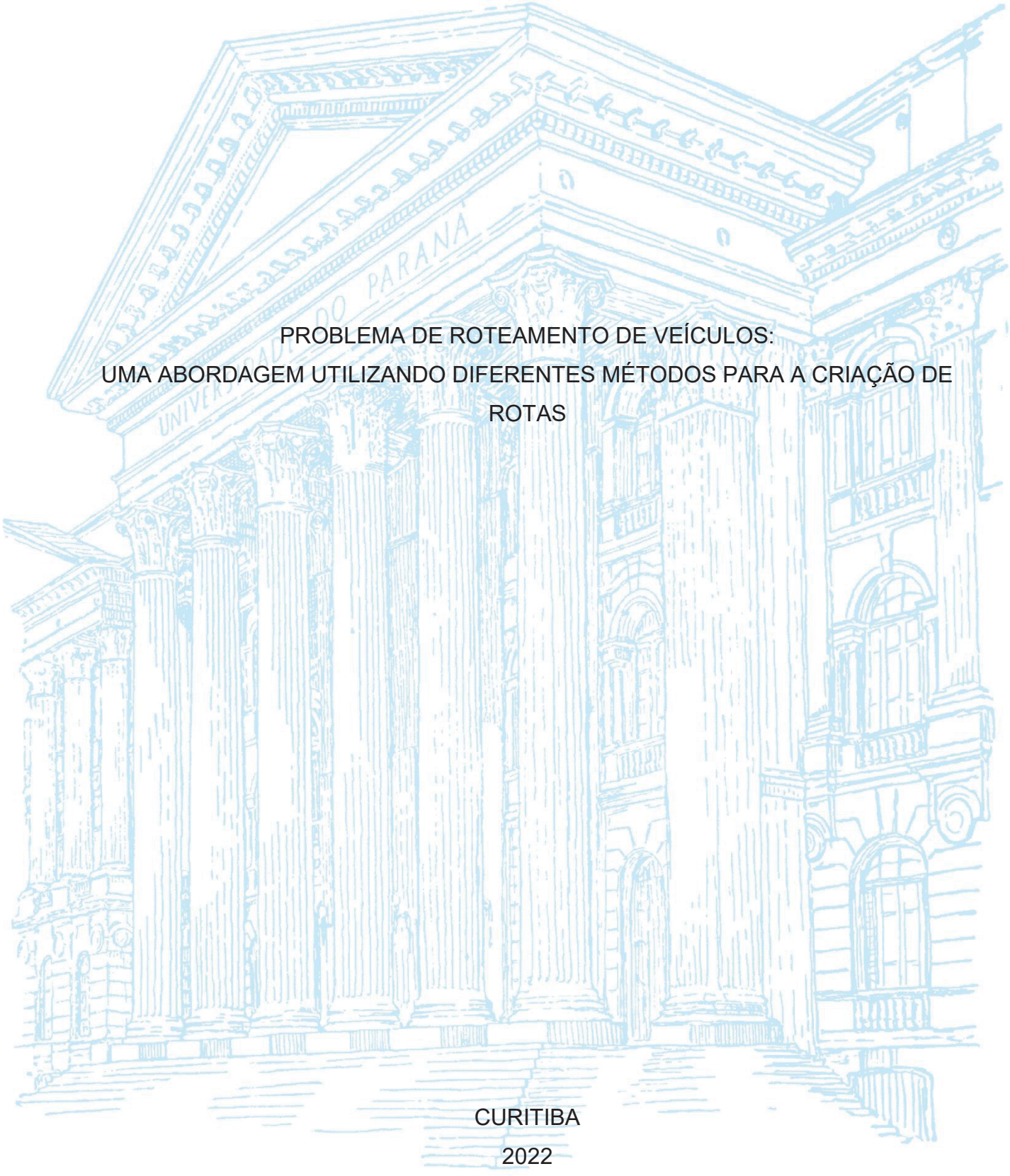
UNIVERSIDADE FEDERAL DO PARANÁ

MATHEUS MORSELLI GYSI



PROBLEMA DE ROTEAMENTO DE VEÍCULOS: UMA ABORDAGEM UTILIZANDO DIFERENTES MÉTODOS PARA A CRIAÇÃO DE ROTAS

CURITIBA

2022

MATHEUS MORSELLI GYSI

PROBLEMA DE ROTEAMENTO DE VEÍCULOS: UMA ABORDAGEM UTILIZANDO DIFERENTES MÉTODOS PARA A CRIAÇÃO DE ROTAS

Trabalho de Conclusão de Curso apresentado ao curso de Especialização em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada.

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño

CURITIBA

2022

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INTELIGÊNCIA ARTIFICIAL APLICADA da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **MATHEUS MORSELLI GYSI** intitulada: **PROBLEMA DE ROTEAMENTO DE VEICULOS: UMA ABORDAGEM UTILIZANDO DIFERENTES METODOS PARA A CRIACAO DE ROTAS**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 25 de Novembro de 2022.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

DIEVAL GUIZELINI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Problema de roteamento de veículos: Uma abordagem utilizando diferentes métodos para a criação de rotas

Matheus Morselli Gysi
Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná
Curitiba, Brasil
esc.matheus@gmail.com

Razer Anthom Nizer Rojas Montaño
Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná
Curitiba, Brasil
razer@ufpr.br

***Resumo*—Os problemas de otimização são comumente vistos e suas soluções tem grande importância na indústria, bem como no meio acadêmico. São exploradas cinco soluções para um problema que consiste em gerar, ou melhorar, rotas para n-vendedores (ou n-dias) os quais devem percorrer o maior número de casas no menor tempo possível. As soluções escolhidas para o problema foram ordenação, aleatória, solução gerada pelo método do ponto proximal (MPP), *Adaptive Large Neighborhood Search* (ALNS) e algoritmo genético. Foram usados 1383 pontos do bairro Porto Novo da cidade de Cariacica no Espírito Santo - Brasil. Após a realização de agrupamentos e limpezas de dados foram roteados 352 pontos com um total de 1200 casas atendidas. Para esse atendimento os algoritmos trouxeram rotas com necessidade entre 5 e 8 vendedores e tempo de percurso entre 24 horas e 30 minutos. Três dos melhores resultados obtidos foram com o uso do algoritmo genético, porém com um tempo de processamento entre 47 e 969 vezes maior que o método do ponto proximal - isso, sem levar em consideração o gasto de memória - que ficou em quarto lugar juntamente com o algoritmo ALNS.**

