



UNIFOR

**UNIVERSIDADE DE FORTALEZA
CENTRO DE CIÊNCIAS TECNOLÓGICAS
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

VICTOR TIEZZI HENRIQUES

**HEURÍSTICAS RÁPIDAS PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS
COM COLETA E ENTREGA SIMULTÂNEAS E FROTA HETEROGÊNEA**

FORTALEZA – CEARÁ

2020

VICTOR TIEZZI HENRIQUES

HEURÍSTICAS RÁPIDAS PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM
COLETA E ENTREGA SIMULTÂNEAS E FROTA HETEROGÊNEA

Trabalho de Conclusão de Curso apresentado ao
Curso de Graduação em Ciência da Computação
do Centro de Ciências Tecnológicas da Univer-
sidade de Fortaleza, como requisito parcial à
obtenção do grau de bacharel em Ciência da
Computação.

Orientador: Napoleão Vieira Nepomuceno

FORTALEZA – CEARÁ

2020

Ficha catalográfica da obra elaborada pelo autor através do programa de geração automática da Biblioteca Central da Universidade de Fortaleza

Henriques, Victor.

Heurísticas rápidas para o problema de roteamento de veículos com coleta e entrega simultâneas e frota heterogênea / Victor Henriques. - 2020
49 f.

Trabalho de Conclusão de Curso (Graduação) - Universidade de Fortaleza. Curso de Ciência Da Computação, Fortaleza, 2020.
Orientação: Napoleão Nepomuceno.

1. Algoritmo Guloso. 2. Problema de Roteamento de Veículo.
3. Semi-guloso. 4. Concorrente. 5. Parada Repentina. I.
Nepomuceno, Napoleão. II. Título.

**HEURÍSTICAS RÁPIDAS PARA O PROBLEMA DE ROTEAMENTO DE
VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEAS E FROTA
HETEROGÊNEA**

VICTOR TIEZZI HENRIQUES

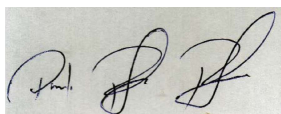
PARECER: APROVADO

Data: 25 / 06 / 2020

BANCA EXAMINADORA:

Rommel Dias Saraiva

ROMMEL DIAS SARAIVA, Dr.



PLÁCIDO ROGERIO PINHEIRO, Dr.

Napoleão V. Nepomuceno

NAPOLEÃO VIEIRA NEPOMUCENO, Dr.
Orientador(a)



Liádina Camargo Lima
Coordenadora do Curso Ciência da Computação

AGRADECIMENTOS

Primeiramente aos meus pais e irmão, pelo amor, incentivo e apoio incondicional.

Aos meus amigos, Fernanda Caroline, Rommel Sousa e Thiago Bochensky por me apoio durante todo o curso.

A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. A palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos.

RESUMO

O Problema de Roteamento de Veículo com Coleta e Entrega Simultânea e Frota Heterogênea (do inglês *Heterogeneous Vehicle Routing Problem with Simultaneous Pickup and Delivery*, (HVRPSPD)), tem como objetivo gerar uma rota para uma frota de veículos heterogênea minimizando os custos e realizando o atendimento de toda a demanda dos clientes. Esse problema é comumente aplicado em empresas que desejam diminuir seus custos de logística. O objetivo deste trabalho é criar heurísticas gulosas que gerem soluções viáveis com custos baixos para o HVRPSPD, por esse motivo, foram desenvolvidas 3 heurísticas baseadas no *Nearest-Neighbor-Based Randomized Algorithm* (NNRA) com as estratégias: Concorrente, Parada Repentina e Semi-Gulosa. As heurísticas desenvolvidas são comparadas ao algoritmo NNRA quanto à eficácia na resolução de instancias da literatura. Os resultados encontrados mostraram que a heurística da parada repentina destacou-se por apresentar melhoria significativa na qualidade das soluções.

Palavras-chave: Algoritmo Guloso. Problema de Roteamento de Veículo. Semi-guloso. Concorrente. Parada Repentina

ABSTRACT

The Heterogeneous Vehicle Routing Problem with Simultaneous Pickup and Delivery (HVRPSPD) aims to generate a route for a heterogeneous vehicle fleet minimizing costs and meeting all customer demand. This problem is commonly applied to companies that want to reduce their logistics costs. The aim of this work is to create greedy heuristics that generate viable solutions with low costs for HVRPSPD, for this reason, 3 heuristics were developed based on the Nearest-Neighbor-Based Randomized Algorithm (NNRA): Concurrent, Sudden Stop and Semi-Greedy. The performance of the developed heuristics are compared to the NNRA. The results found showed that the heuristic of the sudden stop stood out for presenting significant improvement in the quality of the solutions.

Keywords: Greedy Algorithm. Vehicle Routing Problem. Nearest-Neighbor-Based Randomized Algorithm. Sudden Stop. Semi-Greedy. Competitive

LISTA DE ILUSTRAÇÕES

| | |
|---|-----------|
| Figura 1 – Exemplificando um grafo | 15 |
| Figura 2 – Exemplificando da montagem de um grafo | 16 |
| Figura 3 – Um grafo exemplificando o HVRPSPD | 17 |
| Figura 4 – A rota Vizinho Mais Próximo à esquerda e a rota ótima à direita | 18 |
| Figura 5 – Um mapa rodoviário simplificado de parte da Romênia | 21 |
| Figura 6 – Distâncias em linha reta até Bucharest | 21 |
| Figura 7 – Gráfico representativo do funcionamento da calibração | 29 |
| Figura 8 – Média de solução | 44 |
| Figura 9 – Média de Iteração | 44 |

LISTA DE TABELAS

| | |
|---|-----------|
| Tabela 1 – instâncias | 41 |
| Tabela 2 – Menor solução - NNRA Concorrente | 42 |
| Tabela 3 – Menor solução - NNRA Parada Repentina | 45 |
| Tabela 4 – Menor solução - NNRA Semi-Guloso | 46 |

LISTA DE ALGORITMOS

| | |
|---|-----------|
| Algoritmo 1 – Dar o Troco | 19 |
| Algoritmo 2 – Nearest-Neighbor-Based Randomized Algorithm (NNRA) | 33 |
| Algoritmo 3 – Algoritmo Base | 34 |
| Algoritmo 4 – Código de Calibração | 35 |
| Algoritmo 5 – NNRA Concorrente - algoritmo | 36 |
| Algoritmo 6 – NNRA Parada Repentina - algoritmo | 37 |
| Algoritmo 7 – NNRA Semi-Guloso - algoritmo | 38 |
| Algoritmo 8 – NNRA Semi-Guloso - Escolha e Validação do Cliente | 39 |

LISTA DE ABREVIATURAS E SIGLAS

| | |
|---------|--|
| CVRP | Problema de Roteamento Capacitado de Veículos |
| HVRPSPD | <i>Heterogeneous Vehicle Routing Problem with Simultaneous Pickup and Delivery</i> |
| NNRA | <i>Nearest-Neighbor-Based Randomized Algorithm</i> |
| PCV | Problema do Caixeiro Viajante |
| VMP | Vizinho Mais Próximo |
| VRP | <i>Vehicle Routing Problem</i> |
| VRPPD | Problema de Roteamento de Veículos com Coleta e Entrega |
| VRPSPD | Problema de Roteamento de Veículos com Coleta e Entrega Simultânea |

SUMÁRIO

| | | |
|--------------|---|-----------|
| 1 | INTRODUÇÃO | 13 |
| 1.1 | MOTIVAÇÃO | 13 |
| 1.2 | OBJETIVOS | 14 |
| 1.2.1 | Objetivo Geral | 14 |
| 1.2.2 | Objetivos Específicos | 14 |
| 2 | FUNDAMENTAÇÃO TEÓRICA | 15 |
| 2.1 | GRAFOS | 15 |
| 2.2 | PROBLEMA DE ROTEAMENTO DE VEÍCULO | 15 |
| 2.2.1 | Problema de Roteamento de Veículo com Coleta e Entrega Simultânea e Frota Heterogênea(HVRPSPD) | 17 |
| 2.3 | ALGORÍTIMOS GULOSOS | 18 |
| 2.3.1 | Problema do Caixeiro Viajante e a Heurística do Vizinho Mais Próximo | 18 |
| 2.3.2 | Problema do Troco ou Problema de Frobenius | 19 |
| 2.3.3 | Algoritmos semi-gulosos | 20 |
| 2.4 | MÉTODOS DE BUSCA | 20 |
| 2.4.1 | Busca Cega ou Busca sem Informação | 20 |
| 2.4.2 | Busca Heurística ou Busca com Informação | 21 |
| 3 | TRABALHOS RELACIONADOS | 23 |
| 4 | METODOLOGIA | 25 |
| 4.1 | NNRA | 25 |
| 4.2 | OS ALGORITMOS | 26 |
| 4.2.1 | A Base | 27 |
| 4.2.2 | A Calibração | 28 |
| 4.2.3 | NNRA Concorrente | 30 |
| 4.2.4 | NNRA Parada Repentina | 31 |
| 4.2.5 | NNRA Semi-Guloso | 31 |
| 5 | RESULTADOS | 40 |
| 5.1 | RESULTADOS DO NNRA CONCORRENTE | 41 |
| 5.2 | RESULTADOS DA PARADA REPENTINA | 42 |
| 5.3 | RESULTADOS DO SEMI-GULOSO | 43 |
| 6 | CONCLUSÕES | 47 |

| | | |
|-----|--|-----------|
| 6.1 | CONTRIBUIÇÕES DO TRABALHO | 47 |
| 6.2 | LIMITAÇÕES E TRABALHOS FUTUROS | 47 |
| | REFERÊNCIAS | 48 |

1 INTRODUÇÃO

O Problema de Roteamento de Veículo (do inglês, *Vehicle Routing Problem* (VRP)), é um problema onde o veículo precisa gastar a menor quantidade de recursos para realizar um caminho passando por todos os seus clientes e retornando ao ponto de origem. Este problema possui muitas variantes para se adequar a diferentes contextos do mesmo problema, como o problema de roteamento de veículos multi-depósito, onde nele há mais de um depósito no problema, ou o problema de roteamento de veículos com limite de distância onde há uma restrição na distância que o veículo pode percorrer.

Este trabalho terá foco no Problema de Roteamento de Veículo com Coleta e Entrega Simultânea e Frota heterogênea (do inglês, *Heterogeneous Vehicle Routing Problem with Simultaneous Pickup and Delivery* (HVRPSPD)), que é um problema onde veículos com capacidades variadas realizam a coleta e entrega de conteúdo em uma única visita a seus clientes. O seu desafio está em criar um conjunto de rotas para os veículos de forma que sua capacidade não seja excedida durante o trajeto e que esse conjunto de rotas consigam ter um custo baixo em relação aos custos fixos e variáveis dos tipos dos caminhões e das rotas utilizadas.

Neste trabalho, foi realizado um estudo de heurísticas alternativas à proposta apresentada no trabalho de Nepomuceno, Saboia e Pinheiro (2019), utilizando da base de dados de Avci e Topaloglu (2016), para melhorar os custos totais gerados pela solução no intuito de aproximar do ótimo global.

