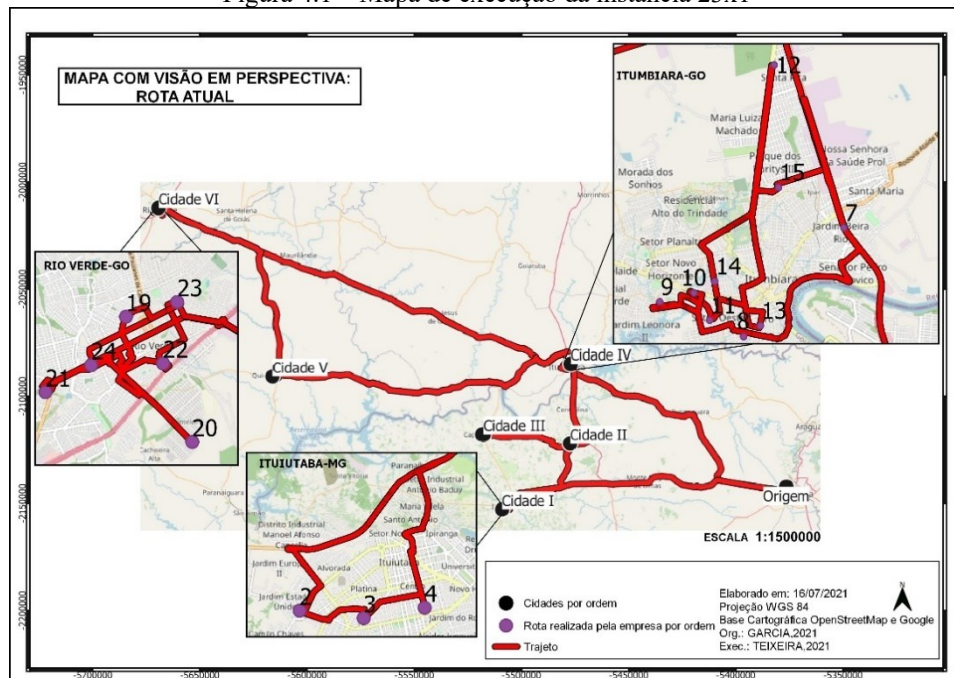
Gráfico 4.1 – Comparação das instâncias otimizadas



Fonte: Autoria própria.

No Gráfico 4.1, percebe-se inicialmente que a meta-heurística conseguiu otimizar todas as rotas analisadas. As instâncias 24x3 e 13x3 tiveram uma maior otimização relativa pelo fato de o algoritmo ter conseguido diminuir de 3 para 2 viagens em ambas. Já as instâncias 23x1, 47x1 e 17x1 foram as que tiveram clientes mais afastados da cidade de origem, e, portanto, tiveram maior otimização absoluta, variando de 154 a 35 km. As instâncias 8x1 e 13x1 foram executadas na mesma cidade da origem e levaram apenas uma viagem, portanto a otimização ficou entre 6 e 15 km.

