CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Unidade Leopoldina

Engenharia de Computação

COMPUTAÇÃO EVOLUCIONISTA

BUSCA ALEATÓRIA

EMILLY RAMOS TEIXEIRA

Samuel da Costa Alves Basílio

Leopoldina, MG, Brasil

Abril de 2024

1 Introdução Teórica

A **otimização** consiste em usar um conjunto de técnicas para calcular os melhores valores extremos de funções, sejam eles máximo ou mínimo. Este valor representa um fator chave para construir ou analisar um sistema. Ao simular internamente o problema em questão a **função objetivo** nos retorna o valor chave para cada um dos possíveis valores que as **variáveis de decisão** podem assumir. O **domínio** dentro do qual a operação será realizada é composto por um conjunto de valores diferentes disponíveis que as variáveis podem receber. As **restrições** são condições impostas às variáveis dependentes.

2 Objetivos

O presente trabalho visa **minimizar** a seguinte **função objetivo** situada na página cinco do arquivo em formato PDF que contém os problemas do CEC 2006. A otimização deverá ser feita dentro do domínio $\mathbf{-10 \leq x_i \geq 10}$. Onde x_i corresponde a cada uma das variáveis de decisão presentes no problema g07.

$$f(\vec{x}) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ +2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

Figura 1 – Função objetivo.

Dentre as restrições propostas para que **f(x)** seja factível, estão:

$$\begin{aligned}
\mathrm{g}_1(\vec{x}) &= -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0 \\
\mathrm{g}_2(\vec{x}) &= 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \\
\mathrm{g}_3(\vec{x}) &= -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0 \\
\mathrm{g}_4(\vec{x}) &= 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\
\mathrm{g}_5(\vec{x}) &= 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\
\mathrm{g}_6(\vec{x}) &= x_1^2 + 2(x_2 - 2)^2 - 2x_1 x_2 + 14x_5 - 6x_6 \leq 0 \\
\mathrm{g}_7(\vec{x}) &= 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \\
\mathrm{g}_8(\vec{x}) &= -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0
\end{aligned}$$

Figura 2 – Restrições.

A solução ótima **24.30620906818** é obtida no momento em que cada uma das variáveis de decisão assumem respectivamente os valores **2.17199634142692, 2.3636830416034, 8.77392573913157, 5.09598443745173, 0.990654756560493, 1.43057392853463, 1.32164415364306, 9.82872576524495, 8.2800915887356, 8.3759266477347**

3 Metodologia

A linguagem de programação Python foi a escolhida para implementar o problema de otimização proposto. Dentre as bibliotecas utilizadas estão random, matplotlib, numpy, math e statistics.

```
import random
import matplotlib.pyplot as plt
import numpy as np
import math
import statistics
```

Figura 3 – Bibliotecas.

Para gerar seeds diferentes a cada execução, a variável de controle da estrutura de repetição é passada como parâmetro para a função random.

```
for i in range(10):
  # Gerando uma nova semente a cada iteração
  random.seed(i)
```

Figura 4 – Seeds.

O próximo passo foi definir dentro do código a função a ser otimizada.

```
def f(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10):
    return math.pow(x1, 2) + math.pow(x2, 2) + (x1*x2) - (14*x1) - (16*x2) + (math.pow(x3-10, 2)) + 4*(math.pow(x4-5, 2)) + (math.pow(x5-3, 2))
```

Figura 5 – Definição da função objetivo.

Percebe-se que devido a sua grande extensão não foi possível visualizar todo o código. Portanto, basta clicar aqui para acessar o ambiente do Google Colab em que está armazenado o programa.

