USO DA FERRAMENTA OR-TOOLS PARA RESOLVER O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELAS DE TEMPO COM DADOS GEORREFERENCIADOS

USE OF OR-TOOLS TO SOLVE THE VEHICLE ROUTING PROBLEMS WITH TIME WINDOWS WITH GEOREFERENCED DATA

Fred Daniel Varelo da Silva
Orientador: Diego Rocha Lima

RESUMO

O Problema de Roteamento de Veículos com Janela de Tempos (PRVJT) é uma extensão do Problema de Roteamento de Veículos (PRV), onde existe no mínimo um depósito, mais de um ponto a ser visitado, e por fim, mais de um veículo para o cumprimento das rotas, porém, o PRVJT vai além, onde o veículo tem que passar por todos os pontos em sua rota, em suas respectivas janelas de tempo, que pode ser algo que o cliente (ponto) pode determinar, onde está janela de tempo pode ser o horário em que o cliente está disponível para receber a visita, ou algo em que o próprio programa irá determinar para que tenha o máximo de tempo otimizado, onde partindo do ponto inicial (deposito), ele irá calcular todas as rotas possíveis para todos os pontos, e selecionar a melhor rota levando em conta a janela de tempo. E para resolver este problema, foi utilizado o *Google OR-Tools* onde ele disponibiliza algoritmos para a computação destas rotas, e para o seu funcionamento é utilizado a matriz de duração e a janela de tempo, onde a matriz de duração foi feita utilizando o *Google Matrix API*, onde passado os pontos, sendo possível utilizar latitude e longitude, ou endereços reais, é retornado a matriz. Com isso busca-se verificar a potencialidade da *OR-Tools* utilizando esses algoritmos disponíveis, e para essa verificação foi utilizado métricas de atratividade visual de rotas, onde busca verificar a facilidade do entendimento da rota pelo usuário. E com isso é visto que a *OR-Tools* é uma ferramenta bastante competente, pois é disponibilizado diversos algoritmos para a resolução do PRV como um todo, onde inclui também o PRVJT, além disso é disponibilizado esses algoritmos em diferentes linguagens de programação, e o resultado das rotas obtidas é de fácil entendimento, fazendo assim, ser possível o uso das métricas de atratividade visual escolhidas, podendo assim averiguar como esses algoritmos se comportam em diferentes cenários de centros urbanos, número de pontos e número de veículos.

Palavras-chave: Problema de Roteamento de Veículos. Janela de Tempo. OR-Tools. Atratividade Visual. Dados Georreferenciados.

;

ABSTRACT

The Vehicle Routing Problem with Time Windows (VRPTW) is an extension of the Vehicle Routing Problem (VRP), where there is at least one deposit, more than one point to be visited, and finally, more than one vehicle to fulfill the routes, however, the VRPTW goes further, where the vehicle has to pass by all the points in its route, in their respective time windows, which can be something that the customer (point) can determine, where this time window can be the time when the customer is available to receive the visit, or something that the program itself will determine to have the maximum optimized time, where starting from the initial point (deposit), it will calculate all possible routes to all points, and select the best route taking into account the time window. To solve this problem, the Google OR-Tools was used, where it provides algorithms for computing these routes, and for its operation is used the duration matrix and the time window, where the duration matrix was made using the Google Matrix API, where the points are passed, and it is possible to use latitude and longitude, or real addresses, the matrix is returned. With this we seek to verify the potential of OR-Tools using these available algorithms, and for this verification we used metrics of visual attractiveness of routes, which seeks to verify the ease of understanding of the route by the user. And so it is seen that OR-Tools is a very competent tool, because it is available several algorithms for solving the VRP as a whole, which also includes the VRPTW, and these algorithms are available in different programming languages, and the result of the routes obtained is easy to understand, making it possible to use the metrics of visual attractiveness chosen, and thus be able to see how these algorithms behave in different scenarios of urban centers, number of points and number of vehicles.

Keywords: Vehicle Routing Problem. Time Windows. OR-Tools. Visual Attractiveness. Georeferenced data.

1 INTRODUÇÃO

O Problema de Roteamento de Veículos (PRV) (TOTH; VIGO, 2014), tem como seu objetivo principal encontrar as menores rotas possível dado diversos pontos em um mapa ou gráfico, e no mínimo mais de um veículos para cumprir essas rotas, onde essas rotas iram iniciar e terminar em um depósito, sendo que cada um desses pontos irão ser atendidos exclusivamente uma única vez.

O PRV em janela de tempos (PRVJT) (KIM; KIM; SAHOO, 2006), é como uma extensão do PRV básico, porém, com tempos estimados de quando o veículo deverá passar por cada um desses pontos, isso faz com que ele resolva o caso onde um cliente que é um ponto no mapa,

especifique que estará disponível entre certa janela de tempo para receber a entrega, fazendo com que possamos calcular uma rota onde o veículo irá passar por esse ponto, nessa janela de tempo em que o cliente informa estar disponível, evitando que a entrega não seja efetuada com sucesso.

Alguns métodos que são utilizados são: Métodos Exatos, Heurística (MARTÍ; REINELT, 2022) e Meta-Heurística (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018). Um dos Métodos Exatos utilizados é a programação dinâmica, ela é aplicável a problemas nos quais a solução ótima pode ser computada a partir de uma solução previamente calculada e memorizada que é o caso do PRV, pois sempre iremos procurar a rota ótima (solução ótima).

tem a Heurística orientada para o tempo do vizinho mais próximo *(Time-oriented, nearest-neighbour heuristic)* (YU et al., 2016), onde ela é iniciada a partir do cliente mais próximo do depósito, a cada interação na busca do próximo cliente irá calcular geograficamente e temporalmente com base no último ponto, ele encontrando um ponto, uma nova rota é criada, e novamente ele faz a interação até voltar novamente ao depósito.

Já do lado da Meta-Heurística tem a Pesquisa Tabu (LAGUNA, 2018), um procedimento adaptativo auxiliar, onde ele guia um algoritmo de busca local na exploração contínua dentro de um espaço de busca. A partir de uma solução inicial, tenta avançar para uma outra solução (melhor que a anterior) na sua vizinhança até que se satisfaça um determinado critério de parada, além disso, a Busca Tabu não é confundida pela ausência de vizinhos aprimorantes, pois o método é construído de forma a evitar o retorno a um ótimo local previamente visitado. Esta característica faz com que o método seja capaz de superar a otimalidade local e atingir um resultado ótimo ou próximo ao ótimo global.

Na área do PRVJT tem alguns trabalhos onde tem atratividade visual da rota (facilitar o entendimento da rota, onde conseguimos ver com clareza qual a rota que o veículo deve passar, evitando assim que ele se confunda com sua rota estabelecida, com a rota de outro veículo), onde é muito importante, pois como mostrado no (ROSSIT et al., 2019). Foi visto que, se reduzir o número de vezes em que as rotas de cruzem, melhorar a compactação das rotas e reduzir a complexidade das rotas, eles geralmente tinha os seguintes efeitos: Incremento na satisfação dos profissionais e especialização do motorista, redução dos custos de implementação, incremento na distancia total, e o incremento no tempo total de operação.

Assim podemos ver que apesar da atratividade visual da rota é algo que pode ser usado em diversos modelos de PRV, no caso do PRVJT pode vir a ser um ponto negativo, visto que é aumentado a distância total das rotas para que ela não se cruzem, fazendo assim que consequentemente também aumentar o tempo total das rotas, o que pode vir a ser um problema, pois como tem janelas de tempo para cada um dos pontos no gráfico, pode acabar prejudicando essas janelas, fazendo até mesmo uma rota que não consiga passar por um certo cliente no tempo estipulado, sendo assim necessário ou uma versão em que demos prioridade para o tempo, e em segunda plano a atratividade visual da rota, ou até mesmo o aumento da frota para que possamos ter o melhor dos dois, a janela de tempo sendo feita com sucesso, e tendo uma atratividade visual para o operador do sistema.

Já no programa sera feita a utilização da *Google OR-Tools* (PERRON; FURNON, 2019-7-19) disponibilizada pelo *Google*, onde ele já nos disponibiliza exemplos de códigos em diversas linguagens, explica o que certo modelo busca fazer, mostra como o programa se comporta quando ele está rodando, além de explicar como fazer cada coisa no programa, no exemplo do PRVJT, ele mostra como criar dados para testarmos o programa ou até mesmo para ser utilizado no programa final, como adicionar as janelas de tempo nos nossos pontos, função principal, o *print* com algumas informações básicas do programa, como o tempo em que passamos em cada um dos pontos e qual a janela de tempo desses pontos, podendo assim ser verificado se foi possível passar por esses pontos em suas janelas de tempo específicas, o tempo total da rota, entre outros.

Apesar do *Google OR-Tools* nos ensinar a fazer os dados fictícios, o programa será feito com dados reais, para que possamos mostrar como o programa irá se comportar com uma situação real, e para que possamos comparar também a mesma situação real, onde não será utilizado o programa com o PRVJT, e sim onde a própria pessoa decide a sua rota baseado em sua experiência na sua área de trabalho.

2 TRABALHOS RELACIONADOS

Esta seção visa apresentar alguns trabalhos relacionados à Roteamento de Veículos com os tópicos citados anteriormente: PRVJT, *Google OR-Tools* e Atratividade Visual das Rotas.

No artigo (KIRCI, 2016), apresenta um sistema usado na Turquia utilizando Roteamento de Veículos com Janela de Tempos, e mostra dados reais comparando um motorista utilizando apenas o seu conhecimento e um *GPS* convencional para fazer as rotas, e depois o mesmo motorista fazendo a mesma rota, porém utilizando o sistema com o PRVJT, e mostra uma grande diferença na distancia total da rota, chegando a ser até mesmo mais de duas vezes menor quando utilizado em uma areá de diversas cidades ao mesmo tempo.

O Artigo (BRAEKERS; RAMAEKERS; NIEUWENHUYSE, 2016), compara 277 artigos no tempo de 2009 até junho 2015, onde ele faz uma taxonomia desses artigos, organizando eles com base no seu modelo de PRV, além de também separa-los pelos métodos que foram usados. E como ele mesmo cita, fazendo com que já possamos ter em mente o que podemos tem em mente no que usar quando fazer o sistema, como pegando um método que está sendo mais utilizado, fazendo assim que possamos ter mais exemplos do que pode ser feito no sistema.

Já para o *Google OR-Tools* foi escolhido o (NAZARI et al., 2018), pois como é citado no artigo, o *Google OR-Tools* é um solucionador de PRV *open-source*, que já disponibiliza algoritmos para calcular as rotas, ou seja, o que nos buscamos uma forma fácil de resolvermos o problema de PRV, e ele já vem com exemplos de algoritmos para essa resolução, para o PRVJT com algoritmos de Meta-Heurística, que é o método mais utilizado em problemas de PRV.

Já no (ROCHA et al., 2022), trata da atratividade visual, mostrando que o problema citado anteriormente sobre a atratividade visual possa ser resolvido, que seria, ao mesmo tempo que deixar as rotas visivelmente atrativas para o usuário, também fazer o seu papel de diminuir o máximo possível das rotas e reduzir o custo total das operações.