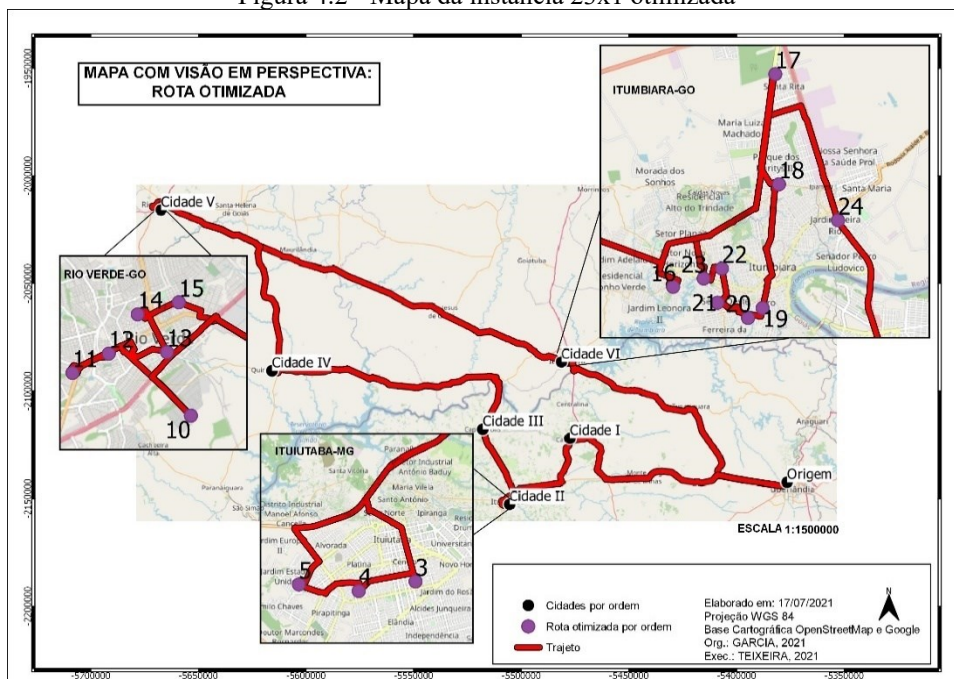
Figura 4.1 – Mapa de execução da instância 23x1



Fonte: Autoria própria.

Na Figura 4.1 vemos exatamente como a instância 23x1 é executada pela empresa. A primeira observação é que não há um padrão bem definido de qual deve ser a sequência de cidades a serem atendidas. Vemos que a cidade IV e a cidade II são atendidas duas vezes. O segundo ponto interessante é que dentro das cidades, também não se vê um padrão de roteamento bem definido.

Figura 4.2 - Mapa da instância 23x1 otimizada



Fonte: Autoria própria.

Após a otimização, como vemos na Figura 4.2, houve um novo arranjo da instância 23x1. A primeira modificação indica que a meta-heurística inverteu a ordem inicial das cidades. Como consequência, vemos que não há repetição de cidades na rota. Outro ponto interessante é ressaltar que a nova rota segue um padrão bem definido no entendimento dos clientes dentro das cidades.

Contudo, este padrão sofre alguma conturbação na otimização local dos clientes dentro das cidades. Vemos que na cidade VI da Figura 4.2 a ordem foi feita no sentido horário, passando pelo cliente 17 que está muito afastado da cidade. Se o algoritmo fizesse este mesmo trecho no sentido opostos, uma pequena melhoria ainda poderia ser alcançada. Felizmente este pequeno desvio é observado apenas em instâncias onde as cidades estão muito distantes e há muitos clientes.

De forma geral, verifica-se que há harmonia na otimização desta instância pois um círculo semelhante a uma “pétala” se forma sem que haja cruzamentos desnecessários no que diz respeito a cidades e clientes. Este comportamento é exatamente o que a teoria do PRV prevê.

