TECH CHALLENGE - FIAP

Autores :

Amanda Mayara Menino Alves - RM356386

Allan Alves de Castro - RM355533

Diogo Torres de Lima - RM353817

Charles Igor Bandeira - RM354501

Igor Nesralla Ribeiro - RM354769

1. O Projeto

- **Otimização de hotspots de internet em uma área:** Otimização de utilização de antenas em uma área de fazenda.
- **Descrição Geral:** Nosso projeto consiste na otimização na distribuição de uma rede de antenas em uma área produtiva de uma fazenda, possibilitando uma série de benefícios envolvendo ferramentas e tecnologia dependente de sinal de internet dentro dessa área, o que pode resultar em um aumento de produtividade para a fazenda.

2. Introdução

- **O Problema:**
    - Em termos gerais o problema é a cobertura de sinal de internet em uma área de fazenda que é muito deficitária.
    - Com uma maior cobertura de sinal podemos obter uma série de benefícios para o proprietário daquela área, possibilitando dados em tempo real em campo, comunicação, instalação de equipamentos dependentes de sinal de internet, entre outros. Dessa forma é possível aumentar a produtividade de uma área de fazenda, além de permitir soluções avançadas como automatização de colheitas e maquinário.

- **Objetivos:**
    - O Objetivo é posicionar quatro antenas dentro de uma área quadrada, de modo a maximizar a área de cobertura de sinal. Para tal, utilizaremos Algoritmos Genéticos para realizar essa otimização e solucionar o problema.

3. Metodologia

- **Descrição do Problema:**
    - Definido como um problema a cobertura de hotspots na area de fazenda.
    - Restrições incluem capacidade e alcance dos hotspots de transmissão.
- **Definição das constantes do problema:**
    - Com base no problema a ser resolvido, foram eleitas quatro constantes:
        - **NUM_ANTENAS**: Quantidade de antenas a serem distribuídas pelo terreno
        - **NUM_POPULACAO:** Quantidade de lugares possíveis de se instalar antenas
        - **NUM_GERACOES**: Ciclos de cálculo da distribuição das antenas. Caso algum resultado retorne que a área foi completamente coberta, o algoritmo é interrompido sem prosseguir para as gerações seguintes.
        - **NUM_ELITISMO**: Quantas entre as melhores áreas de cobertura irão para a próxima geração
        - **AREA_LADO**: O tamanho do lado do terreno quadrado
        - **RAIO_ALCANCE**: O raio de alcance da antena a ser instalada
        - **DISTANCIA_MIN**: A distância mínima entre as antenas
        - **DISTANCIA_MAX**: A distância máxima entre as antenas
        - **PROB_MUTACAO**: A probabilidade de alterar uma das antenas de um grupo para gerar variação no resultado.

- **Fluxo e algoritmos utilizados:**

  Com as constantes e funções principais definidas, seguimos com o fluxo de algoritmos genéticos para executar os algoritmos e alcançar os resultados. Abaixo as etapas e algoritmos utilizados

  Plotagem de antenas:

  Para simular a área com as antenas que serão utilizadas no projeto, criamos uma plotagem das antenas na área, e geramos uma imagem para representar visualmente.

```python
import matplotlib.pyplot as plt
import numpy as np
import math
import random
from PIL import Image

# Função para calcular a distância entre dois pontos (antenas)
def distance(p1, p2):
    return math.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)

# Gerar posições aleatórias para as antenas
def generate_random_position(side_length, population_size):
    return [(random.uniform(0, side_length), random.uniform(0, side_length)) for _ in range(population_size)]

# Função para plotar as antenas e suas áreas de cobertura
def plot_antennas(positions, side_length, range_radius, show_plot=False):
    fig, ax = plt.subplots(figsize=(8, 8))

    # Plotar as antenas
    for i, position in enumerate(positions):
        ax.plot(position[0], position[1], marker='o', markersize=8, color='blue')
        ax.text(position[0], position[1], f'Antena {i+1}', fontsize=12, ha='center', color='white')

        # Colore o fundo de verde para contar a área não coberta pelas antenas
        ax.set_facecolor((0, 1, 0))

        # Calcular os pontos para desenhar o círculo da área de cobertura
        circle = plt.Circle(position, range_radius, color='blue', fill=True)
        ax.add_artist(circle)

        # Plotar área de cobertura da antena
        theta = np.linspace(0, 2*np.pi, 100)
        x_cover = position[0] + range_radius * np.cos(theta)
        y_cover = position[1] + range_radius * np.sin(theta)
        ax.plot(x_cover, y_cover, linestyle='--', color='gray', linewidth=1)

    # Configurações de exibição do gráfico
    ax.set_xlim(0, side_length)
    ax.set_ylim(0, side_length)
    ax.set_aspect('equal')
    ax.set_title('Posição das Antenas e Áreas de Cobertura')
    ax.set_xlabel('X (metros)')
    ax.set_ylabel('Y (metros)')
    ax.grid(True)

    # Salva a imagem para contar o espaço não coberto pelas antenas
    plt.savefig('./scatter_plot.png', bbox_inches='tight', pad_inches=0)

    if show_plot == True:
        plt.show()

    plt.close(fig)

def calculate_uncovered_area():
    with Image.open('./scatter_plot.png') as image:
        image_np = np.array(image)

    green_mask = (image_np[:, :, 0] == 0) & (image_np[:, :, 1] == 255) & (image_np[:, :, 2] == 0)
    uncovered_area = np.count_nonzero(green_mask)

    return uncovered_area
```

Inicialização: Foi gerada uma população inicial aleatória, respeitando as restrições de distância mínima e máxima entre as antenas

```python
def start_population(pop_size, side_length, antenna_qty, min_distance, max_distance):
    population = []

    for _ in range(pop_size):
        individual = []

        # posição aleatória
        individual.append((random.uniform(0, side_length), random.uniform(0, side_length)))

        for _ in range(1, antenna_qty):
            while True:
                new_x = random.uniform(0, side_length)
                new_y = random.uniform(0, side_length)
                new_position = (new_x, new_y)

                # Verificar se a nova posição respeita a distância mínima e maxima em relação às antenas já posicionadas
                if all(distance(new_position, individual[j]) <= max_distance for j in range(len(individual))):
                    if all(distance(new_position, individual[j]) >= min_distance for j in range(len(individual))):
                        individual.append(new_position)
                        break

        population.append(individual)

    return population
```

Avaliação: Foi calculado o fitness de cada indivíduo na população, baseada na área não coberta pelo sinal das antenas.

```python
def calculate_fitness(individual, side_length, range_radius):
    antenna_positions = [(ind[0], ind[1]) for ind in individual]

    # Executa primeiro o plotting para gerar a imagem
    antenna.plot_antennas(antenna_positions, side_length, range_radius)

    # Depois calcula a área em verde (sem cobertura pelas antenas)
    return antenna.calculate_uncovered_area()
```

Seleção: Foram selecionados pares de pais baseados no fitness, utilizando-se o método de torneio

```python
def selection(population, fitness_pop):
    parents = []
    for _ in range(2):
        # Seleciona 3 indivíduos aleatórios
        candidates = random.sample(list(enumerate(population)), 3)

        # Seleciona o melhor dos 3
        parent_index = max(candidates, key=lambda x: fitness_pop[x[0]])[0]
        parents.append(population[parent_index])
    return parents
```

Crossover: São gerados novos indivíduos (filhos) a partir dos pais, utilizando crossover

```python
def crossover(parent_1, parent_2, antenna_qty):
    cut = random.randint(1, antenna_qty - 1)
    son_1 = parent_1[:cut] + parent_2[cut:]
    son_2 = parent_2[:cut] + parent_1[cut:]
    return son_1, son_2
```

