**Problema a ser resolvido**

Uma transportadora tem um caminhão com capacidade de 10m³ e no seu estoque tem 14 tipos de produtos para ser transportados, sendo que desses 14 alguns produtos tem a quantidade maior que um e todos os produtos já tem seu tamanho definido em m³ e também seus valores.

A transportadora precisa realizar a entrega, mas para isso ela precisa de uma forma para decidir quais produtos irá levar para que maximize o valor total dos produtos, ou seja os produtos que são mais caros tem a prioridade maior para levar dentro do caminhão, porem esse caminhão tem a capacidade máxima de 10m³, logo dependendo da quantidade de produtos que tenha em estoque não terá como levar tudo de uma vez.

A imagem abaixo mostra todos os itens com as quantidades que temos em estoque:



Para resolver esse problema, vamos criar passo a passo um algoritmo genético que vai receber esses produtos, esses dados e o próprio algoritmo no processo evolutivo vai descobrir qual a melhor combinação de produtos ou quais são as melhores combinações desses produtos, que a transportadora poderá levar para maximizar o valor da carga.

Vamos começar com a codificação...

1. Baixar e instalar o Anaconda;
2. Baixar e instalar o MySQL (opcional);
3. Instalar o Flask dentro do anaconda;
4. Codificação.

Código do algoritmo genético.

from flask import Flask, render\_template, request

from random import random

import matplotlib.pyplot as plt

import pymysql

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return render\_template('home.html')

@app.route('/resultado', methods = ['POST', 'GET'])

def send\_data():

rugtt = []

#p1 = Produto("Iphone 6", 0.0000899, 2199.12)

lista\_produtos = []

conexao = pymysql.connect(host='localhost', user='root', passwd='123456', db='produtos')#fazendo a conexão com a base de dados para recuperar os produtos cadastrados

cursor = conexao.cursor()

cursor.execute('select nome, espaco, valor, quantidade from produtos')

for produto in cursor:

#print(produto[3])

for i in range(produto[3]):

lista\_produtos.append(Produto(produto[0], produto[1], produto[2]))

cursor.close()

conexao.close()

espacos = []

valores = []

nomes = []

for produto in lista\_produtos:

espacos.append(produto.espaco)

valores.append(produto.valor)

nomes.append(produto.nome)

limite = 10

ind = 1

tamanho\_populacao = 20

taxa\_mutacao = 0.01

numero\_geracoes = 100

ag = AlgoritmoGenetico(tamanho\_populacao)

resultado = ag.resolver(taxa\_mutacao, numero\_geracoes, espacos, valores, limite)

for i in range(len(lista\_produtos)):

if resultado[i] == '1':

item = str(ind)+" -- Produto: "+lista\_produtos[i].nome + " - Valor: R$ "+str(lista\_produtos[i].valor)

rugtt.append(item)

ind = ind+1

gerr = ger

vr = valor

ep = espaco

cr = croo

grafico = ag.lista\_solucoes

return render\_template('home.html',rugtt = rugtt,gerr = gerr, vr=vr, ep=ep, cr=cr, grafico=grafico)

class Produto():

def \_\_init\_\_(self, nome, espaco, valor):

self.nome = nome

self.espaco = espaco

self.valor = valor

class Individuo():

def \_\_init\_\_(self, espacos, valores, limite\_espacos, geracao=0):

self.espacos = espacos

self.valores = valores

self.limite\_espacos = limite\_espacos

self.nota\_avaliacao = 0

self.espaco\_usado = 0

self.geracao = geracao

self.cromossomo = []

for i in range(len(espacos)):

if random() < 0.5:

self.cromossomo.append("0")

else:

self.cromossomo.append("1")

def avaliacao(self):

nota = 0

soma\_espacos = 0

for i in range(len(self.cromossomo)):

if self.cromossomo[i] == '1':

nota += self.valores[i]

soma\_espacos += self.espacos[i]

if soma\_espacos > self.limite\_espacos:

nota = 1

self.nota\_avaliacao = nota

self.espaco\_usado = soma\_espacos

def crossover(self, outro\_individuo):

corte = round(random() \* len(self.cromossomo))

filho1 = outro\_individuo.cromossomo[0:corte] + self.cromossomo[corte::]

filho2 = self.cromossomo[0:corte] + outro\_individuo.cromossomo[corte::]

filhos = [Individuo(self.espacos, self.valores, self.limite\_espacos, self.geracao + 1),