A seguinte função possui por objetivo armazenar em um vetor cada uma das expressões que representam as restrições a serem satisfeitas para que a solução encontrada seja considerada factível.

```
def factible(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10):
    g = [None, None, None, None, None, None, None, None]
    g[0] = -105 + (4*x1) + (5*x2) - (3*x7) + (9*x8)
    g[1] = (10*x1) - (8*x2) - (17*x7) + (2*x8)
    g[2] = (-8*x1) + (2*x2) + (5*x9) - (2*x10) - 12
    g[3] = 3*math.pow(x1-2, 2) + 4*math.pow(x2-3, 2) + 2*math.pow(x3, 2) - (7*x4) -120
    g[4] = 5*math.pow(x1, 2) + (8*x2) + math.pow(x3-6, 2) - (2*x4) - 40
    g[5] = math.pow(x1, 2) + 2*math.pow(x2-2, 2) - (2*x1*x2) + (14*x5) - (6*x6)
    g[6] = 0.5*math.pow(x1-8, 2) + 2*math.pow(x2-4, 2) + 3*math.pow(x5, 2) - x6 -30
    g[7] = -(3*x1) + (6*x2) + 12*math.pow(x9-8, 2) - (7*x10)

    return g
```

Figura 6 – Código referente às restrições.

A função responsável por realizar a busca aleatória recebe como parâmetro o número de iterações a serem feitas, o limite inferior e superior. Em cada execução as variáveis dependentes assumem um novo valor dentro do intervalo estabelecido.

```
limites_inferior = [-10, -10, -10, -10, -10, -10, -10, -10, -10, -10]
limites_superior = [10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
```

Figura 7 – Limites.

O valor retornado pela função objetivo é comparado com o número considerado como melhor imagem, caso ele seja menor assumirá o seu lugar. Dentro dessa mesma estrutura condicional é verificado se a solução é considerada factível e , consequentemente, a variável que armazena a quantidade de soluções factíveis é incrementada em uma unidade. É de suma importância ressaltar que é aplicada uma **penalidade** aos elementos que não satisfazem a condição imposta.

```
if melhor_w > w and all(i >= 0 for i in factible(X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)):
    melhor_w = w

    nFactivel+=1

    melhor_x1 = X1
    melhor_x2 = X2
    melhor_x3 = X3
    melhor_x4 = X4
    melhor_x5 = X5
    melhor_x6 = X6
    melhor_x7 = X7
    melhor_x8 = X8
    melhor_x9 = X9
    melhor_x10= X10

else:
  w+=2*24.000
```

Figura 8 – Melhor solução.

A cada execução do bloco de código é gerado um gráfico que mostra a convergência usando a biblioteca Matplotlib

```
plt.title('\nConvergência')
plt.xlabel('\nIterações')
plt.ylabel('\nMelhor valor')
plt.scatter(range(len(melhores_imagens)), melhores_imagens,color='r')
plt.grid(True)
plt.show()
```

Figura 9 – Convergência.

Antes de realizar o cálculo da melhor mediana, média e desvio padrão é feita uma remoção dos valores do tipo infinito e NAN(not a number) para evitar erros no código. Além disso, também há a verificação da existência de mais de um elemento dentro da variável melhores_imagens

```python
# Remover valores infinitos e NaN da lista melhores_imagens
melhores_imagens = [x for x in melhores_imagens if x != float('inf') and not math.isnan(x)]

# Calculando desvio padrão se houver mais de um elemento em melhores_imagens
if len(melhores_imagens) > 1:
    mediana = statistics.median(melhores_imagens)
    media = statistics.mean(melhores_imagens)
    desvio_padrao = statistics.stdev(melhores_imagens)
else:
    desvio_padrao = 0
    media = 0
    mediana = 0


print("\nMediana: ", mediana)
print("\nMédia: ", media)
print("\nDesvio padrão: ", desvio_padrao)
```

Figura 10 – Mediana, média e desvio padrão.

4 Análise dos Resultados

Ao executar o código que realiza a busca aleatória adotando o número de 8000 interações obtém-se as seguintes saídas:

```
Execução:  0

Soluções factíveis:  4

Melhor x1:  -0.2990577373090275
Melhor x2:  9.41726643118512
Melhor x3:  7.97422845659851
Melhor x4:  7.291335653438253
Melhor x5:  5.219344414072554
Melhor x6:  3.608873863820591
Melhor x7:  -5.324849409402006
Melhor x8:  5.863469246704485
Melhor x9:  1.46723599392082
Melhor x10:  3.954592425116555
Melhor w:  509.4595174397773
```