1.1 MOTIVAÇÃO

Segundo o IBGE - Instituto Brasileiro de geografia e estatística (2014), O transporte de cargas no Brasil é predominantemente realizado por meios de rodovias. Cerca de 61,1% da carga transportada, segundo a Confederação Nacional de Transportes (2018), utilizou o modal rodoviário, cuja rede tem maior difusão pelo território nacional. Tendo isso em mente, 55% dos custos totais das empresas correspondem aos custos logísticos, esses custos totais, representaram 12,3% do Produto Interno Bruto (PIB) brasileiro em 2016.

Isso mostra a importância de mitigar o VRP, pois está presente em muitos sistemas de entrega no mundo. No trabalho de Melo e Ferreira Filho (2001) é mostrado na prática que as soluções de VRP são benéficas para as empresas que adotarem esses sistemas como, aumentar a quantidade de produtos vendidos, diminuir a quantidade de veículos da frota de caminhões, diminuir a ociosidade dos veículos da frota de caminhões, e reduzir a quantidade de quilômetros

rodados dos veículos da frota de caminhões. No Brasil as empresas que fabricam produtos como geladeira, pilhas, computadores, entre outros, são responsáveis pelos resíduos industriais provenientes da fabricação seus produtos de acordo com a Lei Nº 12.305, de 2 de agosto de ago. 2010 (STABELINI, 2018). Essas empresas adotam a logística reversa para amenizar o impacto ecológico e social. Esta logística se encaixa bem com o HVRPSPD que é o problema que este trabalho se dedica a otimizá-lo.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Este trabalho tem como objetivo de pesquisar e implementar heurísticas para HVRPSPD e conseguir menores custos totais gerados pela solução em comparação com os resultados de Nepomuceno, Saboia e Pinheiro (2019) no intuito de aproximar do ótimo global.

1.2.2 Objetivos Específicos

- a) Realizar pesquisas sobre heurísticas que se adequam ao HVRPSPD
- b) Implementar as heurísticas para o HVRPSPD no intuito de alcançar melhores soluções, escapando de ótimos locais fracos
- c) Realizar uma bancada de testes e registrar os resultados
- d) Analisar e comparar o desempenho dos algoritmos propostos com resultados de Nepomuceno, Saboia e Pinheiro (2019)

2 FUNDAMENTAÇÃO TEÓRICA

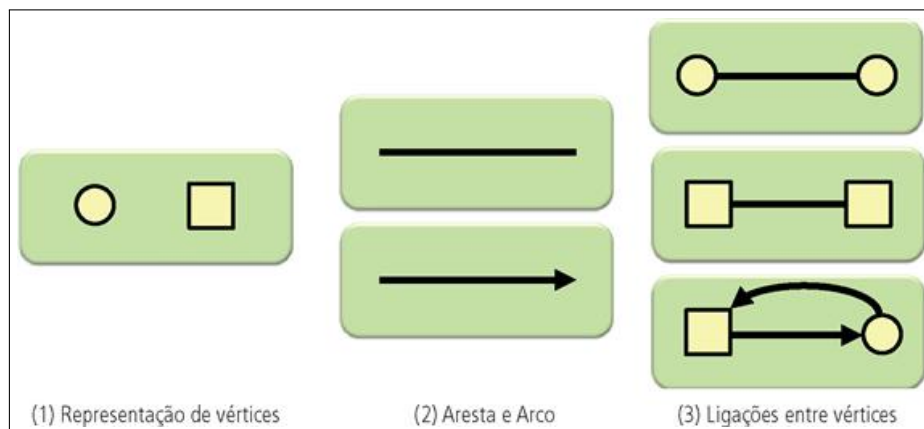
2.1 GRAFOS

De acordo com Goldbarg e Elizabeth (2012), um grafo é uma estrutura de abstração onde formaliza relações de interdependência entre elementos de um conjunto. Ele é bastante útil na representação e solução de diversos tipos de problema.

Na Figura 1, exibe as possíveis formas de vértices, que representam os elementos do conjunto, exibe as possíveis formas de arestas ou arcos, que representam as conexões dos elementos do conjunto, e por fim, exibe como os vértices se conectam através das arestas ou arcos.

É demonstrado o processo de modelagem de um grafo baseado em um mapa do Rio Grande do Norte na Figura 2, onde as cidades são elementos de um conjunto e eles estão representadas por vértices e suas rodovias que fazem a conexão entre as cidades são representadas por arestas, assim criando um grafo.

Figura 1 – Exemplificando um grafo

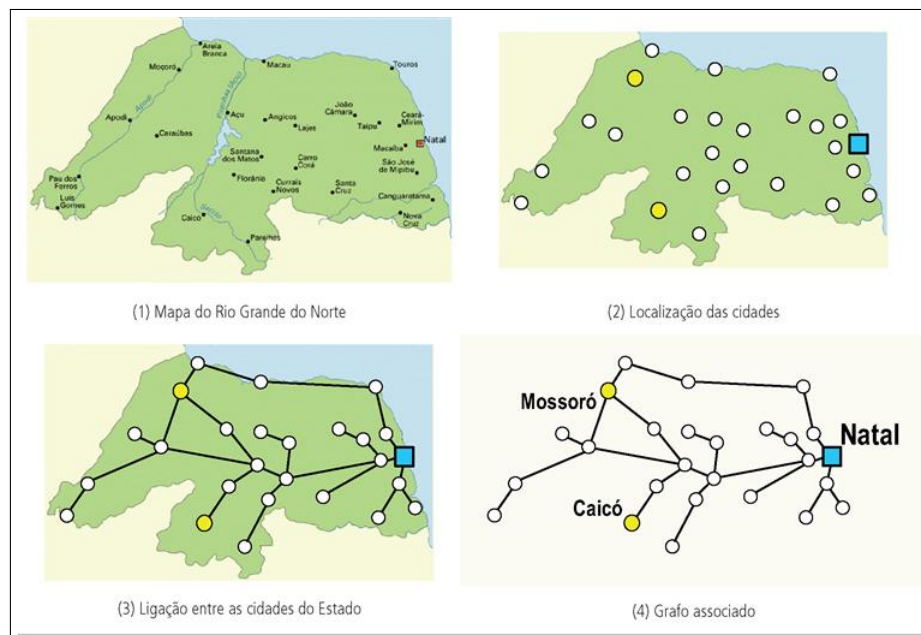


Fonte – Goldbarg e Elizabeth (2012)

2.2 PROBLEMA DE ROTEAMENTO DE VEÍCULO

O Problema de Roteamento de Veículo (do inglês, VRP), estabelece "decisões que minimizem os custos de estocar produtos em depósitos geograficamente dispersos e transportar esses produtos através de diferentes vias empregando diferentes veículos para atender uma demanda não homogênea e também geograficamente dispersa"(GOLDBARG; GOLDBARG; LUNA, 2016).

Figura 2 – Exemplificando da montagem de um grafo



Fonte – Goldberg e Elizabeth (2012)

O VRP é formulado por 1 depósito onde é o ponto de partida do caminhão, clientes que devem ser visitados apenas uma vez para entregar o produto, e a rota que o caminhão deverá realizar para visitar todos os clientes e retornar ao seu ponto de origem. O objetivo é encontrar o menor caminho para sua rota. O VRP possui uma grande variação de problemas que são definidas por suas restrições. Aqui estão alguns deles:

- Problema de Roteamento Capacitado de Veículos (CVRP) - Os veículos passam a ter uma capacidade máxima.
- Problema de Roteamento de Veículos com Coleta e Entrega (VRPPD) - Os veículos passam a realizar a coleta e entrega para seus clientes.
- Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (VRPSPD) - Os clientes passam a ser visitados em uma única vez para realizar a coleta e entrega.
- Problema de Roteamento de Veículo com Coleta e Entrega Simultânea e Frota Heterogênea (HVRPSPD) - Os veículos passam a possuir diferentes capacidades para realizar as suas rotas.

2.2.1 Problema de Roteamento de Veículo com Coleta e Entrega Simultânea e Frota Heterogênea(HVRPSPD)

O HVRPSPD pode ser ilustrado utilizando a Figura 3. Existem 5 pontos denominados 0, 1, 2, 3 e 4, onde 0 é o depósito (*depot*) e os demais pontos são os clientes, os quais possuem uma demanda de entrega d_i e coleta p_i . Todos os pontos são interligados com seu custo de travessia ao lado. Neste exemplo, utilizamos 2 caminhões, o primeiro com 5 de capacidade máxima e o segundo com 10. Nenhum caminhão excedeu a sua capacidade máxima, fazendo assim o seguinte conjunto de rotas uma solução viável.

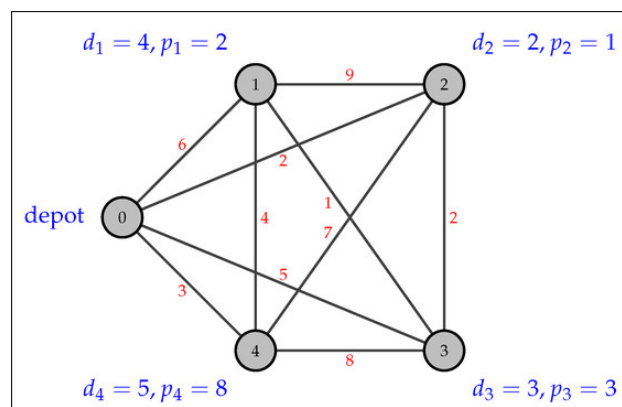
O primeiro caminhão com 5 de capacidade máxima, realiza a seguinte rota:

- Sai do ponto 0 para o ponto 3 com 5 para entrega e com 0 de coleta, tendo 5 de carga total;
- Sai do ponto 3 para o ponto 2 com 2 para entrega e com 3 de coleta, tendo 5 de carga total;
- Sai do ponto 2 para o ponto 0 com 0 para entrega e com 4 de coleta, tendo 4 de carga total.

E então, o segundo caminhão com 10 de capacidade máxima, realiza a seguinte rota:

- Sai do ponto 0 para o ponto 1 com 9 de entrega e com 0 de coleta, tendo 9 de carga total;
- Sai do ponto 1 para o ponto 4 com 5 de entrega e com 2 de coleta, tendo 7 de carga total;
- Sai do ponto 4 para o ponto 0 com 0 de entrega e com 10 de coleta, tendo 10 de carga total.

