UNIVERSIDADE FEDERAL DE VIÇOSA

CAMPUS FLORESTAL

DÉLIO CORNÉLIO QUEIROZ DA COSTA

PROBLEMA DE CAIXEIRO VIAJANTE COM JANELA DE TEMPO: PROPOSTA DE APLICAÇÃO NA MADEIREIRA SÃO GERALDO DE ITAÚNA-MG

FLORESTAL – MINAS GERAIS

2017

DÉLIO CORNÉLIO QUEIROZ DA COSTA

PROBLEMA DE CAIXEIRO VIAJANTE COM JANELA DE TEMPO: PROPOSTA DE APLICAÇÃO NA MADEIREIRA SÃO GERALDO DE ITAÚNA-MG

Monografia, apresentada ao Curso de Ciência da Computação da universidade Federal de Viçosa como requisito para obtenção do título de bacharel em Ciência da Computação.

Orientador: Marcus Henrique Soares Mendes

FLORESTAL – MINAS GERAIS

2017

DÉLIO CORNÉLIO QUEIROZ DA COSTA

PROBLEMA DE CAIXEIRO VIAJANTE COM JANELA DE TEMPO: PROPOSTA DE APLICAÇÃO NA MADEIREIRA SÃO GERALDO DE ITAÚNA-MG

Monografia, apresentada ao Curso de Ciência da Computação da universidade Federal de Viçosa como requisito para obtenção do título de bacharel em Ciência da Computação.

_____

Marcus Henrique Soares Mendes

_____

Maria Amélia Lopes Silva

_____

Ronan D. Mendonça

FLORESTAL – MINAS GERAIS

2017

RESUMO

O Problema do Caixeiro Viajante com Janela de Tempo (PCVJT) é um problema de otimização muito estudado atualmente devido à economia gerada nos gastos do transporte e também no preço final dos produtos. Este trabalho apresenta uma proposta de resolução de um PCVJT real identificado na empresa Madeireira São Geraldo de Itaúna Ltda, mais especificamente no serviço de entregas. Todos os pedidos possuem janelas de tempo referentes ao tempo de entrega, como antes ou depois de um determinado horário. A solução para o problema é encontrada por meio de um algoritmo desenvolvido utilizando os conceitos de Algoritmos Genéticos (AGs). Para demonstrar a eficiência do algoritmo, são feitos três experimentos diferentes, variando tamanho da população e número de indivíduos. No primeiro, o resultado é comparado a um *benchmark* com características similares; e nos outros dois, os resultados obtidos para o problema real são comparados aos resultados de um algoritmo guloso simples. No primeiro experimento o algoritmo desenvolvido encontrou soluções bem próximas ou iguais à ótima. Já no segundo e terceiro, os custos encontrados pelo algoritmo são até 56,55% da solução gulosa.

Palavras Chave: Roteamento de veículos com janela de tempo, Algoritmos Genéticos, Otimização.

ABSTRACT

The Traveling Salesman Problem with Time Window (TSPTW) is an important optimization problem because of the savings generated in transportation costs and the final price of the products. This paper presents a proposal for the resolution of a real identified TSPTW in the company Madeireira São Geraldo de Itaúna Ltda., more specifically in the delivery service. All requests have time windows for delivery time, such as before or after a certain time. A solution to the problem is found through an algorithm developed using the concepts of Genetic Algorithms (GAs). To demonstrate the efficiency of the algorithm, three different experiments are performed, varying the population size and number of individuals. In the first, the result is compared to a benchmark with similar characteristics; And in the two other, the obtained results for the real problem are compared to the result of a simple greedy algorithm. In the first experiment the algorithm found solutions close to or equal to optimal. In the second and third experiments, the costs found by the algorithm are 56.55% of the greedy solution.

Keywords: Vehicle Routing Problem with Time Window, Genetic Algorithms, Optimization.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Muitas empresas possuem serviço de entrega próprio, e o custo para oferecer esse serviço pode ser relativamente alto, como é o caso da Madeireira São Geraldo de Itaúna Ltda. Esta é uma empresa de pequeno porte, com onze funcionários e está no mercado desde 2011. Nesta empresa, as entregas são feitas utilizando caminhões próprios e, dado o peso do produto vendido, o gasto com combustível é alto. Consequentemente, causa um aumento nas despesas e no preço final do produto. Quanto à roteirização, esta é feita manualmente, o que requer muito trabalho, além de um funcionário dedicado exclusivamente para esta tarefa.

Vieira cita [BRÄYSY;GENDREAU; 2005] em seu trabalho dizendo “dentre todos os processos envolvidos na cadeia logística, o transporte é aquele que absorve a maior parcela do custo, atingindo de um a dois terços do valor total” [VIEIRA, 2013, p.1].

Tendo isso em vista, algumas formas de minimizar os gastos com esse serviço são: escolher a melhor rota possível e agrupar mais de um pedido em uma mesma viagem, reduzindo assim o número total de viagens. Neste trabalho aborda-se apenas a primeira estratégia. Porém os pedidos possuem uma janela de tempo para entrega, portanto deve-se considerar também o tempo gasto no percurso. O que classifica o problema como um Problema do Caixeiro Viajante com Janela de Tempo (PCVJT) é o fato de o sistema considerar apenas uma única rota para tratar o problema. Sendo uma variante do clássico PCV (Problema de Caixeiro Viajante), o PCVJT se diferencia dos demais exatamente por considerar também as janelas de tempo para que os clientes sejam atendidos.

Devido ao grande número de soluções possíveis, o custo computacional para encontrar a melhor solução é muito alto. Porém, através de técnicas heurísticas é possível encontrar uma solução no mínimo boa, próxima ao ótimo, em um tempo satisfatório. Dentre vários métodos para solucionar esse problema estão os Algoritmos Genéticos (AGs), que foi o método utilizado. Outros métodos comuns na literatura também são demonstrados a seguir.

1.1. Objetivo

Este trabalho propõe o desenvolvimento de um algoritmo para auxiliar na organização de entregas na Madeireira São Geraldo de Itaúna Ltda. O software objetiva fornecer qual a melhor rota a ser utilizada para entrega de uma lista de pedidos fornecida pelo usuário, de modo que minimize a distância percorrida e o tempo de atraso.

1.1.1. Objetivos específicos

Para atingir o objetivo geral do trabalho, destacam-se os seguintes objetivos específicos:

- Levantar bibliografia sobre resolução do problema de roteamento de veículos.
- Elaborar o grafo que representa o mapa de Itaúna.
- Implementar o sistema.
- Testar o sistema utilizando parâmetros de um *benchmark* e comparação dos resultados.

1.2. Justificativa

Com a utilização de técnicas computacionais, é possível reduzir o gasto da empresa com as entregas, visto que a escolha de uma boa rota impacta diretamente no consumo de combustível, o que gera economia para a mesma. Além disso, como o tempo limite para entrega dos pedidos também é levado em consideração para escolha da rota a ser utilizada, os atrasos também são reduzidos, o que consequentemente aumenta a satisfação do cliente com o serviço prestado.

Existem exemplos de aplicações reais de soluções computacionais na solução de problemas de roteamento, como é o caso do artigo [CARMO; GOMES; BARROS NETO, 2003]. Os autores utilizaram um Sistema de Informações

Geográficas para Transporte (SIG-T), chamado TransCad 3.0, para reduzir as distâncias e os tempos das rotas de entregas e, consequentemente, o custo da operação.

Vale citar também o trabalho [RIBEIRO; RUIZ; DEXHEIMER, 2001]. Neste trabalho os autores utilizaram o algoritmo proposto por Clarke e Wright (1964), que “baseia-se no conceito de ‘ganho’ que pode ser obtido ao se ligar dois nós de forma sucessiva em um roteiro” [RIBEIRO; RUIZ; DEXHEIMER, 2001]. Eles optaram por este método devido a sua simplicidade de implementação e por fornecer bons resultados.

Em ambos os casos os resultados encontrados foram satisfatórios, ou seja, os sistemas geraram soluções melhores para seus respectivos problemas. No primeiro exemplo os autores também aproveitaram o estudo para avaliar a ferramenta TransCad 3.0 e identificaram duas “deficiências” no sistema e propuseram soluções para as mesmas, a fim de aperfeiçoar a ferramenta.

Além disso, o algoritmo pode atender outras empresas. Se necessário, pode também ser facilmente adaptado para outras regiões alterando apenas o grafo (mapa) utilizado.

1.3. Organização do Trabalho

Este trabalho está organizado da seguinte forma: O capítulo 2 apresenta o que é o PRV, suas variações e sua importância, bem como os métodos para resolvê-lo. O Capítulo 3 apresenta o funcionamento do algoritmo desenvolvido. O Capítulo 4 apresenta os resultados obtidos, tanto para nos testes para o bechmark quanto para o problema real. Por fim, o Capítulo 5 finaliza com a conclusão.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica acerca dos PRVs e do PCV, explicitando também alguns tipos mais comuns de ambos. Além disso, traz também as técnicas de soluções comumente utilizadas na literatura, conceitos de AGs e seu funcionamento. Por fim, o capítulo apresenta uma seção de trabalhos relacionados.

2.1. Problema de Roteamento de Veículos

O Problema de Roteamento de Veículos é, de forma simples, um problema que envolve a distribuição de bens ou serviços de um deposito central para os usuários. Proposto em [DANTIZ; RAMSER, 1959], vem sendo bastante estudado desde então, devido a sua alta complexidade e grande variedade de problemas reais a ele associados. Como exemplos de aplicações reais estão:

- A escolha de rotas de transporte escolar ou de empresas.
- A distribuição de produtos de centros de deposito.
- Entrega em domicilio de produtos comprados *on-line*.

Suponha que tenha a disposição uma frota de veículos para transportar mercadorias demandadas ou ofertadas por um grupo de clientes. Além disso, suponha também que cada veículo esteja em um depósito. O Problema de Roteamento de Veículos (PRV) consiste em determinar a rota que cada veículo deve seguir, de modo que todos os clientes sejam atendidos, e que cada veículo volte ao depósito de origem no fim do processo. O objetivo é minimizar o custo total, que é dado pelo somatório dos custos de cada rota. [VIEIRA, 2013]. A Figura 1 mostra um exemplo de solução para um PRV.

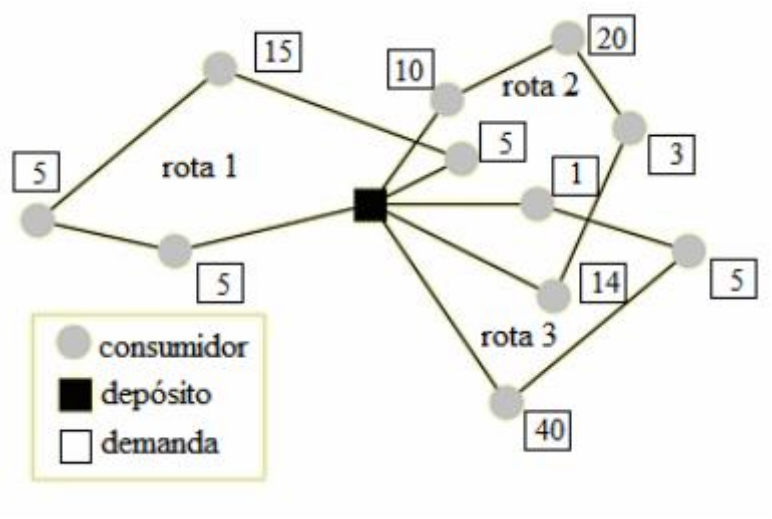

Figura 1: Possível solução para um PRV. Fonte: [Oliveira, 2007]

Seja um grafo G = (V, A) completo, em que A é o conjunto de arcos que representam os caminhos que ligam os clientes entre si e ao depósito, e V é o conjunto de $n+1$ vértices (0, ..., n), convencionalmente 0 representa o depósito e os demais representam os n clientes. A cada aresta $(i \mid j)$ é associado um custo não negativo $c_{i,j}$, que representa o custo de viagem do vértice i ao vértice j.