Figura 11 – Execução zero, quantidade de soluções factíveis e melhores valores

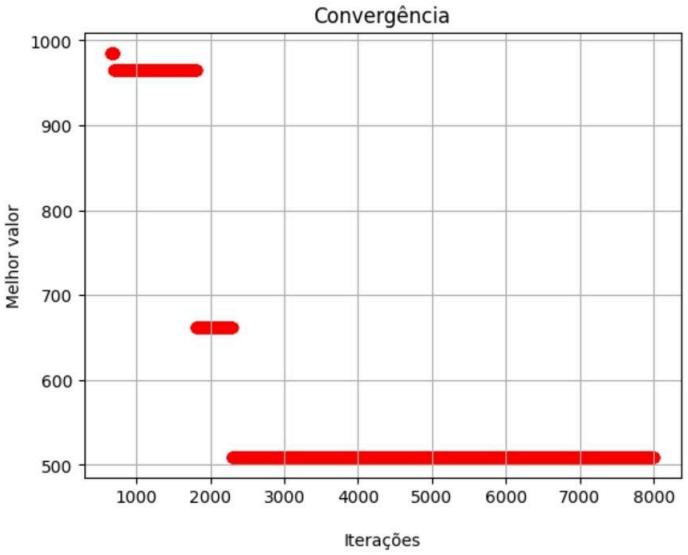

Figura 12 – Gráfico 0.

```
Mediana:  509.4595174397773

Média:  591.0539968444996

Desvio padrão:  165.80529010793734
```

Figura 13 – Mediana, média e moda da execução zero.

```
Execução:  1

Soluções factíveis:  5

Melhor x1:  -0.4385034512521919
Melhor x2:  9.106462513346905
Melhor x3:  9.879806001258132
Melhor x4:  7.59747216987995
Melhor x5:  -1.406210240371223
Melhor x6:  0.15716948025870714
Melhor x7:  -4.478966190556568
Melhor x8:  9.611622295163645
Melhor x9:  7.324779606585945
Melhor x10:  -1.4931316596744715
Melhor w:  232.6454191248278
```

Figura 14 – Execução um, quantidade de soluções factíveis e melhores valores.

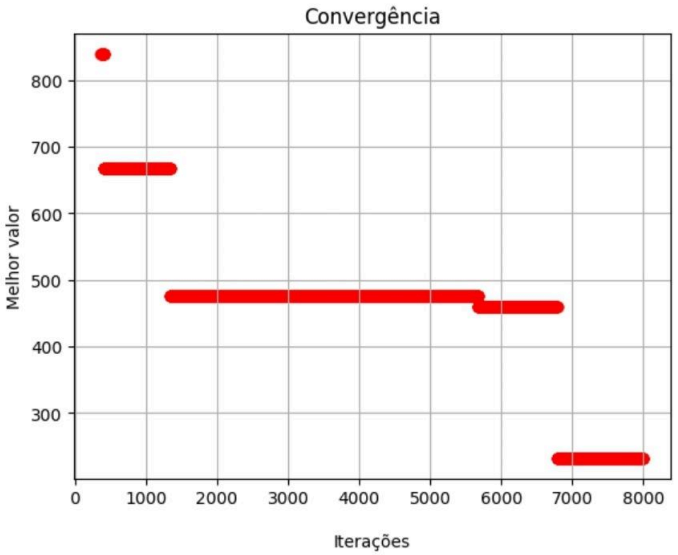

Figura 15 – Gráfico 1.

```
Mediana:  476.86329671147774

Média:  460.81705460730757

Desvio padrão:  120.24537659899343
```

Figura 16 - Mediana, média e moda da execução um.

```
Execução:  2

Soluções factíveis:  3

Melhor x1:  0.859803796152649
Melhor x2:  9.037949503280057
Melhor x3:  -3.12829839067356
Melhor x4:  1.6681663625161143
Melhor x5:  5.685910213025391
Melhor x6:  1.4734324387355358
Melhor x7:  -4.738926781813428
Melhor x8:  8.187022542700909
Melhor x9:  6.7083816316849045
Melhor x10:  6.036794754889328
Melhor w:  393.24358004534383
```

Figura 17 – Execução dois, quantidade de soluções factíveis e melhores valores.