Com isso podemos ver a Tabela 1, onde se tem na primeira coluna, o que se será comparado entre os artigos, PRV (se no artigo explica o funcionamento do PRV ou cita ele), PRVJT (se no artigo explica o funcionamento do PRVJT ou cita ele), Programa (se no artigo é citado a criação de um programa para a resolução da problemática tanto do PRV ou PRVJT), Mapa (se no artigo é mostrado mapas para demonstrar as rotas criadas), Diversos Algoritmos (se foi utilizado mais de um tipo de algoritmo para a resolução do PRV ou PRVJT) e por fim Métricas (métricas que foram utilizadas para verificar a potencialidade das feramentas e algoritmos utilizados).

	Kirci	Braekers; Ramaekers; Nieuwenhuyse	Nazari	Rocha	Fred
PRV	x	x	x	x	x
PRVJT	x	x	x	x	x
Programa	x		x		x
Mapa	x			x	x
Diversos Algoritmos		x	x		x
Métricas	x		x	x	x

Tabela 1 – Tabela de comparação entre artigos

Com isso é visto que no primeiro artigo o único ponto ao qual falta, é a utilização de outros tipos de algoritmos para a verificação de qual algoritmo é melhor para qual tipo de cenário. Já no segundo artigo como ele é focado na junção de diversos artigos e ver qual tipo de PRV é o mais buscado e estudado atualmente, ele não mostra programas mapas ou métricas, porém cita o PRV e o PRVJT e também os tipos de algoritmos mais utilizados para esses problemas. no terceiro artigo é utilizado a *OR-Tools*, porém não é mostrado mapas para as demonstrações, e os algoritmos e métricas, são apenas citados. No quarto artigo sobre a atratividade visual não é citado um programa feito ou diversos algoritmos, porém ele cita as métricas de atratividade visual, mapas com atratividade visual e sem atratividade visual, e tudo isso com os diversos tipos de PRV. Por fim vem este artigo, onde se busca apresentar o PRV, apresentar e amostrar o PRVJT, fazer um programa para a resolução do problema, mostrar mapas dos resultados obtidos, mostrar como o programa se comporta com diversos algoritmos, e por fim a utilização de métricas para a verificação de o que o programa esta priorizando para resolução do problema.

3 FUNDAMENTAÇÃO TEÓRICA

Com o objetivo de criar a rota mais otimizada possível, nós também ganhamos alguns aspectos importantes para o nosso sistema, sendo eles a diminuição de custos, tanto do veículo, manutenção reduzida por fazer rotas significativamente reduzidas, fazendo assim que os desgastes de peças seja também reduzido, e também acabe diminuindo a redução do consumo de gasolina.

Além de oferecer uma plataforma mais objetiva para os motoristas, com rotas mais objetivas, e fáceis de entender com a atratividade visual implantada, e com isso a empresa que utilizar o sistema ira se beneficiar bastante, pois além de contar com sua frota padrão de funcionários contratados, pode até mesmo contar com frotas de autônomos, um exemplo é o que a *Amazon* tem feito, onde eles oferecem um serviço parecido com o *Uber*, onde qualquer pessoa pode ser um funcionário autônomo da empresa, assim escolhendo quando e por quanto tempo trabalhar, onde eles utilizam de CPRVJT para o seu sistema de rotas.

O Problema de Roteamento de Veículos Capacitados com Janela de Tempo (PRVCJT) (TANEL et al., 2022) é a junção de 3 PRVs, o PRV, onde busca fazer a menor rota possível a partir do deposito passando por todos os pontos e voltando para o deposito, o PRVJT, onde um veículo tem que passar por todos os pontos da rota do PRV base, em uma janela de tempo especifica, e o Problema de Roteamento de Veículos Capacitados (PRVC) (UCHOA et al., 2017), onde tem que efetuar a rota do PRV, porém com um veículo que tem um máximo de capacidade que ele pode carregar. Esse problema é um dos mais estudados, pois ele é algo que acontece no mundo real, onde tem um veículo que não pode carregar infinitas mercadorias/pessoas, ele é um problema que pode ser utilizado tanto com mercadorias, por exemplo, busca e entrega de objetos feitas por transportadoras, outro que ele pode ser utilizado também é o transporte de pessoas, por exemplo, ônibus, transportes escolares e empresariais.

No PRVC, nós tem o nosso depósito como o ponto 0, ou seja o ponto inicial, e n outros pontos que são os clientes, referidos como N = {1, 2, . . . , n}. Para sabermos quais mercadorias tem que entregar para cada um dos clientes, e seus respectivos pesos, tem o peso dado como (q_i) e a busca pelas x mercadorias a serem entregue com i ∈ N.

A frota de veículos é considerada homogênea K = {1, 2, . . . , |K|}, onde todos vão ter a mesma capacidade Q > 0, ou seja se a empresa tiver mais de um tipo de veículo, deve levar em conta de fazer outra lista/frota para outro tipo de veículo, o veículo sempre vai de um ponto i para j, então tem o custo de viajem como (C_{ij}).

A rota que o veículo ira percorrer tem uma rota r = $\{i_0, i_1, ..., i_s, i_{s+1}\}$, onde o i_0 e o i_{s+1} é o deposito, assim tem que ele ira iniciar e voltar a ele, e o conjunto S = $\{i_0, ..., i_s\} \subseteq N$, serão os clientes que o veículo ira deixar a mercadoria, assim tem S ⊆ N.

Com isso tem o nosso veículo (K), com capacidade (Q), onde ele fara uma rota (r), com clientes (i ∈ {1, . . . , n}) com (q_i) de mercadoria. Com isso, nós tem que construir uma rota, para que o veículo a partir do depósito, passe por todos os clientes uma única vez, e volte ao depósito sem exceder o limite de carga, e ao mesmo tempo percorrendo o menor caminho possível. Com isso nos tem o nosso modelo matemático definido como G = (V,A), onde G é um grafo dirigido, V = {0, 1, . . . , n, n + 1} seja o conjunto de nós no grafo, onde novamente tem 0 e n + 1 sejam respectivamente o inicio e o fim da rota, ou seja o nosso depósito, e {1, . . . , n} sejam os nossos clientes, já as distancias são dadas como uma matrix (C_{ij}), onde, i,j ∈ {0, . . . , n + 1}.

Como tem nossa rota (r) do nó 0 ao nó n + 1, então o nosso caminho é dado como $\overline{r} =$ (v_0, v_1, . . . , v_h, v_{h+1}), onde v_i, para i ∈ {0, . . . , h + 1} são os nós visitados da rota, ou seja v_0 = 0 e v_{h+1} = n + 1, onde h é o número de clientes visitados na rota. Assim para satisfazer a

necessidade de capacidade, é escrita como:

$$\sum_{i=1}^{h} q_{vi} \leq Q \tag{1}$$

já o nosso custo $c_{\overline{r}}$ de uma rota $\overline{r}$ é dado por:

$$c_{\overline{r}} = \sum_{i=0}^{h} c_{vi,vi+1} \tag{2}$$

Assim tendo R o conjunto de todas as rotas possíveis e ($a_{i,\overline{r}}$) seja uma matriz booleana com n linhas e |R| colunas. tem $a_{i,\overline{r}}$ = 1 caso a rota $\overline{r}$ servir o cliente i. Então tem o CPRV formulado da seguinte forma:

$$\textbf{min} \sum_{\overline{r} \in R} c_{\overline{r}} x_{\overline{r}} \tag{3}$$

sujeito a:

$$\sum_{\overline{r} \in R} a_{i,\overline{r}} x_{\overline{r}} = 1 \qquad \forall i \in \{1, \ldots, n\} \tag{4}$$

$$\sum_{\overline{r} \in R} x_{\overline{r}} = m \tag{5}$$

$$x_{\overline{r}\in}\{0,1\} \qquad \overline{r} \in \mathrm{R}$$

Com isso a função (3) seleciona as rotas possíveis que minimiza a soma do custo da rota, já a equação (4) certifica de que todos os clientes são servidos exatamente uma vez e a equação (5) assegura que são utilizados exatamente m veículos. Algumas versões da (5) faz com que seja seguido a risca m veículos, já em outros casos ele disponibiliza ilimitados números de veículos.

Com o CPRV também podemos limitar o tempo máximo de cada rota, fazendo assim cumprir a carga horaria de cada cliente de maneira mais correta, ou seja, se um motorista trabalha 8 horas por dia, então podemos delimitar o tempo máximo de por exemplo, 2 rotas onde cada rota tem o tempo máximo de 4 horas, fazendo assim com que o motorista não cumpra horas extras, assim fazendo com que empresa gaste menos ainda. Uma maneira de fazer isso, é pegando a media de distancia por tempo que aquele motorista faz, assim tem a distancia media que ele faz por certa quantidade de tempo, assim podemos criar uma medida de distância $d_{i,j}$ em que pode não ser diferente de $c_{i,j}$ que é o modelo padrão. Assim fazemos com que nenhuma rota seja superior a D, assim fazemos que as visitas aos clientes $v_0, \ldots, v_{h+1}$ no caminho possível $\overline{r}$, devem satisfazer:

$$\sum_{i=0}^{h} d_{vi,vi+1} \leq D \tag{6}$$

A restrição (6) também pode ser vista como um limite do tempo gasto na rota e os tempos de serviço nos clientes podem ser incorporados em ($d_{i,j}$).

Agora para o *PRVJT (Vehicle Routing Problem With Time Windows)* que é o Problema de Roteamento de Veículos Com Janela de Tempo, o modelo a seguir pode ser matematicamente afirmado como um problema de fluxo de rede com janelas de tempo e restrições de capacidade:

$$min \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \quad s.t., \tag{7}$$

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1 \quad \forall i \in C, \tag{8}$$

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ijk} \leq q \quad \forall k \in V, \tag{9}$$

$$\sum_{j \in N} x_{0jk} = 1 \quad \forall k \in V, \tag{10}$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0 \quad \forall h \in C, \ \forall k \in V, \tag{11}$$

$$\sum_{i \in N} x_{i,n+1,k} = 1 \quad \forall k \in V, \tag{12}$$

$$x_{ijk}(S_{ik} + t_{ij} - s_{jk}) \leq 0 \quad \forall i, \ j \in N, \ \forall k \in V, \tag{13}$$

$$a_i \leq s_{ik} \leq b_i \quad \forall i \in N, \ \forall k \in V, \tag{14}$$

$$x_{ijk} \in \{0,1\} \quad \forall i, \ j \in N, \ \forall k \in V. \tag{15}$$

A função objeto (7) ira minimizar o custo total da viajem, a (8) certifica de que cada cliente ira ser visitado exatamente uma vez, e a (9) fala que cada veículo poderá carregar apenas até sua carga máxima especificada. Depois as equações (10), (11) e (12) indica que cada veículo deve sair do deposito número *0*, depois que o veículo chegar no cliente ele tem que sair para outro destino, ou seja outro cliente, porém quando chegar no cliente *n*, ele deve ir para o deposito *n + 1*. No (13) ele afirma que a janela de tempo está sendo seguida, e no (14) são as restrições de integridade, vale ressaltar que um veículo não utilizado irá andar na rota vazia *(0, n + 1)*, onde ele sai do deposito *0* e volta ao deposito *(n + 1)*.

