Roteirizador

Roteirizador criado com ortools e gmaps para corridas de moto e bicicletas na cidade de São Paulo

Vamos utilizar tecnicas de VRP para desenvolver essa solução. Algumas variações são:

- Problema de roteamento de veículos capacitados (PRVC)
- Problema de roteamento de veículos com janela de tempo (PRVJT)
- Problema de roteamento de veículos com coleta e entrega
- Problema de roteamento de veículos com múltiplos depósitos
- Problema de roteamento de veículos periódico (PRVP)
- Problema de roteamento de veículos periódico com janela de tempo
- Problema de roteamento de veículos com entregas particionadas

Instalando e importando bibliotecas

```
%%capture
!pip install ortools
```

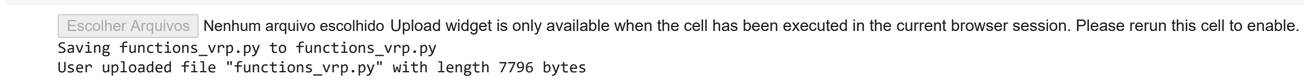
```
%%capture
!pip install gmaps;
```

Importando bibliotecas necessárias

```
from sklearn.neighbors import DistanceMetric

from google.colab import files
from google.colab import output

import os
from math import radians

import pandas as pd
import numpy as np

from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp

import gmaps
import random
import time

output.enable_custom_widget_manager()
```

Arquivo de funções personalizadas para o vrp

```
funcoes_vrp = files.upload()

for fn in funcoes_vrp.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(funcoes_vrp[fn])))
```




```
Saving functions_vrp.py to functions_vrp.py
User uploaded file "functions_vrp.py" with length 7796 bytes
```

```
import functions_vrp as fvrp
```

Configurando o Google Maps

```
gmaps.configure(api_key = fvrp.google_maps_api_key())
```

Entregas

Carregando o arquivo de entregas

```
# código para subir arquivo durante a execução
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Escolher Arquivos Nenhum arquivo escolhido Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Lendo os dados de entregas

```
# lendo os dados
base = pd.read_excel('/content/EntregasNormalizada.xlsx', sheet_name='EntregasNormalizadas')
data = base.copy()
```

```
data.head()
```

	Cod	Nome	Endereço	Latitude	Longitude	SKU	Peso (quilos)
0	or-01	CD Villa Lobos	Av. Manuel Bandeira, 360 - Vila Leopoldina, Sã...	-23.540787	-46.73362	0	0
1	des-01	Pão de Açúcar	R. Teodoro Sampaio, 1933 - Pinheiros, São Paul...	-23.563480	-46.68722	55777	2
2	des-02	Rockambole	R. Belmiro Braga, 119 - Pinheiros, São Paulo -...	-23.559540	-46.68738	51003	4
3	des-03	Bar Moela	Rua Cardeal Arcoverde, 2320 - Pinheiros, São P...	-23.564580	-46.69298	75671	4
4	des-04	Oba Hortifruti FARM	R. Teodoro Sampaio, 1424 - Pinheiros, São Paul...	-23.560720	-46.68347	54155	2

Calculando a matriz de distâncias

```
data['Latitude'] = np.radians(data['Latitude'])
data['Longitude'] = np.radians(data['Longitude'])

dist = DistanceMetric.get_metric('haversine')
distance_matrix = pd.DataFrame(dist.pairwise(data[['Latitude','Longitude']].to_numpy())*6373)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_distance_metric.py:14: FutureWarning: sklearn.neighbors.DistanceMetric has been moved to sklearn.metrics.DistanceMetric in 1.0. This import path will be removed in 1.3
  category=FutureWarning,
```