Tabela 4.1 – Utilização dos veículos

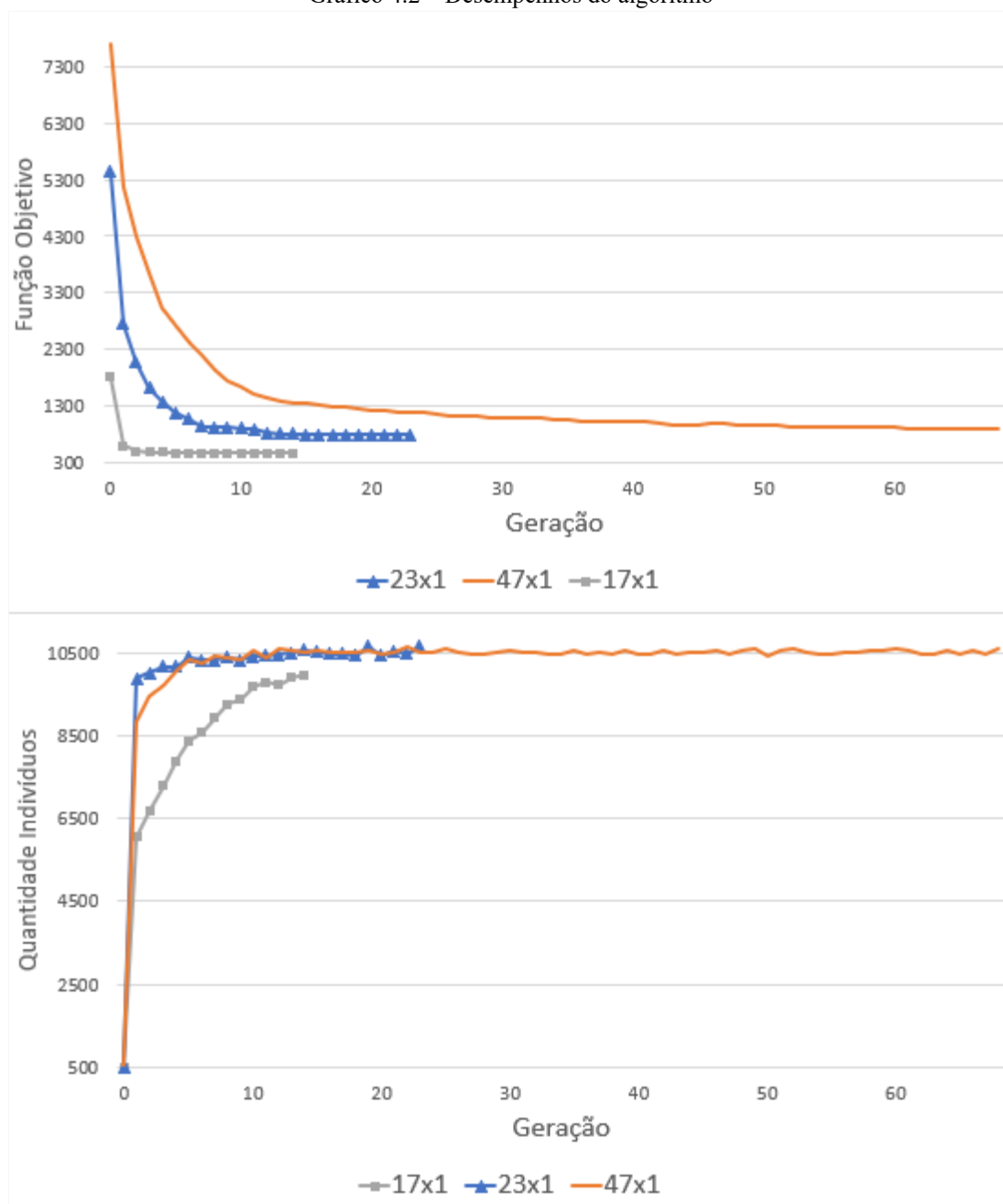
Rota atual				
Instância	Distância (Km)	Número de Veículos	Carga (Kg)	Capacidade (Kg)
8x1	98	1	480	870
23x1	981	1	3421	6220
47x1	865	1	3985	4240
17x1	561	1	723	870
24x3	378	3	178 - 823 - 556	680 - 870 - 870
13x1	78	1	870	870
13x3	361	3	98 - 682 - 322	680 - 680 - 870
Rota otimizada				
Instância	Distância (Km)	Número de Veículos	Carga (Kg)	Capacidade (Kg)
8x1	92	1	480	870
23x1	824	1	3421	6220
47x1	834	1	3985	4240
17x1	489	1	723	870
24x3	297	2	674 - 883	680 - 870
13x1	63	1	870	870
13x3	270	2	344 - 758	680 - 870

Fonte: Autoria própria.

Como vemos na Tabela 4.1 a utilização dos veículos foi otimizada nas instâncias 24x3 e 13x3 que passaram a utilizar apenas duas rotas ao invés de três. Consequentemente as capacidades também foram melhor distribuídas. Outra característica importante observada é

que na instância 24x3 otimizada, a Meta-Heurística aceitou uma ligeira punição ao ultrapassar 13 quilos de carga no veículo 2. Este tipo de comportamento já era esperado pois permite ao usuário algum grau de ajuste e maior flexibilidade no momento da implementação das rotas sugeridas.