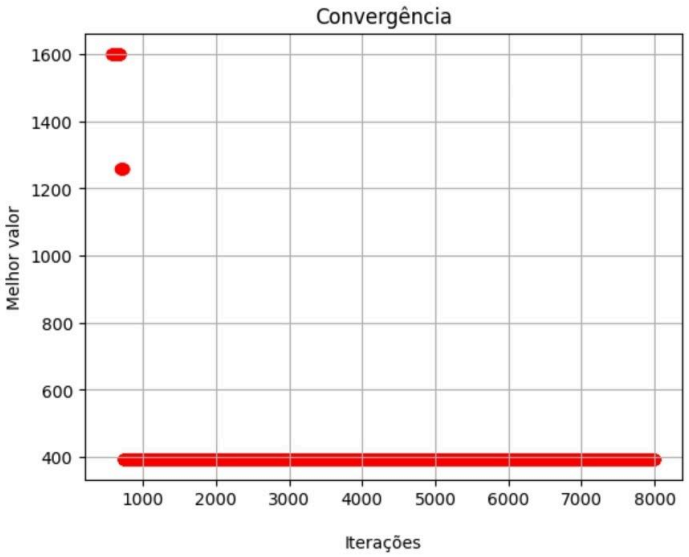

Figura 18 – Gráfico 2.

```
Mediana:  764.4200430226498

Média:  728.2631580074691

Desvio padrão:  145.68888106788427
```

Figura 19 - Mediana, média e moda da execução dois.

```
Execução:  3

Soluções factíveis:  4

Melhor x1:  6.508089218366937
Melhor x2:  6.4078596431522
Melhor x3:  5.888586836453275
Melhor x4:  3.3965813555320334
Melhor x5:  4.803041850276097
Melhor x6:  8.898100188424209
Melhor x7:  -0.4043210784466069
Melhor x8:  7.617177758080395
Melhor x9:  9.351245303888703
Melhor x10:  -7.718159676414724
Melhor w:  430.06603756236944
```

Figura 20 - Execução três, quantidade de soluções factíveis e melhores valores.

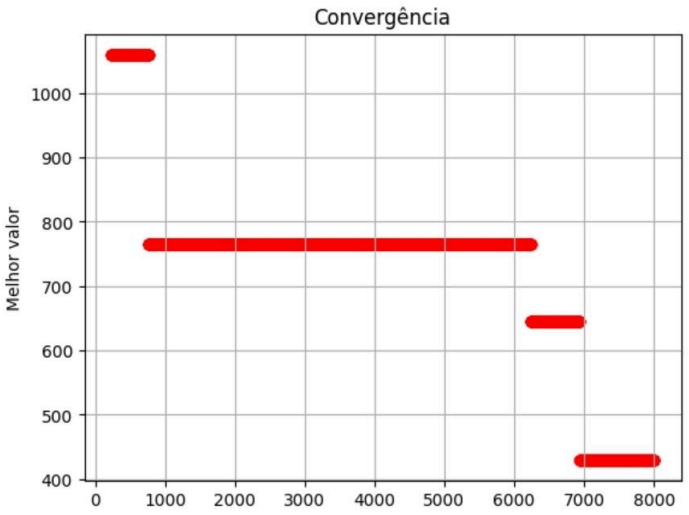

Figura 21 - Gráfico 3

```
Mediana:  764.4200430226498

Média:  728.2631580074691

Desvio padrão:  145.68888106788427
```

Figura 22 - Mediana, média e moda da execução três.

```
Execução:  4

Soluções factíveis:  6

Melhor x1:  -3.2326610384796917
Melhor x2:  9.113059714827678
Melhor x3:  0.9754208435770444
Melhor x4:  4.182273213097407
Melhor x5:  2.9437588112783004
Melhor x6:  -4.207491392000604
Melhor x7:  -5.207739417861593
Melhor x8:  9.572713000975721
Melhor x9:  1.179245609495215
Melhor x10:  6.682353831789726
Melhor w:  452.41875597340635
```

Figura 23 - Execução quatro, quantidade de soluções factíveis e melhores valores.