***Palavras-chave*—problema de roteamento, VRP, algoritmo genético, ALNS, método do ponto proximal.**

***Abstract*—Optimization problems are commonly seen and their solutions are of huge importance in industry as well as in academia. Five solutions was explored for a problem that consists of generating, or improving, routes for n-vendors (or n-days) who must travel the largest number of houses in the shortest possible time. The solutions chosen for the problem were sorting, random, solution generated by proximal point method (PPM), Adaptive Large Neighborhood Search (ALNS) and genetic algorithm. 1383 points were used in the Porto Novo neighborhood of the city of Cariacica in Espírito Santo - Brazil. After grouping and cleaning data, 352 points were routed with a total of 1200 homes served. For this service, the algorithms brought up routes with the need for between 5 and 8 salesman and travel time between 24 hours and 30 minutes. Three of the best results obtained were with the use of the genetic algorithm, but with a processing time between 47 and 969 times greater than the greedy algorithm - without taking into account the memory expenditure - which was in fourth place together with the ALNS algorithm.**

***Index Terms*—routing problem, VRP, genetic algorithm, ALNS, proximal point method.**

I. DESENVOLVIMENTO

O problema de geração de rotas vem sendo estudado há muitos anos, e de acordo com G. Laporte e I. Osman [1], as quatro principais categorias de solução do problema são: o caixeiro-viajante (TSP, do inglês *traveling salesman problem*); o problema de roteamento de veículos (VRP, do inglês, *Vehicle Routing Problem*); o carteiro chinês e o carteiro rural. Devido à natureza do problema, neste artigo, será apresentada a solução para o problema de roteamento de veículos.

De acordo com B. Eksioglu, A. V. Vural, e A. Reisman [2], o problema de roteamento de veículos é algo comum e pode ser considerado uma generalização do problema do caixeiro-viajante, uma vez que o TSP pode ser considerado como uma das rotas geradas pelo VRP.

A principal diferença entre eles, de acordo com W. Herdianti, A. Gunawan e S. Komsiyah [3], é que o TSP, gera apenas uma rota, considerando, portanto, um vendedor visitando todos os pontos e retornando ao ponto inicial. Enquanto o VRP, cria diversas rotas saindo do ponto de origem, visitando alguns pontos e voltando ao de partida. A figura 1 representa melhor essa diferença.

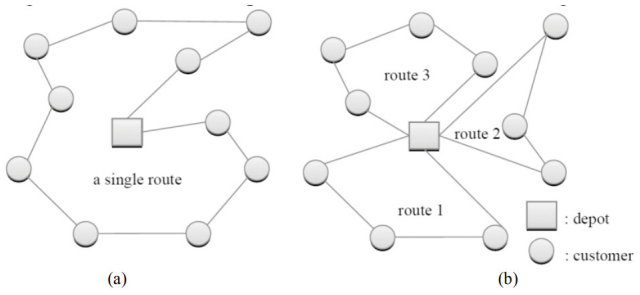


Figura 1 – Mostra a diferença entre o TSP(a), o qual gera apenas uma rota visitando todos os pontos e o VRP(b), que gera diversas rotas, como se fossem diversos caixeiros-viajantes.

Fonte: W. Herdianti, A. Gunawan, e S. Komsiyah,2021 [3]

O problema de roteamento de veículos, foi proposto pela

primeira vez em 1954 por G. Dantzig, R. Fulkerson e S. Johnson [4] e tem como intuito criar as rotas para n-veículos, reduzindo a distância e/ou tempo de percurso para visitar todos os pontos, sem a necessidade de testar todas as soluções possíveis.

Embora o VRP, traga em seu nome o termo veículos, ele pode ser aplicado para gerar rotas indiferentemente do meio de transporte a ser usado. Neste artigo serão considerados vendedores de porta em porta que se direcionam a pé aos locais previamente estabelecidos, com o objetivo de oferecer serviços de telecomunicações.

São apresentadas cinco soluções para o problema usando métodos diferentes. São eles:

- Ordenado (I-B1): Organização dos dados formando uma rota da forma como os dados foram recebidos.
- Aleatório (I-B2): Uma permutação aleatória dos dados, não utilizando nenhuma heurística, ou meta-heurística para o roteamento, usado apenas para gerar as soluções iniciais;
- método do ponto proximal (MPP) (I-B3): é uma heurística popular na resolução de problemas conexos e não convexos, neste o algoritmo sempre busca o ponto mais próximo do atual, até completar todos os pontos [5];
- *Adaptive Large Neighborhood Search* (ALNS (I-B4)): De acordo com [6], [7] e [8], o ALNS é uma meta-heurística usada para solucionar problemas combinatórios complexos, baseado em heurísticas de destruição e reparo de solução, usando um critério de aceitação;
- Algoritmo genético (I-B5): é uma meta-heurística baseada em processos naturais, como cruzamento e mutações, pertencente aos algoritmos evolucionários usada principalmente em problemas de otimização [9] e [10].

O principal objetivo é determinar as melhores rotas para que esses vendedores possam realizar caminhando, minimizando o tempo de percurso entre os pontos a serem visitados. Considerando um tempo fixo de paradas em cada endereço, comparando o tempo de deslocamento nas rotas e o tempo de execução dos algoritmos.

Algumas considerações são importantes sobre o problema:

- Todos os vendedores possuem um ponto de partida comum, e será atribuído pelo usuário, ou pelo próprio algoritmo (utilizando o ponto médio);
- O tempo médio necessário para cada visita é de noventa segundos (tempo atribuído pela média dos atendimentos);
- Os vendedores trabalham 7 horas e trinta minutos todos os dias úteis (segunda a sexta);
- O problema visa minimizar o tempo em rota, para que dessa forma o maior número de clientes possam ser atendidos em um dia.

A. *Descrição dos dados*

Os dados usados para a geração das rotas teste do problema, foram obtidos de uma empresa do ramo de telecomunicações. Para preservar a integridade e segurança dos dados, foram utilizados apenas os endereços (estado, município, bairro, nome da rua, número de fachada e complemento) e pontos de latitude e longitude sem a identificação dos clientes.

Tabela I – Exemplo dos dados necessários que foram recebidos aos quais deverão ser roteados. Devido a LGPD, algumas colunas foram removidas para preservar o anonimato dos dados, pelo mesmo motivo a quantidade de algarismos significativos foram reduzidos nas colunas LATITUDE e LONGITUDE.

UF	MUNICÍPIO	LATITUDE	LONGITUDE
ES	CARIACICA	-20.293	-40.371
ES	CARIACICA	-20.293	-40.371
ES	CARIACICA	-20.311	-40.389
ES	CARIACICA	-20.311	-40.389
ES	CARIACICA	-20.310	-40.390

Fonte: De autoria própria.