Gráfico 4.2 – Desempenhos do algoritmo



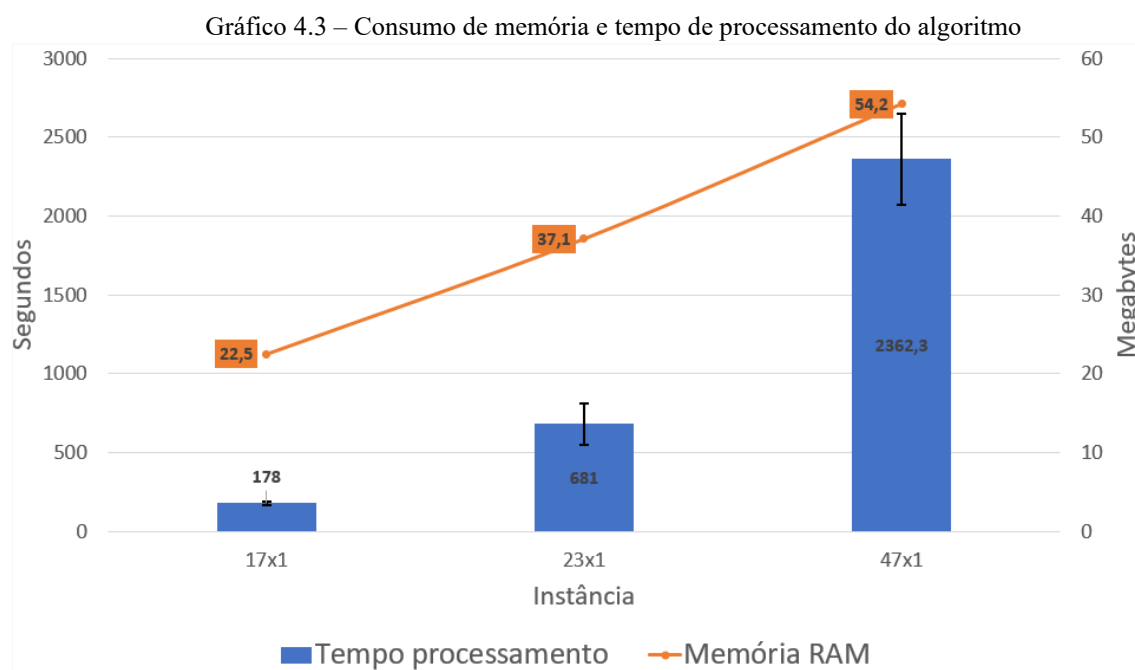
Fonte: Autoria própria.

No Gráfico 4.2 vemos que o Algoritmo Genético se comporta muito bem ao varrer grandes espaços de soluções nas gerações iniciais visto que uma curva muito acentuada é percebida no começo do gráfico de FO por Geração. Contudo, quando se chega a um certo

patamar vemos que há uma grande dificuldade em se manter a taxa de otimização mesmo com um acréscimo considerável na população.

Percebe-se que as curvas do gráfico FO por Gerações se assemelham a uma função logarítmica, e sua tendência ao longo das gerações é de permanecer em algum ótimo local, o que indica que o Algoritmo Genético convergiu para uma solução otimizada.

Outro ponto interessante a se observar é a quantidade de gerações criadas até a parada do algoritmo. Quanto maior for a quantidade de clientes da instância analisada, maior será também o número de gerações avaliadas até que se convirja para uma solução otimizada. Este comportamento implica em mais consumo de recursos computacionais e tempo. O Gráfico 4.3 mostra como foi o comportamento do algoritmo em relação ao tempo de processamento e utilização de memória RAM para tais instâncias.



Um dado interessante é observar que houve maior variabilidade nos tempos de processamento à medida que se tem instâncias mais complexas. O Mesmo comportamento é observado em relação às médias e desvios das FO's coletadas após várias execuções das mesmas instâncias. Portanto, quanto maior for a FO e o número de clientes da instância, maior será a variabilidade da solução otimizada.