Existem muitas extensões para o PRV, que consideram a capacidade dos veículos (PRVC), intervalos de tempo (PRVJT), frotas heterogêneas (PRVFH), entre outros, que são mostrados a seguir.

2.1.1. Problema de Roteamento de Veículos Capacitado (PRVC)

Uma frota K de veículos idênticos, com capacidade C, é alocada a um mesmo depósito. Cada cliente i é associado a uma demanda não negativa m_i. Para o depósito é definido $m_0 = 0$. O objetivo é encontrar um conjunto de K rotas, a fim de minimizar o custo total do transporte e obedecendo as seguintes restrições:

- Toda rota deve ter inicio e fim no deposito;
- Cada cliente deve ser visitado apenas uma vez por um único veículo;
- A demanda total de cada rota não deve ser maior que a capacidade do veículo.

2.1.2. Problema de Roteamento de Veículos com Frota Heterogênea (PRVFH)

Este problema diferencia-se do PRVC principalmente porque cada veículo da frota pode ter sua própria capacidade C. Existem duas variações para este problema, que são com a quantidade de veículos de cada tipo pré-estabelecida ou não. Na primeira é preciso apenas atribuir uma rota para cada veículo. Porém na segunda, além da rota, é necessário também determinar quantos veículos serão utilizados. Essa ultima variação é denominada Problema de Dimensionamento e Roteamento de Frota Heterogênea de Veículos (ou *Fleet Size and Mix Vehicle Routing Problem*) [VIEIRA, 2013].

2.1.3. Problema de Roteamento de Veículos com Coleta e Entrega (PRVCE)

Nesta versão do PRV, os clientes podem receber itens, entregar itens ou até mesmo ambas. Além disso, o item coletado em um cliente pode ser reaproveitado e entregue a outro cliente. Por isso não há obrigatoriedade de que todas as entregas sejam feitas antes das coletas ou vice-versa.

A cada cliente i são associados dois valores, m_i e p_i, que correspondem respectivamente à quantidade de produtos a serem entregues e coletados pelo veículo. Associam-se também a cada cliente outros dois valores: O_i e D_i. O primeiro indica de quem o cliente i receberá seu pedido, já o segundo indica o destino do produto entregue pelo cliente i. Portanto, o cliente O_i deve ser

atendido antes do cliente i, que por sua vez deve preceder o cliente D_i. Quanto ao depósito, pode ser definido tanto como cliente O_i quanto como cliente D_i [VIEIRA, 2013].

2.2. Problema de Roteamento de Veículos com Janela de Tempo (PRVJT)

> "Assume-se que cada consumidor de recursos possui um tempo pré-determinado para ser atendido por um dos veículos da frota do depósito central. Isso faz com que o atendimento sobre cada consumidor ocorra em uma 'janela de tempo', sendo este o fator que nomeia o problema." [OLIVEIRA, 2007].

Marinho define o PRVJT como "uma variação do PRVC em que são adicionadas restrições relacionadas ao instante de tempo em que cada cliente deve ser atendido" [MARINHO, 2013].

Para todo cliente i, existe uma demanda q_i que deve ser atendida. Também é associado a cada cliente um intervalo de tempo $[a_i, b_i]$, denominada janela de tempo, que indica o horário em que o atendimento ao cliente i deve iniciar, bem como o tempo de serviço s_i, que é o tempo gasto para concluir o atendimento ao cliente. A cada aresta (i, j) é definido um tempo t_{ij} que representa o tempo gasto no percurso entre os clientes i e j. O tempo de serviço s_i só é contabilizado dentro da janela de tempo do cliente e após a chegada do veículo ao mesmo. Caso o veículo chegue antes ao cliente i, ele pode esperar até o início da janela de tempo do respectivo cliente.

Assim como para os clientes, define-se também uma janela de tempo $[a_0, b_0]$ para o depósito i=0, que representa o instante em que o depósito abre e fecha respectivamente. Com isso temos também um intervalo de tempo r que é o tempo de duração máxima das rotas.

Dada uma frota de veículos homogênea M, com capacidade de carga Q, o PRVJT busca encontrar o conjunto de rotas com menor custo. Dentre as restrições do problema, além das citadas no PRVC, destacam-se as seguintes:

- Caso o veículo chegue ao cliente antes da abertura da janela de tempo, o veículo deve aguardar, o que caracteriza um tempo de espera.
- Ao visitar um cliente o veículo deve aguardar até o fim do tempo de serviço s_i.
- Todos os veículos devem voltar ao depósito até o instante b_0.

2.3. Problema do Caixeiro Viajante

"Dado um conjunto de n cidades e uma matriz de distâncias d_{ij} entre elas, o Problema do Caixeiro Viajante (PCV), ou *Traveling Salesman Problem* (TSP), consiste em estabelecer uma rota para um Caixeiro, iniciando seu percurso em uma cidade, chamada cidade origem, passar por todas as demais $n-1$ cidades uma única vez e retornar à cidade origem percorrendo a menor distância possível". [SOUZA, 2009]

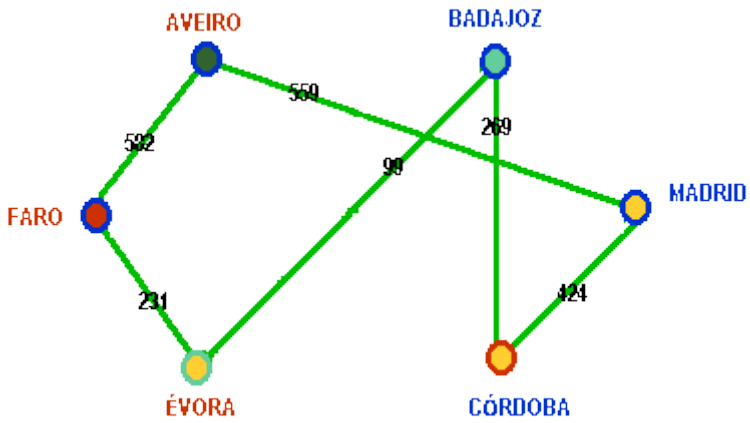

Figura 2: Exemplo de solução para um PCV.

Assim como no PRV, o PCV também é um problema clássico da computação e possuí variações, e algumas delas são citadas a seguir.

- Problema dos m-Caixeiros Viajantes:

Nesta variante do PCV há m Caixeiros e o objetivo é minimizar a distância total percorrida por todos eles.

- Problema do Caixeiro Viajante com Coleta Seletiva de Prêmios (PCVCSP):

O problema pode ser associado a um caixeiro viajante que coleta um prêmio p_k, não negativo, em cada cidade k que ele visita e paga uma penalidade γ_k para cada cidade k que ele não visita, com um custo c_{ij} de deslocamento entre as cidades i e j. O problema encontra-se em minimizar o somatório dos custos da viajem e penalidades, enquanto inclui na sua rota um número suficiente de cidades que o permita coletar um prêmio mínimo p_{min} pré-estabelecido.

- Problema do Caixeiro Viajante com Janela de tempo (PCVJT):

Para cada cliente i atribui-se uma janela de tempo $[a_i, b_i]$. Para cada aresta (i, j) atribui-se um tempo t_{ij}, que corresponde ao tempo gasto para percorrê-la.

2.4. Técnicas de Solução

Na literatura existem diversas técnicas e estratégias para solucionar o PRVJT, dentre as quais algumas são citadas a seguir. Podemos dividir as técnicas em três grupos:

- Métodos Exatos: *Branch-and-Bound*;
- Heurísticas: *Savings, Push-foward Insertion Heuristic*
- Meta-heurísticas: Busca Tabu, *Simulated Annealing,* Algoritmos Evolucionários, Meta-heurísticas híbridas e Algoritmos genéticos.

> "Segundo [CORMEN *et al*, 2001], um problema pode ou não possuir um algoritmo exato para sua solução. Existindo esse algoritmo, o mesmo pode não encontrar um ótimo em tempo hábil,

> ou seja, o algoritmo pode demorar décadas para encontrar a solução adequada. Quando isso ocorre, diz-se que o mesmo é inviável para a instância abordada. Por esta razão, as abordagens exatas devem ser utilizadas para pequenas instâncias do problema, pois para os demais elas demorariam muito tempo de processamento [BEZERRA, 2005]" [SOUZA Jr, 2007].

Para instâncias maiores e mais complexas, devemos utilizar algoritmos heurísticos, que são aqueles que exploram apenas uma pequena parte do espaço de soluções, fornecendo uma solução boa qualidade, com um custo computacional baixo [VIEIRA, 2013].

[MARINHO, 2013] define meta-heurística como "um conjunto de conceitos que podem ser utilizados para definir métodos heurísticos que podem ser aplicados a um vasto conjunto de problemas diferentes. Em outras palavras, uma meta-heurística pode ser vista, de modo geral, como um algoritmo aplicado a diferentes problemas de otimização. Esses algoritmos podem sofrer modificações, visando à adaptação para um problema específico".

Segundo [OLIVEIRA, 2007] a principal diferença entre os algoritmos exatos e os heurísticos é que os exatos possuem a garantia de chegar ao resultado ótimo do problema tratado, mesmo que em tempo exponencial.

2.4.1. Branch-and-Bound

"Os métodos de *Branch-and-Bound* foram desenvolvidos a partir do trabalho pioneiro de Land e Doing[LaDo 60]. O termo '*branch-and-bound*'foi empregado pela primeira vez em [LiMuSwKa 63]"[MACULAN, 2004].

O método baseia-se em desenvolver uma enumeração inteligente das soluções candidatas à solução ótima inteira de um problema. Apenas uma fração das soluções factíveis é realmente examinada. Em cada passo do método as variáveis inteiras são relaxadas e o subproblema resultante resolvido por um método da programação linear. Se a solução desse subproblema tiver todas as

variáveis inteiras, então os descendentes desse ramo analisado estão, naturalmente, implicitamente enumerados.[SOUZA, 2009].

2.4.2. Savings

Segundo Vieira, a heurística foi desenvolvida por Clark e Wright para resolver um problema de roteamento com restrição de capacidade. Essa heurística consiste em conectar duas rotas factíveis formando uma única rota, também factível. A escolha das rotas que serão combinadas é feita de modo que a nova rota gerada tenho um custo menor. Por exemplo, supunha duas rotas R_1 e R_2. Essas duas rotas podem ser combinadas em uma única se removermos de cada uma, uma aresta que incide no depósito e ligando os nós pendentes (as arestas $(i,0) \in R_1$ e $(0,j) \in R_2$ são removidas e então adiciona-se a aresta (i,j) à nova rota). [VIEIRA, 2013].

"A cada iteração as rotas são organizadas em conjuntos de pares. Um ponto i e um ponto j podem formar o par (i,j) se:

- Os pontos de entrega i e j não estão na mesma rota;
- A capacidade de carga do veículo não é violada;
- Nem i nem j são *pontos interiores*[1] em uma rota.

2.4.3. Busca Tabu

> "Busca Tabu é um método heurístico genérico, também chamado de meta-heurística, proposto por [GLOVER, 1986] e descrito em detalhes em [GLOVER; LAGUNA, 1997]. Este método guia um procedimento heurístico de busca local pela utilização de características da solução corrente e da história de busca para explorar o espaço de soluções além da otimalidade local." [Lopes *et. al.,* 2013, p.33]

[1] Pontos interiores em rotas são aqueles que não são precedidos nem sucedidos pelo depósito.