A tabela I mostra apenas as cinco primeiras linhas do conjunto de dados, sem as informações de logradouro, número de fachada, complemento e outras informações a respeito dos clientes. Reduzindo também o número de casas decimais das latitudes e longitudes.

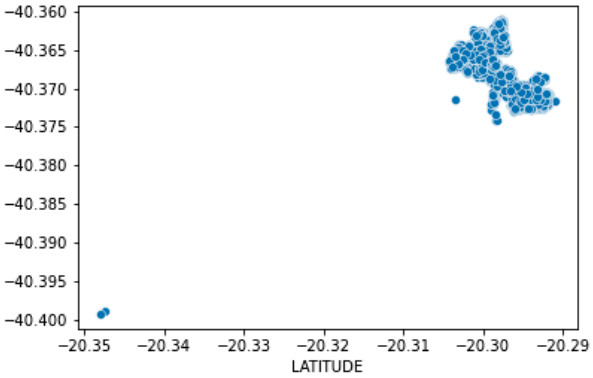


Figura 2 – Gráfico de dispersão dos pontos de latitude e longitude sem tratamentos. É possível verificar que existem pontos distantes do bairro, e esses deverão ser removidos com o objetivo de melhorar as rotas.

Fonte: Dados fornecidos para a determinação das rotas pela área de vendas de uma empresa de telecomunicações.

O problema em questão, é construir as rotas aos vendedores de porta a porta. Foi escolhido para esse estudo o bairro Porto Novo da cidade de Cariacica no Espírito Santo. Inicialmente a base possuía 1383 pontos a serem roteirizados, e a distribuição desses pontos pode ser vista na Figura 2.

Ao receber os dados, foi observado que os pontos de latitude e longitude estavam sendo interpretados como *string*, e então foram transformados em *float*[1]. Em seguida, foi necessário remover os pontos com problemas (pontos em meio ao oceano, em meio a lagoas, ou até mesmo em outros bairros ou

[1]Uso do numpy float64, em vista de não perder a precisão dos dados.

cidades). A correção foi realizada por meio de um algoritmo de clusterização[2] baseado em densidade (DBSCAN)[11], fazendo com que os pontos muitos distantes fossem removidos. O gráfico de dispersão resultante pode ser visto na Figura 3

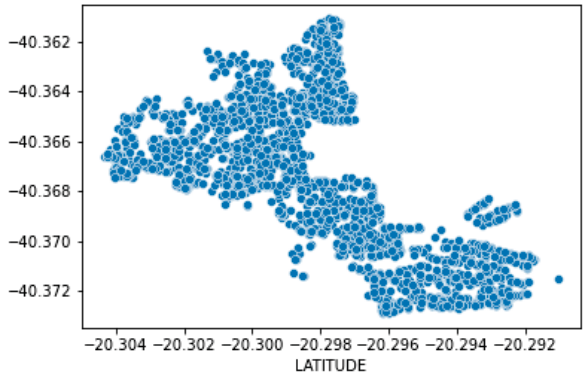


Figura 3 – Gráfico de dispersão dos pontos de latitude e longitude sem os *Outliers*.

Fonte: De autoria própria.

Após esse tratamento inicial, foram agrupados todos os pontos que compartilhavam o mesmo endereço e número, porém com complementos diferentes, pois os vendedores informaram que não faziam a diferenciação entre os complementos na hora das visitas.

Por motivo de reduzir o processamento os dados foram novamente agrupados por proximidade, mantendo apenas um ponto por quadra, para que dessa forma sejam reduzidas as quantidades de pontos a serem roteirizados. É importante ressaltar que para cada ponto agrupado, os tempos de visitas são somados, ou seja, se em um ponto agrupado existirem cinco pontos distintos, o tempo de atendimento é somado para cada um dos cinco clientes.

Após realizados os tratamentos e pré-processamentos, a base ficou com 1.200 pontos totais, e 817 endereços distintos (logradouro, número de fachada, porém complemento diferente, conforme pode ser visto na Figura 4. Após o último agrupamento obtiveram-se 352 pontos os quais foram roteirizados com o uso de cinco métodos que são descritos a seguir.

B. Métodos

Foram desenvolvidos cinco formas para a realização do roteamento dos vendedores. Sendo as duas primeiras apenas a ordenação e o embaralhamento dos dados, e as outras três foram os algoritmos pelo método do ponto proximal, ALNS e genético.

Para determinar as métricas de tempo de percurso e distância percorrida foi usado uma instância da API do *Open*

[2]Foi usado o algoritmo do *scikit-learn*. a sua documentação pode ser encontrada em https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

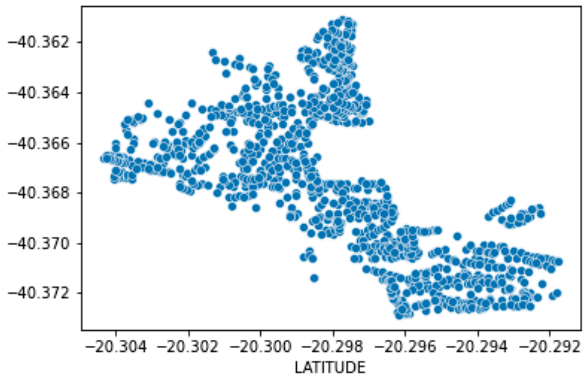


Figura 4 – Gráfico de dispersão dos pontos de latitude e longitude do bairro Porto Novo da cidade de Cariacica, com os Logradouros e números de fachada agrupados.

Fonte: De autoria própria.

Source Routing Machine (OSRM)[3] que gera as matrizes de distância e tempo, considerando o percurso entre os pontos.

Por se tratar de um problema em que o vendedor está a pé, essas matrizes foram corrigidas para que os elementos espelhados tivessem a mesma distância e tempo (caminho do ponto A ao ponto B deve ser igual ao caminho do ponto B ao ponto A).

Todos os métodos dividem os pontos em sub-rotas de acordo com a janela de tempo possível para cada um dos vendedores, após gerar uma rota.

Esses métodos são:

1) Ordenado: Para a geração da rota ordenada foi usada a ordenação padrão dos dados recebidos, a qual era em ordem alfabética dos logradouros e em ordem crescente do número de fachada.

2) Aleatório: Para a criação de rotas aleatórias e permutação inicial dos pontos, foi usada a função *shuffle* do pacote *random*[4] do python com a semente 42.

Embora não traga um resultado satisfatório, foi usado como rota inicial para os algoritmos descritos a seguir.