Figura 3 – Um grafo exemplificando o HVRPSPD



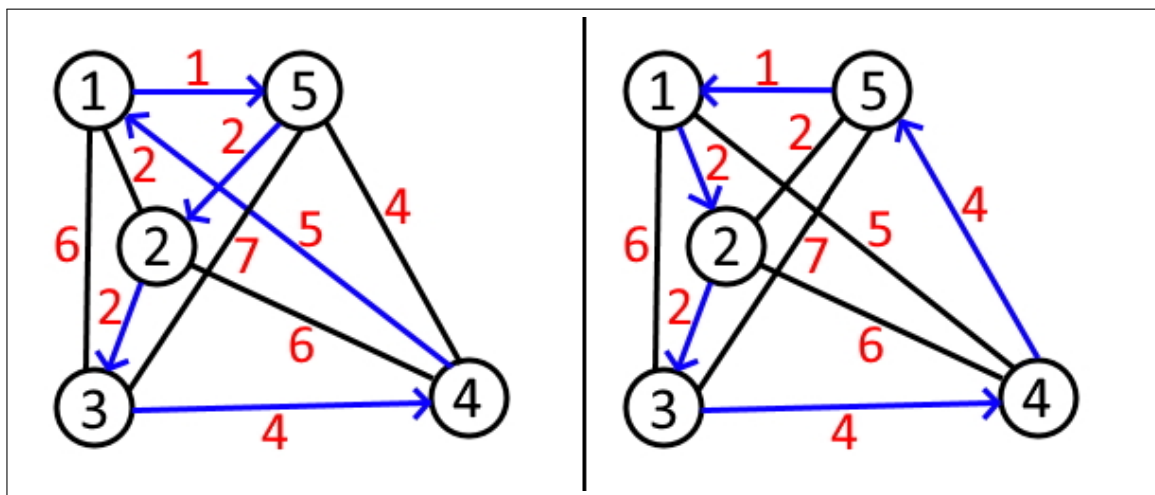
Fonte – Nepomuceno, Saboia, Pinheiro (2019)

2.3 ALGORÍTIMOS GULOSOS

De acordo com Cormen *et al.* (2012), "Os algoritmos gulosos são uma abordagem heurística que sempre buscam fazer a melhor escolha local na esperança de encontrar a solução ótima global", ou seja, a partir de uma sequência de escolhas, com foco na melhor opção de escolha a cada iteração, esse algoritmo é capaz de obter uma solução viável, não podendo provar que a solução seja ótima.

2.3.1 Problema do Caixeiro Viajante e a Heurística do Vizinho Mais Próximo

Figura 4 – A rota Vizinho Mais Próximo à esquerda e a rota ótima à direita



Fonte – Elaborado pelo autor

De acordo com Applegate *et al.* (2011), “O Problema do Caixeiro Viajante (PCV) é, dado a lista de cidades com o custo de deslocamento entre todas elas, achar o menor custo visitando todas as cidades, sem revisitá-las e retornando à cidade de origem”. A ordem de visita das cidades é chamada de rota ou solução. Uma abordagem gulosa para esse problema seria a heurística do Vizinho Mais Próximo (VMP), onde o viajante escolhe a cidade mais próxima, ou seja, a de menor custo para ele naquele dado momento que o mesmo se encontra, sem poder voltar atrás de sua escolha até realizar seu objetivo. Um exemplo disso seria a Figura 4 onde a rota criada usando a heurística do VMP tem custo 14 em comparação com a rota ótima com o custo 13.

2.3.2 Problema do Troco ou Problema de Frobenius

O Problema do Troco, ou Problema de Frobenius, é um problema onde dado o valor, é realizada a troca do mesmo por um conjunto de números pré-definidos. Por exemplo, supondo que há 3 números diferentes, denominados dentro deste problema como moedas, onde seus valores são, respectivamente, 5, 2 e 1 e tem-se como objetivo de gerar o troco para o valor 8, é possível produzir as soluções {5, 2, 1}, {5, 1, 1, 1}, {2, 2, 2, 2}, {2, 2, 2, 1, 1}, entre outras.

Algoritmo 1: Dar o Troco

Entrada: n = Valor do troco, C = Conjunto de moedas

Saída: S = Conjunto de moedas

início

$S \leftarrow \phi$;

$s \leftarrow 0$;

enquanto $s \neq n$ **faça**

$x \leftarrow$ o maior número em C sendo que $s + x \leq n$;

se o número não existe **então**

retorna "Solução não encontrada";

fim

$S \leftarrow S \cup \{\text{a moeda de valor } x\}$;

$s \leftarrow s + x$;

fim

retorna S ;

fim

Fonte - Brassard, Bratley (1995)

O Algoritmo 1 é um exemplo de algoritmo guloso, onde em cada iteração é sempre escolhida a moeda com maior valor dentro do conjunto de moedas disponíveis no problema, sem se preocupar com a qualidade dessa escolha e essas iterações irão ocorrer até uma solução ser encontrada ou o algoritmo não ser capaz de encontrar qualquer solução.

O objetivo do Algoritmo 1 é entregar uma solução para o problema do troco, mas dependendo do conjunto de moedas e do número a dar o troco, talvez o algoritmo não encontre uma solução mesmo que existam respostas para o problema abordado no algoritmo. Tomando como exemplo um conjunto de duas moedas, sendo elas 7 e 8, onde o valor ao qual se deseja gerar troco seja 14, o Algoritmo 1 não será capaz de apresentar uma solução, uma vez que irá

escolher na primeira iteração a moeda com maior valor, nesse caso 8, o que ocasiona que, na próxima iteração, o algoritmo já não seja capaz de gerar uma combinação de moedas que tenham valor igual a 14.

2.3.3 Algoritmos semi-gulosos

De acordo com Hart e Shogan (1987), os algoritmos semi-gulosos possui uma característica de realizar várias escolhas gulosas e a partir disso realizar uma escolha aleatória entre as gulosas para diversificar os resultados.

Usando como referência a Figura 4, no ponto 1, o algoritmo semi-guloso poderia indicar os dois vizinhos mais próximos, os nós 2 e 5, e realizar uma escolha aleatória entre eles, e assim, realizar uma escolha aleatória entre elas. Com isso, a rota ótima poderá ser encontrada após o algoritmo ser executado algumas vezes.

2.4 MÉTODOS DE BUSCA

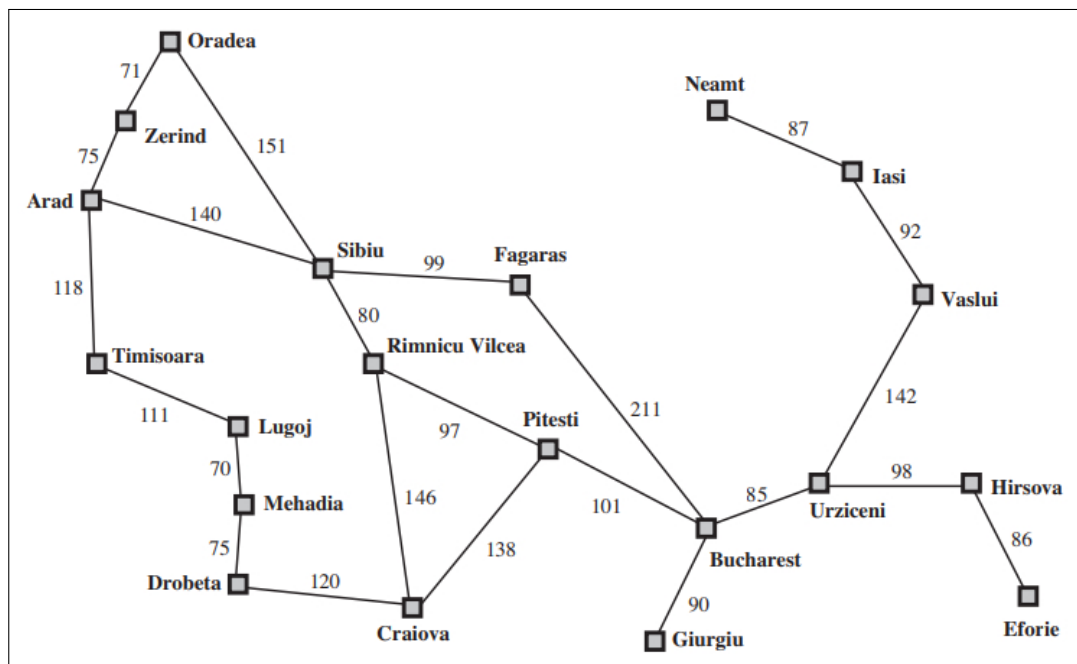
De acordo com Russell e Norvig (2004), “Um algoritmo de busca recebe um problema como entrada e retorna uma solução sob a forma de uma sequência de ações”. Existem 2 subgrupos de métodos de buscas, sendo esses as buscas cegas e as buscas heurísticas.

2.4.1 Busca Cega ou Busca sem Informação

A busca sem informação "não tem nenhuma informação adicional sobre os estados, além daquelas fornecidas na definição do problema. Tudo o que elas podem fazer é gerar sucessores e distinguir um estado objetivo de um estado não-objetivo"(RUSSELL; NORVIG, 2004).

Um exemplo de busca cega seria a busca em grafo, onde todos os nós recém-descobertos são guardados em uma fila e após expandidos, são mandados para uma lista de nós já visitados. Utilizando-se da Figura 5 e da lógica da busca em grafo, o caminho de Arad até Bucharest é {Arad, Sibiu, Fagaras, Bucharest} e apesar dessa ser uma solução para o problema, essa solução não é ótima.

Figura 5 – Um mapa rodoviário simplificado de parte da Romênia



Fonte – Russell, Norvig (2004)

2.4.2 Busca Heurística ou Busca com Informação

A busca com informação é “uma estratégia que utiliza o conhecimento específico do problema, além da definição do próprio problema” e que “pode encontrar soluções de forma mais eficiente que uma estratégia sem informação” (RUSSELL; NORVIG, 2004).

Figura 6 – Distâncias em linha reta até Bucharest

| | | | |
|------------------|-----|-----------------------|-----|
| Arad | 366 | Mehadia | 241 |
| Bucharest | 0 | Neamt | 234 |
| Craiova | 160 | Oradea | 380 |
| Drobeta | 242 | Pitesti | 100 |
| Eforie | 161 | Rimnicu Vilcea | 193 |
| Fagaras | 176 | Sibiu | 253 |
| Giurgiu | 77 | Timisoara | 329 |
| Hirsova | 151 | Urziceni | 80 |
| Iasi | 226 | Vaslui | 199 |
| Lugoj | 244 | Zerind | 374 |

Fonte – Russell, Norvig (2004)

Utilizando-se da Figura 5 e da Figura 6, o caminho de Arad até Bucharest, tomando como base o funcionamento da heurística do vizinho mais próximo da seção 2.3.1, é {Arad, Sibiu, Fagaras, Bucharest}, o mesmo caminho feito pela busca sem informação. A diferença

dessa heurística seria a utilização da Figura 6 para determinar os custos da rota é que ela expande menos nós e não os guarda em memória.

3 TRABALHOS RELACIONADOS

No artigo de Melo e Ferreira Filho (2001), mostra quão importante é aderir um sistema de roteirização para redução de custos operacionais, aumentar a qualidade de serviço, fidelidade de clientes e aumento no lucro da empresa.

O trabalho de Armon, Avidor e Schwartz (2010) introduz e realiza o estudo do problema do caixeiro viajante com variantes cooperativas, o qual apresenta como característica a participação dos clientes como, por exemplo, a partir do cliente se aproximando do caixeiro ou atrás da venda de mercadoria para outros clientes. Considerando a cooperação dos clientes e dos objetivos de interesse, como minimizar distância percorrida ou minimizar o tempo para a realização de todas as entregas, esse trabalho que dentre os problemas alguns apresentam algoritmo de aproximação constante, muitos outros esquema de aproximação em tempo polinomial e outros possuem soluções em tempo polinomial.