Basicamente, a Busca Tabu consiste em explorar o espaço de soluções de forma iterativa, movendo de uma dada solução para outra vizinha. A grande diferença deste método é que ele aceita soluções piores, o que pode gerar ciclos. Por este motivo, a fim de evitar ciclos, as soluções já avaliadas são incluídas na *lista tabu.*

Uma vez incluídas nessa lista, as soluções podem deixar de serem tabus em determinadas circunstancias (normalmente tempo ou número de iterações – prazo tabu [Schopf *et al.*]), chamadas de critérios de aspiração. Esses critérios podem ser necessários devido à possibilidade da lista tabu impedir movimentos para soluções ainda não exploradas.

“A maioria dos artigos nos quais a busca tabu é usada para resolver o PRVJT tem como objetivo principal reduzir o número de rotas.” [VIEIRA, 2013, p.31].

2.4.4. *Simulated Annealing*

“Esse método foi introduzido por Metropolis *et al.* e adaptado aos problemas de otimização por Kirkpatrick *et al.* A primeira aplicação para PRVJT foi feita por Chiang and Russel” [VIEIRA, 2013, p.32].

O *Simulated Annealing* (Recozimento Simulado em português) é uma meta-heurística inspirada no processo físico de recozimento (*annealing*) de um sólido, que é bastante usado na metalurgia. O recozimento consiste em aquecer um sólido até atingir sua temperatura de fusão, e em seguida, resfriá-lo lentamente para obter a matéria no estado cristalino (estado de baixa energia).

No algoritmo AS, uma nova solução x_j será aceita sempre que sua energia for menor que a solução anterior x_i, ou seja, $f(x_j) < f(x_i)$. Porém, para evitar que o algoritmo fique preso em mínimos locais, soluções onde $f(x_j) \geq f(x_i)$ também podem ser aceitas com uma probabilidade p definida pelo *critério de Metropolis* (Equação 2).

$$p = e^{(\frac{\Delta}{k_b T})} \quad (2)$$

Onde:

- Δ : É diferença entre as duas soluções ($f(x_i) - f(x_j)$).
- k_b: Constante física chamada de constante de Boltzmann.
- T : Temperatura.

Basicamente, quanto maior a temperatura T, maior a chance de uma solução pior ou igual à solução atual ser aceita. Isso ocorre para que no início haja maior diversificação. Já no fim do processo a chance de aceite é menor para aumentar a intensificação.

2.4.5. Push-foward Insertion Heuristic (PFIH)

"O PFIH é um eficiente algoritmo de construção de rotas proposto por Solomon (1987) que realiza inserções sequenciais de clientes em uma rota" [REINA, 2012]. O algoritmo funciona acrescentando um cliente por vez, escolhendo sempre, dentre as possíveis rotas, aquela que apresenta o menor custo. Esse custo é determinado por três fatores:

- Posição geográfica do cliente,
- Fim de sua janela de tempo, e
- O ângulo polar entre o cliente e o depósito.

Por ser um método sequencial, naturalmente, a ordem em que os clientes são inseridos afeta diretamente a qualidade da solução final. Para resolver isso, Solomon desenvolveu em seu trabalho [Solomon, 1987] uma fórmula para determinar a ordem em que os clientes devem ser inseridos (Equação 3). Esta equação fornece o custo (C_i) da inserção de cada cliente i na rota e, através deste custo, é possível determinar a ordem de inserção dos clientes. [OLIVEIRA, 2007].

$$C_i = -\alpha \, . \, d_{0i} + \beta . \, l_i + \gamma \left[\left(\frac{p_i}{360} \right) d_{0i} \right] \quad (3)$$

Onde:

- $\alpha = 0{,}7$; $\beta = 0{,}1$; $\gamma = 0{,}2$; definidas empiricamente em [Solomon, 1987].
- d_{0i}= distância do depósito ao cliente i.
- l_i = limite superior da janela de tempo de chegada ao cliente i.
- p_i = ângulo da coordenada polar do cliente i, referente ao depósito.

2.4.6. Algoritmos Evolucionários

Entende-se por Algoritmos Evolucionários todos aqueles que utilizam conceitos baseados na evolução das espécies de Darwin, que diz que os indivíduos mais bem adaptados têm maiores chances de sobrevivência. Como dito por Vieira em [VIERA, 2013], “Darwin identificou três princípios básicos da evolução: reprodução, seleção natural e diversidade dos indivíduos. É com base nesses princípios que essa classe de técnicas foi desenvolvida, modelando os processos de seleção, reprodução e mutação.” Estes algoritmos trabalham com uma população de soluções, ao invés de uma única, e a cada indivíduo atribui-se um valor que representa sua adaptação ao problema, ou seja, o quão bom aquele indivíduo é. Dentre os principais métodos que seguem esse princípio estão os Algoritmos Genéticos (AG), que foi o escolhido para ser aplicado neste trabalho e será detalhado na seção 2.5; a Programação Genética e as Estratégias Evolucionárias.

2.4.7. Meta-heurísticas híbridas

Assim como as aplicações do Problema de Roteamento de Veículos possui variações, os métodos para resolvê-los também apresentam várias formas diferentes de serem aplicados. As técnicas podem ser adaptadas para cada

caso e até mesmo combinadas, dependendo das características específicas do problema a ser resolvido.

Como exemplo temos “Um Modelo Híbrido Estocástico para Tratamento do Problema de Roteamento de Veículos com Janela de Tempo” [OLIVEIRA, 2007], onde o autor apresenta um algoritmo para solucionar o PRVJT através da combinação de três métodos diferentes: uma variação do Recozimento Simulado (*Simulated Annealing*), Subida na Encosta (*Hill Climbing*) e Reinício Aleatório (RA).

Outro exemplo é “Roteirização de Veículos com Janela de Tempo Utilizando Algoritmos Genéticos” [REINA, 2012]. O autor utiliza o *push-foward insertion heuristic* para gerar parte da população inicial e então aplica os conceitos de algoritmos genéticos para encontrar a solução.

2.5. Algoritmos Genéticos (AGs)

Esta meta-heurística foi desenvolvida por John Holland [HOLLAND, 1975] e seus alunos na Universidade de Michigan na década de 60, e é dentre os algoritmos evolutivos a mais popular. Lucas [LUCAS, 2002] diz em seu trabalho “Com base no modo como o darwinismo explica o processo de evolução das espécies, Holland [HOLLAND, 1975] decompôs o funcionamento dos AGs em sete etapas: inicialização, avaliação, seleção, cruzamento, mutação, atualização e finalização;” como mostra a Figura 2.

Figura 3: Estrutura básica de um AG. Fonte: [LUCAS, 2002]

- Inicialização: criação da população inicial.
- Avaliação: calculo do valor *fitness* dos indivíduos.
- Seleção: escolha dos indivíduos para reprodução.
- Cruzamento: recombinação das características (genes) de dois indivíduos (pais) para gerar novos indivíduos (filhos).
- Mutação: Adição de uma perturbação às características de alguns indivíduos para aumentar a variedade da população.
- Atualização: inserção dos novos indivíduos na população.
- Finalização: avalia se as condições de parada foram atendidas, caso positivo exibe a melhor solução na tela e encerra o programa; caso negativo retorna para a etapa de avaliação.

Um dos mais importantes fatores para a construção de um bom AG, assim como em outros Algoritmos Evolucionários, é a forma como os indivíduos são representados [OLIVEIRA, 2007]. A representação é a forma de traduzir a informação do problema real para o computador de maneira viável. Ela afeta

diretamente o desempenho do algoritmo, pois influencia em como serão feitas as operações genéticas e até mesmo na possibilidade de gerar ou não soluções infactíveis. Como dito por Oliveira em [OLIVEIRA, 2007], normalmente, para o PRVJT é utilizada a representação por listas de inteiros sem repetições, onde cada lista representa uma rota. Essa é a mesma representação utilizada neste trabalho.

A inicialização é a etapa onde os primeiros indivíduos são gerados, dando origem à *população inicial*. Esses indivíduos devem sem bem diversificados, para diminuir a convergência prematura, por isso normalmente são gerados de forma aleatória. Porém também é possível utilizar heurísticas como o PFIH para gerar parte da população inicial, como no trabalho de Reina [REINA, 2012]. Neste trabalho a população é criada de unicamente de forma aleatória.

Definida a população inicial, é feita então a Avaliação da mesma. A cada indivíduo é atribuído um valor *fitness*, que é calculado com base na função objetivo ou função de aptidão. Como dito por Pozo *et al.*, através desta função mede-se o quão próximo um indivíduo está da solução desejada ou o quão bom ele é [Pozo *et. al.,* Computação Evolutiva]. No caso do PRV, a função objetivo é o somatório dos pesos das arestas ou a distancia de cada trecho do percurso a ser percorrido.

Após a avaliação, aplicam-se então os três *operadores genéticos,* que são: Seleção, Cruzamento e Mutação. Estes operadores “têm como objetivo manter a diversidade da população e, ao mesmo tempo, transmitir as características adquiridas nas gerações anteriores, provocando uma melhora da aptidão ao longo do processo” [VIEIRA, 2013].

A Seleção é a etapa onde os pais da próxima geração são escolhidos, ou seja, onde os indivíduos com as melhores características são selecionados a fim de transmitir essas características. “Há diversas formas de seleção, entre ele há o método de seleção por Roleta e o método de Seleção por Torneio” [Pozo *et. al.,* Computação Evolutiva].

- Roleta: cada indivíduo da população é representado na roleta proporcionalmente ao índice de aptidão. Assim, para indivíduos com alta

aptidão é dada uma porção maior da roleta, enquanto aos indivíduos de aptidão mais baixa, é dada uma porção relativamente menor (Figura 3).

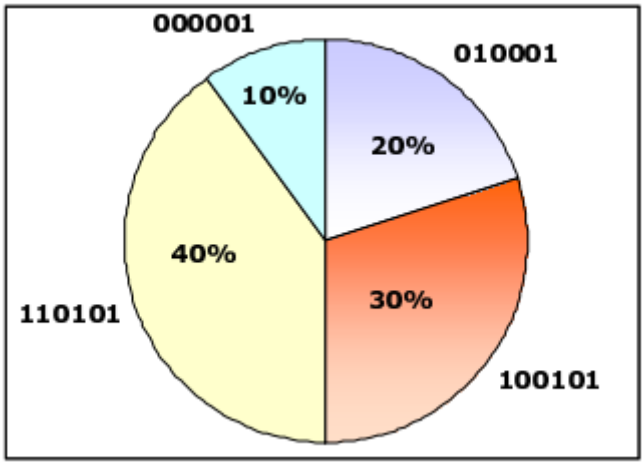

Figura 4: Método de seleção por roleta. Fonte: Computação Evolutiva, 2012.

- Torneio: este método consiste em selecionar aleatoriamente T indivíduos na população, com reposição, sendo $T \geq 2$. Em cada grupo de T indivíduos, os integrantes disputam entre si de acordo com seus valores de *fitness.* Em caso de empate, o vencedor é escolhido de forma aleatória. A pressão de seleção é controlada através do valor de T. Na prática, torneios de tamanho 2 ou 3 são muito bons.[MENDES, M., Notas de Aula]. A Figura 4 mostra um exemplo de torneio com $T = 3$.