3) Método do ponto proximal: O desenvolvimento do método do ponto proximal utilizou as métricas obtidas na matriz de tempo, por estarmos em busca de uma otimização no tempo, embora pudesse utilizar a matriz de distância, caso o objetivo fosse minimizar a distância percorrida. O método parte de um local inicial, fornecido pelo problema, caso não seja, é usado o ponto central (ou qualquer outro), sempre em busca de um ponto mais próximo, excluindo os que já foram visitados.

4) ALNS: Trata-se de uma meta-heurística, formada pela destruição e reparo das soluções baseadas em heurísticas simples. Foram utilizadas duas heurísticas de destruição:

[3]Documentação da API do OSRM disponível em: http://project-osrm.org/docs/v5.24.0/api/#

[4]Documentação da biblioteca *random* disponível em: https://docs.python.org/3/library/random.html

- Aleatória: Destrói aleatoriamente a solução dada;
- Gulosa: Destrói os pontos de maior custo.

Além dessas, foram usadas três heurísticas de reparo:

- Aleatória: Repara a solução colocando os novos pontos aleatoriamente na solução dada;
- Invertida: Repara a solução invertendo os pontos destruídos;
- Gulosa: Repara a solução colocando os pontos no local de menor custo.

Como o algoritmo foi criado para simular a têmpera de metais [8] em diversos momentos são utilizadas temperaturas para nomear os parâmetros.

Deve-se lembrar de que o algoritmo do ALNS utiliza sempre três soluções para o problema, a primeira solução será tratada com a sigla BS (do inglês *best solution*) que é a melhor solução já encontrada, a segunda como de CR (do inglês *current solution*) que é a solução corrente (não necessariamente é a BS) e a terceira como AC (do inglês *actual solution*) que é a solução corrente (CR) após passar pelos mecanismos de destruição e construção (ou reparo).

Foram realizadas algumas alterações no ALNS, em relação aos descritos em A. Santini, S. Ropke e L. M. Hvattum [7] em Windras et. al [12] e em S. Ropke e D. Pisinger [13]. A primeira modificação foi em relação ao critério de aceitação da tempera simulada, o qual foram usados quatro pesos, em vez de três, sendo eles:

- Recompensa 1: AC é melhor que BS;
- Recompensa 2: AC é melhor que a CR, mas pior que BS;
- Recompensa 3: AC não é melhor que BS nem CR, mas é escolhida pelo mecanismo de escolhas;
- Recompensa 4: análogo ao caso 3 porém não é escolhida, usada com peso baixo apenas para evitar que o problema deixe de escolher uma heurística no método da roleta.

A segunda modificação importante é que foram guardadas todas as soluções já atribuídas. O processo compara se a AC já havia sido testada, quando já testada, o próximo passo é destruir e construir até encontrar uma solução nova não tratada. Embora onere o tempo de processamento, principalmente nas iterações finais, evita que trave em mínimos locais, ou em uma mesma solução.

Para obter a solução através do ALNS, foram atribuídos alguns parâmetros iniciais: número de segmentos; número de iterações por segmentos; percentual mínimo e máximo de pontos a serem substituídos em cada heurística de remoção aleatória; número de pontos mínimos e máximos a serem removidos na destruição gulosa; os pesos dados para as heurísticas e recompensas e solução inicial.

A execução do algoritmo se dá, inicialmente, com a criação de uma solução, podendo ser aleatória, ordenada ou com o uso de uma solução gerada pelo método do ponto proximal.

A partir de então são calculadas as temperaturas inicial e a de congelamento e a taxa de resfriamento, as quais foram atribuídas de acordo com o descrito em S. Ropke e D. Pisinger [13] e em G. Mattos Ribeiro e G. Laporte [8]. Sendo as duas primeiras respectivamente, a terça parte e a quingentésima parte do valor, a ser minimizado, encontrado na primeira solução. Já para a taxa de resfriamento é usada a equação 1, onde n_{iter} representa o número de iterações, e n_{seq} o número de segmentos.

$$\alpha = \sqrt[n_{iter} * n_{seq} - 1]{\frac{3}{500}} \tag{1}$$

Para cada segmento são realizadas n iterações, de acordo com os parâmetros iniciais, para cada uma o algoritmo utiliza uma roleta para determinar qual será a heurística de destruição e de reparo, a solução corrente é destruída e, em seguida, construída. Por fim, o algoritmo analisa se a solução AC já existe (caso já exista é gerada uma nova), passa pelo critério de aceitação comparando a solução AC com BS e CR, pontua a heurística de acordo com os pesos, por fim resfria o modelo. Ao final de cada segmento a roleta é atualizada até que atinja o ponto de parada, temperatura de congelamento, ou todos os segmentos sejam iterados.

5) Algoritmo genético: Conforme descrito em W. Lee e H.-Y. Kim [14] e em C. Sheppard [15], foi desenvolvido o algoritmo genético, como uma nova opção para a otimização das rotas.

Esse método é uma meta-heurística baseada na teoria da evolução das espécies de Charles Darwin [15]. Neste os indivíduos de uma população são submetidos a se reproduzirem e eventualmente a sofrerem mutações. Esses indivíduos são formados de cromossomos e em cada cromossomo há os genes, que, nesse caso, representam cada um dos locais que o vendedor irá visitar.

Para a implementação desse algoritmo foi utilizada a abordagem em que cada cada cromossomo do individuo era representado por um numero o qual representava um ponto que deveria ser visitado, ao invés de utilizar a abordagem binária.

Nesse algoritmo podem ser escolhidos como parâmetros iniciais: o ponto de partida; o número de gerações; o algoritmo para a criação da população zero; a quantidade de seus indivíduos na população; e o percentual de indivíduos que irão sofrer mutação por geração.

Esse método evolutivo, começa com a criação da população inicial, que pode ser gerada de diversas formas. Aqui foram geradas quatro rodadas com os testes:

- Todos os indivíduos gerados de forma aleatória;
- Parte dos indivíduos de forma aleatória e a outra através do método do ponto proximal (duas vezes, com parâmetros de quantidade de gerações e população diferentes);
- Soluções obtidas no método do ponto proximal [5], iniciando em cada ponto possível, e uma solução aleatória para os demais indivíduos (caso o número de indivíduos fosse maior que o de pontos).

Foi atribuída uma função que escolhe os n melhores indivíduos, que serão mantidos para a próxima geração, e os n

[5]Porém, foi alterada, para escolher entre qualquer um dos cinco pontos mais próximos, isso fez com que o modelo reduzisse as chances de travar em mínimos locais

piores indivíduos que serão eliminados antes dos cruzamentos e mutações.