Angelelli *et al.* (2014) apresentam uma análise de três variações do problema do caixeiro viajante com a presença de lucro e é feito o estudo da complexidade em casos do problema com e sem tempo de serviço associados aos clientes e, finalmente, apresenta algoritmos polinomiais e esquema de aproximação de tempo polinomial (FPTAS) para, respectivamente, casos polinomialmente solucionáveis e alguns casos considerados NP-Difícil.

No artigo de Avci e Topaloglu (2016), abordam o HVRPSPD criando um algoritmo de busca local híbrida mesclando algoritmo de Aceitação de Limite (do inglês, *Threshold Accepting*) e Busca Tabu (do inglês, *Tabu Search*).

Na dissertação de Saboia (2019) é avaliada a metodologia Gerar-E-Resolver na construção de soluções para o VRPSPD, propondo operadores de vizinhança com o papel de atuarem como geradores de instâncias reduzidas do problema original e após a identificação das instâncias pelo mesmo, essas são resolvidas a partir da utilização do CPLEX com a utilização da heurística de subida de encosta. Por fim, os resultados são comparados a resultados presentes na literatura.

O VRPPD também é abordado por Fernandes (2019) onde, tomando como referência o modelo apresentado por Avci e Topaloglu (2016), é feita uma modelagem do VRPPD utilizando programação linear inteira e, adicionalmente, são realizadas alterações com o objetivo de tornar o mesmo capaz de gerar soluções para um maior número de situações reais. Aditivamente, é desenvolvida uma heurística gulosa e são realizados testes utilizando o modelo e a heurística propostos e instâncias 21 presentes na literatura.

O HVRPSPD foi abordado no artigo de Nepomuceno, Saboia e Pinheiro (2019) onde, tomando como referência o modelo apresentado por Avci e Topaloglu (2016), utilizam um algoritmo *Nearest-Neighbor-Based Randomized Algorithm* (NNRA) para demonstrar que é possível ter uma solução equivalente ou melhor as apresentadas no artigo de Avci e Topaloglu (2016) em menos tempo de execução. Neste algoritmo, foi utilizado busca cega, ou seja, quando uma nova solução é iniciada, ela não possui nenhuma referencia de soluções anteriores, e as rotas dos caminhões são guiada com vizinho mais próximo na qual há uma chance desse guia ser aleatório. Esse caminhão, primeiramente, recebe um cliente aleatório obrigatório e posteriormente clientes mais próximos da última escolha ou aleatórios até que o caminhão não seja capaz de comportar mais clientes. Além disso, em toda geração da rota, existe uma chance de que o próximo cliente escolhido seja aleatório ao invés do vizinho mais próximo, fazendo assim, a solução ser capaz de escapar de ótimos locais. No final da iteração, caso a solução atual seja melhor que a melhor solução encontrada até aquele momento, irá substituí-la e iniciar uma nova iteração.

4 METODOLOGIA

Este capítulo tem como objetivo explicar o comportamento dos algoritmos NNRA Concorrente, NNRA Parada Repentina e NNRA Semi-Guloso que foram desenvolvidos neste trabalho para, em seguida, realizar uma avaliação do desempenho dos algoritmos mencionados acima. Esses algoritmos foram criados baseados no NNRA do artigo de Nepomuceno, Saboia e Pinheiro (2019) e estão disponíveis no site "github.com/VictorTiezzi/HVRPSPD".

4.1 NNRA

O NNRA foi um algoritmo desenvolvido por Nepomuceno, Saboia e Pinheiro (2019) que entrega uma solução viável em pouco tempo de execução. Esse algoritmo utiliza-se de uma heurística gulosa baseada no vizinho mais próximo e, levando em consideração o tempo limite de execução pré-estabelecido, o algoritmo irá encontrar múltiplas soluções e selecionará a melhor solução encontrada.

O algoritmo utiliza-se de busca cega, uma vez que, uma nova solução não possui nenhuma referência de soluções anteriores encontradas pelo algoritmo, e as rotas dos caminhões são geradas a partir do princípio de vizinho mais próximo, considerando-se que há uma chance desse guia ser aleatório.

Esse trabalho destaca-se por, ao utilizar o NNRA, apresentar soluções competitivas com o mesmo tempo de execução e eficácia particularmente boa ao considerar a probabilidade p igual a 0,99 de utilizar o critério de vizinho mais próximo, e apresenta bom desempenho mesmo com instâncias de tamanho consideravelmente grandes.

O NNRA representado no Algoritmo 2 possui como dado de entrada uma probabilidade p que faz referência a se a escolha do cliente for aleatório ou for o vizinho mais próximo, o tempo limite t é utilizado para limitar o tempo de execução do algoritmo. Adicionalmente, na inicialização do algoritmo é criada uma variável chamada "Melhores Rotas", que será responsável por armazenar as melhores rotas encontradas.

Em seguida, é feito o primeiro teste de condição, o qual verifica se o tempo atual ultrapassa o tempo limite o algoritmo, caso a condição seja falsa todo o bloco subordinado será pulado, o que acarretará no encerramento do algoritmo e o retorno das Melhores Rotas, caso contrário, ou seja, a condição seja verdadeira será iniciado uma nova iteração.

Logo após, é realizada a criação de duas variáveis, denominas Rotas Atuais vazias, que irá armazenar todas as rotas dos veículos criados nesta iteração, e lista de clientes da instância

criada de ordem aleatória. Enquanto a lista de clientes não estiver vazia, ou seja, clientes não forem alocados para a rota de algum veículo, será criado um novo veículo e, conseqüentemente, também será inicializada uma rota para o mesmo e todas as marcações de inviabilidade são retiradas dos clientes em "Lista de clientes".

Seguidamente, enquanto o veículo atual ainda for capaz de realizar testes com algum cliente que esteja dentro de "Lista de clientes", será feito o processo de escolha de um cliente para a rota de um determinado veículo.

Prontamente, a variável Cliente escolhido é criada para armazenar um cliente e uma variável responsável por armazenar um número é criada e o valor atribuído a mesma será selecionado de forma randômica com base no intervalo 0 a 1. Caso aconteça de a condição de o cliente atual não for o primeiro cliente da rota e o número aleatório for menor que a probabilidade, como consequência a escolha do cliente será feita a partir da utilização da estratégia do vizinho mais próximo, tendo como referência a posição atual do veículo. Caso a condição seja falsa, o cliente deverá ser escolhido de forma aleatória.

Sem demora, o cliente passa então a ser armazenado em Cliente Escolhido. Na checagem prontamente a seguir verifica-se se o cliente escolhido é viável para a rota do veículo, ou seja, se o cliente está dentro de todas as restrições do HVRPSPD. No caso da condição ser verdadeira o cliente é inserido na rota do veículo e caso contrario, o Cliente é marcado como inviável para a rota do veículo.

Posteriormente, ao término de toda a validação é verificado se ainda há algum cliente válido que pode ser inserido na rota do veículo e, caso não tenha, a rota do veículo será encerrada e, imediatamente, guardada na variável Rotas Atuais. E, finalmente, se a Lista de Clientes estiver vazia, é verificado se Rotas atuais apresenta menor custo total comparado ao custo total de Melhores Rotas, e por fim, se o tempo limite for batido, o NNRA retornará as Melhores Rotas.

4.2 OS ALGORITMOS

Neste trabalho o algoritmo desenvolvido foi subdividido em cinco partes, as quais são denominadas de Base, calibração, NNRA Concorrente, NNRA Parada repentina e NNRA Semi-Guloso. Essa subdivisão foi feita tendo como objetivo manter uma melhor organização e, também, considerando a possibilidade de reutilização do algoritmo desenvolvido para, por exemplo, trabalhos futuros.

As cinco subdivisões, serão devidamente detalhadas no decorrer da metodologia.

4.2.1 A Base

No algoritmo desenvolvido neste trabalho, a subdivisão denominada base representa a essência do mesmo devido ao fato de ser nessa etapa do algoritmo que irá ocorrer o processo de calibração e repetição durante a execução das heurísticas NNRA Concorrente, NNRA Parada repentina e NNRA Semi-Guloso.

As heurísticas desenvolvidas, por terem como base VRP, apresentam como objetivo a minimização dos custos fixos e variáveis. O custo fixo é determinado pela utilização de um veículo na rota de solução, valendo ressaltar que o custo fixo pode ter diferentes valores dependendo dos tipos de veículo que compõem a frota. Por outro lado, o custo variável é um valor característico do veículo que é multiplicado pela distância euclidiana da rota percorrida pelo veículo. O custo total das rotas de solução é obtido através da somatória dos custos fixo e variável dos veículos utilizados na solução encontrada.

A Base representado no Algoritmo 3, observa-se que, dada uma entrada batizada de "Tempo limite", o algoritmo se inicia criando uma variável responsável por armazenar as melhores rotas e no processo de inicialização é feita uma atribuição de valor nulo a mesma. Em seguida é feita a chamada de um método intitulado "Calibração" o qual, a partir dos parâmetros "número de Subida Máxima" e o "número de Testes", o qual irá retornar um número inteiro que representará o "número N Calibrado" do algoritmo.

Os parâmetros "número de Subida Máxima" e "número de Testes" são variáveis inteiras, que, foram criados a partir da execução de testes utilizando as instâncias geradas por Avci e Topaloglu (2016) e, com o objetivo de evitar a extrapolação do tempo limite da fase de calibração, foram atribuído o valores fixados de 4 e 50 respectivamente.

A variável "número N Calibrado", precisa ser calibrada pois os algoritmos NNRA Concorrente, NNRA Parada Repentina e NNRA Semi-Guloso precisam de um número estático para o andamento da lógica, e essa lógica pode trazer melhores soluções dependendo do número escolhido.

No passo da repetição, a variável "número N Calibrado", após ter passado pela fase de calibração, será utilizada na heurística escolhida, a qual será representada pela variável "Rotas Atuais", repetidas vezes até que o limite de tempo de execução seja alcançado. A cada iteração é realizado um processo de checagem, onde se verifica se a variável "Melhores Rotas" estiver vazia ou se o custo do valor encontrado daquela dada iteração for efetivamente melhor do que o custo previamente armazenado então é realizado a atualização de "Melhores Rotas" para o

valor encontrado na iteração. E, por fim, a heurística irá retornar um conjunto de rotas que serão armazenadas em "Melhores Rotas".

4.2.2 A Calibração

A subdivisão denominada calibração é responsável por determinar o valor da variável "Número N Calibrado". Este método é dependente da entrada de 2 números inteiros, os quais são o número de subida máxima e o número de testes para calibração. Essas duas variáveis foram fixadas como 4 e 50 respectivamente, com objetivo de não extrapolar o tempo limite na fase de calibração.

Inicialmente, são instanciadas 3 variáveis denominadas "número N a ser Calibrado", "Mapa" e "Subida". A variável "número N a ser Calibrado" é uma variável auxiliar que será usada como entrada na heurística escolhida e será responsável por determinar o sequenciamento das iterações do laço presente no algoritmo 3 e a variável em questão sofrerá acréscimos durante o decorrer do algoritmo que será descrito em mais detalhes abaixo.

O "Mapa" é armazenado como uma lista de lista, o qual irá funcionar a partir do princípio de uma chave e um valor. A chave será equivalente à variável "número N a ser calibrado", enquanto o valor a uma lista de custo totais da heurística utilizada e irá depender da variável "número N a ser calibrado" a qual foi utilizada na heurística.