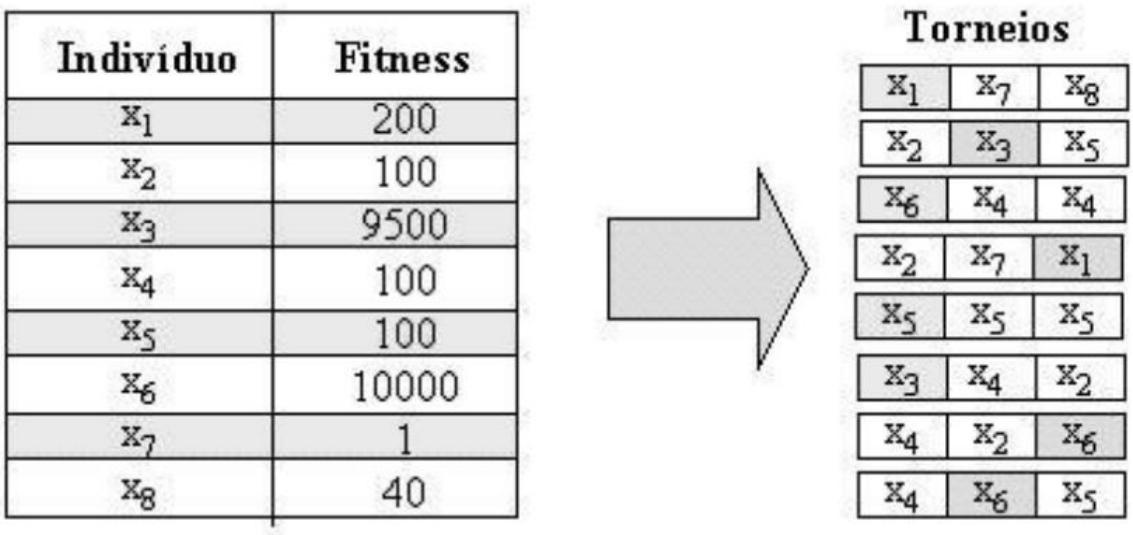

Figura 5: Exemplo de torneio com T=3. Fonte: MENDES, Notas de Aula.

Quanto ao Cruzamento, [OLIVEIRA, 2007] diz que consiste em mesclar as informações de duas ou mais soluções, simulando a reprodução dos seres vivos onde materiais genéticos de dois pais são misturados na geração de sua prole. Alguns tipos bastante utilizados e mais simples são o cruzamento em um ponto e o cruzamento em dois pontos, como mostrado nas Figuras 5 e 6:

- Cruzamento de um ponto: seleciona-se aleatoriamente um ponto de corte do cromossomo de cada pai e cada descendente recebe uma das partes.

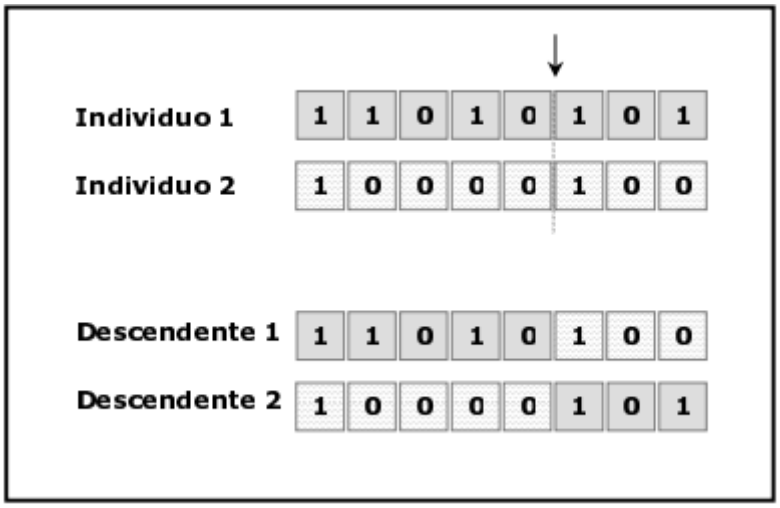

Figura 6: Cruzamento de um ponto. Fonte: Computação Evolutiva, 2012.

- Cruzamento de dois pontos: semelhante ao anterior, seleciona-se aleatoriamente dois pontos de corte do cromossomo de cada pai. “Um dos descendentes fica com a parte central de um dos pais e as partes extremas do outro pai e vice versa” [Pozo *et. al.,* Computação Evolutiva].

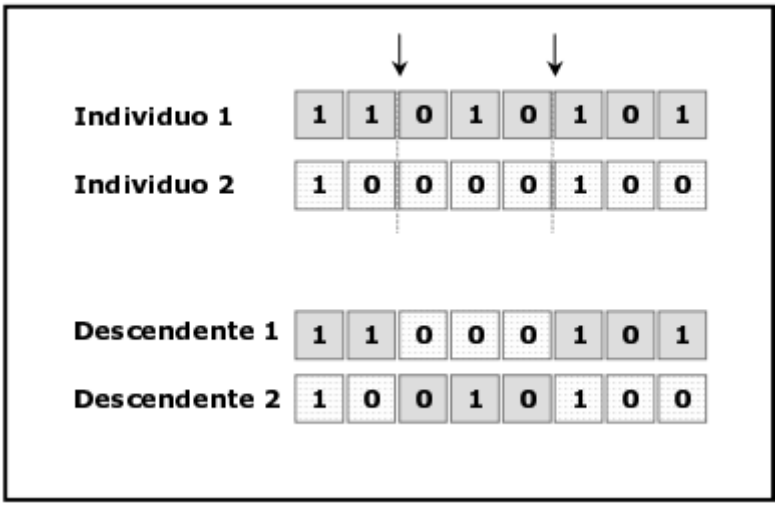

Figura 7: Cruzamento de dois pontos. Fonte: Computação Evolutiva, 2012.

Esses métodos não funcionam para casos onde os indivíduos são modelados como listas sem repetição, como é normalmente feito em PRVJT. Para esses casos, assim como neste trabalho, existem métodos mais elaborados específicos, como o *Order Based Crossover.* Este método utiliza uma máscara binária criada aleatoriamente para gerar dois novos indivíduos (filhos). Nas posições onde a o valor da máscara é 1, os filhos recebem, nessas mesmas posições, os mesmos genes dos pais (filho 1 recebe do pai 1 e filho 2 recebe do pai 2). Para o restante dos genes, onde o valor da máscara é 0, eles são ordenados assim como aparecem no outro pai e inseridos nas posições ainda vazias dos respectivos filhos (Os genes do pai 1 são colocados na mesma ordem que aparecem no pai 2 e inseridos no filho 1). A Figura 7 ilustra exemplifica o resultado desse método.

Pai1	A	B	C	D	E	F	G
Pai2	E	B	D	C	F	G	A
Máscara	0	1	1	0	0	1	0
Filho1	E	B	C	D	G	F	A
Filho2	A	B	D	C	E	G	F

Figura 8: *Order Based Crossover*. Fonte: MENDES, Notas de Aula.

A terceira operação genética é a Mutação, ela “modifica aleatoriamente alguma característica do indivíduo sobre o qual é aplicada” [Pozo *et. al,* Computação Evolutiva]. Estas modificações são importantes para manter a diversidade da população e evitar que o algoritmo convirja prematuramente. Além disso, segundo Pozo *et. al*, “a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca possivelmente não será zero.”

Existem vários métodos de mutação. Como exemplo simples pode-se citar a troca de um gene de um indivíduo codificado como um vetor binário, como mostra a Figura 8.

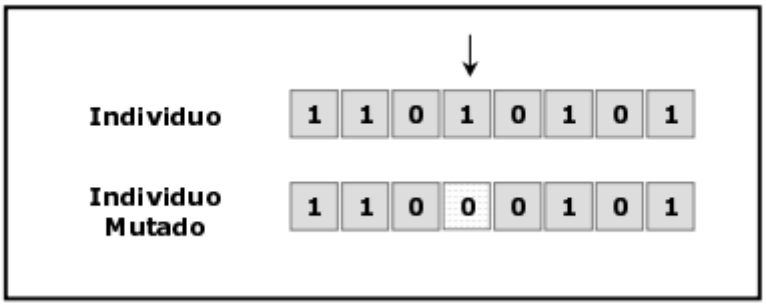

Figura 9: Exemplo de mutação. Fonte: Computação Evolutiva, 2012.

Porém, assim como no cruzamento, a codificação utilizada no PRV (listas sem repetições) exige métodos próprios de mutação. Como exemplo, pode-se citar

o método 2-Opt, que foi adotado neste trabalho. Este método consiste em escolher dois vértices randômicos (*x1* e *x2*), não sendo o primeiro nem o último, e inverter todo o caminho entre esses dois pontos, incluindo eles mesmos. Além da população, este método também recebe como entrada uma constante k, que é a chance, em porcentagem, de um individuo sofrer mutação (k=10 significa 10% de chance de ocorrer mutação). A Figura 9 ilustra o método de mutação 2-Opt:

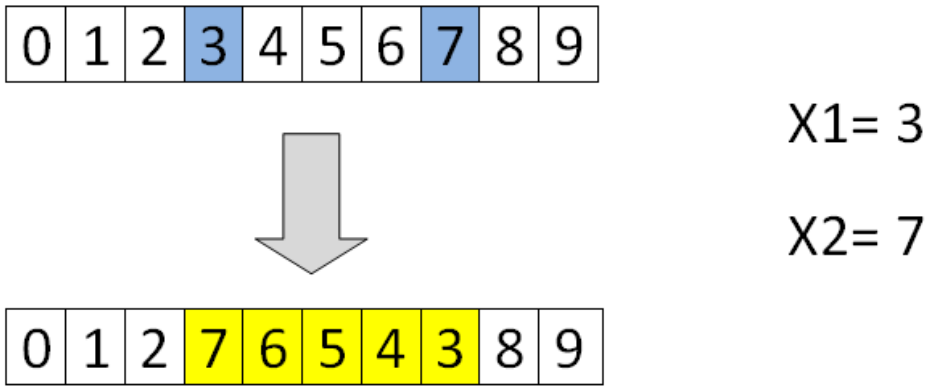

Figura 10: Mutação 2-Opt.

Para não perder as melhores soluções encontradas logo após o cruzamento ou mesmo em gerações anteriores, utiliza-se o processo de elitismo. [SOUZA Jr, 2007] define esse processo da seguinte forma: é feita uma cópia integral da melhor solução na geração sucessora. Neste trabalho o elitismo consiste em armazenar as $^{P}/_{10}$ melhores soluções encontradas durante o processo, sendo P o tamanho da população.

Após a aplicação dos operadores genéticos chegamos a ultima etapa descrita por Holland, a Finalização, onde é feita a verificação da condição de parada do algoritmo. Caso a condição seja satisfeita, o algoritmo para e retorna o melhor individuo. Caso a condição não seja satisfeita, o algoritmo volta aos operadores genéticos e simula mais uma geração (executa mais uma iteração).

2.6. Trabalhos relacionados

Esta seção aborda alguns trabalhos encontrados na literatura que serviram de apoio para o desenvolvimento do trabalho.

Como principal trabalho relacionado e base de apoio encontra-se [VIEIRA, 2013], que aborta vários métodos para resolver o problema, porém com maior ênfase nos Algoritmos Genéticos. Em seu trabalho, a autora aborda diversos métodos de cruzamento e mutação. Para testar seu algoritmo utilizou um conjunto de 56 problemas, cada qual com cem clientes, propostos por Solomon, além de comparar aos melhores resultados encontrados na literatura. Quantos aos resultados, o algoritmo desenvolvido não é tão competitivo em relação aos publicados na literatura devido ao fato de terem focado o algoritmo na minimização da distancia total apenas.

[SOUZA Jr, 2007] apresenta uma possível solução para o problema. Nesse caso, o autor propõe a resolução do problema através de Algoritmos Evolutivos, e cita também outras técnicas conhecidas na literatura, como o *Push-Forward Isertion Heuristic*, Busca Tabu e Algoritmos Genéticos. Assim como no trabalho anterior, para os testes foram utilizadas as instâncias de Solomon. O algoritmo implementado neste trabalho é estático e obteve resultados satisfatórios, conseguindo até mesmo superar os melhores resultados já encontrados anteriormente em alguns casos testados.