Tabela 4.2 – Variabilidade do algoritmo

Rota otimizada				
Instância	Função Objetivo	Desvio Padrão	Distância (Km)	Desvio Padrão
8x1	87,7	±0	92,3	±0
13x1	59,5	±0	62,6	±0
13x3	257,6	±0,8	271,0	±0,8
17x1	465,1	±0	489,6	±0
23x1	784,3	±3,1	827,1	±2,5
24x3	294,3	±12	301,7	±6,8
47x1	858,2	±31,4	857,8	±17,1

Fonte: Autoria própria.

Outra análise importante é a de como foi o comportamento da meta-heurística em relação à dimensão temporal do problema. De acordo com as operações da empresa, considerou-se como janelas de tempo tanto o horário comercial normal como o horário específico que o cliente deseja ser atendido (matutino ou vespertino). Como quase todos os clientes exigem que o atendimento seja apenas durante o horário comercial, aqueles que necessitam de horários mais curtos são considerados críticos para o problema. A Tabela 4.3 mostra o número de atendimentos que deveriam ser feitos em horário comercial e atendimentos críticos com horários específicos.

Tabela 4.3 – Horários de atendimento dos clientes

Rota otimizada				
Instância	Horário comercial	Acertos	Horário específico	Acertos
8x1	8	8	-	-
23x1	18	17	5	5
47x1	47	46	-	-
17x1	17	17	-	-
24x3	23	23	1	1
13x1	13	13	-	-
13x3	13	13	-	-

Fonte: Autoria própria.

Como vemos na Tabela 4.3, a meta-heurística conseguiu organizar tanto os atendimentos em horários comerciais, como aqueles mais críticos, que totalizaram 6 clientes. Nas instâncias 23x1 e 47x1, dois clientes tiveram previsões de atendimentos fora do horário comercial. Isto acontece porque são rotas em cidades diferentes da cidade de origem, consequentemente, os motoristas precisam sair do ponto de origem antes do horário comercial. Contudo, esses atendimentos fora da janela ideal não ultrapassaram 30 minutos de antecipação e, portanto, não fogem do escopo das operações da empresa.

5 CONCLUSÃO

O presente trabalho propôs a otimização das rotas de entrega de uma empresa por meio de um algoritmo computacional meta-heurístico. Tal meta-heurística foi inspirada nos conceitos de Algoritmo Genético aplicados à otimização de distâncias percorridas por uma frota heterogênea em uma empresa que transporta produtos congelados. A metodologia abordada auxilia nas estratégias de rotas e políticas de entregas, além da flexibilidade de ajustar as necessidades operacionais como disponibilidade de veículos, recursos humanos e período de atendimento ao cliente.

Além disso, o algoritmo desenvolvido neste trabalho pôde atender às exigências de implementação da empresa referente ao tempo de otimização, uma vez que as instâncias analisadas são recorrentes e seguem quase sempre as mesmas demandas. O consumo de memória do sistema esteve dentro das exigências de não mais do que 0,5 GB RAM. Os indicadores de distância e tempo são mostrados com exatidão, atendendo às limitações de janelas de tempo, capacidade, disponibilidade dos veículos e previsões de atendimento.

Após a análise dos resultados estimou-se uma otimização total de 1900 km/mês, o que se traduz numa redução de 13,53 % em quilometragem rodada. Estes valores refletem em média uma economia anual de R \$18.300,00 de custos com combustíveis.

Além disso, pôde-se constatar que boa parte das contribuições vieram das instâncias mais complexas que tinham maior quantidade de clientes, cidades mais distantes entre si e múltiplas viagens. Após as otimizações, as instâncias 23x1, 47x1, 24x3 e 13x3 juntas representaram 720 km de economia por mês, cerca de 80% da otimização total obtida.

Para além das vantagens operacionais proporcionadas pelas instâncias otimizadas, este sistema pode atuar também como ferramenta de previsão de custos e estimativas de mão de obra, visto que as rotas são criadas de forma automática em questão de poucos minutos. Com estas simulações disponíveis em tempo real a gerência pode estimar a viabilidade de rotas futuras e seus impactos nas operações da empresa.