A variável "Subida" determina a condição de parada do Algoritmo 4. No caso de essa variável ter valor igual ou superior ao valor armazenado em "número de subida máxima" a condição de parada é atingida.

Em seguida, após a inicialização das variáveis descritas, enquanto o critério de parada não for satisfeito cria-se uma lista denominada "lista de soluções", uma variável auxiliar que armazena os custos totais da bancada de testes para ser inserido no "Mapa".

Considerando o "número máximo de testes", o qual foi informado nos parâmetros do algoritmo, em seguida é, respectivamente, executada a heurística com o "número N a ser calibrado" e o valor encontrado é armazenado "Rotas Atuais". Logo depois, o custo total, ou seja, os custos fixo e variável, vinculados ao valor em "Rotas Atuais" é guardado na "Lista de Custos Totais".

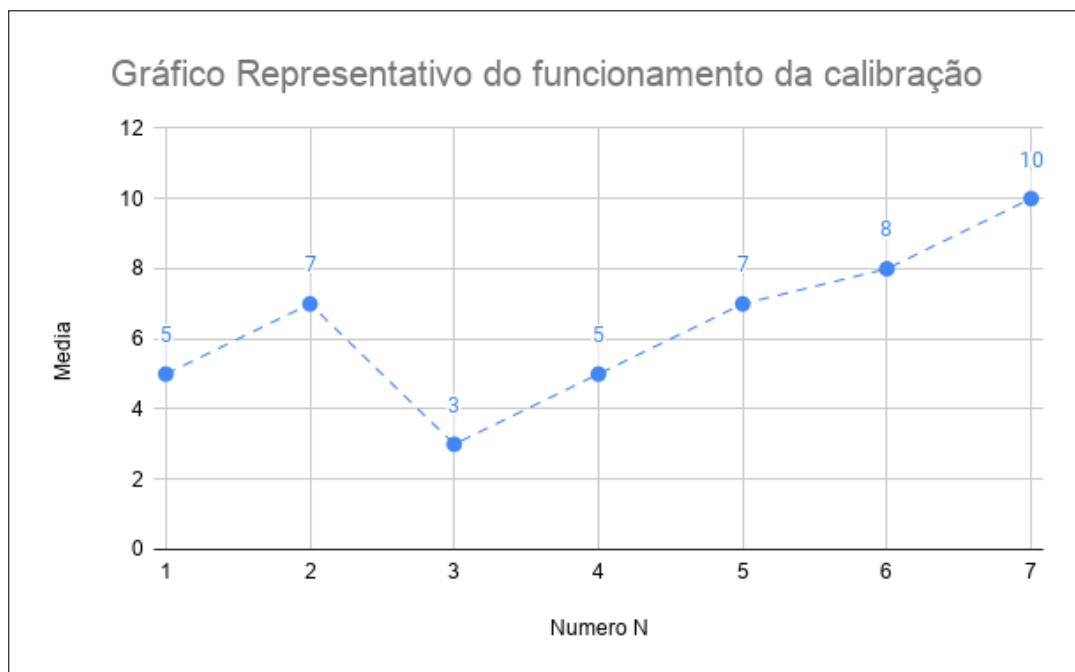
Sem demora, são realizadas algumas verificações, sendo estas a verificação da variável "Melhores Rotas", e caso esta possua valor nulo ou o custo do valor encontrado na iteração atual for, de fato, melhor do que o custo previamente atribuído a "Melhores Rotas", atualiza-se

o valor da mesma com o valor encontrado. Imediatamente depois, a variável "Mapa" recebe a "Lista de Custos Totais", adicionalmente, é atribuído "número N a ser calibrado" como chave, com o objetivo de facilitar o acesso e localização dos dados na lista.

Prontamente, ao término da realização de testes com a variável "número N a ser Calibrado" igual a 2, é calculada a média das soluções da bancada de testes referente à iteração atual é comparada com a média das soluções das bancada anterior. Após realizada essa comparação, se a média atual possuir valor maior do que a média referente à bancada de testes anterior, a variável denominada "Subida" é incrementada em 1, caso contrario, é subtraída em 1, sendo o limite inferior dessa variável igual a 0.

Seguidamente, é verificado o valor de "Subida" e se porventura o valor desta variável for equivalente ao "Número Máximo de Subida", o critério de parada é alcançado, ocasionando no término do algoritmo e, finalmente, é realizada uma procura com o objetivo de encontrar o menor valor de solução entre todos os valores de todas as listas e, a chave, que se encontra na variável mapa, vinculada à lista desse valor, será atribuído ao número "N". Caso o critério de parada não seja alcançado, é feito o incremento de 1 na variável "Subida" e inicia-se uma nova iteração até que, por fim, o critério de parada seja efetivamente alcançado.

Figura 7 – Gráfico representativo do funcionamento da calibração



Fonte – Elaborado pelo autor

Baseando na Figura 3, assumindo valor de N igual a 7, teremos no total 7 iterações, ao verificarmos a segunda iteração, podemos observar que a média dos custos totais apresenta

valor superior do que a média encontrada na primeira iteração, acarretando no acréscimo de uma unidade na variável "subida". Em contrapartida, observando a terceira iteração, na qual o "Número N a ser calibrado" possui valor igual a 3, podemos mirar que a sua média apresenta valor menor quando comparada a iteração anterior, consequentemente é realizado o decréscimo em uma unidade na variável "Subida".

Ao observarmos a terceira até a sétima iterações, podemos contemplar que ocorrerá o acréscimo da "Subida" quatro vezes consecutivas, e, ao considerarmos que a variável que delimita o critério de parada possui valor igual a 4, podemos inferir que na sétima iteração o critério de parada será alcançado, para esse exemplo, como consequência da "Subida" ser igual a variável "número Máximo de Subida" o processo de calibração será encerrado.

Adicionalmente, vale ressaltar que, quando o critério de parada é alcançado, é realizada a eleição do valor da variável "número N" e, tendo como base o exemplo apresentando na Figura 7, partindo do suposto que a solução que apresenta menor valor esteja na lista do mapa referente à chave da variável N igual a 3, o "Número N calibrado" irá receber como retorno dessa função o valor 3.

4.2.3 NNRA Concorrente

Na heurística desenvolvida neste trabalho representada no Algoritmo 5, é criado um NNRA que apresenta uma modificação a partir da inserção de concorrência de veículos no algoritmo. Essa adaptação permite que o NNRA, após a criação da quantidade de veículos, atribua os clientes para a composição das rotas dos veículos e execute esse procedimento de forma concomitante. Na ocorrência de situações onde a lista que armazena os veículos, que irão ser tratados de forma concorrente, estiver vazia, ou seja, os veículos efetivamente já atenderam o número máximo de clientes dentro dos limites de sua capacidade, e ainda existir clientes que não foram visitados, a heurística retorna a trabalhar com atribuição de clientes a veículos de forma sequencial.

Ao comparar o NNRA Concorrente com o NNRA, e observando o Algoritmo 5, podemos atentar que passa a existir uma variável denominada "Lista de Veículos" com N veículos escolhidos de forma aleatória entre os disponíveis da instância. Podemos ressaltar também que, apesar do método de seleção e validação dos clientes para averiguar se o cliente deve ser inserido na rota do veículo é o mesmo em relação ao método do NNRA, há uma pequena alteração, a qual a inserção dos clientes passa a acontecer de acordo com a ordem da variável "Lista de Veículos"

Pode-se destacar que o NNRA Concorrente em comparação ao NNRA, devido ao fato de os clientes serem distribuídos para veículos de forma simultânea, para instâncias de tamanho igual ou superior a 50 clientes, em particular, nota-se que o valor encontrado da média de custo total é ligeiramente melhor ao encontrado pelo NNRA.

4.2.4 NNRA Parada Repentina

Na heurística NNRA Parada Repentina representada em Algoritmo 6, que foi desenvolvida neste trabalho, nota-se a presença de uma nova verificação em comparação ao NNRA, na qual é averiguado se o cliente selecionado é efetivamente válido para ser inserido na lista de clientes de um veículo e, caso o contrario aconteça, ou seja, o cliente falhe no processo de validação o mecanismo de parada repentina é executado, no qual um contador de falhas é incrementado em uma unidade inteira e, na ocorrência de PR falhas de validação, sendo PR um valor predeterminado, o veículo será finalizado mesmo que ainda existam clientes para serem verificados como válidos para inserção na lista de cliente deste veículo. Pode-se verificar que, na heurística de NNRA Parada Repentina, a quantidade de iterações e o número do PR escolhido são variáveis inversamente proporcional.

4.2.5 NNRA Semi-Guloso

A estratégia semi-gulosa representada no Algoritmo 7 é utilizada no desenvolvimento da heurística NNRA Semi-Guloso, na qual é feita a seleção de N clientes mais próximos e será realizada uma escolha aleatória dentre os clientes previamente selecionados, com a objetivo de tentar escapar de ótimos locais.

A modificação realizada para incluir a estratégia semi-gulosa consiste na criação de uma "Lista de Clientes Escolhidos". Na primeira iteração, o primeiro cliente escolhido para rota sempre será aleatório. A partir da segunda iteração em diante, todos os clientes não visitados estarão na "Lista de Clientes Auxiliar" e, nesta lista, será feita uma ordenação com base na distância atual do veículo em relação ao cliente da lista. Ao término da ordenação, os N clientes serão realocados para "Lista de Clientes Escolhidos" a partir de uma estratégia de escalonamento, na qual o primeiro elemento a ser retirado da "Lista de Clientes Auxiliar" será o primeiro a ser inserido na "Lista de Clientes Escolhidos."

No Algoritmo 8 são realizadas a escolha de um cliente e sua validação. Um índice aleatório da lista de clientes escolhidos é selecionado e o cliente armazenado na posição

correspondente ao índice eleito irá para validação de rota do veículo.

Caso ocorra do cliente armazenado no índice selecionado não ser válido para aquela dada rota, o índice será atualizado para o antecedente do atual, e o processo irá se repetir. Na ocorrência do índice selecionado em uma dada iteração representar o início da lista, ou seja, ser igual a 0, o antecessor será o índice que representa o final da lista de clientes escolhidos. O critério de parada ocorrerá quando toda a lista tiver sido percorrida, ou seja, quando houver o retorno ao índice selecionado inicialmente de forma randômica. Vale ressaltar que, na ocorrência de uma lista de clientes escolhidos que apresenta valor nulo, a estratégia semi-gulosa não poderá ser utilizada, logo o algoritmo irá retornar a estratégia do NNRA.