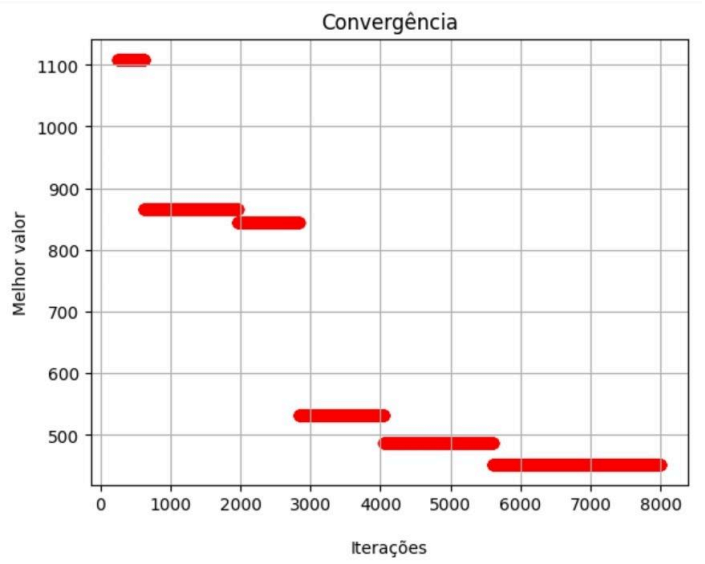

Figura 24 - Gráfico 4

```
Mediana:  487.14176388396226

Média:  619.408699432783

Desvio padrão:  202.9631532761981
```

Figura 25 - Mediana, média e moda da execução quatro.

```
Execução:  5

Soluções factíveis:  3

Melhor x1:  2.780252317262555
Melhor x2:  7.950303637169622
Melhor x3:  4.818583980739405
Melhor x4:  3.6105586750792433
Melhor x5:  2.1550162262018198
Melhor x6:  4.806489652253454
Melhor x7:  -5.030114910987553
Melhor x8:  5.201553336842682
Melhor x9:  8.434855404739764
Melhor x10:  5.4676383079999304
Melhor w:  405.28622920458855
```

Figura 26 - Execução cinco, quantidade de soluções factíveis e melhores valores.

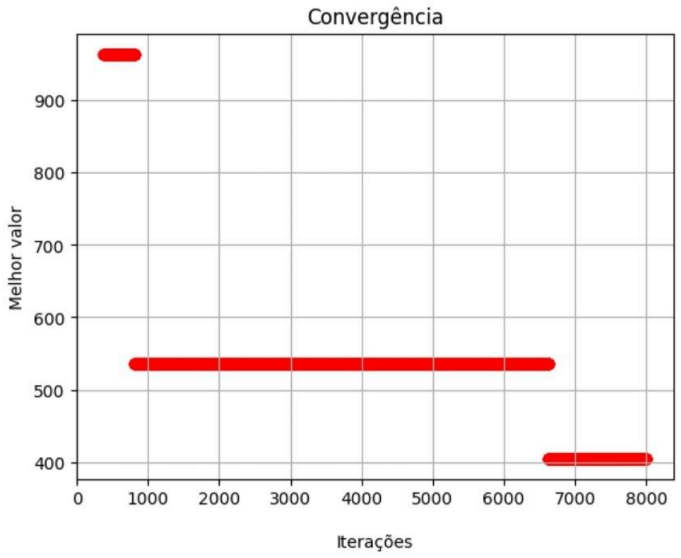

Figura 27 - Gráfico 5

```
Mediana:  535.8278905349953

Média:  536.7986025710625

Desvio padrão:  116.37513190991193
```

Figura 28 - Mediana, média e moda da execução cinco

```
Execução:  6

Soluções factíveis:  3

Melhor x1:  2.7996328921014815
Melhor x2:  9.786006684579597
Melhor x3:  6.373060313246576
Melhor x4:  3.0357190089648896
Melhor x5:  -5.686774664522083
Melhor x6:  -4.765261754086769
Melhor x7:  -3.4004916513290766
Melhor x8:  4.520752221063182
Melhor x9:  7.370784343488868
Melhor x10:  6.362074777477691
Melhor w:  516.6687392579356
```

Figura 29 - Execução seis, quantidade de soluções factíveis e melhores valores.