4 METODOLOGIA

A pesquisa tem por finalidade resolver problemas de rotas para empresas e transportes escolar, transporte de trabalhadores, entre outros, onde no caso de uma empresa podemos utilizar na entrega de suas mercadorias para seus consumidores, ou até mesmo no recebimento da mercadoria de seus consumidores, já no caso de um transporte escolar, ele tem o objetivo de passar por todas as suas paradas regulares para pegar os alunos e deixa-los em suas respectivas escolas, e esse caso também serve para transporte industrial, outra finalidade é diminuir os custos das viagens para as empresas, e aumentar a eficiência dos seus trabalhadores.

E o objetivo é criar um sistema onde podemos aplicar essas finalidades, onde tem esse sistema especifico para uma área, porém com algumas mudanças em seu código, podendo englobar diferentes áreas com problemas de transporte com rotas. O sistema não vai ter uma versão para o cliente, apenas para o motorista e a sua empresa, porém ainda assim o sistema vai buscar ser direto ao ponto, com a decisão do deposito, que será o ponto inicial e o final da rota, as adições dos clientes que serão os pontos em que a rota ira passar, e a imagem com a rota mais simples o possível, para que não tenha problemas em entender as rotas, pois mesmo criando rotas menores, caso as rotas se interliguem muitas vezes pode vir a causar o erro do motorista, então a atratividade visual da rota pode vir a ser algo importante para cumprir o objetivo do sistema, porém no caso do PRVJT pode vir a ser algo negativo que pode vir a impactar negativamente na janela de tempo.

E como foi analisado pela Kirci, nos tem uma grande diminuição do tamanho das rotas, principalmente quando ela testou em uma área maior, e vale resaltar que quando nos tem áreas maiores, nos tem maiores rotas, além de caso um motorista for fazer uma rota sem o sistema, aumenta ainda mais o número de erros que ele pode cometer em pegar caminhos diferentes, e também pegar em ordens erradas de pontos em que ele pode passar, ou seja, vai ficar ainda mais aparente a utilidade de usar um sistema como o PRV para as rotas, pois suponhamos que tem n clientes em uma área a, nos tem então um motorista (A) que não usa o sistema, e uma motorista (B) que utiliza o sistema com o PRV, as chances de que o motorista (A) cometa erro em suas rotas, pegando caminhos maiores, desnecessários, ou ate mesmo indo em pontos em uma ordem que faça aumentar a sua rota pode acontecer seja E, caso nos aumentarmos apenas o número de clientes, vai diminuir o número de erros, já de tem mais pontos no mapa, e o mapa é de mesmo tamanho, porém caso aumentarmos a nossa área consideravelmente, a taxa de erro que o motorista (A) vai cometer, ira aumentar exponencialmente caso o número de clientes também não aumente em uma grande quantidade.

Porém para o sistema PRV isso não é um problema em relação a rota, porém pode vir a ser um problema no tempo que ele ira conseguir processar a rota, por isso não podemos criar um sistema para todo o Brasil por exemplo, pois suas dimensões são tão grandes que para carregar o mapa, para em seguida carregar a sua rota, pode demorar muito, principalmente caso seja usado para calcular em um computador simples, então tem que ver área que iremos fazer o nosso sistema funcionar. Ou seja, mesmo tendo um sistema que é superior a apenas a intuição humana, de qual ponto será o próximo, e qual caminho eu devo ir, que pode ser utilizado por GPS, porém os GPS padrões tem apenas a opção de ponto A e ponto B, onde não considera outros pontos que pode vir a afetar essa rota A e B, tem que levar em conta que o sistema também tem suas limitações, que pode ser otimizado por meio de atualizações.

Utilizando um *package* do *OpenStreetMaps* (OpenStreetMap contributors, 2017), o *OpenStreetMaps NetworkX* (OSMnx) (BOEING, 2017), nele é possível pegar o mapa de qualquer parte do mundo e também qualquer ponto de latitude e longitude, e dentro do próprio OSMnx ele da número para pontos como curvas e interseções, e esses números podem ser chamados por meio desse pacote, fazendo assim que tenhamos diversas informações sobre aquele ponto, como

se ele é uma estrada, rodovia, direção, se é asfaltada ou não, entre outros. E uma das que mais importa a gente, podermos calcular a distancia de um ponto ao outro, e utilizando assim uma função do próprio OSMnx para calcularmos a menor rota entre dois pontos.

Pegamos as latitudes e longitudes mínimas e máximas, e passamos os seus parâmetros, depois nos pegamos os gráficos dessa área e mostrar na tela o mapa dessa área como mostra a imagem a seguir:

```
ox.config(log_console=True, use_cache=True)

min_lat, max_lat, min_long, max_long = -4.582200, -4.548900, -37.787000, -37.7457000

graph = ox.graph_from_bbox(min_lat, max_lat, min_long, max_long, network_type='drive')
graph_projected = ox.project_graph(graph)
###ox.plot_graph(G_projected)
fig, ax = ox.plot_graph(graph_projected, edge_color='#FFF',node_size=0, figsize=(10, 10))
print (type(fig), type(ax))
```

Figura 1 – Exemplo de código para pegar uma área

Esse mapa da Figura 1 pode ser totalmente personalizado, tanto do fundo, das ruas, das interseções, entre outros. Vai por conta da pessoa qual estilo ira ser adotado, o mesmo vale para as rotas, onde podemos aumentar sua largura, cor, os pontos onde ele passa, entre outros.

4.1 Obtenção dos Pontos

O *OSMnx* foi utilizado apenas para a obtenção de pontos aleatórios no mapa, onde tem latitude e longitude mínimas e máximas, e com isso pode ser utilizado um método para pegar pontos aleatórios entre dois pontos, e para isso utilizamos latitude mínima e máxima, e longitude mínima e máxima, e como o *OSMnx* nos trata de pegar todos os pontos da área escolhida, se um ponto for escolhido e ele não existir no mapa, como por exemplo for um rio ou mar, ele ira redirecionar esse ponto para o ponto mais próximo existente.

4.2 Matriz de Duração

Após a obtenção dos pontos aleatórios é então feita a matriz de duração, pois como iremos tratar de janela de tempo, não pode utilizar a matriz de distância que é normalmente utilizada em um PRV normal, e para isso foi utilizado o *Google Maps API*, mais especificamente o *Distance Matrix API*, que faz a matriz de distância, porém com algumas mudanças é possível transformá-la em uma matriz de duração, que ao invés de nos retornar distancias em metros, nos retornara a distancia em forma de segundos ou minutos de um ponto ao outro.

4.3 Obtenção das Rotas

Na obtenção das rotas, é utilizado o *Google OR-Tools*, onde nele iremos passar a nossa matriz de duração obtida anteriormente, passar também o número de veículos que será utilizado,

a janela de tempo para aquela matriz de duração, nessa parte de janela de tempo você tem que fazer ela da sua maneira, a maneira escolhida foi pegar a matriz de duração onde cada coluna representa um ponto em relação aos outros, com isso peguei o tempo mínimo e máximo de todos os outros pontos em relação a cada ponto, com isso eu soube qual era o menor tempo de um ponto qualquer até aquele ponto, e o máximo de tempo de um ponto qualquer até aquele ponto, porém é claro, um cliente pode decidir quando ele quer a janela de tempo, mas como estamos utilizado pontos aleatórios, essa foi a melhor maneira para a criação da janela de tempo.

Com tudo pronto, basta rodar a *Google OR-Tools* para obter a rota daquela matriz, com a quantidade de veículos escolhida, o retorno se da da seguinte forma: veículo n$\rightarrow$ rota = 0 $\rightarrow$ P1 $\rightarrow$ P2 $\rightarrow$ . . . $\rightarrow$ Pn-2 $\rightarrow$ Pn-1 $\rightarrow$ 0, duração total da rota do veículo n = D, caso tenha mais veículos ele irá repetir até o ultimo veiculo, e ao final ele ira dizer o tempo total, somando todas as rotas.

4.3.1 Formas de Calcular a Rota

Porém o *Google OR-Tools* nos disponibiliza diversas formas de calcular as nossas rotas, que são dadas como *Routing Options*, ou seja Opções de Rota, e foram escolhidos duas opções, *First Solution Strategy (FSS)* onde ele é o método que o *Google OR-Tools* utiliza para encontrar a solução inicial, e o *Local Search Options (LSO)* onde ele utiliza meta-heurística para encontrar a solução. O FSO tem treze opções de calcularmos, porém após uma calibração dessas treze opções, foram escolhidas três, o *PATH_CHEAPEST_ARC* que é o modelo padrão do *Google OR-Tools*, onde ele calcula a partir do deposito e compara com todos os outros pontos, e o menor escolhido ele adiciona na rota, e a partir dele ele procura o próximo ponto, a outra opção foi *Path Most Contrained Arc* onde ele é bastante parecido com a primeira opção, porém o arco de um ponto X a um ponto Y, será comparado com o arco do ponto X a um ponto Z, o menor arcos, ou como ele mesmo cita, o arco mais restrito, será escolhido, vale ressaltar que qual tem um arco de um ponto X ao ponto N que seria o nosso deposito, e todos os outros pontos já foram adicionados na rota, ele não ira comparar com outro arco, pois ele entende que é o fim da rota, e por fim a última opção foi *GLOBAL_CHEAPEST_ARC* ele conecta iterativamente dois pontos que produzem o segmento de rota menor, ou como ele mesmo cita, o mais barato.

Já no LSO tem um total de cinco opções, porém uma delas não funciona, e após fazer a calibração das quatro opções restantes, foram escolhidas três opções, a *GREEDY_DESCEND* aceita melhorar os vizinhos de busca local até que um mínimo local seja alcançado, a segunda opção é *GUIDED_LOCAL_SEARCH* ele usa pesquisa local para pegar o mínimo local, ele é geralmente o método mais eficiente de meta-heurística para o Problema de Roteamento de Veículos, por fim tem *TABU_SEARCH* onde ele usa a busca tabu para pegar o mínimo local.

4.3.2 Utilização das Rotas

E para todas essas opções foram utilizadas quatro cidades, São Paulo, Fortaleza, Washington D.C. e Madrid, onde cada um foram escolhidos quinze, trinta e quarenta e cinco pontos

aleatórios, ou seja, São Paulo vai ter uma matriz de quinze pontos, uma de trinta pontos e uma de quarenta e cinco pontos, onde cada uma dessas matrizes vão passar pelas três opções de FSO e as três opções de LSO, isso vale para as outras cidades também.