Mutação: Aplicamos a mutação em um indivíduo, respeitando as restrições de distância mínima e máxima.

```python
# Função de mutação respeitando a distância mínima e maxima entre antenas
def mutation(individual, population, mutation_probability, min_distance, max_distance):
    # Não sofre mutação se o número sorteado estiver abaixo da probabilidade configurada
    if random.random() > mutation_probability:
        return individual

    mutated = False
    antenna_index = random.randint(1, 3)

    # Pega outra antena aleatória da população respeitando a regra de distância
    while mutated == False:
        randomized_index = random.randrange(len(population))

        for new_position in population[randomized_index]:
            if (min_distance <= distance(individual[antenna_index], new_position) <= max_distance):
                individual[antenna_index] = new_position
                mutated = True

    return individual
```

Elitismo: Mantemos os melhores indivíduos de uma geração para a próxima, garantindo a preservação das melhores soluções.

```python
def elitism(elitism_qty, population_qty, population, fitness_pop):
    elitism_index = sorted(range(population_qty), key=lambda x: fitness_pop[x])[:elitism_qty]
    new_population = [population[indice] for indice in elitism_index]

    return new_population
```

- **Definição dos valores das constantes e execução:**

Uma vez que temos as antenas plotadas e os algoritmos genéticos criados, podemos definir os valores iniciais das constantes e executar os algoritmos pare obter os resultados do modelo.

Definição das constantes:

```python
import antennas as an
import genetic_algorithm as ga

NUM_ANTENAS = 4
NUM_POPULACAO = 10
NUM_GERACOES = 10
NUM_ELITISMO = 3
AREA_LADO = 400

RAIO_ALCANCE = 100
DISTANCIA_MIN = 100
DISTANCIA_MAX = 300

PROB_MUTACAO = 0.5

```

Inicialização da população e definição da função do algoritmo genético:

```python
def genetic_algorithm(pop_size, generation_qty):
    population = ga.start_population(pop_size, AREA_LADO, NUM_ANTENAS, DISTANCIA_MIN, DISTANCIA_MAX)

    fitness_pop = []

    for generation in range(generation_qty):
        # Avaliação da populaç
        fitness_pop = [ga.calculate_fitness(individual, AREA_LADO, RAIO_ALCANCE) for individual in population]

        # Se a cobertura do terreno já estiver completa
        if any(x == 0 for x in fitness_pop):
            break

        print(f'Geração: {generation + 1} - Fitness da população: {fitness_pop}')

        parents = [ga.selection(population, fitness_pop) for _ in range(pop_size // 2)]

        # Inicia a nova população com elitismo
        new_population = ga.elitism(NUM_ELITISMO, NUM_POPULACAO, population, fitness_pop)

        # Criando nova geração aplicando crossover e mutação
        for parent_1, parent_2 in parents:
            son_1, son_2 = ga.crossover(parent_1, parent_2, NUM_ANTENAS)

            son_1 = ga.mutation(son_1, population, PROB_MUTACAO, DISTANCIA_MIN, DISTANCIA_MAX)
            new_population.append(son_1)

            if (len(new_population) >= NUM_POPULACAO):
                break

            son_2 = ga.mutation(son_2, population, PROB_MUTACAO, DISTANCIA_MIN, DISTANCIA_MAX)
            new_population.append(son_2)

        # Substituição da população antiga pela nova geração
        population = new_population

    # Retornar a melhor solução encontrada
    best_index = min(range(pop_size), key=lambda x: fitness_pop[x])
    print(f"Melhor fitness calculado: {fitness_pop[best_index]}")
    return population[best_index]

```

Exibição visual dos resultados:

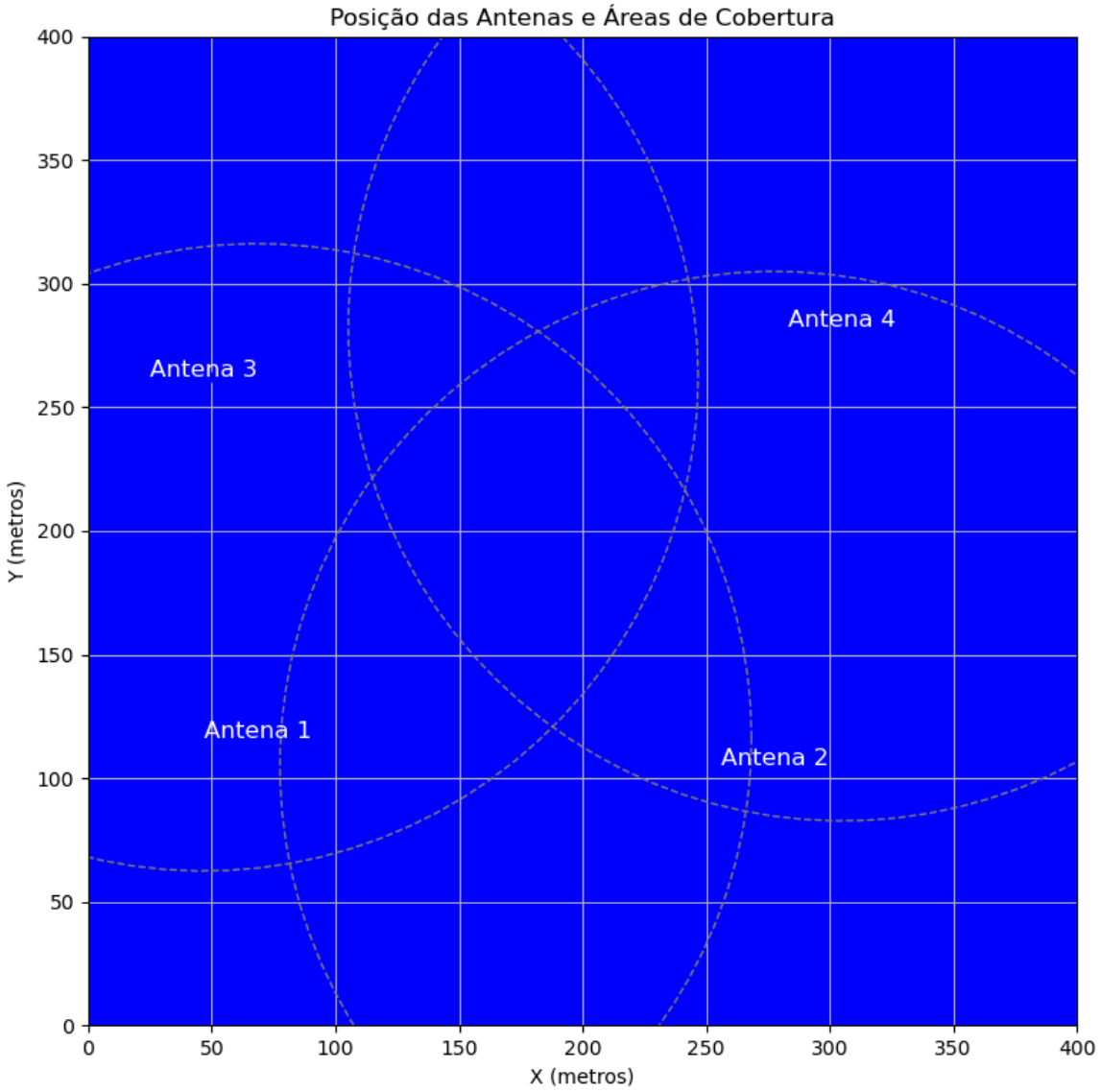
```python
def print_result(solution):
    print("Melhor solução encontrada:")
    for i, position in enumerate(solution):
        print(f"Antena {i+1}: ({position[0]:.2f}, {position[1]:.2f})")

best_solution = genetic_algorithm(pop_size=NUM_POPULACAO, generation_qty=NUM_GERACOES)

print_result(best_solution)
an.plot_antennas(best_solution, AREA_LADO, RAIO_ALCANCE, show_plot=True)
```