Algoritmo 2: Nearest-Neighbor-Based Randomized Algorithm (NNRA)

Entrada: P = Probabilidade, T = Tempo Limite

Saída: Melhores Rotas

início

Melhores Rotas Vazia;

enquanto *tempo atual* $\leq T$ **faça**

Rotas Atuais vazias;

Lista de clientes da instância;

enquanto *Lista de Clientes não estiver vazia* **faça**

Veículo \leftarrow Criar 1 Veículo da instância escolhido aleatoriamente;

Limpe as marcações de Inviabilidade nos clientes;

enquanto *Houverem Clientes para o Veículo* **faça**

Cliente Escolhido Vazio;

número Aleatório entre 0 e 1;

se *Não for o primeiro cliente do veículo* **E** *número Aleatório* $< P$ **então**

Cliente Escolhido \leftarrow um cliente baseado no vizinho mais próximo que não seja um cliente já visitado por este veículo;

fim

senão

Cliente Escolhido \leftarrow um cliente qualquer da lista de clientes que não seja um cliente já visitado por este veículo;

fim

se *o Cliente Escolhido for Viável* **então**

Adicione o cliente escolhido na rota do veículo;

fim

senão

Marque esse cliente como inviável;

fim

Marque o Veículo como "Sem clientes para o Veículo";

para todo *Cliente em Lista de Clientes* **faça**

se *Houver Clientes não visitados* **então**

Marque o Veículo como "Com clientes para o Veículo";

fim

fim

fim

Feche a rota do Veículo;

Rotas Atuais \leftarrow Rota do Veículo;

fim

se *Melhores Rotas = Nulo* **OU** *custo(Rotas Atuais) < custo(Melhores Rotas)* **então**

Melhores Rotas \leftarrow Rotas Atuais

fim

fim

retorna *Melhores Rotas*

fim

Algoritmo 3: Algoritmo Base

Entrada: tempo limite**Saída:** Melhores Rotas**início** Melhores Rotas \leftarrow nulo; número N Calibrado \leftarrow calibração(número de Subida Máxima, número de Testes para a Calibração); **enquanto** $tempo\ atual \leq tempo\ limite$ **faça** Rotas Atuais \leftarrow algoritmo(número N Calibrado); **se** $Melhores\ Rotas = Nulo$ **OU** $custo(Rotas\ Atuais) < custo(Melhores\ Rotas)$ **então** Melhores Rotas \leftarrow Rotas Atuais **fim** **fim** **retorna** *Melhores Rotas*;**fim**

Algoritmo 4: Código de Calibração

Entrada: número Máximo de Subida, número Máximo de Testes para a Calibração

Saída: número N calibrado

início

número N a ser calibrado = 1;

Mapa \leftarrow HashMap<número N a ser calibrado, Lista de Custos Totais>;

Subida = 0;

enquanto *Critério de parada não for atingido* **faça**

 Lista de Custos Totais;

para *cada número Máximo de Testes para a Calibração* **faça**

 Rotas Atuais \leftarrow algoritmo(número N a ser calibrado);

 Lista de Custos Totais \leftarrow custo(Rotas Atuais);

se *Melhores Rotas = Nulo* **OU** *custo(Rotas Atuais) < custo(Melhores Rotas)* **então**
 | Melhores Rotas \leftarrow Rotas Atuais

fim

fim

 Mapa \leftarrow (número N a ser calibrado, Lista de Custos Totais);

se *número N a ser calibrado $\neq 1$ E Media(Mapa(número N a ser calibrado)) -*
 Media(Mapa(número N a ser calibrado - 1)) ≥ 0 **então**

 | Subida \leftarrow Subida + 1;

fim

senão

 Subida \leftarrow Subida - 1;

se *Subida < 0* **então**

 | Subida \leftarrow 0;

fim

fim

se *Subida \geq número Máximo de Subida* **então**

 | Critério de parada atingido;

 | número N calibrado \leftarrow número N a ser calibrado com o menor resultado do Mapa;

fim

senão

 | número N a ser calibrado \leftarrow número N a ser calibrado + 1;

fim

fim

retorna *número N calibrado*

fim

Algoritmo 5: NNRA Concorrente - algoritmo

Entrada: N = número De Veículos

Saída: Lista de Rotas

início

Lista de Rotas vazias;

Lista de clientes da instância;

Lista de Veículos \leftarrow Criar N Veículos da instância escolhidos aleatoriamente;

enquanto *Lista de Clientes não estiver vazia* **faça**

se *Lista De Veículos estiver vazia* **então**

 Lista de Veículos \leftarrow Criar 1 Veículo da instância escolhido aleatoriamente;

fim

para todo *Veículo na Lista de Veículos* **faça**

 Cliente Escolhido Vazio;

 número Aleatório entre 0 e 1;

se *Não for o primeiro cliente do veículo E número Aleatório < 0.99* **então**

 Cliente Escolhido \leftarrow um cliente baseado no vizinho mais próximo que não seja um cliente já visitado por este veículo;

fim

senão

 Cliente Escolhido \leftarrow um cliente qualquer da lista de clientes que não seja um cliente já visitado por este veículo;

fim

se *Nenhum cliente for escolhido* **então**

 Marque este Veículo para não receber mais clientes;

 Feche a rota do Veículo;

 Lista de Rotas \leftarrow Rota do Veículo;

 Pule para o próximo Veículo da Lista de Veículos;

fim

se o *Cliente Escolhido for Viável* **então**

 Adicione o cliente escolhido na rota do veículo;

fim

senão

 Marque esse cliente como inviável para este veículo;

fim

fim

 Todos os Veículos marcados para não receber mais clientes são retirados da Lista de Veículos;

fim

para todo *Veículo na Lista de Veículos* **faça**

 Feche a rota do Veículo;

 Lista de Rotas \leftarrow Rota do Veículo;

fim

retorna *Lista de Rotas*

fim

Algoritmo 6: NNRA Parada Repentina - algoritmo

Entrada: PR = Parada Repentina

Saída: Lista de Rotas

início

Lista de Rotas vazias;

Lista de clientes da instância;

enquanto *Lista de Clientes não estiver vazia* **faça**

 Veículo \leftarrow Criar 1 Veículo da instância escolhido aleatoriamente;

 número PR_Atual = 0;

 Limpe as marcações de Inviabilidade nos clientes;

enquanto *Houverem Clientes para o Veículo* **faça**

 Cliente Escolhido Vazio;

 número Aleatório entre 0 e 1;

se *Não for o primeiro cliente do veículo E número Aleatório < 0.99* **então**

 Cliente Escolhido \leftarrow um cliente baseado no vizinho mais próximo que não seja um cliente já visitado por este veículo;

fim

senão

 Cliente Escolhido \leftarrow um cliente qualquer da lista de clientes que não seja um cliente já visitado por este veículo;

fim

se *o Cliente Escolhido for Viável* **então**

 Adicione o cliente escolhido na rota do veículo;

fim

senão

 Marque esse cliente como inviável;

fim

se *Cliente foi marcado como Inviável* **então**

 PR_Atual \leftarrow PR_Atual + 1;

se *PR_Atual \geq PR* **então**

 Termine o Enquanto;

fim

fim

 Marque o Veículo como "Sem clientes para o Veículo";

para todo *Cliente em Lista de Clientes* **faça**

se *Houver Clientes não visitados* **então**

 Marque o Veículo como "Com clientes para o Veículo";

fim

fim

fim

 Feche a rota do Veículo;

 Lista de Rotas \leftarrow Rota do Veículo;

fim

retorna *Lista de Rotas*

fim

Algoritmo 7: NNRA Semi-Guloso - algoritmo

Entrada: N = número De Clientes

Saída: Lista de Rotas

início

Lista de Rotas vazias;

Lista de clientes da instância;

enquanto *Lista de Clientes não estiver vazia* **faça**

 Veículo ← Criar 1 Veículo da instância escolhido aleatoriamente;

 Limpe as marcações de Inviabilidade nos clientes;

enquanto *Houverem Clientes para o Veículo* **faça**

 Cliente Escolhido Vazio;

 Lista de Clientes Escolhidos é criado;

se *Não for o primeiro cliente do veículo* **então**

 Lista de clientes auxiliar é criado;

para todo *cliente em lista* **faça**

 Lista auxiliar ← um cliente baseado no vizinho mais próximo que não seja um cliente já visitado por este veículo;

fim

 Organizar em ordem crescente a lista auxiliar;

 Separar os "N" primeiros da lista e alocar na lista de escolhidos;

fim

senão

 Cliente Escolhido ← um cliente qualquer da lista de clientes que não seja um cliente já visitado por este veículo;

fim

 Escolha e Validação do Cliente em algoritmo 8;

 Marque o Veículo como "Sem clientes para o Veículo";

para todo *Cliente em Lista de Clientes* **faça**

se *Houver Clientes não visitados* **então**

 Marque o Veículo como "Com clientes para o Veículo";

fim

fim

fim

 Feche a rota do Veículo;

 Lista de Rotas ← Rota do Veículo;

fim

retorna *Lista de Rotas*

fim

Algoritmo 8: NNRA Semi-Guloso - Escolha e Validação do Cliente

```

início
  se Lista de escolhidos não estiver vazia então
    número Aleatório entre 0 ao tamanho máximo da lista de escolhidos;
    número  $\acute{I}ndex \leftarrow$  número Aleatório;
    enquanto número Aleatório diferente de  $\acute{I}ndex$  faça
      Escolha o cliente da Lista de Clientes Escolhidos na posição do  $\acute{I}ndex$ ;
      se o Cliente Escolhido for Viável então
        | Adicione o cliente escolhido na rota do veículo;
      fim
      senão
        | Marque esse cliente como inviável;
        | decmente 1 em  $\acute{I}ndex$ ;
        se  $\acute{I}ndex < 0$  então
          |  $\acute{I}ndex \leftarrow$  tamanho da Lista de Clientes Escolhidos - 1;
        fim
      fim
    fim
  fim
  senão
    se o Cliente Escolhido for Viável então
      | Adicione o cliente escolhido na rota do veículo;
    fim
    senão
      | Marque esse cliente como inviável;
    fim
  fim
fim

```

5 RESULTADOS

O ambiente computacional utilizado foi um *Desktop* rodando o sistema operacional Windows 10 versão 2004, Ryzen 5 de segunda geração 2600X Six-Core com 19MB de Cache e 3.6 GHz de clock, uma placa de vídeo dedicada Radeon RX 590 com 8GB GDDR5 de memória Gráfica, com 16 GB (2x8GB) de memória RAM DDR4 com 2400 MHz. Adicionalmente, todas as heurísticas foram desenvolvidas utilizando a linguagem de programação Java 13.0.2.

Todas as instâncias, as quais foram cedidas por Avci e Topaloglu (2016), como apresentado na Tabela 1, são agrupadas em 2 conjuntos. O primeiro conjunto contém as instâncias tituladas de 101 a 114, as quais são consideradas pequenas devido ao fato de possuir uma quantidade de clientes que varia de 10 a 100. O segundo conjunto contém as instâncias tituladas 201 a 214, sendo elas consideradas grandes como consequência da quantidade de clientes variar entre 150 a 550, salientando que a posição de todos os clientes alteram-se entre as instâncias e, adicionalmente, há 1 depósito em cada instância, o qual delimita o início e término das rotas.

A quantidade de veículos que compõem a frota para cada instância testada varia de 2 a 4. Cada veículo possui como características um custo fixo, que está relacionado à utilização do veículo, um custo variável, o qual é dependente da distância euclidiana, sendo esta calculada a partir da multiplicação entre custo variável e a distância, e a capacidade total que ele pode transportar durante o trajeto. Vale ressaltar também que, em relação aos dados de tipos de veículos apresentados na Tabela 1, como custos fixo e variável, variam de instância para instância assim como a posição de todos os clientes.