Outro trabalho que teve grande influencia neste foi [OLIVEIRA, 2007], onde o autor apresenta um algoritmo para solucionar o PRVJT através da combinação de três heurísticas diferentes: Recozimento Simulado Não Monotônico (RSNM), Subida na Encosta (SE) e Reinício Aleatório (RA). Além disso, o trabalho também oferece um arcabouço de métodos estatísticos para ajustar parâmetros de sistemas estocásticos de otimização. Os resultados dos experimentos foram comparados com os melhores resultados conhecidos alcançados pelas técnicas individualmente nas instâncias de Solomon. O algoritmo implementado mostrou-se eficaz e robusto no tratamento do PRVJT , pois os resultados obtidos superaram ou igualaram 37 das 56 instâncias testadas.

3. METODOLOGIA

Inicialmente, utilizando um mapa impresso da cidade Itaúna de 2012, foi feito a identificação, separação e listagem dos bairros. Neste primeiro momento foram listados 65 bairros. A partir daí, considerando cada bairro como um nó de um grafo, foram inseridas as arestas, utilizando as principais ruas que fazem as ligações entre os bairros. Por se tratar de um problema real, as arestas foram desenhadas no mapa de forma a manterem a coerência com a realidade. Feito isso, como o mapa estava em escala, foi calculado o peso de cada aresta com base no seu comprimento no mapa, em centímetros. Feito isso se chegou à primeira versão do mapa de Itaúna.

A partir daí foram feitas mais algumas alterações no grafo, como incremento de nós para representar novos bairros e pontos de interseção entre algumas arestas, a fim de deixa-lo mais coerente com a realidade. Feitas as modificações, o grafo final possui 72 nós e 128 arestas, como mostra a Figura 10:

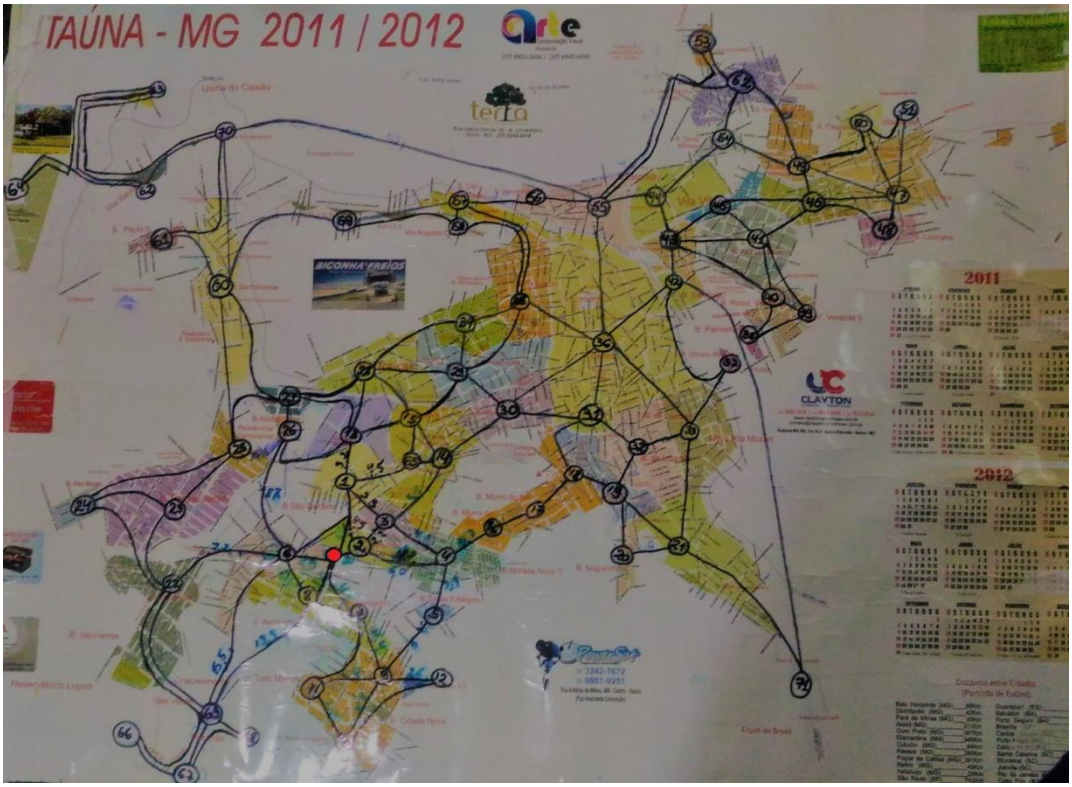

Figura 11: foto do mapa de Itaúna.

O ponto vermelho marcado no mapa indica o depósito central. A relação entre os vértices com os bairros que representam pode ser encontrada no Apêndice A deste trabalho.

Quanto ao software, foi desenvolvido em linguagem C, utilizando a IDE CodeBlocks 13.12, utilizando técnicas e conceitos de algoritmos genéticos.

Neste caso, cada indivíduo é modelado como um vetor de inteiros com 72 posições, que representa o caminho pelo qual o caminhão passará, como mostra a Figura 11:

0	1	5	8	10	[...]	14	6	2	3

Figura 12: representação de um indivíduo.

O problema resolvido não é capacitado, ou seja, a capacidade dos veículos não é considerada pelo algoritmo. O valor *fitness* é calculado através do somatório do valor das arestas entre os nós percorridos, como é mostrado na equação 4, porém existe uma penalidade caso o tempo de entrega de algum pedido seja excedido. A penalidade é calculada adicionando ao *fitness* o tempo de atraso de cada pedido, como mostra a equação 5. O tempo atual (Equação 6) é calculado multiplicando a distancia percorrida por três (valor arbitrário), enquanto que o tempo limite de cada pedido é fornecido pelo usuário.

$$fitness = \sum_{i=0}^{n} d_{ij} + \sum_{i=0}^{n} penalidade_i \quad (4)$$

Sendo:

- *n:* é o número de nós no grafo.
- *d:* é a distancia entre os vértices i e j, ou seja, o tamanho da aresta (i,j).

$$Penalidade = TempoAtual - TempoLimite \quad (5)$$

Sendo:

$$TempoAtual = \sum_{i=0}^{n} d_{ij} * 3 \quad (6)$$

Os primeiros indivíduos são gerados utilizando um algoritmo guloso, a partir de um vértice inicial que é escolhido de forma randômica. No caso do mapa de Itaúna, não existem arestas entre todos os nós, apenas entre os vizinhos. Portanto, a essas arestas que não existem no mundo real é atribuído um valor arbitrário relativamente alto para evitar que estas sejam utilizadas. O algoritmo guloso funciona escolhendo a menor aresta a cada iteração, sem se preocupar com consequências futuras. Além disso, um algoritmo guloso nunca se arrepende ou volta atrás, as decisões tomadas são absolutas. O Pseudoalgoritmo 1 mostra o algoritmo guloso utilizado neste trabalho.

Pseudoalgoritmo 1: IndividuoGuloso (*estrutura){

```
1  escolhe o primeiro vértice aleatoriamente; // indivíduo[0]
2  para i=0 até N, faça{
3     encontra a menor aresta (individuo[ i ], j) ainda não percorrida.
4  Indivíduo[i+1] = j;
5  }
```

Após gerar toda a população, dá-se inicio à parte evolutiva do algoritmo, começando pela seleção dos indivíduos que serão os pais da próxima geração. Essa seleção é feita a partir de um torneio binário, onde dois indivíduos escolhidos aleatoriamente “disputam” entre si, e aquele que possuir o melhor *fitness* vence. O Pseudoalgoritmo 2 mostra um torneio binário em que N indivíduos são selecionados a partir de uma população P.

Pseudoalgoritmo 2: Seleção (*estrutura){

```
1  enquanto (contador< (P/2))
2     escolha 2 indivíduos randômicos rand1 e rand2;
3     se (rand1>=rand2)
4        filho == rand1;
5     se não
6        filho == rand2;
```

```
7       contador++;
8   }
```

O próximo passo é o cruzamento, que mescla dois indivíduos (pais) para criar um novo (filho). Neste trabalho, os indivíduos são uma lista de vértices a ser percorrida, uma rota, portanto os métodos mais simples, como o Cruzamento de um ponto não podem ser utilizados, pois podem gerar indivíduos inválidos (vértices repetidos ou não incluídos). A fim de evitar indivíduos inválidos foi utilizado o método *Order Based Crossover.* O Pseudoalgoritmo 3 mostra como ele funciona.

Pseudoalgoritmo 3: Cruzamento(*estrutura):

```
1   Mask -> vetor de bits aleatórios com tamanho N.
2   enquanto (contador< (P/4)){
3       escolha 2 indivíduos randômicos pai1 e pai2;
4       para i=0 até N, faça{
5           se (Mask[i]==0){
6               filho1[i] == pai1[i];
7               filho2[i] == pai2[i];
8           }
9       }
10  L1 <- lista de genes do pai1 que estão na posição i em que
    Mask[i]=1;
11  L2 <- lista de genes do pai2 que estão na posição i em que
    Mask[i]=1;
12  Ordenar os genes de L1 da mesma forma que aparecem no pai2;
13  Ordenar os genes de L2 da mesma forma que aparecem no pai1;
14  Preencher os genes vazios do filho1 com L1;
15  Preencher os genes vazios do filho1 com L1;
16  contador++;
17  }
```

Feito o cruzamento, o *fitness* é recalculado. Caso alguns destes novos indivíduos superem algum dos elites anteriores, estes são guardados na elite, a fim de que esses indivíduos sejam mantidos para a próxima geração. O tamanho da elite é sempre 10% do tamanho da população total, para que os melhores indivíduos não corram o risco de serem perdidos, e ao mesmo tempo, para evitar que o algoritmo convirja prematuramente.

O último operador genético a ser aplicado é o de mutação, e para este utilizamos o 2-Opt, com k = 10, ou seja, a chance de ocorrer mutação no indivíduo é de 10%. O Pseudoalgoritmo 4 mostra como esse operador foi implementado, onde P é o tamanho da população total.

Pseudoalgoritmo 4: Mutação(*estrutura, int k);

```
1   Para i=0 até P, faça{
2       Escolher um numero randômico X entre 0 e 100;
3       se(X<k){
4           Escolha 2 indivíduos randômicos(x1 e x2);
5           Para i=0 até x1, faça
6               filho[i] == pai[i];
7           Para i=x1 até x2, faça
8               filho[i] == pai[x2 – (i - x1)];
9           Para i=x2 até N, faça
10              filho[i] == pai[i];
11      }
12  }
```

Após a mutação o fitness é recalculado e todos os indivíduos são recomparados com a elite para garantir que, caso a mutação consiga gerar um ou mais indivíduos melhores dos que os que estão atualmente na elite, estes não sejam perdidos. O último passo do ciclo, antes de conferir se as condições de parada foram atendidas é inserir os indivíduos da elite na população. Essa inserção é feita substituindo outros indivíduos, escolhidos aleatoriamente.

Ao fim destes passos, são avaliadas as condições de parada, e caso não sejam cumpridas, o AG volta ao método de Seleção e executa todos os passos mais uma vez. Caso as condições sejam satisfeitas, o programa exibe na tela qual a melhor solução encontrada, junto com seu valor *fitness.* Neste caso, existe apenas uma condição de parada, que é o número de gerações. O Pseudoalgoritmo 5 mostra a estrutura deste AG, onde N é o número de indivíduos.