Notou-se que após as análises, a meta-heurística apresentou dificuldades para sair de ótimos locais. Este comportamento ficou mais evidente nas instâncias mais complexas que consumiram mais tempo de processamento e recursos computacionais. Este comportamento foi

de certa forma mitigado pelo fato de as otimizações terem superado as expectativas. Mas esperava-se que o algoritmo evitasse com maior facilidade este tipo de obstáculo.

Outro passivo deste trabalho foi a estimativa para o tempo de atendimento dos clientes. Devido à escassez de dados históricos dos horários de atendimento, não foi possível realizar um estudo estatístico mais aprofundado desta variável. Diante disto, os autores se limitaram a realizar tais cálculos levando em conta apenas o volume de produtos entregues.

Dito isto, novos estudos podem ser realizados com o desenvolvimento de Algoritmos Genéticos Híbridos com recursos mais avançados que visem contornar a problemática de ótimos locais. O uso de heurísticas de busca local é um bom caminho para solucionar o problema da otimização de clientes dentro das.

Outras oportunidades também podem ser exploradas com o uso de processamento paralelo para reduzir o tempo computacional de execução do algoritmo. Com isto, é possível realizar maiores relaxamentos nos parâmetros de controle populacional.

REFERÊNCIAS

Agência IBGE notícias. Editoria: Estatísticas Sociais, data 03 de abril de 2020. **Disponível em:** <http://www.ibge.gov.br>. Acessado em 21 set. 2020.

ANDRADE M. D., ARAGÃO JR. D. P. SANTANA R. **Um algoritmo para o problema de Roteirização de veículos com frota heterogênea.** XXXVIII ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO “A Engenharia de Produção e suas contribuições para o desenvolvimento do Brasil” - Maceió, Alagoas, Brasil, data: 16 a 19 de outubro de 2018. **Disponível em:** <http://www.abepro.org.br/>. Acessado em

ARENALES, M. et al. **Pesquisa Operacional: Para Cursos de Engenharia.** 2ª. ed. Rio de Janeiro: Elsevier: ABEPRO, v. I, 2015.

BALLOU, R. H. **Gerenciamento da Cadeia de Suprimentos/Logística Empresarial.** Tradução de Raul Rubenlch. 5ª. ed. Porto Alegre: Bookman, v. I, 2006.

BARBOZA, ANGELA OLANDOSKI. **Simulação e técnicas da computação evolucionária aplicadas a problemas de programação linear inteira mista.** 2005. 217 f. Tese (Doutorado em Engenharia Elétrica e Informática Industrial) – Universidade Tecnológica Federal do Paraná, Curitiba, 2005.

BATISTUS, D. R. **O problema de Roteirização de Veículos com Restrições de Janelas de Tempo: Uma Proposta de Solução por Meio da Metaheurística Scatter Search.** Tese de doutorado em Engenharia de Produção da Universidade Federal de Santa Catarina. Florianópolis, p. 126. 2009. Disponível em: <http://repositorio.ufsc.br/xmlui/handle/123456789/92680>.

BERTAGLIA, P. R. **Logística e Gerenciamento da Cadeia de Abastecimento.** 3. ed. São Paulo: Saraiva, v. I, 2016.

BEVILAQUA, A. **Algoritmo memético paralelo baseado em ilhas com busca local Lin-Kernighan para solução do Problema de Roteamento de Veículos Heterogêneos em Duas Fases inspirado em um modelo de distribuição usado por atacadistas.** Tese de Doutorado em Engenharia Elétrica da Universidade Federal de Uberlândia. Uberlândia, p. 108. 2019. Disponível em: <http://dx.doi.org/10.14393/ufu.te.2019.5>.

BOWERSOX, D. J.; CROSS, D. J.; COOPER, B. M. **Gestão da Cadeia de Suprimentos e Logística.** Rio de Janeiro: Elsevier, 2007.

CHOPRA, S.; MEINDL, P. **Gerenciamento da Cadeia de Suprimentos.** 8. ed. São Paulo: Person Prentice Hall, v. I, 2003.