4.4 Métricas

Após isso vai ser colocado em pratica duas métricas, a primeira métrica é a de Compactação e a segunda métrica é a de Cruzamento. Essas métricas tem o intuito de verificar como o *Google OR-Tools* está fazendo essas rotas, o que ele esta priorizando quando cria essas rotas, como por exemplo, ele está priorizando minimizar o tamanho da rota ou está priorizando a atratividade visual da rota, ou então, ele está priorizando a atratividade visual da rota ou ele não se importa se tiver rotas com interseções.

4.4.1 Métrica de Compactação

A métrica de compactação é uma das mais usadas quando é falado de atratividade visual das rotas, por conta disso ela pode nos mostrar o quão atrativa é a nossa rota, e como citado anteriormente, a atratividade visual tem um grande problema, que caso você busque o máximo de atratividade pessoal ela sempre ira impactar negativamente no tamanho da rota, pois ele ira tentar melhorar a atratividade visual da rota as custas do tamanho que irá aumentar, e como também citado anteriormente o PRVJT ele tem o foco principal na priorização da janela de tempo, então com a Compactação poderemos ver se ele está colocando algum tipo de atratividade visual na rota, ou se está focando apenas na priorização da janela de tempo.

4.4.2 Métrica de Cruzamento

Já na métrica de Cruzamento nos iremos ver se as nossas rotas tem Cruzamento, o que pode nos mostrar se a nossa rota esta fazendo caminhos por onde já passamos previamente, e como o PRVJT prioriza a janela de tempo, essa métrica poderá nos mostrar se o *Google OR-Tools* irá priorizar a janela de tempo a ponto de fazer a rota se cruzar, o que pode ser mostrando melhor caso aumentem o número de veículos pros mesmos pontos, pois teremos mais veículos, mostrando se ele ira evitar algumas inter ceções ou não.

5 RESULTADOS

Levando em conta o artigo da Pinar Kirci: (KIRCI, 2016), onde ela fez testes em 4 cidades em que ela não cita, porém ela cita dois locais em que foi testado o seu algoritmo, em Istambul com 5.343 km² de área e o segundo local em que engloba uma parte da Turquia, incluindo Istambul e com o deposito também em Istambul com 63.955,46 km² de área, com os dados que foram passados podemos ter uma média para sabermos quão melhor foi o algoritmo dela comparado a um caso onde não foi utilizado o algoritmo, para assim termos uma base de quão eficiente pode ser o nosso.

No primeiro caso tem aproximadamente 250 km percorridos em uma rota A por um motorista, onde ele quem decide qual sera o próximo ponto a ser visitado, já com o algoritmo sendo utilizado foram aproximadamente 150 km percorridos na mesma rota A, podemos assim dizer que o algoritmo nessa área é 40% mais eficiente do que quando não utilizamos ele.

No segundo caso tem aproximadamente 3.500 km percorridos em uma rota B por um motorista, onde ele quem decide qual sera o próximo ponto a ser visitado, já com o algoritmo sendo utilizado foram aproximadamente 1.650 km percorridos na mesma rota A, podemos assim dizer que o algoritmo nessa área é 53% mais eficiente do que quando não utilizamos ele.

Quanto maior a área utilizada para calcular a rota, maior a eficiência do algoritmo em relação a quando não utilizamos, pois quando não utilizamos o algoritmos e deixamos que uma pessoa que escolha a rota, ela pode cometer diversos erros tanto na ordem dos pontos a serem visitados, como também no caminho em que ele ira utilizar, e com base nisso é esperado que tenhamos um algoritmo entre 5% até 20% de eficiência em rotas que tenham as mesmas proporções de áreas, porém feitas por um individuo comum.

Porém para saber se estamos nessa média de eficiência precisaríamos obter dados de alguma empresa que trabalha diversos clientes que devem ser visitador em uma rota, e que tenha janela de tempo, então como uma medida diferente, foi criados pontos aleatórios em algumas cidades, onde o deposito sempre se encontra no centro da cidade, e para a verificação da usabilidade da *Google OR-Tools* foi utilizados alguns algoritmos, inclusive Meta-Heurística que é o que tem como base por ser o mais utilizado.

5.1 Mapas Plotados

Como meio de demonstração das rotas e algoritmos que ira ser citado na próxima seção, algumas figuras a seguir foram plotados mapas utilizando as rotas da *OR-Tools* em conjunto com as coordenadas e o algoritmo de plotagem do *OSMnx*, vale ressaltar que a opção de algoritmo escolhido para a resolução destas rotas, foi o *PATH_CHEAPEST_ARC*, ao qual é o modelo padrão da ferramenta *OR-Tools*.

Na Figura 2, tem-se o mapa plotado da cidade de São Paulo, o maior centro urbano que sera utilizado como teste, nele foi utilizado três veículos para o cumprimento de rotas para quarenta e cinco pontos, pode-se notar que para o cumprimento dessas rotas, nenhuma delas optou por traçar uma rota pelo centro da cidade, pois o algoritmo verifica as ruas com maior velocidade média, ou seja, ele ira dar preferencia por avenidas ao invés de ruas.

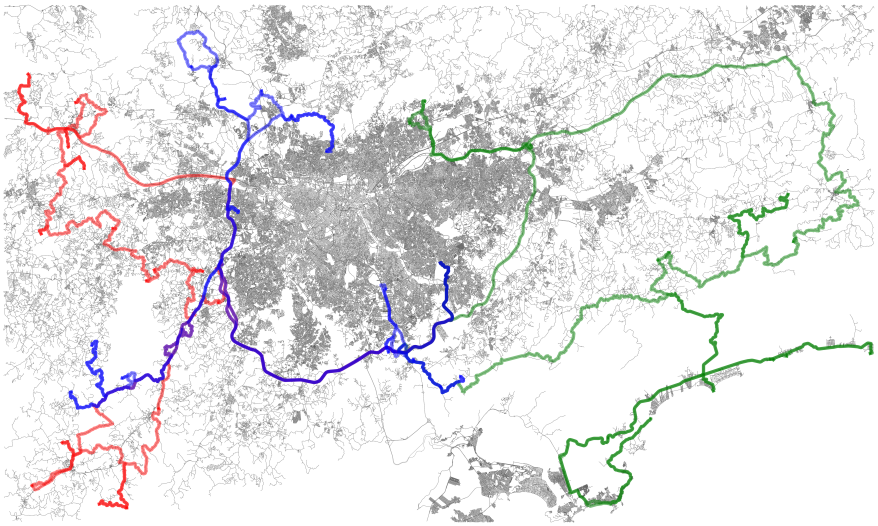

Figura 2 – Mapa Plotado de São Paulo 45 pontos

Agora na Figura 3, temos um centro urbano relativamente menor em comparação a São Paulo, onde agora nesse, foi testado novamente três veículos, porém desta vez foram escolhidos apenas quinze pontos a serem visitados, nessa figura pode-se analisar que tem um menor indicio de cruzamento de rotas, tanto em relação a rotas de cores diferentes, ou seja, de veículos diferentes, como também em relação a cruzamentos em sua própria rota.

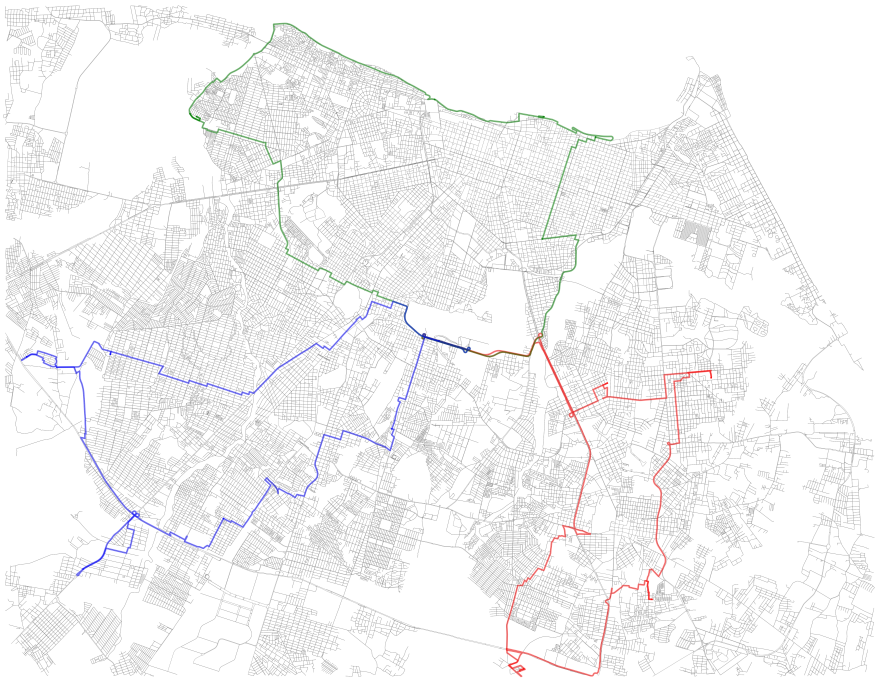

Figura 3 – Mapa Plotado de Fortaleza 45 pontos

Já na Figura 4, temos a cidade de Washington D.C., onde tem proporcionalmente, uma área parecida com a de Fortaleza, também testado com três veículos e quinze pontos, e como pode ser visto, as rotas não tiveram cruzamentos, porém algo que pode ser visto também, é que as rotas estão muito grandes para apenas três veículos.

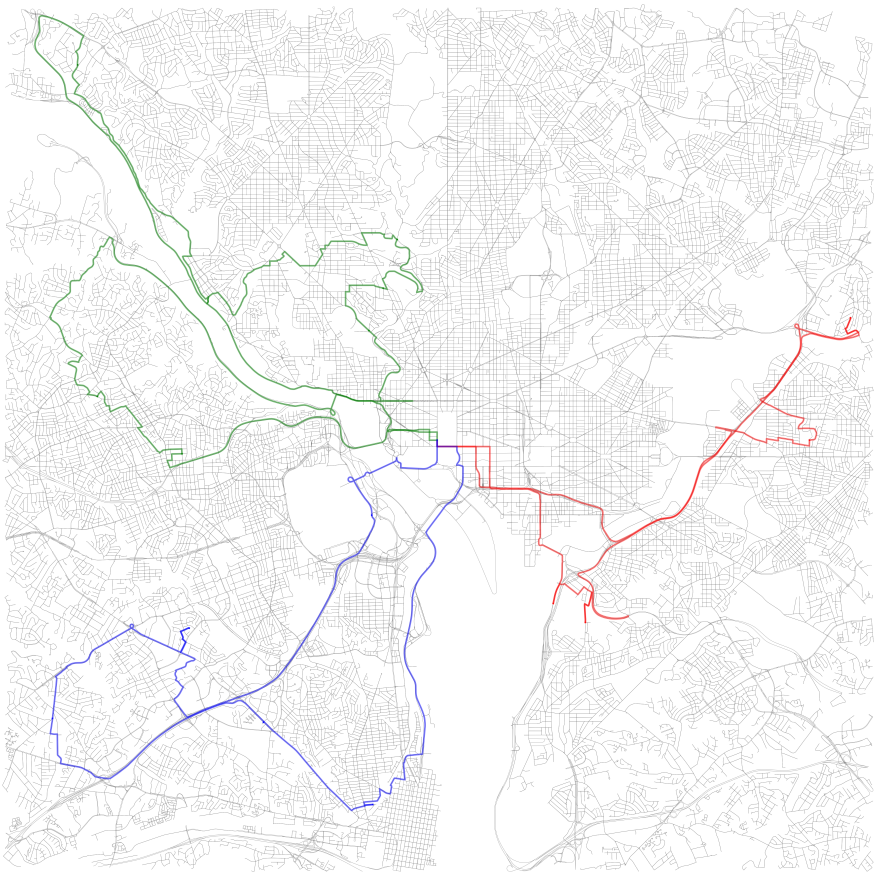

Figura 4 – Mapa Plotado de Washington 15 pontos

Na Figura 5, foi aumentado o número de pontos, passando assim a ser quarenta e cinco pontos a serem visitados, porém continuando com os mesmos três veículos, e o que pode-se notar é que, foi aumentado ainda mais o tamanho das rotas dos veículos, e também aumentou o número de cruzamento entre rotas de veículos diferentes.