Individuo(self.espacos, self.valores, self.limite\_espacos, self.geracao + 1)]

filhos[0].cromossomo = filho1

filhos[1].cromossomo = filho2

return filhos

def mutacao(self, taxa\_mutacao):

#print("Antes %s " % self.cromossomo)

for i in range(len(self.cromossomo)):

if random() < taxa\_mutacao:

if self.cromossomo[i] == '1':

self.cromossomo[i] = '0'

else:

self.cromossomo[i] = '1'

#print("Depois %s " % self.cromossomo)

return self

class AlgoritmoGenetico():

def \_\_init\_\_(self, tamanho\_populacao):

self.tamanho\_populacao = tamanho\_populacao

self.populacao = []

self.geracao = 0

self.melhor\_solucao = 0

self.lista\_solucoes = []

def inicializa\_populacao(self, espacos, valores, limite\_espacos):

for i in range(self.tamanho\_populacao):

self.populacao.append(Individuo(espacos, valores, limite\_espacos))

self.melhor\_solucao = self.populacao[0]

def ordena\_populacao(self):

self.populacao = sorted(self.populacao,

key = lambda populacao: populacao.nota\_avaliacao,

reverse = True)

def melhor\_individuo(self, individuo):

if individuo.nota\_avaliacao > self.melhor\_solucao.nota\_avaliacao:

self.melhor\_solucao = individuo

def soma\_avaliacoes(self):

soma = 0

for individuo in self.populacao:

soma += individuo.nota\_avaliacao

return soma

def seleciona\_pai(self, soma\_avaliacao):

pai = -1

valor\_sorteado = random() \* soma\_avaliacao

soma = 0

i = 0

while i < len(self.populacao) and soma < valor\_sorteado:

soma += self.populacao[i].nota\_avaliacao

pai += 1

i += 1

return pai

#aqui mostra a geracao

def visualiza\_geracao(self):

melhor = self.populacao[0]

print("G:%s -> Valor: %s Espaço: %s Cromossomo: %s" % (self.populacao[0].geracao,

melhor.nota\_avaliacao,

melhor.espaco\_usado,

melhor.cromossomo))

def resolver(self, taxa\_mutacao, numero\_geracoes, espacos, valores, limite\_espacos):

global ger

global valor

global espaco

global croo

self.inicializa\_populacao(espacos, valores, limite\_espacos)

for individuo in self.populacao:

individuo.avaliacao()

self.ordena\_populacao()

self.melhor\_solucao = self.populacao[0]

self.lista\_solucoes.append(self.melhor\_solucao.nota\_avaliacao)

self.visualiza\_geracao()

for geracao in range(numero\_geracoes):

soma\_avaliacao = self.soma\_avaliacoes()

nova\_populacao = []

for individuos\_gerados in range(0, self.tamanho\_populacao, 2):

pai1 = self.seleciona\_pai(soma\_avaliacao)

pai2 = self.seleciona\_pai(soma\_avaliacao)

filhos = self.populacao[pai1].crossover(self.populacao[pai2])

nova\_populacao.append(filhos[0].mutacao(taxa\_mutacao))

nova\_populacao.append(filhos[1].mutacao(taxa\_mutacao))

self.populacao = list(nova\_populacao)

for individuo in self.populacao:

individuo.avaliacao()

self.ordena\_populacao()

self.visualiza\_geracao()

melhor = self.populacao[0]

self.lista\_solucoes.append(melhor.nota\_avaliacao)

self.melhor\_individuo(melhor)

print("\nMelhor solução -> G: %s Valor: %s Espaço: %s Cromossomo: %s" %

(self.melhor\_solucao.geracao,

self.melhor\_solucao.nota\_avaliacao,

self.melhor\_solucao.espaco\_usado,

self.melhor\_solucao.cromossomo))

ger = str(self.melhor\_solucao.geracao)

valor = str("{:10.2f}".format(self.melhor\_solucao.nota\_avaliacao))

espaco = str("{:10.3f}".format(self.melhor\_solucao.espaco\_usado))

croo = str(self.melhor\_solucao.cromossomo)

return self.melhor\_solucao.cromossomo

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

código do html(flask)