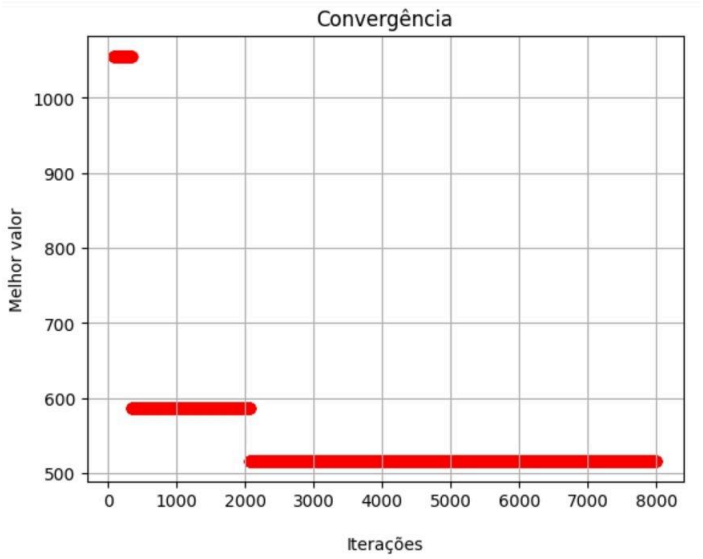

Figura 30 - Gráfico 6

```
Mediana:  516.6687392579356

Média:  549.231185200415

Desvio padrão:  96.74245343825743
```

Figura 31 - Mediana, média e moda da execução seis.

```
Execução:  7

Soluções factíveis:  5

Melhor x1:  -0.9386494140270116
Melhor x2:  8.95493273653933
Melhor x3:  9.279714484280035
Melhor x4:  6.449566781312132
Melhor x5:  5.344792962565217
Melhor x6:  6.330490695966837
Melhor x7:  -6.722647054385793
Melhor x8:  9.567340425886655
Melhor x9:  7.957669692268855
Melhor x10:  5.391248631891905
Melhor w:  310.0464298917502
```

Figura 32 - Execução sete, quantidade de soluções factíveis e melhores valores.

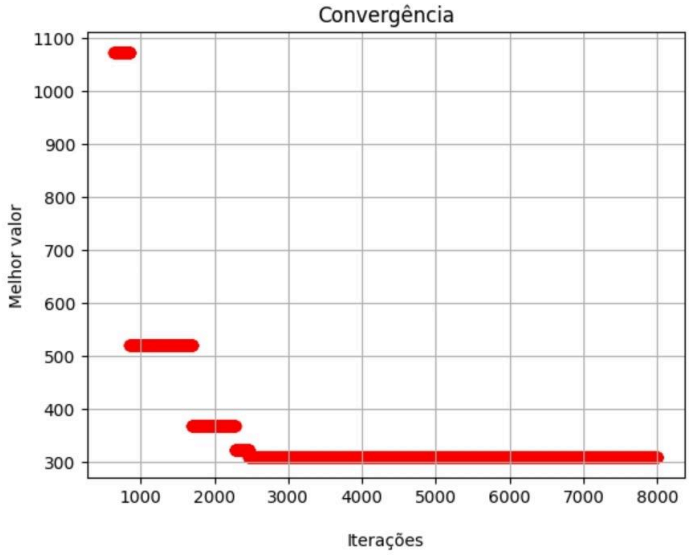

Figura 31 - Gráfico 7

```
Mediana:  310.0464298917502

Média:  361.06252673794256

Desvio padrão:  139.648634571107
```

Figura 32 - Mediana, média e moda da execução sete.

```
Execução:  8

Soluções factíveis:  4

Melhor x1:  3.4923920066999834
Melhor x2:  7.543313754760845
Melhor x3:  9.455775795486609
Melhor x4:  3.668661379312553
Melhor x5:  9.282973852928475
Melhor x6:  -1.8040153955974176
Melhor x7:  -1.638408163153679
Melhor x8:  6.661384712443159
Melhor x9:  9.286560851483024
Melhor x10:  0.6163609239401211
Melhor w:  220.39870354306973
```

Figura 33 - Execução oito, quantidade de soluções factíveis e melhores valores.