Todos os indivíduos não eliminados são cruzados, baseados em uma distribuição meia normal, escolhendo-se aleatoriamente o ponto de corte entre os cromossomos dos indivíduos para o cruzamento.

Após realizados os cruzamentos, é escolhida uma amostra da base para ser realizada a mutação, essas podem ocorrem de três formas:

- Por inversão: Tomam-se dois pontos aleatórios e inverte a sequência. Por exemplo:
  4 5 **6** 1 2 3 **0** 4
  4 5 **0** 3 2 1 **6** 4
- Por permutação: Altera-se a posição de dois pontos. Por exemplo:
  4 5 **6** 1 2 3 **0** 4
  4 5 **0** 1 2 3 **6** 4
- Por embaralhamento: Tomam-se dois pontos e em seguida embaralha. Por exemplo:
  4 5 **6** 1 2 3 **0** 4
  4 5 **2** 0 3 6 **1** 4

Finalizadas as mutações, são geradas as rotas e sub-rotas e calculadas os tempos e distâncias para todos os indivíduos e o processo é repetido até a n-ésima geração.

C. Tecnologias

Para o desenvolvimento dessas soluções foi usado um notebook pessoal com processador i7(10ª geração), 32GB de memória RAM, como linguagem de programação o python (versão 3.10.4). As bibliotecas usadas foram:

- matplotlib versão 3.4.3;
- numpy versão 1.22.4;
- pandas versão 1.4.2;
- random (biblioteca padrão do python);
- requests versão 2.26.0;
- scipy versão 1.8.1;
- seaborn versão 0.11.2;
- sklearn versão 1.1.1.

Para o uso da API do OSRM foi criada uma maquina virtual com o uso do Docker.

II. RESULTADOS E DISCUSSÕES

Para esse problema o tempo mínimo para atender todos os 1200 clientes, considerando o tempo médio de 90 segundos por parada[6], totalizando 108.000 segundos em atendimentos. Os dados referentes ao problema, podem ser vistas na Tabela II.

A. Medida de qualidade dos modelos

Com os dados tratados e com todo o pré-processamento necessário, foi dado para cada ponto um número entre 0 e 351, com isso, foi gerada uma primeira solução de forma ordenada (ordenação em que os dados foram recebidos). A partir disso foi obtido um tempo de deslocamento de 77.178 segundos com a necessidade de sete vendedores para a realização da rota. Essa rota pode ser observada na Figura 5.

[6]Dados de tempo médio foram fornecidos pela área responsável.

Tabela II – Quadro resumo: Apresentando os principais dados do problema.

Número de pontos	352	
Número de clientes	1.200	
Tempo total de atendimento	108.000	segundos

Fonte: De autoria própria.

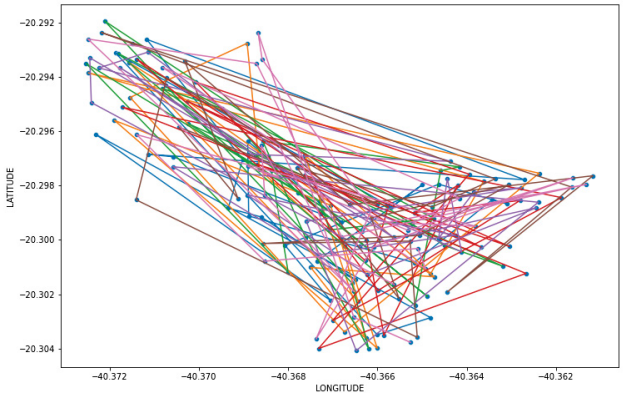


Figura 5 – Rota gerada utilizando a ordem em que os dados foram recebidos.

Fonte: De autoria própria.

Com isso, foi gerada uma rota aleatória, utilizando a semente quarenta dois, para a randomização dos pontos. Para esse resultado foi obtida uma rota com processamento rápido, mas com o pior resultado. O tempo em rota do vendedor foi de 86.385 segundos, e seriam necessários oito pessoas para realizar a rota. O mapa da rota na Figura 6, mostra a rota aleatória obtida.

Para o método do ponto proximal foram geradas 352 rotas, uma para cada ponto inicial. Para essas rotas houveram respostas variando entre 5.549 e 14.200 segundos de tempo de percurso, com uma média de 9.989 segundos, resultando assim em um tempo andando de 5.549 segundos, o que mostra uma melhora significativa para o problema. A rota gerada pode ser vista na Figura 7.

Para o uso do ALNS foi necessário inicialmente uma calibração dos parâmetros de quantidade miníma e máxima de pontos a serem removidos e reparados, assim como foi necessário determinar os melhores pesos para cada uma das heurísticas. Os valores podem ser vistos na Tabela III.

Os valores das temperaturas inicial e de congelamento, bem como a taxa de resfriamento, são calculadas automaticamente pelo modelo.

Foram criadas quatro parametrizações para o algoritmo do ALNS todas utilizando 100 segmentos, mudando número de

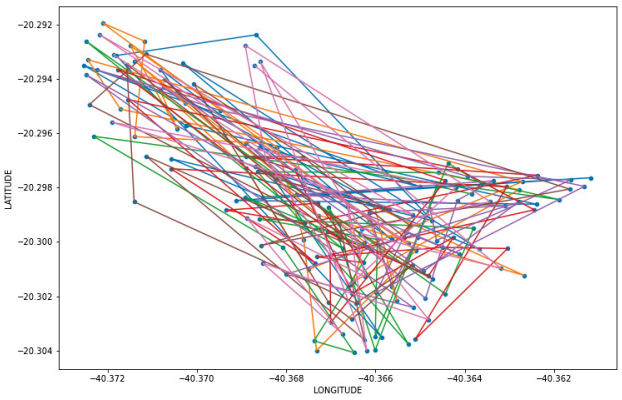


Figura 6 – Rota gerada pelo embaralhamento dos dados, usando uma semente conhecida.

Fonte: De autoria própria.

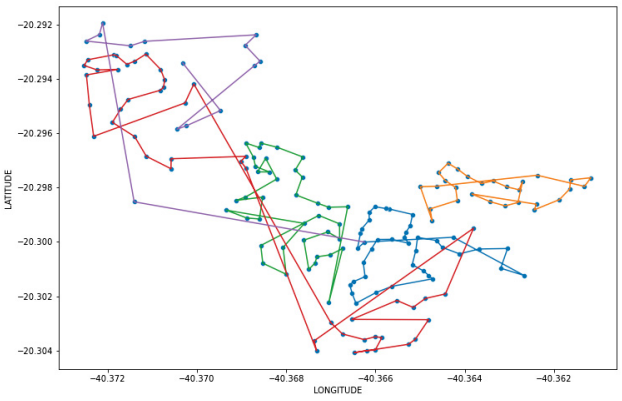


Figura 7 – Melhor rota gerada pelo método do ponto proximal. Cada cor representa um dia ou vendedor para a rota.