Em relação aos parâmetros do algoritmo, os mesmos apresentam valores pré-definidos. Na variável "Número Máximo de subida", responsável por delimitar a quantidade de subidas realizados no algoritmo 4.2.2, foi designado o valor 4 para que não ocorresse uma quantidade maior de testes do que o necessário durante a fase de calibração.

A variável "Número Máximo de Testes para a Calibração", encarregada pela quantidade de iterações por laço, foi designado o valor 50 para que não extrapolasse o tempo limite de 1 minuto.

A variável "Probabilidade", é incumbida de determinar a probabilidade de se utilizar uma estratégia VRP ou aleatória na escolha do cliente a ser adicionada na rota de um veículo, possui valor 0,99, pois no artigo de Nepomuceno, Saboia e Pinheiro (2019) ficou comprovado que quanto mais próximo do valor 1, melhores rotas são apresentadas. A variável "tempo limite",

é responsável por limitar o tempo de execução do algoritmo 4.2.1, foi designado o valor de 60 segundos para comparar com os resultados de Nepomuceno, Saboia e Pinheiro (2019).

Foram feitos 10 testes de cada algoritmo em cada instância, levando em conta que foram feitos testes do NNRA no ambiente computacional deste trabalho e os testes do artigo de Nepomuceno, Saboia e Pinheiro (2019)

Tabela 1 – instâncias

| Instância | Tipos de Veículos | Clientes |
|-----------|-------------------|----------|
| 101 | 2 | 10 |
| 102 | 2 | 10 |
| 103 | 3 | 15 |
| 104 | 4 | 15 |
| 105 | 3 | 20 |
| 106 | 4 | 20 |
| 107 | 3 | 35 |
| 108 | 3 | 35 |
| 109 | 3 | 50 |
| 110 | 2 | 50 |
| 111 | 3 | 75 |
| 112 | 2 | 75 |
| 113 | 2 | 100 |
| 114 | 2 | 100 |
| 201 | 3 | 150 |
| 202 | 3 | 150 |
| 203 | 3 | 200 |
| 204 | 2 | 200 |
| 205 | 3 | 250 |
| 206 | 2 | 250 |
| 207 | 3 | 300 |
| 208 | 2 | 300 |
| 209 | 3 | 350 |
| 210 | 2 | 350 |
| 211 | 2 | 400 |
| 212 | 2 | 400 |
| 213 | 2 | 500 |
| 214 | 2 | 550 |

Fonte – Avci e Topaloglu (2016)

5.1 RESULTADOS DO NNRA CONCORRENTE

A Tabela 2 apresenta os resultados encontrados utilizando a heurística NNRA Concorrente. Para cada instância utilizada, temos como colunas os valores encontrados utilizando o NNRA, executado no ambiente computacional descrito neste capítulo, os resultados do NNRA presentes no artigo de Nepomuceno, Saboia e Pinheiro (2019) e o do NNRA Concorrente, heurística desenvolvida neste trabalho.

A heurística NNRA Concorrente, como mostra a Tabela 2, foi capaz de produzir soluções melhores para as instâncias 106, 107, 207, 209 e 212 dentre todas as instâncias testadas, consequentemente, tivemos soluções melhores em 17,8% das instâncias testadas. E finalmente, nota-se que não há melhora significativa entre os algoritmos no quesito de qualidade de solução.

Pode-se afirmar que, no caso da heurística concorrente, o tamanho da instância, ou seja, a quantidade de clientes e a quantidade de tipos de veículo, não influencia diretamente na capacidade do algoritmo de gerar melhores soluções do que o NNRA gerado para este .

Tabela 2 – Menor solução - NNRA Concorrente

| Instância | NNRA | NNRA (Artigo) | NNRA Concorrente |
|-----------|----------|---------------|------------------|
| 101 | 620,23 | 620,23 | 620,23 |
| 102 | 588,53 | 588,53 | 588,53 |
| 103 | 445,13 | 445,13 | 445,13 |
| 104 | 448,03 | 442,51 | 448,03 |
| 105 | 500,82 | 500,82 | 500,82 |
| 106 | 577,14 | 572,21 | 565,47 |
| 107 | 1169,47 | 1155,92 | 1154,21 |
| 108 | 1672,33 | 1663,92 | 1672,93 |
| 109 | 983,95 | 980,31 | 991,18 |
| 110 | 1240,84 | 1230,00 | 1254,94 |
| 111 | 1638,73 | 1620,72 | 1659,32 |
| 112 | 1029,24 | 1011,99 | 1039,64 |
| 113 | 1388,91 | 1375,51 | 1375,89 |
| 114 | 1600,57 | 1595,26 | 1616,07 |
| 201 | 1638,11 | 1626,96 | 1639,09 |
| 202 | 2420,81 | 2425,65 | 2435,99 |
| 203 | 3694,35 | 3746,27 | 3703,26 |
| 204 | 2518,13 | 2453,82 | 2454,44 |
| 205 | 2726,65 | 2683,77 | 2747,79 |
| 206 | 2614,44 | 2528,85 | 2611,11 |
| 207 | 2934,73 | 2911,20 | 2902,47 |
| 208 | 2943,80 | 2867,69 | 2934,39 |
| 209 | 3966,59 | 3997,14 | 3933,20 |
| 210 | 2692,87 | 2622,71 | 2647,31 |
| 211 | 3818,67 | 3821,15 | 3878,49 |
| 212 | 3481,43 | 3441,52 | 3385,37 |
| 213 | 8711,20 | 8518,00 | 8623,76 |
| 214 | 10455,83 | 10417,10 | 10438,70 |

Fonte – Elaborado pelo autor

5.2 RESULTADOS DA PARADA REPENTINA

A Tabela 3 apresenta os resultados encontrados utilizando a heurística NNRA Parada Repentina, juntamente com, na primeira e segunda coluna, as soluções encontradas a partir do NNRA e do NNRA do artigo de Nepomuceno, Saboia e Pinheiro (2019).

Como demonstrado pela Tabela 3, a heurística NNRA Parada Repentina, obteve soluções piores no primeiro conjunto das instâncias de teste, com exceção das instâncias 108, 112 e 113, porém as soluções não se distanciaram significativamente dos resultados obtidos pelo NNRA. Em relação ao segundo conjunto, observa-se bom desempenho ao encontrar soluções melhores em todas as instâncias, ou seja, da instância 201 a 214.

Levando em consideração as soluções apresentadas, podemos destacar que, para as instâncias testadas, têm-se uma taxa de 61% de soluções melhores encontradas e, em contrapartida, uma taxa de 25% para soluções consideradas piores. Consequentemente, podemos afirmar que para instâncias de tamanho superior a 150 e inferior a 550, a heurística NNRA Parada Repentina demonstra melhor desempenho quando comparada ao NNRA para esses casos.

Ao compararmos a heurística de Parada Repentina com a Concorrente, podemos constatar que a Parada Repentina apresenta desempenho superior pois apresenta solução melhor em 22, solução igual em 4, sendo estas as instâncias 101, 102, 103 e 105, e, solução pior em somente 2 das 28, as quais são as 106 e 107. Consequentemente, verifica-se 79% de melhoria de desempenho.

A Figura 8 foi gerada a partir da média dos resultados encontrados utilizando como base uma única instância que continha 550 clientes, na qual, para cada valor dentro do intervalo de 1 a 50, foram realizados 20 testes.

Percebe-se a influência que o número de paradas tem em relação à qualidade dos resultados encontrados. Nos casos em que o Número de Parada encontra-se entre 1 e 6, podemos notar uma melhoria na Média de soluções ilustrada pela queda brusca da linha de tendência. Enquanto isso, para valores superiores a 6, a linha de tendência sinaliza o crescimento gradativo da média de soluções para esses valores.

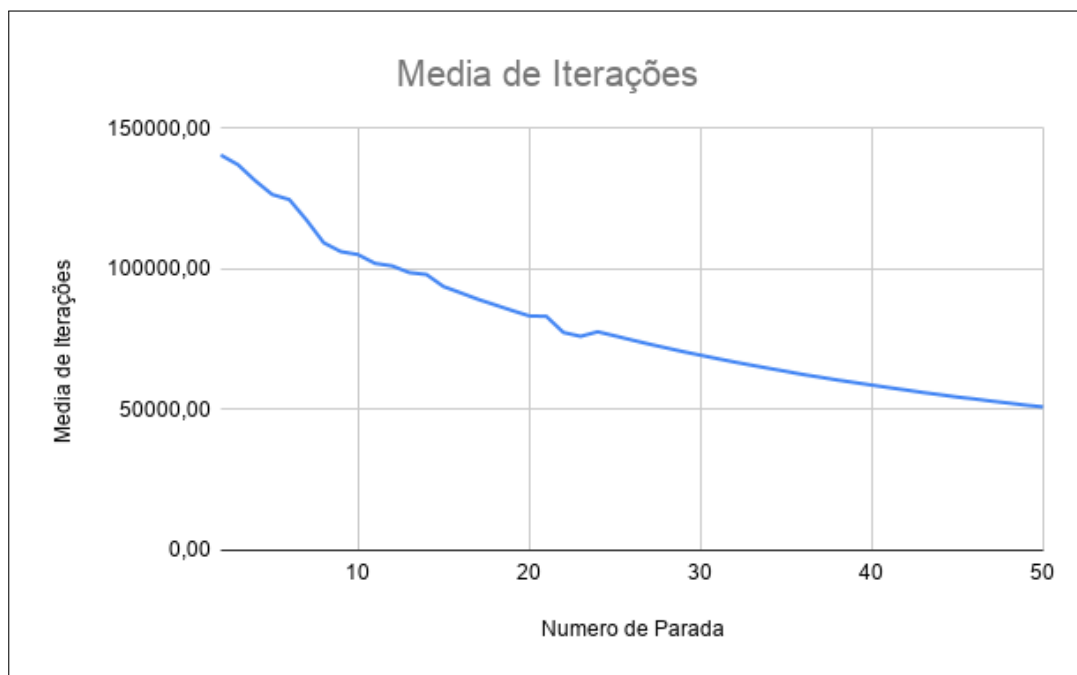
Enquanto isso, a Figura 9 relaciona o Número de Parada com a quantidade média de iterações. Nota-se que a relação entre o número de parada e a quantidade de iterações é inversamente proporcional devido à quanto maior o número da parada maior será a quantidade de falhas que deverão ocorrer para o critério de parada ser alcançado, consequentemente mais demoradas serão as iterações do algoritmo considerando a presença de um tempo limite.

5.3 RESULTADOS DO SEMI-GULOSO

A Tabela 4 expõe os resultados obtidos utilizando a heurística NNRA Semi-Guloso e, adicionalmente, também exibe os resultados do NNRA gerados no ambiente computacional e

Figura 8 – Media de solução

Fonte – Elaborado pelo autor

Figura 9 – Media de Iteração

Fonte – Elaborado pelo autor

o NNRA do artigo.