```
distance_matrix.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.000000	5.362358	5.155688	4.916808	5.573520	4.661585	4.950413	5.067510	5.241548	4.976538	4.938076	4.754533	5.438756	5.575333	5.762310	7.533732	7.742534	8.069212	7.790750	7.702569
1	5.362358	0.000000	0.438549	0.599870	0.490333	0.708800	0.455886	0.479214	0.716080	0.878411	1.064771	0.985586	1.325361	1.488773	0.886221	2.688731	2.697532	3.117091	2.862744	2.917922
2	5.155688	0.438549	0.000000	0.800163	0.419706	0.556660	0.273166	0.088356	0.321714	0.439865	0.626441	0.564450	0.952935	1.133541	0.713160	2.619667	2.702276	3.094722	2.826482	2.836444
3	4.916808	0.599870	0.800163	0.000000	1.060407	0.540332	0.573686	0.766622	1.119694	1.142443	1.318295	1.131136	1.747270	1.931395	1.445154	3.287363	3.295060	3.716411	3.462595	3.515637
4	5.573520	0.490333	0.419706	1.060407	0.000000	0.961341	0.662893	0.508016	0.435863	0.707020	0.855013	0.900697	0.915506	1.053840	0.395928	2.245244	2.299617	2.701970	2.438325	2.468847

Lendo os veículos disponíveis

```
# lendo os dados
veiculos = pd.read_excel('/content/EntregasNormalizada.xlsx', sheet_name='Veículos Disponíveis')
```

```
veiculos.head()
```

	Veiculos	Capacidade (quilos)	Qtd disponivel
0	Bike	2	1

Roterizador

Definindo variáveis iniciais para o modelo

```
# quantidade de veículos disponíveis
num_veiculos = veiculos['Qtd disponivel'].sum()

# capacidade dos veículos
capacidade_veiculos = list(veiculos['Capacidade (quilos)'])

# demanda, peso dos itens para serem entregues
demanda = list(data['Peso (quilos)'])
```

Criando os dados do modelo

```
# dados
model_data = fvrp.criando_modelo(distance_matrix, demanda, capacidade_veiculos)

# variáveis do modelo
manager = pywrapcp.RoutingIndexManager(len(model_data['distance_matrix']), model_data['num_vehicles'], model_data['depot'])
routing = pywrapcp.RoutingModel(manager)
```

Rodando o modelo para obter a solução

```
solucao = fvrp.solver_rotas(model_data, manager, routing, 10, pywrapcp, routing_enums_pb2)
```

Obtendo as rotas, caso haja solução

```
if solucao:
  rotas = fvrp.obtendo_rotas(10, manager, routing, solucao)
else:
  print("sem solução")
```

Imprimindo a roterização, caso haja solução

```
if solucao:
  fvrp.imprimindo_solucao(model_data, manager, routing, solucao)
else:
  print("sem solução")
```

```
Pontos de entregas não realizados: 2 6 8 17 18 19
Rota do veículo 0:
Ponto 0 -> Ponto 1 (descarregar: 2 quilos)
Distância da rota: 10m
Carga Total: 2

Rota do veículo 1:
Ponto 0 -> Ponto 4 (descarregar: 2 quilos)
Distância da rota: 10m
Carga Total: 2

Rota do veículo 2:
Ponto 0 -> Ponto 12 (descarregar: 2 quilos) -> Ponto 7 (descarregar: 2 quilos)
Distância da rota: 10m
Carga Total: 4

Rota do veículo 3:
Ponto 0 -> Ponto 14 (descarregar: 3 quilos) -> Ponto 13 (descarregar: 2 quilos)
Distância da rota: 10m
Carga Total: 5

Rota do veículo 4:
Ponto 0 -> Ponto 3 (descarregar: 4 quilos)
Distância da rota: 8m
Carga Total: 4

Rota do veículo 5:
```

```
Ponto 0 -> Ponto 5 (descarregar: 3 quilos)
Distância da rota: 8m
Carga Total: 3

Rota do veículo 6:
Ponto 0 -> Ponto 15 (descarregar: 3 quilos) -> Ponto 16 (descarregar: 2 quilos)
Distância da rota: 14m
Carga Total: 5

Rota do veículo 7:
Ponto 0 -> Ponto 11 (descarregar: 4 quilos)
Distância da rota: 8m
Carga Total: 4

Rota do veículo 8:
Ponto 0 -> Ponto 9 (descarregar: 3 quilos)
Distância da rota: 8m
Carga Total: 3

Rota do veículo 9:
Ponto 0 -> Ponto 10 (descarregar: 2 quilos)
Distância da rota: 8m
Carga Total: 2

Distância total de todas as rotas: 94m
Carga total de todas as rotas: 34
```