**home.html**

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

<link rel="stylesheet" href="static/css/style.css">

<title>Algoritmos Genéticos</title>

<style>

#img\_problema{

width:100%;

padding:10px;

}

</style>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<div class="container">

<a class="navbar-brand" href="/"><img id="logo-principal" src="static/images/connectst.jpeg"/> Startup Connects </a>

</div>

<form role="form" action="{{url\_for('send\_data')}}" method="POST">

<input type="submit" class="btn btn-secondary btn-sm" value="Gerar">

</form>

</nav>

<div class="container">

{%if rugtt != null%}

<h4 class="text-center">Melhor solução</h4>

<table class="table table-sm table-hover">

<thead class="thead-light">

<tr>

<th scope="col"class="text-center">Geração:</th>

<th scope="col"class="text-center">Valor</th>

<th scope="col"class="text-center">Espaço</th>

</tr>

</thead>

<tbody>

<tr>

<td class="text-center">{{gerr}}</td>

<td class="text-center">R$ {{vr}}</td>

<td class="text-center">{{ep}}m³</td>

</tr>

</tbody>

</table>

<table class="table table-sm table-hover">

<thead class="thead-light">

<tr>

<th scope="col" class="text-center">Cromossomo:</th>

</tr>

</thead>

<tbody>

<tr>

<td>{{cr}}</td>

</tr>

</tbody>

</table>

{%for ru in rugtt%}

<h6>{{ru}}</h6>

{% endfor %}

<div class="col-sm-12">

<canvas class="line-charts container"></canvas>

</div>

{%endif%}

{%if rugtt == null%}

<h3 class="text-center">Problema a ser resolvido</h3>

<img id="img\_problema" title="Problema a ser resolvido" src="static/images/problema.png"/>

<h6><b>Atividade realizada por:</b></h6>

<span>Jacson Gomes</span><br>

<span>Nailson Melo</span>

{%endif%}

</div>

<!--geracao do grafico-->

<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.7.3/Chart.min.js"></script>

<script>

var ctx = document.getElementsByClassName("line-charts");

var chartGraph = new Chart(ctx, {

type: 'line',

data: {

labels:["0 ","1 ",

"2 ",

"3 ",

"4 ",

"5 ",

"6 ",

"7 ",

"8 ",

"9 ",

"10 ",

"11 ",

"12 ",

"13 ",

"14 ",

"15 ",

"16 ",

"17 ",

"18 ",

"19 ",

"20 ",

"21 ",

"22 ",

"23 ",

"24 ",

"25 ",

"26 ",

"27 ",

"28 ",

"29 ",

"30 ",

"31 ",

"32 ",

"33 ",

"34 ",

"35 ",

"36 ",

"37 ",

"38 ",

"39 ",

"40 ",

"41 ",

"42 ",

"43 ",

"44 ",

"45 ",

"46 ",

"47 ",

"48 ",

"49 ",

"50 ",

"51 ",

"52 ",

"53 ",

"54 ",

"55 ",

"56 ",

"57 ",

"58 ",

"59 ",

"60 ",

"61 ",

"62 ",

"63 ",

"64 ",

"65 ",

"66 ",

"67 ",

"68 ",

"69 ",

"70 ",

"71 ",

"72 ",

"73 ",

"74 ",

"75 ",

"76 ",

"77 ",

"78 ",

"79 ",

"80 ",

"81 ",

"82 ",

"83 ",

"84 ",

"85 ",

"86 ",

"87 ",

"88 ",

"89 ",

"90 ",

"91 ",

"92 ",

"93 ",

"94 ",

"95 ",

"96 ",

"97 ",

"98 ",

"99 "

],

datasets:[

{

label: "Acompanhamento dos valores",

data:{{grafico}},

borderWidth: 4,

borderColor: 'rgba(77,166,253,0.85)',

backgroundColor: 'transparent'

},

]

},

options:{

title:{

display:true,

fontSize:30,

text: "GRAFICO DE GERAÇÃO"

},

labels:{

fontStyle: "bold"

}

}

});

</script>

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous"></script>

</body>

</html>

Agora você pode executar no python e abrir no browser.