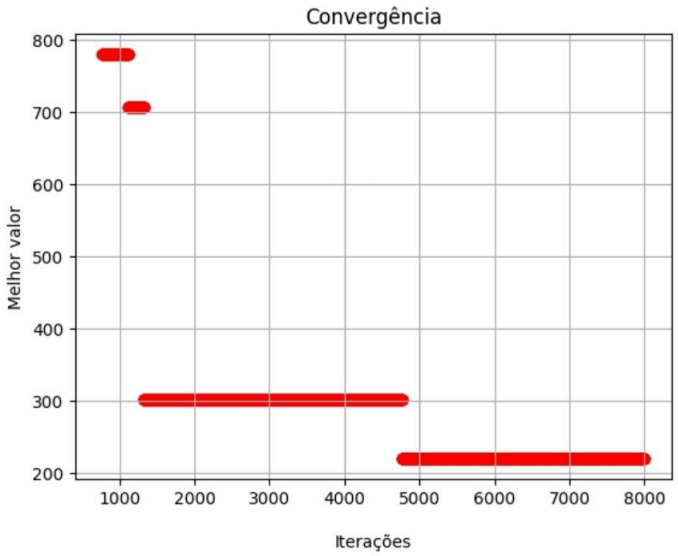

Figura 34 - Gráfico 8

```
Mediana:  302.3085563952272

Média:  300.2272692813648

Desvio padrão:  136.61178285792016
```

Figura 35 - Mediana, média e moda da execução oito.

```
Execução:  9

Soluções factíveis:  2

Melhor x1:  0.6420747588923739
Melhor x2:  7.409076060904965
Melhor x3:  8.247182046817176
Melhor x4:  8.609043724334331
Melhor x5:  2.5634228887554187
Melhor x6:  6.309995302474778
Melhor x7:  -6.1827965586805345
Melhor x8:  9.76599242138056
Melhor x9:  5.486412965275891
Melhor x10:  3.1562097897272245
Melhor w:  346.5993905704678
```

Figura 36 - Execução nove, quantidade de soluções factíveis e melhores valores.

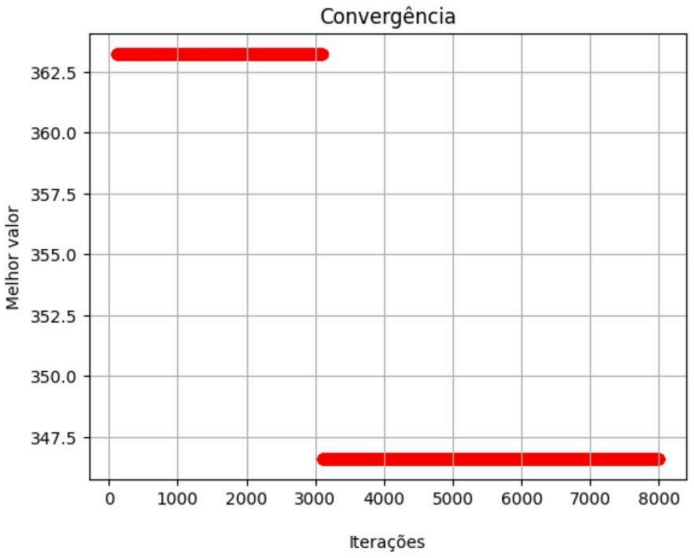

Figura 37 - Gráfico nove

```
Mediana:  346.5993905704678
Média:  352.912466374393
Desvio padrão:  8.075700820039179
```

Figura 38 - Mediana, média e moda da execução nove.