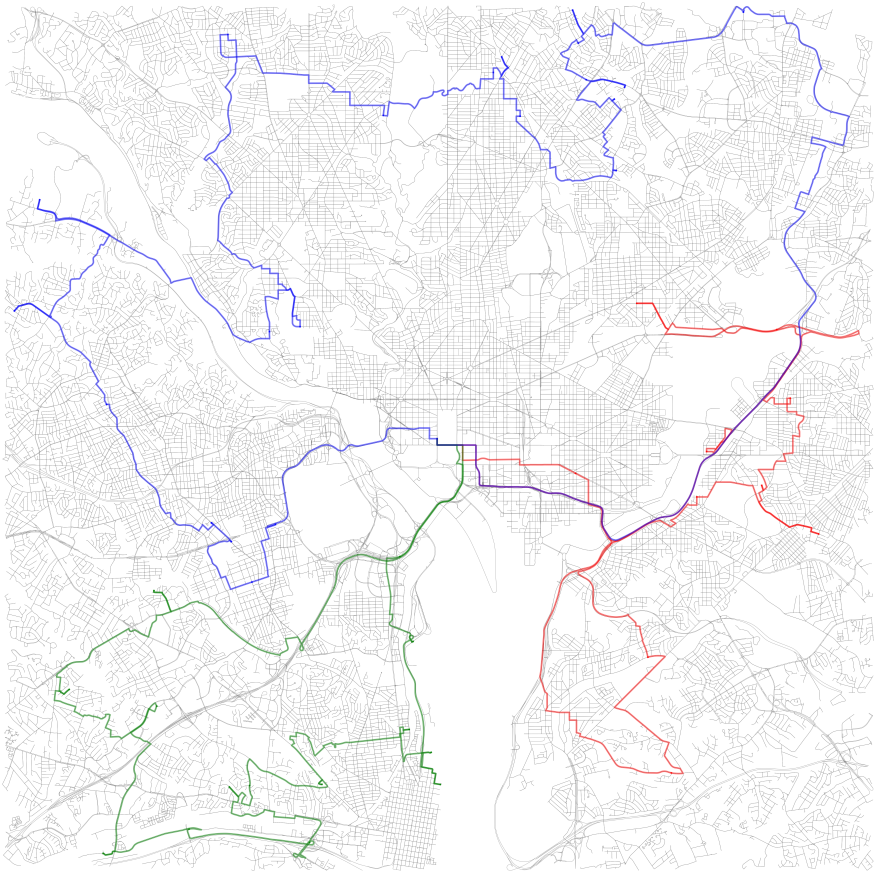

Figura 5 – Mapa Plotado de Washington 45 pontos

Por fim, na Figura 6, tendo os mesmo quarenta e cinco pontos da Figura 5, porém agora as rotas foram feitas com cinco veículos, e o que pode ser notado é que, adicionando mais veículos para o cumprimento das rotas, faz com que o tamanho médio das rotas seja diminuído, porém o tamanho total, somando todas as rotas seja maior, além de custar a atratividade visual das rotas, que como pode ser visto, temos diversos cruzamentos entre as rotas, e como pode ser visto, tanto essa figura como as outras, podemos ver que elas seguem o mesmo principio da figura de São Paulo, onde ele está dando prioridade em fazer o caminho das rotas por avenidas ao invés de de centros onde pode se encontrar mais engarrafamento, o que afetaria a janela de tempo a ser seguida.

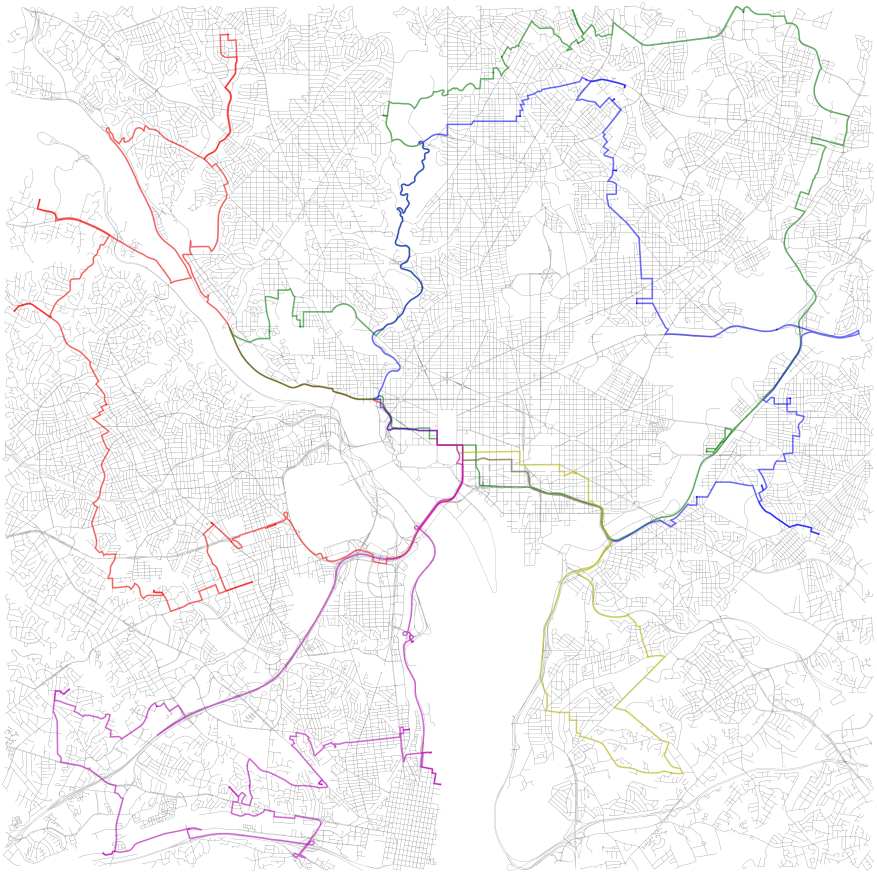

Figura 6 – Mapa Plotado de Washington 45 pontos com 5 veículos

Com isso pode-se ter uma noção inicial de como o *OSMnx* faz a plotagem dos mapas com as rotas adquiridas com os algoritmos da *OR-Tools*, ressaltando que foram utilizados os mesmos pontos e rotas que vão ser mostrados nas próximas seções, porém por meio de tabela para que seja mais didático verificar a diferença dos algoritmos para a resolução das rotas.

5.2 Início da Calibração

Para isso, como citado anteriormente, foi utilizado quatro cidades, São Paulo, Fortaleza, Washington e Madrid, onde em cada cidade foram utilizados quinze, trinta e quarenta e cinco pontos aleatórios, sem contar o deposito que é no centro de cada cidade. Nas tabelas a seguir tem cinco colunas, a primeira (Nº de Pontos) sendo o número de pontos a serem calculados, a (*Option*) que são os algoritmos utilizados que a *OR-Tools* disponibiliza, em seguida as colunas com o número de veículos utilizados, onde se tem o tempo total das rotas para aquele algoritmo, ou seja, um exemplo com três veículos, é a soma dos três veículos para aquela quantidade de pontos, e aquele algoritmo utilizado. Os algoritmos utilizados foram dois, o FSO e o LSO, onde ambos tem diversas opções dentre eles, além disso foi também feito a calibração de ambos os algoritmos, onde essa calibração nos testamos todas as opções de algoritmos, e apenas os melhores serão utilizados em todas as cidades, a cidade de São Paulo foi utilizada para fazer a

calibração por conta de ser a com a maior área, com cerca de 1.521 (mil quinhentos e vinte e um) quilômetros quadrados, primeiramente o *First Solutin Strategy* (FSS) como mostrado na Tabela 2 com quinze pontos, na Tabela 3 com trinta pontos e por fim na Tabela 4 com quarenta e cinco pontos. Depois foi feito a calibração com o LSO como mostrado na Tabela 5 com quinze pontos, na Tabela 6 com trinta pontos e por fim na Tabela 7 com quarenta e cinco pontos.

Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
15	PATH_CHEAPEST_ARC	1381min	1376min	1505min
15	PATH_MOST_CONSTRAINED_ARC	1420min	1376min	1505min
15	EVALUATOR_STRATEGY	NO RETURN	NO RETURN	NO RETURN
15	SAVINGS	NO RETURN	NO RETURN	NO RETURN
15	SWEEP	NO RETURN	NO RETURN	NO RETURN
15	CHRISTOFIDES	NO RETURN	NO RETURN	NO RETURN
15	ALL_UNPERFORMED	NO RETURN	NO RETURN	NO RETURN
15	BEST_INSERTION	NO RETURN	NO RETURN	NO RETURN
15	PARALLEL_CHEAPEST_INSERTION	NO RETURN	NO RETURN	NO RETURN
15	LOCAL_CHEAPEST_INSERTION	NO RETURN	NO RETURN	1505min
15	GLOBAL_CHEAPEST_ARC	1381min	1405min	1505min
15	LOCAL_CHEAPEST_ARC	1381min	TIME LIMIT	TIME LIMIT
15	FIRST_UNBOUND_MIN_VALUE	1381min	1409min	TIME LIMIT

Tabela 2 – Calibração São Paulo FSS 15 pontos

Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
30	PATH_CHEAPEST_ARC	1599min	1696min	1810min
30	PATH_MOST_CONSTRAINED_ARC	1619min	1571min	1645min
30	EVALUATOR_STRATEGY	NO RETURN	NO RETURN	NO RETURN
30	SAVINGS	1651min	1651min	1751min
30	SWEEP	NO RETURN	NO RETURN	NO RETURN
30	CHRISTOFIDES	NO RETURN	1664min	1759min
30	ALL_UNPERFORMED	NO RETURN	NO RETURN	NO RETURN
30	BEST_INSERTION	NO RETURN	NO RETURN	NO RETURN
30	PARALLEL_CHEAPEST_INSERTION	1557min	1656min	NO RETURN
30	LOCAL_CHEAPEST_INSERTION	NO RETURN	1644min	1710min
30	GLOBAL_CHEAPEST_ARC	1571min	1638min	1749min
30	LOCAL_CHEAPEST_ARC	TIME LIMIT	TIME LIMIT	TIME LIMIT
30	FIRST_UNBOUND_MIN_VALUE	TIME LIMIT	TIME LIMIT	TIME LIMIT