CUNHA, C. B. **Aspectos Práticos da Aplicação de Modelos de Roteirização de Veículos a Problemas Reais.** Revista Transportes, Rio de Janeiro, v. 8, n. 2, p. 51-74, 2000.

GOLDBARD, M. C.; LUNA, H. P. L. **Otimização Combinatória e Programação Linear: Modelos e Algoritmos.** 2ª. ed. Rio de Janeiro: Elsevier, 2005.

GOLDBERG, D. E. Genetic Algorithms: in Search, Optimization & Machine Learning. 1ª. ed. [S.l.]: Addison-Wesley Publishing Company, 1989. Disponível em: http://www2.fiit.stuba.sk/~kvasnicka/Free%20books/Goldberg_Genetic_Algorithms_in_Search.pdf

HILLIER, F. S.; LIEBERMAN, G. J. **Introdução à Pesquisa Operacional**. Tradução de Ariovaldo Griesi. 8ª. ed. São Paulo: McGraw-Hill, v. I, 2006.

MARQUES, CÍCERO FERNANDES. **Pesquisa sobre a distribuição de alimentos resfriados e congelados**. Dissertação (mestrado) - Universidade Federal de Santa Catarina, centro Tecnológico. Programa de Pós-graduação em Engenharia de Produção, p. 89. 2008. Disponível em: <http://repositorio.ufsc.br/xmlui/handle/123456789/92127>.

MELO, A. C.; FILHO, V. J. F. Sistemas de Roteirização e Programação de Veículos. **Revista Brasileira de Pesquisa Operacional**, Rio de Janeiro, v. 21, n. 2, p. 223-232, julho 2001. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382001000200007&lng=pt&nrm=iso.

LOPES H. S., RODRIGUES L. A., STTEINER M. T. A.. **Meta-heurística em Pesquisa Operacional** /- Curitiba, PR: Omnipax, 2013. 472 p.:il 30cm. Vários autores / Inclui bibliografias ISBN: 978-85-64619-10-4 e ISBN: 978-85-64619-11-1. Citados: RESENDE M.G.C. & SILVA R.M.A. Disponível em: <http://www.omnipax.com.br>.

PLATT, ALLAN AUGUSTO. **Logística e cadeia de suprimentos**. 2. ed. – Florianópolis: Departamento de Ciências da Administração/ UFSC, 2013. Disponível em: http://arquivos.eadadm.ufsc.br/EaDADM/UAB_2011_1/Modulo_6/Logistica_Cadeia_Suprimentos/material_didatico/logistica_e_cadeia_de_suprimentos-online.pdf. Acesso em 30 de set. de 2020.

PRINS, C. **A simple and effective evolutionary algorithm for the vehicle routing problem**. Computers and Operations Research. Elsevier. v.31. 2004

REEVES, C. GENETIC ALGORITHMS. In: GLOVER, F.; KOCHENBERGER, G. A. Handbook of Metaheuristics. [S.l.]: Kluwer Academic Publishers, 2003. Cap. 3, p. 55-82.
REINA, C. D. **Roteirização de Veículos com Janelas de Tempo Utilizando Algoritmo Genético**. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia e Transportes. São Paulo, p. 90. 2012. Doi: 10.11606/D.3.2012.tde-06062013-162636.

REIS, J. V. A. et al. **Estudo de Viabilidade de Uma Nova Política de Entrega para uma Fábrica de Salgados Utilizando Problema de Roteirização de Veículos**. Encontro Nacional de Engenharia de Produção, Santos, 15 a 18 de outubro de 2019.

REIS, J. V. A. **Meta-Heurísticas Baseadas em Busca em Vizinhança Variável Aplicadas a Problemas de Operação de Transportes**. Tese (Doutorado em Engenharia de Transportes) - Escola Politécnica, Universidade de São Paulo. São Paulo, p. 221. 2013. Doi: 10.11606/T.3.2013.tde-22092014-150121.

SOLOMON, M.M. **Algorithms for vehicle routing and scheduling problems with time window constraints**. Operations Research. Informs. v.35. 1987. Doi: <https://doi.org/10.1287/opre.35.2.254>.