A partir da mesma tabela podemos averiguar que, ao compararmos a heurística semi-gulosa com o NNRA, apenas para as instâncias 104, 105 e 106 foi encontrado uma solução com custo total melhor, entretanto, para todas as demais instâncias testadas, a solução adquirida

Tabela 3 – Menor solução - NNRA Parada Repentina

| Instância | NNRA | NNRA (Artigo) | NNRA Parada Repentina |
|-----------|----------|---------------|-----------------------|
| 101 | 620,23 | 620,23 | 620,23 |
| 102 | 588,53 | 588,53 | 588,53 |
| 103 | 445,13 | 445,13 | 445,13 |
| 104 | 448,03 | 442,51 | 447,99 |
| 105 | 500,82 | 500,82 | 500,82 |
| 106 | 577,14 | 572,21 | 575,85 |
| 107 | 1169,47 | 1155,92 | 1165,32 |
| 108 | 1672,33 | 1663,92 | 1640,18 |
| 109 | 983,95 | 980,31 | 980,33 |
| 110 | 1240,84 | 1230,00 | 1244,52 |
| 111 | 1638,73 | 1620,72 | 1630,97 |
| 112 | 1029,24 | 1011,99 | 1007,29 |
| 113 | 1388,91 | 1375,51 | 1357,95 |
| 114 | 1600,57 | 1595,26 | 1597,42 |
| 201 | 1638,11 | 1626,96 | 1621,81 |
| 202 | 2420,81 | 2425,65 | 2335,15 |
| 203 | 3694,35 | 3746,27 | 3615,88 |
| 204 | 2518,13 | 2453,82 | 2442,49 |
| 205 | 2726,65 | 2683,77 | 2631,47 |
| 206 | 2614,44 | 2528,85 | 2465,33 |
| 207 | 2934,73 | 2911,20 | 2786,24 |
| 208 | 2943,80 | 2867,69 | 2836,25 |
| 209 | 3966,59 | 3997,14 | 3884,08 |
| 210 | 2692,87 | 2622,71 | 2599,17 |
| 211 | 3818,67 | 3821,15 | 3667,78 |
| 212 | 3481,43 | 3441,52 | 3352,41 |
| 213 | 8711,20 | 8518,00 | 8318,48 |
| 214 | 10455,83 | 10417,10 | 9992,35 |

Fonte – Elaborado pelo autor

foi consideravelmente pior, apresentado crescimento de até, no máximo, 9,08% em cima do valor adquirido usando o NNRA. Podemos inferir que os custos totais obtidos crescerão de maneira diretamente proporcional ao tamanho das instâncias, em particular, em relação a quantidade de clientes.

Ao compararmos o desempenho da Heurística Semi-Gulosa com a de Parada Repentina, pode-se salientar que o Semi-Guloso apresenta melhor desempenho em apenas 3 das 28 instâncias utilizadas como base de teste, sendo estas 104, 105 e 106, ou seja, 11% das instâncias, e, em contra partida, apresenta desempenho pior em 79% das demais instâncias, com poucas ocorrências de soluções iguais para ambas as heurísticas.

Em comparação a Heurística Concorrente, observamos o mesmo comportamento, o qual temos, novamente, somente as instâncias 104, 105 e 106 apresentando valores de solução melhor, representando apenas 11% do total de instâncias utilizadas para teste.

Tabela 4 – Menor solução - NNRA Semi-Guloso

| Instância | NNRA | NNRA (Artigo) | NNRA Semi-Guloso |
|-----------|----------|---------------|------------------|
| 101 | 620,23 | 620,23 | 620,23 |
| 102 | 588,53 | 588,53 | 588,53 |
| 103 | 445,13 | 445,13 | 445,13 |
| 104 | 448,03 | 442,51 | 442,39 |
| 105 | 500,82 | 500,82 | 495,00 |
| 106 | 577,14 | 572,21 | 553,04 |
| 107 | 1169,47 | 1155,92 | 1216,19 |
| 108 | 1672,33 | 1663,92 | 1729,75 |
| 109 | 983,95 | 980,31 | 1048,33 |
| 110 | 1240,84 | 1230,00 | 1306,54 |
| 111 | 1638,73 | 1620,72 | 1856,80 |
| 112 | 1029,24 | 1011,99 | 1118,70 |
| 113 | 1388,91 | 1375,51 | 1508,62 |
| 114 | 1600,57 | 1595,26 | 1743,68 |
| 201 | 1638,11 | 1626,96 | 1958,09 |
| 202 | 2420,81 | 2425,65 | 2758,51 |
| 203 | 3694,35 | 3746,27 | 4132,30 |
| 204 | 2518,13 | 2453,82 | 2817,25 |
| 205 | 2726,65 | 2683,77 | 3199,61 |
| 206 | 2614,44 | 2528,85 | 3062,88 |
| 207 | 2934,73 | 2911,20 | 3475,70 |
| 208 | 2943,80 | 2867,69 | 3383,16 |
| 209 | 3966,59 | 3997,14 | 4640,89 |
| 210 | 2692,87 | 2622,71 | 3309,02 |
| 211 | 3818,67 | 3821,15 | 4411,36 |
| 212 | 3481,43 | 3441,52 | 4002,58 |
| 213 | 8711,20 | 8518,00 | 9291,53 |
| 214 | 10455,83 | 10417,10 | 11115,24 |

Fonte – Elaborado pelo autor

6 CONCLUSÕES

6.1 CONTRIBUIÇÕES DO TRABALHO

Neste trabalho, foi abordado o HVRPSPD, o qual é um problema no qual veículos, com capacidades variadas, realizam a coleta e entrega de um produto ou serviço em uma única visita a seus clientes e tem como objetivo encontrar uma rota na qual todos os clientes sejam atendidos.

Foram desenvolvidas 3 heurísticas, sendo cada uma delas, respectivamente, baseadas nas estratégias de concorrência, de parada repentina e semi-gulosa. Para o processo de testes foram utilizadas 28 instâncias, as mesmas presentes no artigo de Avci e Topaloglu (2016). Após a realização dos testes observou-se que, ao compararmos cada uma das heurísticas com o NNRA, a heurística de Parada repentina obteve uma melhoria de, aproximadamente, 61% em encontrar soluções melhores e, também, heurística Semi-Gulosa apresentou o pior desempenho, sendo este de 79%. Complementarmente, ao compararmos apenas as 3 heurísticas, heurística de Parada repentina destaca-se com os melhores resultados.

6.2 LIMITAÇÕES E TRABALHOS FUTUROS

Um limite que deve ser apontado é que a fase de calibração das heurísticas demanda tempo de processamento, então dependendo da instância que for usada, o teste pode bater o tempo limite sem passar pela fase de repetição. As instâncias de Avci e Topaloglu (2016), utilizadas neste trabalho para efeitos comparativos, não representam situações reais pois foram criados artificialmente. Outro limite sobre as heurísticas, mesmo que o ambiente computacional, usando como data referente a publicação desse trabalho, seja acima da média, as heurísticas não aproveitam sua total capacidade de processamento utilizando apenas 1 núcleo de processamento.

Para trabalhos futuros, adaptar as abordagens criadas neste trabalho para outros VRPs, a implementação para uso de processamento paralelo, melhor abordagem de calibração de variáveis dos algoritmos deste trabalho e o uso de instâncias que simulem situações reais.

REFERÊNCIAS

- ANGELELLI, E.; BAZGAN, C.; SPERANZA, M. G.; TUZA, Z. Complexity and approximation for traveling salesman problems with profits. **Theoretical Computer Science**, v. 531, p. 54 – 65, 2014. ISSN 0304-3975. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0304397514001728>>.
- APPLEGATE, D.; BIXBY, R.; CHVÁTAL, V.; COOK, W. **The Traveling Salesman Problem: A computational study**. Princeton University Press, 2011. (Princeton Series in Applied Mathematics). ISBN 9781400841103. Disponível em: <<http://books.google.com.br/books?id=zfIm94nNqPoC>>.
- ARMON, A.; AVIDOR, A.; SCHWARTZ, O. Cooperative tsp. **Theoretical Computer Science**, v. 411, n. 31, p. 2847 – 2863, 2010. ISSN 0304-3975. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0304397510002112>>.
- AVCI, M.; TOPALOGLU, S. A hybrid metaheuristic algorithm for heterogeneous vehicle routing problem with simultaneous pickup and delivery. **Expert Systems with Applications**, v. 53, p. 160 – 171, 2016. ISSN 0957-4174. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0957417416000610>>.
- BRASIL. **Lei Nº 12.305, de 2 de ago. de 2010**: Política nacional de resíduos sólidos. Brasília, DF: ago. 2010. <http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2010/lei/l12305.htm>.
- BRASSARD, G.; BRATLEY, P. **Fundamentals of Algorithmics**. Nova Iorque: Prentice Hall, 1995.
- CNT - CONFEDERAÇÃO NACIONAL DE TRANSPORTES. **Plano CNT de Transporte e Logística**. Brasília, 2018. Disponível em: <<https://planotransporte.cnt.org.br/#download>>.
- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos: teoria e pratica**. Rio de Janeiro: Elsevier, 2012. ISBN 978-85-352-3699-6.
- FERNANDES, F. C. A. Extensões do problema de roteamento de veículo com coleta e entrega. 2019.
- GOLDBARG, E.; GOLDBARG, M.; LUNA, H. **Otimização combinatória e meta-heurísticas: algoritmos e aplicações**. Rio de Janeiro: Elsevier Brasil, 2016. ISBN 978-85-352-7812-5. Disponível em: <<https://www.evolution.com.br/epubreader/otimizao-combinatoria-e-metaheuristics-algoritmos-apliacaes-1ed>>.
- GOLDBARG, M.; ELIZABETH. **Grafos: Conceitos, algoritmos e aplicações**. [S.l.: s.n.], 2012. ISBN 978-85-352-5716-8.
- HART, J.; SHOGAN, A. W. Semi-greedy heuristics: An empirical study. **Operations Research Letters**, v. 6, n. 3, p. 107 – 114, 1987. ISSN 0167-6377. Disponível em: <<http://www.sciencedirect.com/science/article/pii/0167637787900216>>.
- IBGE – INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Nota técnica da Logística dos Transportes no Brasil 2014**. Rio de Janeiro, 2014. Disponível em: <<https://www.ibge.gov.br/geociencias/organizacao-do-territorio/redes-e-fluxos-geograficos/15793-logistica-dos-transportes.html?t=acesso-ao-produto>>.

MELO, A. C. d. S.; FERREIRA FILHO, V. J. M. Sistemas de roteirização e programação de veículos. **Pesquisa Operacional**, scielo, v. 21, p. 223 – 232, 07 2001. ISSN 0101-7438. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382001000200007&nrm=iso>.

NEPOMUCENO, N. V.; SABOIA, R. B.; PINHEIRO, P. R. A fast randomized algorithm for the heterogeneous vehicle routing problem with simultaneous pickup and delivery. **Algorithms**, Multidisciplinary Digital Publishing Institute, v. 12, n. 8, p. 158, Aug 2019. ISSN 1999-4893. Disponível em: <<https://doi.org/10.3390/a12080158>>.

RUSSELL, S. J.; NORVIG, P. **Inteligência artificial**. Rio de Janeiro: Elsevier, 2004.

SABOIA, R. B. Aplicação da metodologia gerar-e-resolver para o problema de roteamento de veículos com coleta e entrega simultâneas e frota heterogênea. 2019.

STABELINI, D. **Logística reversa**: o que é, como funciona e como aplicar. 2018. <<https://blog.texaco.com.br/ursa/logistica-reversa-o-que-e-como-funciona/>>.