Pseudoalgoritmo 5: AG

```
1  Gera população inicial
2  Seleciona elite.
3   Avaliação
4  Para i=0 até N, faça{
5      Seleção
6      Cruzamento
7      Avaliação
8      Guarda na elite caso haja algum indivíduo melhor
9      Mutação
10     Avaliação
11     Guarda na elite caso haja algum indivíduo melhor
12     Atualiza a população
13 }
14 Exibe o indivíduo com o menor fitness
```

4. RESULTADOS

Foram feitos vários testes no algoritmo, variando o número de indivíduos e também o número de gerações. No total foram feitos 600 testes (10 para cada instancia do problema), 300 para o benchmark eil76 e outros 300 para o problema real. Quanto aos demais parâmetros utilizados na configuração do AG, temos:

- Número de indivíduos: 5, 10, 25, 50, 100.
- Número de gerações: 1, 10, 25, 50, 100, 150.
- Taxa de cruzamento: 50%.
- Taxa de mutação: 10%.
- Número de pedidos: 71 (um em cada nó, com exceção do depósito)
- Elitismo: 10% da população total. Nos testes com população P=5 adotou-se elitismo como 20% de P, ou seja, apenas um indivíduo.
- Quantidade de veículos: 1.

O eil76 é um benchmark para o TSP (*Traveling Salesman Problem*) cuja solução ótima é 538 e foi escolhido para ser comparado com este trabalho pelo numero de nós ser próximo ao do problema real (76). Este bechmark foi retirado da TSPLIB, que é uma biblioteca de instâncias de exemplo para o TSP (e problemas relacionados) de várias fontes e de vários tipos. A TSPLIB é mantida pela *Heideilberg University*, Alemanha.

4.1. *Benchmark* Eil76

Para resolver o *benchmark* considerou-se um pedido em cada nó, com janelas de tempo iguais à do depósito central, ou seja, todos os pedidos podem ser entregues a qualquer momento, sem nenhuma prioridade; e apenas um veículo. Esses parâmetros foram adotados a fim de adaptar o problema para que ficasse do tipo PCVJT.

Tabela 1: Resultados dos testes utilizando os parâmetros do benchmark.

pop\gerações	1	10	25	50	100	150
5	622	622	582	601	639	679
5	584	659	625	584	610	615
5	629	633	592	623	622	589
5	627	592	594	628	625	594
5	589	593	659	592	640	623
5	582	615	584	589	599	622
5	580	629	635	648	593	623
5	599	630	643	599	595	599
5	589	633	623	615	600	622
5	623	599	633	596	582	629
10	588	589	585	576	577	577
10	584	581	587	584	587	588
10	570	600	565	587	589	596
10	577	592	584	579	622	589
10	580	582	589	591	572	596
10	580	584	600	573	580	573
10	576	570	589	577	588	599
10	597	587	576	600	588	568
10	588	625	577	588	589	587
10	574	598	587	582	584	574
25	580	582	560	565	567	571
25	568	583	571	568	561	559
25	565	552	570	574	572	575
25	579	569	584	574	583	570
25	569	580	566	571	567	562
25	592	561	568	558	561	561
25	573	566	584	558	551	554
25	558	571	569	573	566	568
25	574	554	582	565	564	563
25	564	560	549	561	559	580
50	539	540	545	541	552	550
50	549	538	546	551	541	540
50	543	548	544	549	544	565
50	550	540	550	539	539	542
50	548	542	541	538	538	547
50	544	543	544	545	546	562
50	567	543	542	541	555	549
50	547	541	539	549	546	541
50	542	543	543	539	549	539
50	560	547	544	541	542	548
100	582	540	539	542	540	543
100	551	546	538	540	539	540

100	549	541	542	539	542	541
100	542	539	545	540	541	539
100	560	552	539	538	538	538
100	543	540	560	539	540	542
100	555	538	541	541	543	541
100	541	539	540	538	539	539
100	580	541	540	543	541	540
100	540	542	542	539	540	540

Na Tabela 1 temos os resultados obtidos nos testes feitos utilizando os parâmetros do bechmark eil76. Conforme o tamanho da população e o número de gerações crescem, nota-se a melhora das soluções encontradas. Essa melhora nos resultados pode ser mais facilmente observada nas tabelas 2 e 3, que resumem melhor os resultados obtidos.

Tabela 2: Médias, modas e medianas dos testes do benchmark.

	pop\gerações	**1**	**10**	**25**	**50**	**100**	**150**
media	**5**	602,4	620,5	617	607,5	610,5	619,5
moda		589	633	-	-	-	623
mediana		594	625,5	624	600	605	622
media	**10**	581,4	590,8	583,9	583,7	587,6	584,7
moda		588	-	587	-	589	596
mediana		580	588	586	583	587,5	587,5
media	**25**	572,2	567,8	570,3	566,7	565,1	566,3
moda		-	-	584	565	567	-
mediana		571	567,5	569,5	566,5	565	565,5
media	**50**	548,9	542,5	543,8	543,3	545,2	548,3
moda		-	543	544	541	546	-
mediana		547,5	542,5	544	541	545	547,5
media	**100**	554,3	541,8	542,6	539,9	540,3	540,3
moda		-	540	539	539	540	540
mediana		551	541	541	540	540	540

Tabela 3: Mínimos e máximos dos testes do benchmark.

	pop\gerações	1	10	25	50	100	150
min	5	580	592	582	584	582	589
max		629	659	659	648	640	679
min	10	558	552	549	558	551	554
max		597	625	600	600	622	599
min	25	558	552	549	558	551	554
max		592	583	584	574	583	580
min	50	539	538	539	538	538	539
max		567	548	550	551	555	565
min	100	540	538	538	538	538	538
max		582	552	560	543	543	543

A partir da análise das Tabelas 2 e 3 pode-se observar que o algoritmo cumpre seu propósito, pois quando número de indivíduos e gerações é suficientemente grande, retorna como resultado mais frequente (moda) uma solução (540) bem próxima à ótima conhecida. Além disso, consegue encontrar em alguns testes o resultado ótimo que é 538, como mostra a linha min da Tabela 3 para população com 100 indivíduos. Nota-se também que à medida que o número de gerações aumenta, a discrepância entre o valores mínimos e máximos encontrados diminuem significativamente. A partir de 50 gerações com 100 indivíduos, até mesmo o pior resultado encontrado, que foi 543.

- Melhor rota encontrada: 1, 33, 63, 16, 3, 44, 32, 9, 39, 72, 58, 10, 31, 55, 25, 50, 18, 24, 49, 23, 56, 41, 43, 42, 64, 22, 61, 21, 47, 36, 69, 71, 60, 70, 20, 37, 5, 15, 57, 13, 54, 19, 14, 59, 66, 65, 38, 11, 53, 7, 35, 8, 46, 34, 52, 27, 45, 29, 48, 30, 4, 75, 76, 67, 26, 12, 40, 17, 51, 6, 68, 2, 74, 28, 62, 73.
- *Fitness:* 538.

4.2. Problema real

Para resolver o problema real foram feitos dois experimentos em duas situações diferentes. Normalmente as entregas não são definidas em janelas de tempo específicas, mas sim em limites de tempo, como até determinado horário ou após uma hora específica. Existe também a possibilidade de um pedido ser marcado como a primeira entrega do dia. Nesse caso sua janela de tempo abre junto com a do depósito e é a primeira a ser encerrada. Em ambos os casos o pedido referente ao nó 8 foi definido como sendo a primeira entrega.

4.2.1. Experimento 1

Neste primeiro experimento utilizou-se uma lista de pedidos hipotética com um pedido em cada nó, assim como feito no benchmark, porém apenas alguns possuem janelas de tempo diferentes do depósito central, que são listadas na Tabela 4.

Tabela 4: Lista de pedidos com janelas de tempo diferentes do experimento 1.

Cliente	Janela de Tempo
1	Entre 12 e 16 h
3	Após 16 h
8	Primeira entrega
15	Até 15h30m
22	Até 12 h
23	Até 10 h
27	Após 9 h
50	Até 15 h
65	Após 13 h

Tabela 5: Resultados obtidos nos testes utilizando os parâmetros do experimento 1.

pop\gerações	1	10	25	50	100	150
5	2128,60	1925,10	2103,80	2036,40	2200,00	2113,00
5	2103,10	2201,00	2199,30	1939,60	2106,30	2008,60

5	2106,30	2106,00	2017,30	2008,60	2109,20	2008,60
5	2107,20	2106,60	2096,80	2109,20	2008,60	2108,00
5	2106,10	2103,80	2103,00	2105,80	2034,40	2103,30
5	2036,40	2103,30	2106,10	2194,30	2008,60	2105,80
5	2102,00	2034,60	2017,80	2112,00	2113,00	2096,80
5	2197,60	2110,00	2103,10	2115,80	2102,00	2034,60
5	2199,30	2110,00	2008,60	1925,90	2017,80	2105,80
5	2103,00	2105,80	1925,90	2112,00	2105,80	2106,00
10	2100,80	2017,00	2017,80	2099,90	2099,40	1923,70
10	1920,90	2099,30	2009,50	1920,90	1939,60	2008,60
10	2010,60	2098,10	2005,30	2017,00	2105,80	2012,80
10	2102,00	2022,50	2017,30	2027,40	2006,20	2101,30
10	2098,80	2017,00	1925,90	2100,90	1930,30	2009,80
10	2017,80	2034,60	2017,80	1937,40	2091,00	2012,30
10	2006,40	2015,60	2017,30	2006,20	1922,70	2103,00
10	2008,60	2100,70	2008,60	2017,00	2102,00	2008,60
10	2112,00	2106,30	2031,10	2105,80	2014,80	1939,60
10	2100,10	2012,30	2029,40	1925,90	2097,00	1939,60
25	2102,00	1823,90	1925,90	1939,60	1823,70	2008,60
25	1918,10	1999,40	2089,90	2008,60	1920,30	1999,50
25	2034,60	2002,70	2011,30	1824,90	2001,20	1919,60
25	2010,90	2089,20	1999,10	1997,80	2001,50	2028,80
25	2008,60	1939,60	1925,90	1939,60	2008,60	1837,60
25	2000,50	1920,10	2012,30	1920,50	1933,70	1824,60
25	2001,90	2095,80	1919,30	2017,00	2003,40	1808,40
25	2008,60	2017,00	1999,70	2013,30	2036,40	1919,00

25	1912,60	1834,30	1911,40	2026,40	1820,20	1932,40
25	1925,90	1916,90	1935,10	1998,40	1933,30	1823,70
50	1865,00	1883,40	1812,60	1719,30	1872,40	1816,50
50	2008,60	1901,30	1976,10	1801,80	1798,80	1804,40
50	1915,80	1885,60	1702,80	1704,30	1673,00	1883,50
50	1879,90	1726,60	1996,80	1925,90	1802,00	1703,6
50	1804,00	1711,30	1821,40	1689,00	1796,20	1791,90
50	1906,50	1880,70	1900,60	1687,50	1800,20	1700,30
50	1939,60	1785,10	1799,20	1993,30	1876,00	1597,60
50	1990,70	1973,30	1869,70	1795,20	1687,50	1812,30
50	1818,20	1707,80	1822,20	1888,40	1678,40	1717,30
50	1825,20	1912,40	1793,00	1786,60	1893,80	1813,20
100	1805,40	1667,50	1670,00	1674,00	1587,30	1577,70
100	1801,10	1767,10	1708,10	1648,80	1579,60	1594,20
100	1800,00	1660,90	1570,00	1577,70	1578,40	1590,80
100	1775,10	1668,60	1590,80	1631,30	1571,10	1588,30
100	1757,70	1902,00	1670,90	1577,50	1561,70	1593,30
100	1807,40	1718,60	1697,40	1660,10	1608,00	1561,70
100	1763,70	1677,80	1673,20	1617,30	1568,40	1571,10
100	1780,00	1758,70	1566,30	1673,80	1577,70	1568,80
100	1697,60	1669,60	1687,40	1678,20	1594,80	1561,70
100	1779,60	1674,40	1597,20	1676,90	1595,30	1577,50

Na Tabela 5 temos os resultados obtidos nos testes feitos utilizando os parâmetros do experimento 1. Assim como no teste do benchmark, conforme o tamanho da população e o número de gerações crescem, nota-se a melhora das soluções encontradas. Essa melhora nos resultados pode ser mais

facilmente observada nas tabelas 6 e 7, que resumem melhor os resultados obtidos.