Tabela 3 – Calibração São Paulo FSS 30 pontos

Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
45	PATH_CHEAPEST_ARC	2008min	1993min	2143min
45	PATH_MOST_CONSTRAINED_ARC	1940min	2091min	2185min
45	EVALUATOR_STRATEGY	NO RETURN	NO RETURN	NO RETURN
45	SAVINGS	NO RETURN	1994min	2199min
45	SWEEP	NO RETURN	NO RETURN	NO RETURN
45	CHRISTOFIDES	NO RETURN	NO RETURN	NO RETURN
45	ALL_UNPERFORMED	NO RETURN	NO RETURN	NO RETURN
45	BEST_INSERTION	NO RETURN	NO RETURN	NO RETURN
45	PARALLEL_CHEAPEST_INSERTION	1921min	NO RETURN	2245min
45	LOCAL_CHEAPEST_INSERTION	NO RETURN	2128min	2151min
45	GLOBAL_CHEAPEST_ARC	1921min	2009min	2166min
45	LOCAL_CHEAPEST_ARC	1923min	TIME LIMIT	TIME LIMIT
45	FIRST_UNBOUND_MIN_VALUE	TIME LIMIT	TIME LIMIT	TIME LIMIT

Tabela 4 – Calibração São Paulo FSS 45 pontos

Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
15	GREEDY_DESCENT	1381min	1375min	1505min
15	GUIDED_LOCAL_SEARCH	1381min	1376min	1505min
15	SIMULATED_ANNEALING	1381min	1405min	1505min
15	TABU_SEARCH	1381min	1376min	1505min

Tabela 5 – Calibração São Paulo LSO 15 pontos

Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
30	GREEDY_DESCENT	1557min	1665min	1810min
30	GUIDED_LOCAL_SEARCH	1557min	1585min	1669min
30	SIMULATED_ANNEALING	1599min	1696min	1747min
30	TABU_SEARCH	1557min	1585min	1709min

Tabela 6 – Calibração São Paulo LSO 30 pontos

Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
45	GREEDY_DESCENT	1918min	2066min	2242min
45	GUIDED_LOCAL_SEARCH	1914min	2003min	2144min
45	SIMULATED_ANNEALING	2008min	2049min	2144min
45	TABU_SEARCH	1917min	2015min	2151min

Tabela 7 – Calibração São Paulo LSO 45 pontos

A calibração é necessária para que utilizemos apenas as opções de algoritmos mais eficazes, e foi escolhida três opções de cada algoritmo, primeiramente FSS como mostrado na Tabela 8 onde tem a soma total com cada opção de algoritmo com quinze, trinta e quarenta e cinco pontos, porém algumas opções não funcionaram, as nomeadas *NO RETURN* onde a *Google OR-Tools* não retornou nenhum valor, fazendo assim excluir elas automaticamente, pois em alguns casos funcionam e em outros casos não, já as nomeadas *TIME LIMIT* são as que extrapolaram o tempo limite de execução, onde foi colocado como trinta minutos no máximo, vale ressaltar que todas as outras tiveram no máximo cinco minutos de tempo de execução, com média de um ponto cinco minutos de média. Após isso tem apenas três que conseguiram rodar em todos os casos, como mostrado na Tabela 9 com três, quatro e cinco veículos, e na Tabela 8 somando todos os veículos.

OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
PATH_CHEAPEST_ARC	4988min	5065min	5458min
PATH_MOST_CONSTRAINED_ARC	4979min	5038min	5335min
EVALUATOR_STRATEGY	NO RETURN	NO RETURN	NO RETURN
SAVINGS	NO RETURN	NO RETURN	NO RETURN
SWEEP	NO RETURN	NO RETURN	NO RETURN
CHRISTOFIDES	NO RETURN	NO RETURN	NO RETURN
ALL_UNPERFORMED	NO RETURN	NO RETURN	NO RETURN
BEST_INSERTION	NO RETURN	NO RETURN	NO RETURN
PARALLEL_CHEAPEST_INSERTION	NO RETURN	NO RETURN	NO RETURN
LOCAL_CHEAPEST_INSERTION	NO RETURN	NO RETURN	NO RETURN
GLOBAL_CHEAPEST_ARC	4873min	5052min	5420min
LOCAL_CHEAPEST_ARC	TIME LIMIT	TIME LIMIT	TIME LIMIT
FIRST_UNBOUND_MIN_VALUE	TIME LIMIT	TIME LIMIT	TIME LIMIT

Tabela 8 – Calibração São Paulo FSS Final

OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
PATH_CHEAPEST_ARC	4988min	5065min	5458min
PATH_MOST_CONSTRAINED_ARC	4979min	5038min	5335min
GLOBAL_CHEAPEST_ARC	4873min	5052min	5420min

Tabela 9 – Calibração São Paulo FSS 3 Melhores

OPTION	TOTAL TIME
PATH_CHEAPEST_ARC	15511min
PATH_MOST_CONSTRAINED_ARC	15352min
GLOBAL_CHEAPEST_ARC	15345min

Tabela 10 – Calibração São Paulo FSS 3 Melhores Total

Agora com o LSO como mostrado na Tabela 11 onde tem a soma total com cada opção de algoritmo com quinze, trinta e quarenta e cinco pontos, porém aqui nenhuma teve *NO RETURN*

ou *TIME LIMIT*, com isso agora precisamos fazer a soma total com todos os veículos como mostrado na Tabela 12, por fim tem-se os três melhores como mostrado na Tabela 13 e na Tabela 14.

OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
GREEDY_DESCENT	4856min	5106min	5557min
GUIDED_LOCAL_SEARCH	4852min	4964min	5318min
SIMULATED_ANNEALING	4988min	5150min	5396min
TABU_SEARCH	4855min	4976min	5365min

Tabela 11 – Calibração São Paulo LSO Final

GREEDY_DESCENT	15519min
GUIDED_LOCAL_SEARCH	15134min
SIMULATED_ANNEALING	15534min
TABU_SEARCH	15196min

Tabela 12 – Calibração São Paulo LSO Final Total

OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
GREEDY_DESCENT	4856min	5106min	5557min
GUIDED_LOCAL_SEARCH	4852min	4964min	5318min
TABU_SEARCH	4855min	4976min	5365min

Tabela 13 – Calibração São Paulo LSO 3 Melhores

GREEDY_DESCENT	15519min
GUIDED_LOCAL_SEARCH	15134min
TABU_SEARCH	15196min

Tabela 14 – Calibração São Paulo LSO 3 Melhores Total

5.2.1 Resultado das Cidades

Começando com o algortimo FSS , começando em São Paulo, que foi a cidade ao qual fizemos a calibração anteriormente, podemos ver que esses algoritmos em alguns casos fazer as mesmas rotas, e por conta disso em alguns casos tem o mesmo resultado, também podemos ver que o primeiro algoritmo *PATH_CHEAPEST_ARC* que é o modelo padrão do *Google OR-Tools* ele é o melhor quando se trata de poucos pontos, que no caso seria a instância de 15 pontos, quanto pulamos para a instância de 30 pontos podemos ver que ela começa a decair a sua eficiência principalmente quando colocamos mais veículos, porém quando chegamos a instância de 45 veículos, podemos ver que os três algoritmos já ficam mais próximos, oscilando com base na quantidade de veículos, como mostra na Tabela 15.

CIDADE	Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
SÃO PAULO	15	PATH_CHEAPEST_ARC	1381min	1376min	1505min
	15	PATH_MOST_CONSTRAINED_ARC	1420min	1376min	1505min
	15	GLOBAL_CHEAPEST_ARC	1381min	1405min	1505min
	30	PATH_CHEAPEST_ARC	1599min	1696min	1810min
	30	PATH_MOST_CONSTRAINED_ARC	1619min	1571min	1645min
	30	GLOBAL_CHEAPEST_ARC	1571min	1638min	1749min
	45	PATH_CHEAPEST_ARC	2008min	1993min	2143min
	45	PATH_MOST_CONSTRAINED_ARC	1940min	2091min	2185min
	45	GLOBAL_CHEAPEST_ARC	1921min	2009min	2166min

Tabela 15 – Tabela São Paulo FSS Final

Agora na cidade de Fortaleza por ser uma cidade relativamente muito menor do que São Paulo, podemos ver que em quase todos os casos tem os mesmos caminhos como podemos ver pelos tempos, isso se deve por conta da área da cidade, pois como ela é bem menor do que São Paulo, também tem menas opções de caminhos a fazer, como podemos ver na 16 na instância com 15 pontos a unica diferença se da por conta do *PATH_MOST_CONSTRAINED_ARC* com cinco veículos, na instância de 30 pontos podemos ver novamente o que foi citado anteriormente com relação ao primeiro algoritmo *PATH_CHEAPEST_ARC*, que vai perdendo sua eficiência quando vamos adicionando mais pontos.

CIDADE	Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
FORTALEZA	15	PATH_CHEAPEST_ARC	276min	308min	308min
	15	PATH_MOST_CONSTRAINED_ARC	276min	308min	328min
	15	GLOBAL_CHEAPEST_ARC	276min	308min	308min
	30	PATH_CHEAPEST_ARC	372min	424min	425min
	30	PATH_MOST_CONSTRAINED_ARC	362min	401min	419min
	30	GLOBAL_CHEAPEST_ARC	376min	389min	441min
	45	PATH_CHEAPEST_ARC	491min	500min	499min
	45	PATH_MOST_CONSTRAINED_ARC	481min	481min	542min
	45	GLOBAL_CHEAPEST_ARC	494min	481min	519min

Tabela 16 – Tabela Fortaleza FSS Final

Na cidade de Washington D.C. como é uma cidade de menor tamanho comparado a São Paulo, pode-se ver um cenário parecido que ocorreu em Fortaleza, porém Washington por ter maior densidade de ruas em um mesmo espaço tem mais rotas que podem ser criadas, e como podemos ver elas se diferenciam por alguns minutos como podemos ver na Tabela 17.

CIDADE	Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
WASHINGTON	15	PATH_CHEAPEST_ARC	215min	223min	247min
	15	PATH_MOST_CONSTRAINED_ARC	222min	223min	247min
	15	GLOBAL_CHEAPEST_ARC	215min	224min	247min
	30	PATH_CHEAPEST_ARC	331min	353min	403min
	30	PATH_MOST_CONSTRAINED_ARC	331min	350min	366min
	30	GLOBAL_CHEAPEST_ARC	341min	350min	366min
	45	PATH_CHEAPEST_ARC	409min	428min	464min
	45	PATH_MOST_CONSTRAINED_ARC	405min	434min	486min
	45	GLOBAL_CHEAPEST_ARC	406min	453min	437min

Tabela 17 – Tabela Washington FSS Final

Por fim na cidade de Madrid os resultados não se diferem muito dos obtidos previamente

em Fortaleza e em Washington, onde os algoritmos não tem muitas ruas como opção para realizarem as suas rotas como pode ser visto na Tabela 18.

Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
15	PATH_CHEAPEST_ARC	281min	306min	344min
15	PATH_MOST_CONSTRAINED_ARC	281min	305min	344min
15	GLOBAL_CHEAPEST_ARC	281min	320min	346min
30	PATH_CHEAPEST_ARC	399min	412min	442min
30	PATH_MOST_CONSTRAINED_ARC	408min	407min	425min
30	GLOBAL_CHEAPEST_ARC	387min	425min	445min
45	PATH_CHEAPEST_ARC	495min	509min	529min
45	PATH_MOST_CONSTRAINED_ARC	490min	513min	527min
45	GLOBAL_CHEAPEST_ARC	506min	508min	519min

Tabela 18 – Tabela Madrid FSS Final

Já os algoritmos de LSO, começando novamente com São Paulo que foi a cidade que fizemos a calibração, podemos ver que os algoritmos LSO são bastante parecidos principalmente quando tem poucas instâncias como é o caso com a instância de 15 pontos, onde desde três veículos até cinco veículos, todos fazer as mesmas rotas, como podemos ver na Tabela 19, porém como podemos ver, quando vamos adicionando mais instâncias e testando elas com mais veículos, podemos ver esses se divergindo um dos outros, onde o *GUIDED_LOCAL_SEARCH* que foi dado pelo *Google OR-Tools* como o que geralmente se sai melhor com roteamento de veículos, ele foi realmente o melhor nos casos em que tem maior quantidade de instâncias e veículos, e logo após ele vem o *TABU_SEARCH*, e por fim o pior dos três o *GREEDY_DESCENT*.

CIDADE	Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
SÃO PAULO	15	GREEDY_DESCENT	1381min	1375min	1505min
	15	GUIDED_LOCAL_SEARCH	1381min	1376min	1505min
	15	TABU_SEARCH	1381min	1376min	1505min
	30	GREEDY_DESCENT	1557min	1665min	1810min
	30	GUIDED_LOCAL_SEARCH	1557min	1585min	1669min
	30	TABU_SEARCH	1557min	1585min	1709min
	45	GREEDY_DESCENT	1918min	2066min	2242min
	45	GUIDED_LOCAL_SEARCH	1914min	2003min	2144min
	45	TABU_SEARCH	1917min	2015min	2151min

Tabela 19 – Tabela São Paulo LSO Final

Na cidade de Fortaleza acontece a mesma coisa que aconteceu anteriormente, quando vamos colocando mais intâncias e mais veículos, vemos eles se diferenciando, com a mesma ordem de melhor para pior, *GUIDED_LOCAL_SEARCH* seguido por *TABU_SEARCH*, e por fim o pior dos três o *GREEDY_DESCENT* como mostrado na Tabela 20.

CIDADE	Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
FORTALEZA	15	GREEDY_DESCENT	276min	308min	342min
	15	GUIDED_LOCAL_SEARCH	276min	308min	342min
	15	TABU_SEARCH	276min	308min	342min
	30	GREEDY_DESCENT	372min	424min	425min
	30	GUIDED_LOCAL_SEARCH	362min	380min	408min
	30	TABU_SEARCH	362min	380min	408min
	45	GREEDY_DESCENT	491min	515min	530min
	45	GUIDED_LOCAL_SEARCH	464min	479min	500min
	45	TABU_SEARCH	482min	481min	499min

Tabela 20 – Tabela Fortaleza LSO Final

Em Washington D.C. como citado nos algoritmos de FSS, apesar de ter um tamanho menor que São Paulo, ele tem maior densidade de ruas e como podemos ver na Tabela 21 desde inicio o *GREEDY_DESCENT* já demostra ser o que pega as maiores rotas em comparação com os outros dois, e acaba seguindo a mesma ordem dos outros, o melhor sendo *GUIDED_LOCAL_SEARCH* seguido por *TABU_SEARCH*.

CIDADE	Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
WASHINGTON	15	GREEDY_DESCENT	215min	223min	254min
	15	GUIDED_LOCAL_SEARCH	215min	223min	253min
	15	TABU_SEARCH	215min	223min	253min
	30	GREEDY_DESCENT	331min	358min	410min
	30	GUIDED_LOCAL_SEARCH	331min	350min	366min
	30	TABU_SEARCH	331min	350min	366min
	45	GREEDY_DESCENT	409min	455min	466min
	45	GUIDED_LOCAL_SEARCH	392min	415min	428min
	45	TABU_SEARCH	403min	445min	448min

Tabela 21 – Tabela Washington LSO Final

Por fim Madrid que como podemos ver na Tabela 22 o *GREEDY_DESCENT* vem demonstrando novamente ser o pior e o *GUIDED_LOCAL_SEARCH* demonstrando ser o melhor dos três.

CIDADE	Nº DE PONTOS	OPTION	3 VEÍCULOS	4 VEÍCULOS	5 VEÍCULOS
MADRID	15	GREEDY_DESCENT	306min	326min	346min
	15	GUIDED_LOCAL_SEARCH	301min	322min	345min
	15	TABU_SEARCH	301min	326min	345min
	30	GREEDY_DESCENT	410min	433min	436min
	30	GUIDED_LOCAL_SEARCH	399min	409min	431min
	30	TABU_SEARCH	400min	409min	431min
	45	GREEDY_DESCENT	522min	525min	529min
	45	GUIDED_LOCAL_SEARCH	506min	503min	521min
	45	TABU_SEARCH	514min	525min	521min

Tabela 22 – Tabela Madrid LSO Final

5.3 Métricas

Com isso podemos agora podemos utilizar nossas duas métricas, a de Compactação para vermos como o *Google OR-Tools* está lidando com a atratividade visual das rotas, e com a métrica

de Cruzamento para vermos como o *Google OR-Tools* está lidando com possíveis interseções, além disso as instâncias estão nomeadas nas tabelas como: (nome da cidade, quantidade de veículos, quantidade de pontos), como por exemplo SP_4V_30, está dizendo que é a instância São Paulo com 4 veículos e 30 pontos.

Para medida de comparação a Métrica de Compactação quanto mais próximo de 1 mais ela está visualmente atrativa, quanto mais próximo de 0 menos visualmente atrativa ela é. Já a Métrica de Cruzamento, levando em conta instâncias de 15 pontos podemos ter de 6 interseções até 119 interseções, já na instância de 30 pontos podemos ter de 20 até 464 interseções, e por fim com 45 instâncias tem de 44 até 990 possíveis interseções, vale ressaltar que isso é levando em conta uma rota miníma até uma rota máxima, onde a mínima seria uma rota em que tem todas as rotas com a mesma quantidade de clientes, e a máxima onde apenas uma rota comporta todos os clientes.

Tendo isso em mente na primeira Tabela 23 tem todas as instâncias de 15 pontos utilizando algoritmo FSS, como podemos ver, em todos os casos a Métrica de Compactação está dando zero, isso pode nos mostrar que o *Google OR-Tools* esta dando importância apenas no segmento da janela de tempo, sem se importar com a atratividade visual da rota, além disso podemos ver que no tem quatro interseções/cruzamento que ocorre em São Paulo com três veículos, considerando as interseções de 15 instâncias onde mostrado que tem 6 cálculos de interseções, 4 delas se cruzam tanto no primeiro algoritmo como no terceiro algoritmo, mostrando assim que nesses casos tem 66% de interseções, isso pode ser pelo fato de São Paulo ser uma cidade muito grande, e com pontos espalhados nele, quando vamos calcular a Cruzamento ele acaba pegando pontos que na rota em si elas não se cruzam, porém quando vamos fazer a mesma rota com linhas retas elas acabam se cruzando.

	PATH_CHEAPEST_ARC		PATH_MOST_CONSTRAINED_ARC		GLOBAL_CHEAPEST_ARC	
INSTÂNCIA	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO
SP_3V_15	0.0	4	0.0	1	0.0	4
SP_4V_15	0.0	2	0.0	0	0.0	2
SP_5V_15	0.0	1	0.0	1	0.0	1
F_3V_15	0.0	0	0.0	0	0.0	0
F_4V_15	0.0	0	0.0	0	0.0	0
F_5V_15	0.0	0	0.0	0	0.0	1
W_3V_15	0.0	2	0.0	3	0.0	1
W_4V_15	0.0	1	0.0	1	0.0	0
W_5V_15	0.0	0	0.0	0	0.0	0
M_3V_15	0.0	3	0.0	3	0.0	3
M_4V_15	0.0	0	0.0	0	0.0	1
M_5V_15	0.0	0	0.0	0	0.0	0

Tabela 23 – Métricas FSS com 15 instâncias/pontos

Já Tabela 24 em que tem ainda 15 instâncias porém agora com LSO, podemos ver que ela repete o mesmo caso da anterior, porém podemos ver que tem uma com cinco interseções, mais especificamente na SP_4V_15 (São Paulo 4 Veículos 15 pontos) com algoritmo *TABU_SEARCH*, onde tem essas cinco interseções, enquanto na *GREEDY_DESCENT* que tínhamos um tempo maior de roda não tem nenhuma Cruzamento, isso pode ser um indicio de que a

GREEDY_DESCENT se importa mais com evitar interseções do que com o tamanho de rota em si, diferente do *TABU_SEARCH* e *GUIDED_LOCAL_SEARCH*

	GREEDY_DESCENT		GUIDED_LOCAL_SEARCH		TABU_SEARCH	
INSTÂNCIA	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO
SP_3V_15	0.0	4	0.0	4	0.0	4
SP_4V_15	0.0	0	0.0	2	0.0	5
SP_5V_15	0.0	1	0.0	1	0.0	1
F_3V_15	0.0	0	0.0	0	0.0	0
F_4V_15	0.0	0	0.0	0	0.0	0
F_5V_15	0.0	0	0.0	0	0.0	0
W_3V_15	0.0	2	0.0	2	0.0	2
W_4V_15	0.0	1	0.0	1	0.0	1
W_5V_15	0.0	1	0.0	1	0.0	1
M_3V_15	0.0	1	0.0	3	0.0	3
M_4V_15	0.0	0	0.0	0	0.0	0
M_5V_15	0.0	0	0.0	0	0.0	0

Tabela 24 – Métricas LSO com 15 instâncias/pontos

Agora para instâncias de 30 pontos, onde tínhamos de 20 até 464 cálculos interseções, vemos que tem menos de 50% de interseções como mostra na Tabela 25 com FSS e também na Tabela 26 com LSO, vemos em ambas as tabelas que apenas de Fortaleza, Washington D.C. e Madrid serem menores que São Paulo, eles tiveram menas interseções, isso é muito por conta de que eles muitas vezes fazem o mesmo caminho e acabam com resultados parecidos, caso fosse calculado interseções não apenas na mesma rota, como estamos fazendo, e sim entre rotas, o vamos de interseções dessas cidades seriam relativamente maiores, pois como iriamos comparar uma rota com outra, e essas cidades tendem a ter rotas bastante similares senão idênticas, teríamos diversas interseções encontradas. E como podemos ver novamente, mesmo quando aumentamos o número de pontos, a Métrica de Compactação ainda está dando zero, mostrando mais uma vez, que o *Google OR-Tools* está realmente dando sua importância total em cumprir a janela de tempo, e descartando a atratividade visual.