Fonte: De autoria própria.

Tabela III – Parâmetros iniciais usados para as heurísticas de construção e destruição do ALNS

Heurística randômica	min	5,00%
	max	50,00%
Heurística gulosa	min	15
	max	100
taxa de atualização de pesos		75,00%

Fonte: De autoria própria.

iterações e o tipo de solução inicial. Os resultados podem ser observados na Tabela IV e suas rotas vistas na Figura 8. Embora essa meta-heurística tenha sido capaz de encontrar boas soluções, não foi capaz de encontrar uma melhor do que a obtida por meio do método do ponto proximal dada inicialmente.

Tabela IV – Resultados obtidos a partir do ALNS

Iterações	Rota inicial	Tempo total	Tempo em rota	Tempo de CPU (s)
50	Aleatória 8a	122.813	14.813	992
50	Método do ponto proximal 8c	113.549	5.549	1.059
100	Aleatória 8b	116.619	8.619	2.171
100	Método do ponto proximal 8c	113.549	5.549	2.532

Fonte: De autoria própria.

Por fim foram realizadas quatro parametrizações do algoritmo genético, alterando-se o numero de gerações, quantidade de indivíduos para a população inicial, e a forma como a população inicial era criada, podendo ser aleatória ou pelo método do ponto proximal tradicional e modificado. Os resultados dessas simulações podem ser analisados na tabela V e na Figura 9.

Na tabela V, tem-se o número de gerações, que representa o numero de vezes que algoritmo ira passar pelos processos de cruzamento e mutação, indivíduos, representa a quantidade de rotas geradas/alteradas por geração, rota inicial, é a forma como a população da primeira geração é criada, tempo total e representa o tempo necessário para a melhor rota ser realizada, já o tempo em rota mostra o tempo em deslocamento, por fim tem-se o tempo usado pelo algoritmo para encontrar essa solução.

Para evitar a mutação de muitos indivíduos, foi atribuído que apenas 1% deles poderiam sofrer esse tipo de alteração. Além disso o algoritmo deveria escolher entre as três formas de mutação criadas: permutação, inversão e embaralhamento, e essas possuíam diferentes probabilidades de serem escolhidas, pois algumas acabam alterando muito a solução.

Nesse foi possível encontrar uma solução melhor que a solução obtida atraves do método do ponto proximal com um tempo de deslocamento de 1.825 segundos, essa pode ser observada em 10.

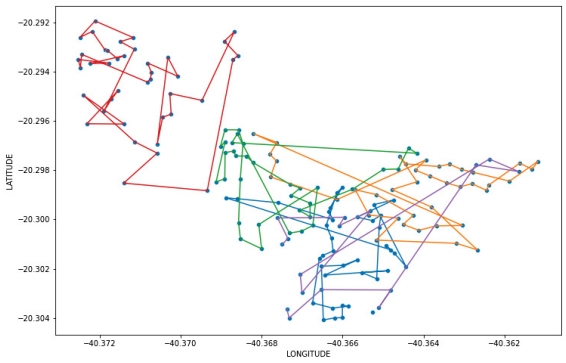
Tabela V – Resultados das rotas geradas pelo algoritmo evolutivo genético

Número de gerações	Indivíduos	Rota inicial	Tempo total (s)	Tempo em rota(s)	Tempo de CPU(s)
10.000	250	Aleatória	123.573	15.573	5.698
1.000	500	100 MPP e 400 aleatórias	111.981	3.981	862
10.000	750	250 MPP e 500 aleatórias	109.825	1.825	14.521
10.000	750	750 MPP* I-B5	110.168	2.168	14.108

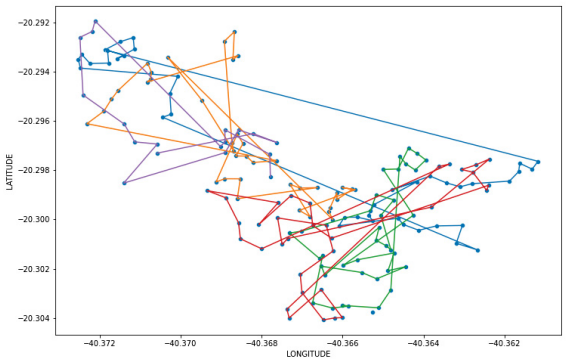
Fonte: De autoria própria.

B. Discussões

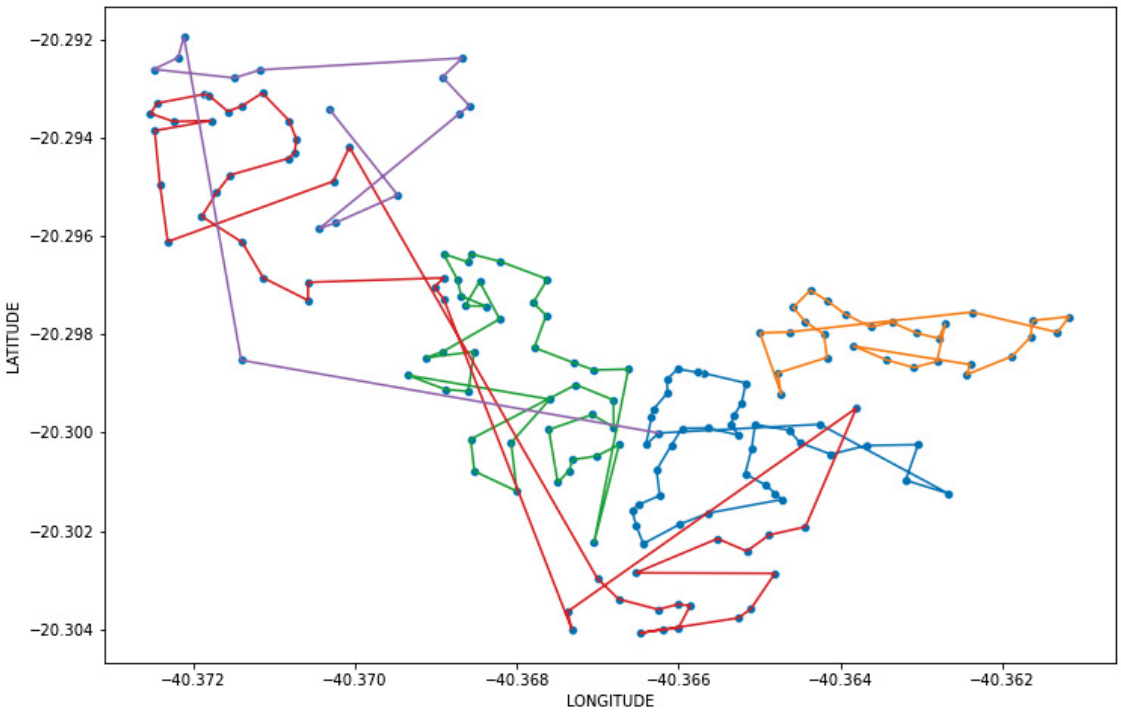
Os resultados podem ser comparados através da leitura da tabela VI. Nesta são apresentados os algoritmos na primeira coluna, seus parâmetros iniciais (quando houverem), os tempos de deslocamento das rotas, o número de vendedores