Tabela 6: Médias, modas e medianas dos testes do experimento 1.

	pop\gerações	1	10	25	50	100	150
media	5	2118,96	2090,62	2068,17	2065,96	2080,57	2079,05
moda		-	2110	-	2112	2008,6	2008,6
mediana		2106,2	2105,9	2099,9	2107,5	2103,9	2104,55
media	10	2047,8	2052,34	2008	2015,84	2030,88	2005,93
moda		-	2017	2017,8	2017	-	2008,6
mediana		2058,3	2028,55	2017,3	2017	2052,9	2009,2
media	25	1992,37	1963,89	1972,99	1968,61	1948,23	1910,22
moda		2008,6	-	1925,9	1939,6	-	-
mediana		2005,25	1969,5	1967,1	1998,1	1967,45	1919,3
media	50	1895,35	1836,75	1849,44	1799,13	1787,83	1764,06
moda		-	-	-	-	-	-
mediana		1893,2	1882,05	1821,8	1790,9	1799,5	1798,15
media	100	1776,76	1716,52	1643,13	1641,56	1582,23	1578,51
moda		-	-	-	-	-	1561,7
mediana		1780	1677,8	1670,9	1660,1	1579,6	1577,7

Tabela 7: Mínimos e máximos dos testes do experimento 1.

	pop\gerações	1	10	25	50	100	150
min	5	2036,4	1925,1	1925,9	1925,9	2008,6	2008,6
max		2199,3	2201	2199,3	2194,3	2200	2113
min	10	1912,6	1823,9	1911,4	1824,9	1820,2	1808,4
max		2112	2106,3	2031,1	2105,8	2105,8	2103

min	25	1912,6	1823,9	1911,4	1824,9	1820,2	1808,4
max		2102	2095,8	2089,9	2026,4	2036,4	2028,8
min	50	1804	1707,8	1702,8	1687,5	1673	1597,6
Max		2008,6	1973,3	1996,8	1993,3	1876	1883,5
min	100	1.698	1660,9	1566,3	1577,5	1561,7	1561,7
max		1990,7	1902	1708,1	1678,2	1608	1594,2

Analisando as Tabelas 6 e 7, observa-se que as soluções encontradas melhoram consideravelmente à medida que o número de gerações e o tamanho da população aumentam. Para comparação dos resultados obtidos, foi feita também a resolução do problema através de um algoritmo guloso, , que neste caso encontrou como *fitness* da solução 2201. Este guloso é semelhante ao utilizado na geração da população inicial e demonstrado no Pseudoalgoritmo 1, porém agora ele é iniciado a partir do depósito central.

- Melhor solução encontrada: 0, 8, 6, 22, 23, 25, 27, 26, 4, 16, 17, 18, 19, 32, 31, 30, 29, 34, 35, 36, 42, 43, 44, 55, 56, 57, 58, 5, 7, 9, 12, 10, 11, 20, 21, 71, 24, 28, 33, 37, 38, 40, 39, 41, 45, 46, 49, 54, 52, 53, 47, 48, 50, 51, 59, 60, 61, 62, 63, 70, 64, 66, 67, 69, 68, 1, 13, 15, 65, 14, 3, 2.
- *Fitness:* 1561,7.

4.2.2. Experimento 2

Para este experimento utilizou-se uma lista também com 71 pedidos, porém desta vez todos eles possuem janelas de tempo diferentes. Os valores das restrições foram obtidos utilizando o gerador de números aleatórios do Google, com valores entre 100 e 1000. Assim como no experimento anterior, o pedido referente ao nó 8 foi definido como sendo a primeira entrega e seu limite de tempo foi obtido também através do gerador de números aleatórios do Google, porém com limites entre 1 e 50. Os valores das restrições de tempo utilizados estão listados na Tabela 8.

Tabela 8: Lista de pedidos e suas janelas de tempo do experimento 2.

Cliente	Janela de Tempo	Cliente	Janela de Tempo	Cliente	Janela de Tempo
1	771	25	516	49	946
2	544	26	453	50	292
3	886	27	568	51	869
4	186	28	414	52	574
5	271	29	532	53	262
6	109	30	833	54	751
7	506	31	374	55	374
8	10 (Primeira Entrega)	32	831	56	631
9	351	33	555	57	871
10	440	34	383	58	239
11	156	35	526	59	870
12	645	36	178	60	619
13	303	37	989	61	161
14	192	38	529	62	945
15	569	39	826	63	513
16	314	40	554	64	850
17	944	41	740	65	722
18	886	42	151	66	244
19	486	43	643	67	909
20	319	44	925	68	156
21	217	45	497	69	165
22	480	46	460	70	831
23	803	47	982	71	246
24	601	48	860		

Tabela 9: Resultados obtidos nos testes utilizando os parâmetros do experimento 2.

pop\gerações	1	10	25	50	100	150
5	1380,40	1288,90	1291,30	1285,90	1293,70	1293,70
5	1368,30	1295,90	1466,30	1372,10	1293,70	1374,70
5	1467,80	1293,70	1291,50	1282,50	1372,10	1291,50

5	1282,50	1370,50	1288,40	1466,30	1468,80	1377,10
5	1284,70	1291,30	1468,40	1295,90	1467,80	1376,10
5	1396,60	1374,90	1283,90	1375,00	1293,30	1468,20
5	1467,80	1284,70	1372,00	1376,50	1469,90	1293,30
5	1277,50	1463,20	1469,90	1466,30	1293,30	1197,00
5	1372,10	1463,20	1288,40	1370,50	1285,90	1285,90
5	1293,70	1288,40	1377,10	1293,30	1285,90	1372,10
10	1189,80	1283,90	1365,70	1279,60	1365,70	1197,00
10	1281,00	1277,50	1368,70	1280,30	1375,20	1275,30
10	1283,90	1193,30	1189,80	1281,10	1288,70	1277,50
10	1197,00	1288,40	1280,30	1284,70	1192,00	1197,00
10	1293,70	1197,00	1282,50	1366,80	1284,70	1291,10
10	1277,50	1282,40	1283,90	1284,70	1280,20	1371,20
10	1368,60	1189,80	1197,00	1369,70	1194,80	1293,30
10	1285,60	1197,00	1372,50	1193,30	1283,70	1197,00
10	1372,80	1378,90	1273,60	1289,60	1275,30	1268,00
10	1293,60	1277,50	1274,20	1197,00	1276,50	1288,40
25	1190,40	1267,30	1275,40	1197,00	1197,00	1275,50
25	1279,80	1262,20	1282,50	1187,30	1270,90	1277,00
25	1268,60	1190,30	1267,40	1095,70	1277,00	1277,50
25	1095,40	1271,20	1268,69	1182,60	1271,90	1270,50
25	1277,90	1271,10	1277,30	1273,70	1276,70	1264,40
25	1276,60	1264,80	1269,30	1277,00	1282,50	1279,60
25	1265,50	1257,10	1278,30	1287,00	1270,20	1268,80
25	1274,20	1273,80	1271,50	1270,50	1189,70	1272,60
25	1285,90	1269,40	1271,20	1267,40	1197,00	1189,80

25	1271,10	1277,50	1189,80	1190,80	1181,70	1189,50
50	1160,70	1146,10	1044,50	966,50	1165,70	862,00
50	990,80	1250,90	1061,20	1076,10	836,30	1172,40
50	1172,00	1257,90	1070,10	1176,10	1123,70	917,00
50	1163,30	1177,00	1163,70	958,20	1183,30	1054,3
50	1170,10	1159,30	1055,10	974,70	1151,10	1162,70
50	1069,20	1153,20	1197,00	1139,10	1149,90	1065,90
50	1063,60	931,00	1176,80	1069,70	1152,40	1197,00
50	1154,60	1040,00	956,80	1041,60	1152,30	1148,90
50	1061,40	1059,10	1166,90	1065,00	1061,00	1151,50
50	1197,00	1053,70	1053,90	962,60	1047,80	1152,70
100	960,60	909,90	914,00	916,50	918,00	831,30
100	1014,30	1159,40	939,80	1048,30	831,30	934,30
100	1050,90	996,50	927,10	908,30	927,20	891,50
100	928,80	1049,70	1014,40	846,30	839,60	831,70
100	957,40	1031,50	942,10	1000,80	834,40	897,80
100	1039,00	947,20	900,40	839,10	854,30	910,00
100	1053,30	950,80	909,60	954,50	905,80	835,70
100	947,50	962,00	846,00	966,00	1014,30	835,90
100	929,20	899,00	849,20	846,90	939,30	934,30
100	931,30	925,00	911,90	929,50	833,10	838,10

Na Tabela 9 temos os resultados obtidos nos testes feitos utilizando os parâmetros do experimento 2. Assim como no teste do benchmark e no experimento 1, conforme o tamanho da população e o número de gerações crescem, nota-se a melhora das soluções encontradas. Essa melhora nos resultados pode ser mais facilmente observada nas tabelas 10 e 11.

Tabela 10: Médias, modas e medianas dos testes do experimento 2.

	pop\gerações	1	10	25	50	100	150
media	5	1359,14	1341,47	1359,72	1358,43	1352,44	1332,96
moda		1467,8	1463,2	1288,4	1466,3	1293,7	
mediana		1370,2	1294,8	1331,75	1371,3	1293,7	1332,9
media	10	1284,35	1256,57	1288,82	1282,68	1281,68	1265,58
moda		-	1277,5	-	1284,7	-	1197
mediana		1284,75	1277,5	1281,4	1282,9	1281,95	1276,4
media	25	1248,54	1260,47	1265,139	1222,9	1241,46	1256,52
moda		-	-	-	-	1197	-
mediana		1272,65	1268,35	1271,35	1232,2	1270,55	1271,55
media	50	1120,27	1122,82	1094,6	1042,96	1108,411	1088,44
moda		-	-	-	-	-	-
mediana		1157,65	1149,65	1065,65	1053,3	1151,1	1150,2
media	100	1023,873	983,1	915,45	925,62	889,73	874,06
moda		-	-	-	-	-	934,3
mediana		960,6	962	914	929,5	976,8	891,5

Tabela 11: Mínimos e máximos dos testes do experimento 2.

	pop\gerações	1	10	25	50	100	150
min	5	1277,5	1284,7	1283,9	1282,5	1285,9	1197
max		1467,8	1463,2	1469,9	1466,3	1469,9	1468,2
min	10	1095,4	1189,8	1189,8	1095,7	1181,7	1189,5
max		1372,8	1378,9	1372,5	1369,7	1375,2	1371,2
min	25	1095,4	1190,3	1189,8	1095,7	1181,7	1189,5
max		1285,9	1277,5	1282,5	1287	1282,5	1279,6
min	50	990,8	931	956,8	958,2	836,3	862

max		1197	1257,9	1197	1176,1	1183,3	1197
min	100	929	899	846	839,1	831,3	831,3
max		1197	1159,4	1014,4	1048,3	1014,3	934,3

Analisando as Tabelas 10 e 11, observa-se que as soluções encontradas melhoram consideravelmente à medida que o número de gerações e o tamanho da população aumentam. Para comparação dos resultados obtidos, assim como no experimento 1, foi feita também a resolução do problema através de um algoritmo guloso, que neste caso encontrou como *fitness* da solução 1469.9.