	PATH_CHEAPEST_ARC		PATH_MOST_CONSTRAINED_ARC		GLOBAL_CHEAPEST_ARC	
INSTÂNCIA	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO
SP_3V_30	0.0	8	0.0	6	0.0	8
SP_4V_30	0.0	5	0.0	8	0.0	6
SP_5V_30	0.0	3	0.0	3	0.0	5
F_3V_30	0.0	3	0.0	1	0.0	3
F_4V_30	0.0	2	0.0	2	0.0	3
F_5V_30	0.0	2	0.0	2	0.0	3
W_3V_30	0.0	1	0.0	1	0.0	1
W_4V_30	0.0	1	0.0	0	0.0	0
W_5V_30	0.0	0	0.0	0	0.0	1
M_3V_30	0.0	5	0.0	2	0.0	1
M_4V_30	0.0	1	0.0	2	0.0	3
M_5V_30	0.0	3	0.0	2	0.0	2

Tabela 25 – Métricas FSS com 30 instâncias/pontos

	GREEDY_DESCENT		GUIDED_LOCAL_SEARCH		TABU_SEARCH	
INSTÂNCIA	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO
SP_3V_30	0.0	8	0.0	5	0.0	5
SP_4V_30	0.0	5	0.0	5	0.0	5
SP_5V_30	0.0	3	0.0	1	0.0	2
F_3V_30	0.0	3	0.0	2	0.0	2
F_4V_30	0.0	2	0.0	1	0.0	1
F_5V_30	0.0	2	0.0	1	0.0	1
W_3V_30	0.0	1	0.0	1	0.0	1
W_4V_30	0.0	0	0.0	0	0.0	0
W_5V_30	0.0	0	0.0	1	0.0	2
M_3V_30	0.0	4	0.0	1	0.0	4
M_4V_30	0.0	3	0.0	1	0.0	0
M_5V_30	0.0	2	0.0	1	0.0	2

Tabela 26 – Métricas LSO com 30 instâncias/pontos

Agora para as instâncias de 45 pontos podemos concluir com certeza de que o *Google OR-Tools* não esta se preocupando com a atratividade visual das rotas, e sim apenas com a janela de tempo, ja que em todas as instâncias com 15, 30 e 45 pontos, todos os resultados de compactação deram zero. Na parte de Cruzamento, como podemos ver o que era esperado aconteceu, que é quando aumentamos o número de pontos, irá por consequência aumentar o número de interseções, e como citado anteriormente com 45 pontos é calculado cerca de 44 até 990 instâncias, porém o maior número que houve de interseções foi de 10, na Tabela 27 com SP_3V_45 na *PATH_CHEAPEST_ARC* e na SP_4V_45 com *PATH_MOST_CONSTRAINED_ARC*, já na Tabela 28 se da por conta do SP_3V_45 com *GREEDY_DESCENT*, o que pode mudar o argumento antes colocado de que o *GREEDY_DESCENT* sacrificava o tempo da rota para priorizar a minimização das interseções.

	PATH_CHEAPEST_ARC		PATH_MOST_CONSTRAINED_ARC		GLOBAL_CHEAPEST_ARC	
INSTÂNCIA	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO
SP_3V_45	0.0	10	0.0	6	0.0	9
SP_4V_45	0.0	7	0.0	10	0.0	3
SP_5V_45	0.0	6	0.0	3	0.0	7
F_3V_45	0.0	4	0.0	2	0.0	8
F_4V_45	0.0	2	0.0	2	0.0	7
F_5V_45	0.0	4	0.0	3	0.0	4
W_3V_45	0.0	1	0.0	1	0.0	5
W_4V_45	0.0	4	0.0	6	0.0	7
W_5V_45	0.0	0	0.0	2	0.0	2
M_3V_45	0.0	8	0.0	9	0.0	6
M_4V_45	0.0	6	0.0	7	0.0	8
M_5V_45	0.0	4	0.0	5	0.0	5

Tabela 27 – Métricas FSS com 45 instâncias/pontos

	GREEDY_DESCENT		GUIDED_LOCAL_SEARCH		TABU_SEARCH	
INSTÂNCIA	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO	COMPACTAÇÃO	CRUZAMENTO
SP_3V_45	0.0	10	0.0	8	0.0	6
SP_4V_45	0.0	4	0.0	7	0.0	5
SP_5V_45	0.0	6	0.0	6	0.0	5
F_3V_45	0.0	4	0.0	2	0.0	5
F_4V_45	0.0	1	0.0	2	0.0	0
F_5V_45	0.0	3	0.0	2	0.0	3
W_3V_45	0.0	1	0.0	3	0.0	0
W_4V_45	0.0	0	0.0	1	0.0	0
W_5V_45	0.0	1	0.0	3	0.0	0
M_3V_45	0.0	8	0.0	7	0.0	7
M_4V_45	0.0	6	0.0	5	0.0	5
M_5V_45	0.0	3	0.0	5	0.0	5

Tabela 28 – Métricas LSO com 45 instâncias/pontos

6 CONCLUSÃO

Com tudo dito acima, nos vimos que a utilização do *Google Or-Tools* é bem ampla pois tem diversos tipos de Problema de Roteamento de Veículos, além de também ter diversos tipos de algoritmos para a execução das rotas, além de também poder escolher a quantidade de veículos, o tempo máximo de cada rota, tempo para a espera do cliente, entre outros. Porém vale ressaltar que a média de tempo de execução do *Google OR-Tools* é de um minuto quando usado o método padrão, e cerca de cinco minutos no pior dos casos com 45 pontos, então como forma de tirar duvida, foi feito um teste com 100 e 200 pontos, com o *PATH_CHEAPEST_ARC* teve um tempo de 4 minutos com 100 pontos e 6 minutos com 200 pontos.

E como vimos nas métricas, nos podemos verificar qual o tipo de algoritmo irá se adequar melhor dependendo da situação, essas duas métricas são apenas alguns tipos de métricas que podem ser feito para validar a capacitação da ferramenta *Google OR-Tools*, tendo muitas outras, como citado na seção de métricas, o modo que a Métrica de Cruzamento foi calculado leva em conta apenas uma rota por vez, e verifica se aquela rota tem cruzamento, porém pode ser calculada também em comparação a outras rotas quando tem mais de um veículo para fazer as rotas de uma matriz de duração/distância.

Com tudo dito acima, nos vimos que utilizando um algoritmo PRV para rotas é uma grande vantagem, para a empresa, pois vai diminuir os custos das viagens, por diminuir a quilometragem das rotas, e por conta disso diminuir os custos de manutenção da frota, poder entregar os produtos para seus clientes no tempo desejado fazendo assim que aumente a satisfação do cliente, e também para o funcionário que ira realizar a rota, pois ele já sabe todo o caminho a percorrer, sem duvidas com relação a rota, e também uma certeza de que irá conseguir cumprir a rota sem grandes problemas, e assim voltar no horário em que acaba o expediente.

E claro, o algoritmo não ser apenas utilizado apenas para empresas de entrega como foi citado nos exemplos acima, mas também pra qualquer tipo de problema que envolva rotas onde um veículo pode passar, como transporte escolar, transporte industrial para levar os funcionários de certos pontos da cidade para a sede da industria, veículos que oferecem serviços como por exemplo de energia onde eles tem uma lista de endereços que eles devem passar e averiguar a

situação e resolver o problema, entre outros. Porém cada um tem um problema em diferente de um para o outro, então é necessário algumas alterações para ser aplicada em áreas diferentes.

Como Trabalhos futuros seria o aumento de instâncias, como por exemplo 100, 200, 500 clientes, e também o teste dessas instâncias com todos os algoritmos, para ver se aqueles que não conseguiram retornar nada e aqueles que extrapolaram o tempo de execução funcionam quando tempos mais pontos no mapa, além de tentar comparar uma rota de uma empresa e compara-la usando o *Google OR-Tools* vendo o quanto o PRVJT é mais eficiente do que uma rota normalmente feita por humanos.

Referências

ABDEL-BASSET, M.; ABDEL-FATAH, L.; SANGAIAH, A. K. Metaheuristic algorithms: A comprehensive review. **Computational intelligence for multimedia big data on the cloud with engineering applications**, Elsevier, p. 185–231, 2018.

BOEING, G. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. **Computers, Environment and Urban Systems**, Elsevier, v. 65, p. 126–139, 2017.

BRAEKERS, K.; RAMAEKERS, K.; NIEUWENHUYSE, I. V. The vehicle routing problem: State of the art classification and review. **Computers & Industrial Engineering**, Elsevier, v. 99, p. 300–313, 2016.

KIM, B.-I.; KIM, S.; SAHOO, S. Waste collection vehicle routing problem with time windows. **Computers & Operations Research**, Elsevier, v. 33, n. 12, p. 3624–3642, 2006.

KIRCI, P. On the performance of tabu search algorithm for the vehicle routing problem with time windows. In: IEEE. **2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)**. [S.l.], 2016. p. 351–354.

LAGUNA, M. Tabu search. In: **Handbook of heuristics**. [S.l.]: Springer, 2018. p. 741–758.

MARTÍ, R.; REINELT, G. Heuristic methods. In: **Exact and Heuristic Methods in Combinatorial Optimization**. [S.l.]: Springer, 2022. p. 27–57.

NAZARI, M. et al. Reinforcement learning for solving the vehicle routing problem. **arXiv preprint arXiv:1802.04240**, 2018.

OpenStreetMap contributors. **Planet dump retrieved from https://planet.osm.org** . 2017. <https://www.openstreetmap.org>.

PERRON, L.; FURNON, V. **OR-Tools**. 2019–7–19. Disponível em: <https://developers.google.com/optimization/>.

ROCHA, D. et al. Visual attractiveness in vehicle routing via bi-objective optimization. **Computers I& Operations Research**, v. 137, p. 105507, 2022. ISSN 0305-0548. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054821002513>.

ROSSIT, D. G. et al. Visual attractiveness in routing problems: A review. **Computers & Operations Research**, Elsevier, v. 103, p. 13–34, 2019.

TANEL, A. et al. Capacitated vehicle routing problem with time windows. In: **Digitizing Production Systems**. [S.l.]: Springer, 2022. p. 653–664.

TOTH, P.; VIGO, D. **Vehicle routing: problems, methods, and applications**. [S.l.]: SIAM, 2014.

UCHOA, E. et al. New benchmark instances for the capacitated vehicle routing problem. **European Journal of Operational Research**, Elsevier, v. 257, n. 3, p. 845–858, 2017.

YU, B. et al. k-nearest neighbor model for multiple-time-step prediction of short-term traffic condition. **Journal of Transportation Engineering**, American Society of Civil Engineers, v. 142, n. 6, p. 04016018, 2016.