(a) ALNS 1 - Rota inicial aleatória e 50 iterações por segmento



(b) ALNS 2 - Rota inicial aleatória e 100 iterações por segmento



(c) ALNS 3 - Rota inicial pelo método do ponto proximal independente do número de segmentos não houve melhoras da rota gerada inicialmente.

Figura 8 – Rotas geradas pela meta-heurística ALNS

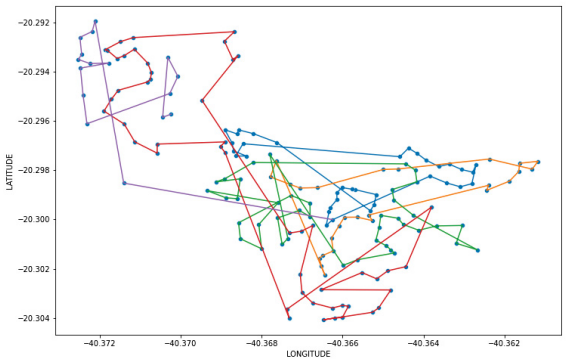
Fonte: De autoria própria.

ou dias necessários para que as rotas sejam completadas, os tempos de processamento (CPU), e, quando necessário, algumas observações.

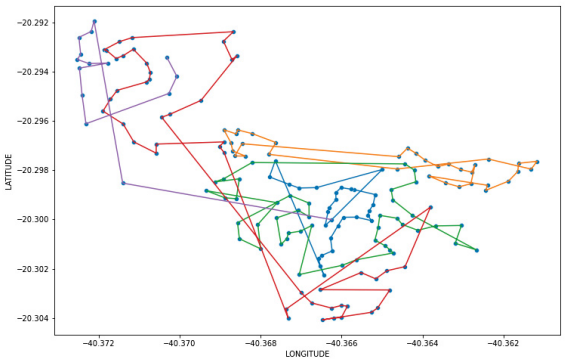
Ainda nessa tabela, o objetivo é encontrar o algoritmo que forneça o menor tempo em rota, e preferencialmente que não utilize muitos recursos computacionais. Ou seja, busca-se a minimização das duas colunas, tempo em rota e tempo de CPU.

Grande parte dos algoritmos testados trouxeram melhorias em redução de tempo de rota, porém sem redução de número de vendedores ou dias para que atendessem todas as casas. Sendo necessário em todos os casos no mínimo cinco vendedores para visitar todas as 1200 casas.

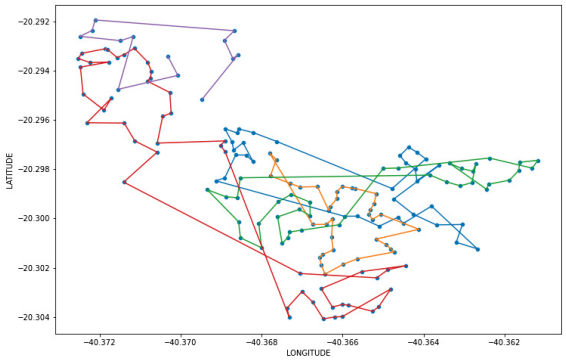
Possivelmente, caso fosse feito o roteamento de todos os



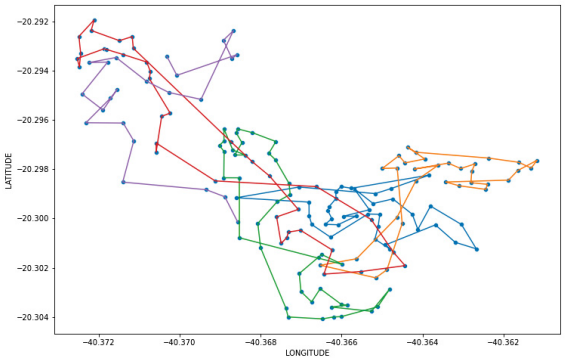
(a) AG 1 - número de gerações: 10.000, número de indivíduos: 250, Rota inicial: Aleatória



(b) AG 2 - número de gerações: 1.000, número de indivíduos: 500, Rota inicial: 100 MPP e 400 aleatórias



(c) AG 3 - número de gerações: 10.000, número de indivíduos: 750, Rota inicial: 250 MPP e 500 aleatórias



(d) AG 4 - número de gerações: 10.000, número de indivíduos: 750, Rota inicial: 750 MPP baseadas em I-B5

Figura 9 – Rotas geradas pelo algoritmo evolutivo genético

Fonte: De autoria própria.

pontos, sem os agrupamentos iniciais, poderia ser encontrada alguma rota melhor, seja preenchendo todos os dias com atendimentos até o máximo possível, ou deixando todos os vendedores com a mesma quantidade de atendimentos. Porém ao realizar esse método, o tempo de processamento se tornou muito elevado, as matrizes de tempo e distancia muito grandes, e para os casos do ALNS e do algoritmo genético o computador esgotou a memória, sem terminar as rotas.

Nota-se ainda que, como era esperado, quanto maior for o número de iterações nos algoritmos maior é o tempo para encontrar as soluções. Em muitos casos foi visto que não foi necessário ter passado por todas as rodadas para encontrar a melhor rota. Para resolver essa questão poderia ser implementada uma função de inércia, que após um determinado número de repetições o algoritmo apresente como sendo a melhor solução, sem a necessidade de passar por todas.

Conforme já mencionado, a rota aleatória é usada apenas para que haja uma, ou mais, rotas iniciais para a permutação utilizando outros métodos. Uma rota criada dessa forma não é utilizada no cotidiano, pois leva a pontos muito distantes entre si, sem nenhuma minimização de tempo ou distância.