SOUZA, M.F.J. **Inteligência computacional para otimização**. Departamento de Computação, Instituto de Ciências Exatas Biológicas, Universidade Federal de Ouro Preto, 35400-000 Ouro Preto, MG. Disponível em: <http://www.decom.ufop.br/prof/marcone,E-mail:marcone@iceb.ufop.br>.

TAHA, H. A. **Pesquisa Operacional**. Tradução de Arlete Simille Marques. 8ª. ed. São Paulo: Pearson Prentice Hall, v. I, 2008.

TORQUATO, M. F. **Proposta de Implementação Paralela de Algoritmo Genético em FPGA**. Dissertação (Mestrado em Engenharia Elétrica e de Computação) - Universidade Federal do Rio Grande do Norte - Centro de Tecnologia. Natal, p. 74. 2017. Disponível em: <https://repositorio.ufrn.br/jspui/handle/123456789/24729>.

VECCHI, T. P. B. **Otimização de Rotas dos Caminhões de Coleta e Resíduos Sólidos Urbanos**. Tese (Doutorado em Engenharia Química) - Universidade Estadual de Maringá. Maringá, p. 136. 2016. Disponível em: <http://repositorio.uem.br:8080/jspui/handle/1/3664>.

APÊNDICE

Apêndice A - Programação em Python da API Google Maps: Coordenadas geográficas.

```
import googlemaps
import pandas as pd

gmaps = googlemaps.Client(key='')

vetor_Colunas = ['ID', 'LOCAL', 'LAT', 'LNG']
matriz = pd.DataFrame(columns=vetor_Colunas)

matriz_Entrada = pd.read_excel('Arquivos_Entrada/Entrada.xlsx')

for index, ponteiro in matriz_Entrada.iterrows():
    geocode_result = gmaps.geocode(ponteiro['DESCRIÇÃO'])
    lat = geocode_result[0]['geometry']['location']['lat']
    lng = geocode_result[0]['geometry']['location']['lng']
    matriz_temporaria =
pd.DataFrame([[ponteiro['ID'], ponteiro['DESCRIÇÃO'], lat, lng]], columns=vetor_Colunas)
matriz = matriz.append(matriz_temporaria, ignore_index=True)

matriz_Saida=
pd.ExcelWriter('Arquivos_Entrada/Saida_Coordeanda_Lat_LNG.xlsx')
matriz.to_excel(matriz_Saida, "Aba_01")
matriz_Saida.save()
```

Apêndice B - Programação em Python da API Google Maps Distance Matrix e Tempo.

```
import googlemaps
import pandas as pd

gmaps = googlemaps.Client(key='')

origem = pd.read_excel('arquivos/origem.xlsx')
destino = pd.read_excel('arquivos/destino.xlsx')
saida = pd.ExcelWriter('arquivos/Saida.xlsx')
Vetor_Colunas =
['ORIGEM', 'DESTINO', "DISTANCIA", 'DURACAO', 'COORDENADA_ORIG', 'COORDENADA_DES
T']
Matriz = pd.DataFrame(columns=Vetor_Colunas)
for index, l_origem in origem.iterrows():
    fazer_busca = False
    for index, l_destino in destino.iterrows():
        if l_destino['ID'] == l_origem['ID']:
            fazer_busca = True
```

```

        if fazer_busca:
            try:
                consulta =
gmaps.distance_matrix(l_origem['COORDENADA'],l_destino['COORDENADA'])
                distancia = consulta ['rows'][0] ['elements'][0]
['distance']['value']
                duracao =
consulta['rows'][0]['elements'][0]['duration']['value']
            except :
                distancia = ''
                duracao = ''
                Matriz_Temporaria = pd.DataFrame([[l_origem['ID'],
l_destino['ID'], distancia,
duracao,l_origem['COORDENADA'],l_destino['COORDENADA']]],columns=Vetor_Colu
nas)
                Matriz = Matriz.append(Matriz_Temporaria,ignore_index=True)

Matriz.to_excel(saida,"Dados")
saida.save()

```