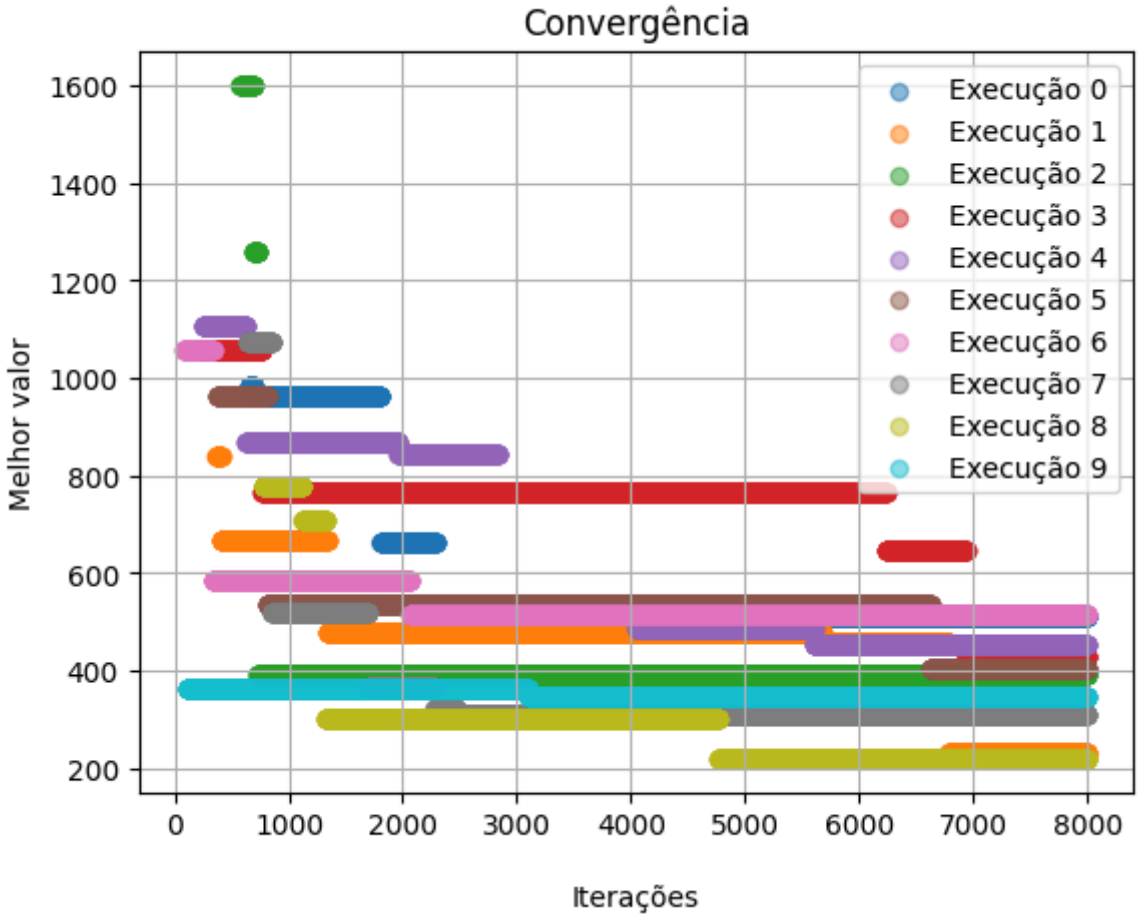

Figura 39 - Gráfico com convergência de todas as nove execuções.

5 Conclusão

Diante dos resultados obtidos percebe-se que o algoritmo da busca aleatória não é uma boa ferramenta para encontrar a solução ótima, definida como **24.30620906818** pelo CEC, uma vez que os valores encontrados estão muito distantes do esperado. Isso ocorre, pois a busca aleatória é um código sem inteligência.

Referências Bibliográficas.

[1] Liang, J. J., Runarsson, T. P., Mezura-Montes, E., Clerc, M., Suganthan, P. N., Coello Coello, C. A., & Deb, K. (2006). Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. In IEEE Congress on Evolutionary Computation (CEC 2006), Vancouver, BC, Canada