Visualizando

Algumas variáveis para a geração das visualizações

```
# obtendo a localização de todas as entrega
localizacao_entregas = list(zip(base.Latitude, base.Longitude))
# localização do armazem
localizacao_base = localizacao_entregas[0]

# só entregas
localizacao_entregas = localizacao_entregas[1:]
```

```
# nome do local de entrega
nome_entregas = list(base.Nome)

# nome do armazem
nome_base = nome_entregas[0]

# só entregas
nome_entregas = nome_entregas[1:]
```

Visualizando os pontos de entrega e do armazém

Agora, vamos mostrar o armazém e todas as entregas a serem realizadas

```
mapa_de_pontos = fvrp.gerar_mapa_pontos('Armazém e Pontos de Entrega', localizacao_base, nome_base, localizacao_entregas, nome_entregas)
mapa_de_pontos
```

```
Armazém e Pontos de Entrega
```

Todas as rotas por veículos

```
def criando_direction_layer(localizacao_base, localizacao_rotas, markers):

    # gerando cores aleatórias para a rota de cada veiculo
    r = random.randint(0, 255)
    g = random.randint(0, 255)
    b = random.randint(0, 255)
    cor = '#%02x%02x%02x' % (r, g, b)

    # ultimo ponto
    u = localizacao_rotas[-1]

    # outros pontos que não a base e o ultimo
    w = localizacao_rotas[1:-1]
    time.sleep(3)
    if w:
      dl = gmaps.directions_layer(localizacao_base, u, waypoints=w, stroke_color=cor, show_markers=markers, stroke_opacity=0.8)
    else:
      dl = gmaps.directions_layer(localizacao_base, u, stroke_color=cor, show_markers=markers, stroke_opacity=0.8)

    return dl
```

```
def criando_mapa_com_rotas(titulo, mapa, localizacao_base, localizacao_entregas, rotas_mapa):
    localizacao_rotas = []

    if isinstance(rotas_mapa, list):
      for id_entrega in rotas_mapa:
        localizacao_rotas.append(localizacao_entregas[id_entrega])

      localizacao_rotas = localizacao_rotas[:-1]
      dl = criando_direction_layer(localizacao_base, localizacao_rotas, True)
      mapa.add_layer(dl)

    else:
      for id_veiculo in rotas_mapa:
        for id_entrega in rotas_mapa[id_veiculo]:
          localizacao_rotas.append(localizacao_entregas[id_entrega])

        localizacao_rotas = localizacao_rotas[:-1]
        dl = criando_direction_layer(localizacao_base, localizacao_rotas, False)
        time.sleep(3)
        mapa.add_layer(dl)
    print(titulo)
    return mapa
```

Todos roteiros

```
mapa_todo_roteiro = fvrp.gerar_mapa_pontos('Armazém e Pontos de Entrega', localizacao_base, nome_base, localizacao_entregas, nome_entregas)
mapa_todo_roteiro = fvrp.criando_mapa_com_rotas('Rotas por veículo', mapa_todo_roteiro, localizacao_base, localizacao_entregas, rotas)
mapa_todo_roteiro
```

Armazém e Pontos de Entrega
Rotas por veículo

```
rota1_base = fvrp.gerar_mapa_pontos('', localizacao_base, nome_base, localizacao_entregas, nome_entregas)
rota1 = criando_mapa_com_rotas('Rota 1', rota1_base, localizacao_base, localizacao_entregas, rotas[0])
rota1
```

Rota 1

```
rota2_base = fvrp.gerar_mapa_pontos('', localizacao_base, nome_base, localizacao_entregas, nome_entregas)
rota2 = criando_mapa_com_rotas('Rota 2', rota2_base, localizacao_base, localizacao_entregas, rotas[1])
rota2
```

Armazém e Pontos de Entrega
Rota 2

```
rota3_base = fvrp.gerar_mapa_pontos('', localizacao_base, nome_base, localizacao_entregas, nome_entregas)
rota3 = criando_mapa_com_rotas('Rota 3', rota3_base, localizacao_base, localizacao_entregas, rotas[2])
rota3
```

Rota 3

```
rota4_base = fvrp.gerar_mapa_pontos('', localizacao_base, nome_base, localizacao_entregas, nome_entregas)
rota4 = criando_mapa_com_rotas('Rota 4', rota4_base, localizacao_base, localizacao_entregas, rotas[3])
rota4
```

Rota 4