No caso das rotas obtidas pelo método do ponto proximal, vale ressaltar que embora não seja a melhor resposta, otimiza o problema de forma rápida, e sem um custo computacional elevado. Com ela foi obtido o quarto melhor resultado, com um tempo de deslocamento de 5.549s. Considerando-se que a resposta foi obtida em quinze segundos, um valor quase mil vezes mais rápido que os algoritmos que possuíram respostas melhores.

Percebe-se que o ALNS não trouxe nenhuma melhoria para a rota inicial usando o método do ponto proximal, possivelmente por ter ficado preso em um mínimo local e, com alterações realizadas pela destruição e reparo da rota, não foi possível sair dele.

Entretanto para rotas iniciais aleatórias, o algoritmo baseado na têmpera simulada, atingiu resultados muito próximos dos obtidos através das soluções do método do ponto proximal.

Identifica-se que o algoritmo genético foi o que trouxe os

Tabela VI – Quadro resumo dos resultados dos cinco algorítimos testados. Nessa deve-se buscar pelo menor tempo em rota. É importante levar em consideração o tempo de CPU, devido ao custo computacional.

	Parâmetros	**Tempo em rota (s)**	**Número de vendedores ou dias**	**Tempo de CPU (s)**	**Observações**
Rota ordenada	-	77.178	7	≤ 0.001	-
rota aleatória	Seed = 42	86.385	8	≤ 0.001	para testes foram geradas mais 10 mil soluções aleatórias, porém todas tiveram resultados semelhantes
rota MPP	-	5.549	5	15	foram geradas todas as rotas para cada um dos pontos
ALNS	Segmentos = 100 Iterações = 50 rota aleatória	14.813	5	992	-
	Segmentos = 100 Iterações = 50 rota MPP	5.549	5	1.059	-
	Segmentos = 100 Iterações = 100 rota aleatória	8.619	5	2.171	-
	Segmentos = 100 Iterações = 100 rota MPP	5.549	5	2.532	-
GA	gerações = 10000 população = 250	15.573	5	5.698	-
	gerações = 1000 população = 500	3.981	5	862	Usadas 100 soluções MPP e 400 aleatórias
	gerações = 10000 população = 750	1.825	5	14.521	Usadas 250 soluções MPP e 500 aleatórias
	gerações = 10000 população = 750	2.168	5	14.108	Usadas 750 soluções MPP modificadas, conforme descrito

Fonte: De autoria própria.

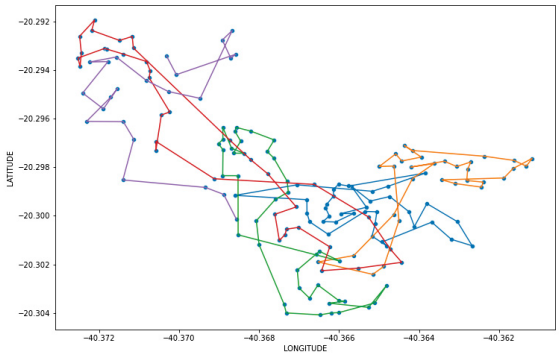


Figura 10 – Melhor rota obtida foi através de um algoritmo genético, com uma população inicial de 750 indivíduos, 10.000 gerações, e as soluções iniciais dadas pelo método do ponto proximal modificado, e descrito em I-B5

Fonte: De autoria própria.

melhores resultados, tendo 3 dos 4 melhores resultados que os demais algoritmos. A melhor rota obtida pode ser vista em 10. Houve uma redução no percurso de mais de uma hora (3.724s), porém com um gasto computacional mais elevado, se comparado com o método do ponto proximal, que apresentou o quarto melhor resultado.

Com isso, percebe-se que o uso de heurísticas e meta-heurísticas trazem melhora nos processos de otimizações de rotas de veículos.

Referências

[1] G. Laporte and I. Osman, "Routing problems: A bibliography," *Annals of Operations Research*, vol. 61, pp. 227–262, 01 1995.

[2] B. Eksioglu, A. V. Vural, and A. Reisman, "The vehicle routing problem: A taxonomic review," *Computers Industrial Engineering*, vol. 57, no. 4, pp. 1472–1483, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835209001405

[3] W. Herdianti, A. Gunawan, and S. Komsiyah, "Distribution cost optimization using pigeon inspired optimization method with reverse learning mechanism," *Procedia Computer Science*, vol. 179, pp. 920–929, 01 2021.

[4] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410, 1954. [Online]. Available: https://doi.org/10.1287/opre.2.4.393

[5] L. Leustean, A. Nicolae, and A. Sipos, "An abstract proximal point algorithm," 2017. [Online]. Available: https://arxiv.org/abs/1711.09455

[6] D. Pisinger and S. Røpke, *Large Neighborhood Search*, 2nd ed. Springer, 2010, pp. 399–420.

[7] A. Santini, S. Ropke, and L. M. Hvattum, "A comparison of acceptance criteria for the adaptive large

neighbourhood search metaheuristic," *Journal of Heuristics*, vol. 24, no. 5, pp. 783–815, Oct 2018. [Online]. Available: https://doi.org/10.1007/s10732-018-9377-x

[8] G. Mattos Ribeiro and G. Laporte, "An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem," *Computers Operations Research*, vol. 39, no. 3, pp. 728–735, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054811001298

[9] F. Gerges, G. Zouein, and D. Azar, "Genetic algorithms with local optima handling to solve sudoku puzzles," in *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, ser. ICCAI 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 19–22. [Online]. Available: https://doi.org/10.1145/3194452.3194463

[10] O. M. Shir, *Niching in Evolutionary Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1035–1069. [Online]. Available: https://doi.org/10.1007/978-3-540-92910-9_32

[11] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: Why and how you should (still) use dbscan," *ACM Trans. Database Syst.*, vol. 42, no. 3, jul 2017. [Online]. Available: https://doi.org/10.1145/3068335

[12] S. T. Windras Mara, R. Norcahyo, P. Jodiawan, L. Lusiantoro, and A. P. Rifai, "A survey of adaptive large neighborhood search algorithms and applications," *Computers Operations Research*, vol. 146, p. 105903, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054822001654

[13] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation Science*, vol. 40, no. 4, pp. 455–472, 2006. [Online]. Available: https://doi.org/10.1287/trsc.1050.0135

[14] W. Lee and H.-Y. Kim, "Genetic algorithm implementation in python," in *Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, 2005, pp. 8–11.

[15] C. Sheppard, *Genetic algorithms with Python*. Smashwords Edition S. l, 2017.