- Melhor solução encontrada: 0, 8, 6, 22, 23, 25, 27, 26, 4, 16, 17, 18, 19, 32, 31, 5, 7, 9, 12, 10, 11, 20, 21, 71, 24, 28, 33, 37, 38, 40, 39, 41, 45, 46, 49, 54, 52, 53, 47, 48, 50, 51, 57, 58, 59, 60, 61, 70, 62, 64, 63, 66, 67, 69, 68, 56, 55, 44, 43, 42, 36, 35, 34, 29, 30, 14, 65, 15, 13, 1, 3, 2.
- *Fitness:* 831,3.

5. CONCLUSÃO

Com base na análise dos resultados dos testes do eil76 obtidos no algoritmo desenvolvido, observa-se que com as configurações de população e número de gerações apropriadas, o software consegue encontrar soluções bem próximas ou iguais à solução ótima conhecida.

Feita esta constatação, o algoritmo então é testado utilizando parâmetros próximos da realidade, para que forneça uma solução para o problema real. Assim como no teste do benchmark, a qualidade da melhor solução encontrada aumenta significativamente à medida que o número de gerações e o número de indivíduos crescem. Para efeito de comparação, o resultado encontrado pra o problema real no experimento 1 utilizando um algoritmo guloso foi 2201. Quando se utilizou o AG o melhor resultado foi 1561,7, que corresponde a 70,95% do resultado obtido com o algoritmo guloso. Já no experimento 2, o resultado encontrado utilizando o algoritmo guloso foi 1469.9 e, quando se utilizou o AG, o melhor resultado foi 831,3, que corresponde a 56,55% do resultado obtido com o algoritmo guloso. Essa diferença da redução do custo da melhor solução encontrada pelo AG se dá devido ao fato de todos os pedidos do experimento 2 possuírem restrições de tempo, pois quanto mais restrições houver no problema, mais os métodos que consideram o tempo irão se sobressair em relação a aqueles que não consideram. Portanto, conclui-se que o software é eficaz na resolução do problema.

6. REFERÊNCIAS

LUCAS, Diogo C., Algoritmos Genéticos: Uma Introdução, 2002. Apostila elaborada sob a orientação de Luís Otávio Álvares, para a disciplina de Ferramentas de Inteligência Artificial. Disponível em: <http://www.inf.ufrgs.br/~alvares/INF01048IA/ApostilaAlgoritmosGeneticos.pdf> - Acesso em: 04 maio 17.

OLIVEIRA, Humberto C. B., Um Modelo Híbrido Estocástico Para Tratamento do Problema de Roteamento de Veículos com Janela de Tempo, 2007. 117 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Pernambuco, Recife, PE, 2007. Disponível em: <http://repositorio.ufpe.br/bitstream/handle/123456789/2678/arquivo6093_1.pdf?sequence=1&isAllowed=y> Acesso em: 04 maio 17.

CORDENONSI, André Z.; SANTOS, Willian B. Resolução do Problema de Roteamento de Veículos Utilizando a Heurística Savings. Revista do Centro de Ciências da Economia e Informática, Bagé – RS, v. 5, n. 7, p. 7-14, 2001.

FRANCO JUNIOR *et. al.,* Problema de Roteamento de Veículos com Janela de Tempo. Disponível em: <http://www.bcc.unifal-mg.edu.br/~humberto/disciplinas/2008_2_po/apresentacoes/roteamento.pdf > Acesso em: 05 maio 17.

SOUZA JUNIOR, Arnaldo T., Proposta Para o Problema de Roteamento de Veículos Dinâmico com Janela de Tempo, 2007. Disponível em: <http://repositorio.ufla.br/bitstream/1/5437/1/MONOGRAFIA_Proposta_para_o_problema_de_roteamento_de_veiculos_dinamico_com_janela_de_tempo.pdf> Acesso em: 07 maio 17.

VIEIRA, Heloísa P., Metaheurística para a Solução de Problemas de Roteamento de Veículos com Janela de Tempo, 2013. 108 f. Dissertação (Mestrado em Matemática Aplicada) – Universidade Estadual de Campinas,

Campinas, SP, 2008. Disponível em: <http://www.ime.unicamp.br/~chico/tese_heloisa.pdf> Acesso em: 07 maio 17.

REINA, Caio D., Roteirização de Veículos com Janela de Tempo Utilizando Algoritmo Genético, 2012, 90 f. Dissertação (Mestrado em Engenharia) – Escola Politécnica da Universidade de São Paulo, São Paulo, SP, 2012. Disponível em: <www.teses.usp.br/teses/disponiveis/3/3138/tde-06062013.../ReinaCD_mestrado.pdf> Acesso em: 09 maio 17.

MARINHO, Cássia J., Um Estudo da Solução do Problema de Roteamento de Veículos com Janela de Tempo Via Metaheurísticas, 2013. 47 f. Dissertação (Mestrado em Modelagem Matemática e Computacional) – Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, MG, 2013. Disponível em: <http://www.files.scire.net.br/atrio/cefet-mg-ppgmmc_upl/THESIS/196/dissertacaocassiadejesusmarinho.pdf> Acesso em: 10 maio 17.

SILVA, Paulo T.G.C., Uma Variação do Push-Forward Insertion Heuristic aplicada ao Problema de Roteamento de Veículos com Janelas de Tempo, XL Simpósio Brasileiro de Pesquisa Operacional (SBPO), 2008, João Pessoa, PB.

Notas de aulas do Prof. Silvio Alexandre de Araujo. Disponível em: <http://wiki.icmc.usp.br/images/3/32/Pi_aula_16_11_finalizacaoPI_mari.pdf> Acesso em: 12 maio 17

MACULAN, Nelson; FAMPA, Marcia H.C., Otimização Linear, 2004. Disponível em: <https://www.researchgate.net/profile/Nelson_Maculan/publication/268292974_Otimizacao_Linear/links/54c3ce3d0cf256ed5a926e42.pdf> Acesso em: 12 maio 17.

SOUZA, Marcone J. F., Otimização Combinatória, Notas de Aula, 2009/2. 106 f. Disponível em: <http://www.decom.ufop.br/marcone/Disciplinas/OtimizacaoCombinatoria/OtimizacaoCombinatoria.pdf> Acesso em 14 maio 17.

LOPES, Heitor S.; RODRIGUES, Luis Carlos A.; STEINER, Maria T. A. (Eds), Meta-Heurísticas em Pesquisa Operacional, 22 ed., Curitiba, PR, Omnipax, 2013. 472 p. ISBN: 978-85-64619-10-4.

POZO *et. al,* Computação Evolutiva, Departamento de Informática, Universidade Federal do Paraná. Disponível em: <http://www.inf.ufpr.br/aurora/tutoriais/Ceapostila.pdf> Acesso em: 16 maio 17.

CARMO, Everton C.; GOMES, Heider A. S.; BARROS NETO, Júlio F. Roteamento de Veículos no Transporte Rodoviário de Cargas: Uma Aplicação para a Distribuição de Jornais, XXXV Simpósio Brasileiro de Pesquisa Operacional (SBPO), 2003, Natal, RN.

RIBEIRO, Gladyston M.; RUIZ, Maria D. V; DEXHEIMER, Letícia. Programa de Roteamento de Veículos Aplicação no Sistema de Coleta dos Correios, Encontro Nacional de Engenharia de Produção (ENEGEP), 2001, Salvador, BA.

TSPLIB está disponível *on-line em:* <https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>.

APÊNDICE A

A tabela a seguir representa a relação entre os vértices do grafo e seus respectivos bairros ou regiões correspondentes:

Vértice	Bairro/região	Vértice	Bairro/região	Vértice	Bairro/região
0	Depósito	24	São Bento	48	Leonane
1	São Geraldo	25	Res. Santanense	49	Santa Mônica II
2	Res. São Geraldo	26	Alaita	50	Piaguassu
3	Morada Nova	27	JK	51	Vila Israel
4	Morada Nova II	28	Tropical	52	Jadir Marinho
5	Santa Edwirges	29	Cerqueira Lima	53	Centenário
6	Garcias	30	Piedade	54	Santa Mônica
7	Aeroporto	31	Santo Antônio	55	Universitário II
8	Aeroporto II	32	Lourdes	56	Universitário
9	Cidade Nova	33	Nova Vila Mozart	57	São Judas Tadeu
10	Murilo Gonçalves	34	Belvedere	58	Vila Augusto Chaves
11	Três Marias	35	Graças	59	NRVGS
12	Cidade Nova II	36	Centro	60	Santanense
13	Chácara do Quitão	37	Olímpio Moreira	61	Paulo II
14	Novo Horizonte	38	Palmeiras	62	Vila Santa Maria
15	Vila Nazaré	39	Veredas II	63	Distrito Industrial
16	Morro do Sol II	40	Veredas	64	Granja Glória
17	Morro do Sol	41	Pe. Eustáquio	65	Bela Vista
18	Pio XII	42	Santiago	66	Lopes
19	Nogueira Machado	43	Vila Tavares	67	Campos
20	Nogueirinha	44	Eldorado	68	Vale dos Ipês
21	Morro do Engenho	45	Vila Vilaça	69	Pirobrás
22	Itaunense	46	Irmãos Auler	70	Trevo Santanense
23	Parque Jardim	47	Várzea da Olaria	71	Itatiaiuçu

APÊNDICE B

A tabela a seguir mostra a lista de arestas do grafo que representa o mapa de Itaúna-MG. Por exemplo, a primeira aresta “0 1 5.5”, representa uma aresta entre os nós 0 e 1 com peso 5.5.

0 1 5.5	0 2 2.6	0 6 4.5	0 7 4.1
0 8 2.8	1 3 3.3	1 65 4.5	1 13 3.2
1 2 4.5	2 3 2.2	2 4 6.0	3 4 4.8
3 14 5.5	4 16 3.7	4 5 3.9	5 7 5.0
6 26 8.8	6 22 7.2	6 8 2.3	6 69 11.1
7 8 3.4	7 9 4.4	7 11 6.0	8 69 8.9
9 11 4.4	9 12 2.6	9 10 4.6	10 11 3.7
10 12 8.0	13 28 4.4	13 15 4.1	14 65 2.0
14 15 3.2	14 30 5.5	15 65 3.0	15 28 4.6
16 17 3.0	17 18 3.3	18 19 2.7	18 31 5.1
19 20 4.0	19 21 5.2	19 32 3.2	20 21 3.9
21 32 7.3	21 71 12.0	22 69 10.6	22 23 6.2
22 24 8.5	22 66 18.6	23 24 7.1	23 25 5.6
24 25 10.6	25 27 8.2	25 60 11.1	26 27 1.8
26 13 6.1	27 28 5.5	27 13 7.1	27 60 9.3
28 29 6.0	28 34 6.8	29 30 3.8	29 34 3.7
29 35 6.9	30 31 5.5	30 36 7.4	31 32 3.5
32 33 3.6	33 37 5.2	33 36 7.7	34 35 3.8
35 36 6.1	35 57 9.4	35 58 9.4	36 42 6.1
36 55 9.0	37 42 8.4	38 40 3.0	38 39 4.0
39 40 2.0	39 41 5.5	40 41 3.8	40 43 7.7
41 43 5.7	41 45 3.0	41 46 4.2	42 43 2.9
42 71 28.0	43 44 2.5	43 45 3.8	44 55 3.3
44 54 5.7	45 46 5.8	46 47 5.3	46 48 5.0
46 49 2.5	47 48 2.0	47 49 6.5	47 50 5.2
47 51 5.4	49 54 5.0	49 52 7.1	49 50 6.0
50 51 3.7	52 54 3.7	52 53 4.0	52 55 15.7
53 55 15.5	55 56 4.3	55 70 26.0	56 57 4.9
57 58 1.5	57 59 7.2	58 59 7.5	59 60 11.5
60 61 6.3	60 70 10.5	61 70 9.8	62 63 14.8
62 64 11.0	63 64 13.2	63 55 47.8	67 66 5.5
67 68 6.6	68 69 2.2	69 67 4.4	70 62 7.0