4. Testes e Resultados

- **TESTE 1Teste 1:**
  - **Interpretação:**
    - Conseguiu cobertura completa antes do término das gerações
  - **Constantes utilizadas:**
    - `NUM_ANTENAS = 4`
    - `NUM_POPULACAO = 10`
    - `NUM_GERACOES = 50`
    - `NUM_ELITISMO = 3`
    - `AREA_LADO = 400`
    - `RAIO_ALCANCE = 200`
    - `DISTANCIA_MIN = 100`
    - `DISTANCIA_MAX = 300`
    - `PROB_MUTACAO = 0.5`
  - **Resultados Obtidos:**
    - `Geração: 1 - Fitness da população: [8665, 22525, 29343, 17848, 22279, 11193, 15916, 8840, 18220, 9361]`
    - `Geração: 2 - Fitness da população: [8665, 8840, 9361, 1374, 64777, 11480, 19235, 22279, 15183, 15916]`
    - `Geração: 3 - Fitness da população: [1374, 8665, 8840, 35480, 287, 27098, 34430, 7167, 54732, 34386]`
    - `Geração: 4 - Fitness da população: [287, 1374, 7167, 10977, 5, 34430, 34430, 9241, 7081, 56686]`
    - `Geração: 5 - Fitness da população: [5, 287, 1374, 9241, 9241, 54490, 26499, 15183, 13245, 34430]`
    - `Geração: 6 - Fitness da população: [5, 287, 1374, 80572, 34430, 80572, 54512, 10977, 7167, 16459]`
    - `Melhor fitness calculado: 0`
    - `Melhor solução encontrada:`
    - `Antena 1: (68.38, 116.29)`
    - `Antena 2: (277.53, 105.09)`
    - `Antena 3: (46.78, 262.38)`
    - `Antena 4: (305.17, 282.65)`
  - **Gráfico resultante:**



- **TESTE 2:**
  - **Interpretação:**
    - A mutação foi alterada para o valor máximo para testar a cobertura de um terreno maior, no entanto a partir da quarta geração os resultados atingiram um *plateau* e pouco se alteraram, possivelmente pela distância máxima e mínima entre as antenas. Visualmente, talvez a posição da antena 2 pudesse estar melhor localizada para maior cobertura.
  - **Constantes utilizadas:**
    - `NUM_ANTENAS = 4`
    - `NUM_POPULACAO = 12`
    - `NUM_GERACOES = 50`
    - `NUM_ELITISMO = 2`
    - `AREA_LADO = 600`

- RAIO_ALCANCE = 200
- DISTANCIA_MIN = 100
- DISTANCIA_MAX = 300
- PROB_MUTACAO = 1

- **Resultados Obtidos:**
  - Geração: 1 - Fitness da população: [137411, 133050, 163497, 136755, 167745, 152664, 138683, 135605, 87471, 153750, 90291, 150244]
  - Geração: 2 - Fitness da população: [87471, 90291, 131673, 167745, 117570, 95746, 101415, 104966, 111054, 110482, 164898, 152561, 191472]
  - Geração: 3 - Fitness da população: [87471, 90291, 102969, 161654, 84977, 84868, 160334, 156844, 133385, 191472, 164898, 164898, 150278]
  - Geração: 4 - Fitness da população: [84868, 84977, 210718, 159102, 89149, 95143, 164898, 210718, 111869, 125162, 96803, 112623, 146751]
  - Geração: 5 - Fitness da população: [84868, 84977, 156844, 147946, 182534, 146751, 165305, 211158, 156844, 147946, 156844, 165646, 120431]
  - Geração: 6 - Fitness da população: [84868, 84977, 135982, 135982, 172376, 147946, 185314, 147946, 172376, 165305, 247151, 247151, 165169]
  - Geração: 7 - Fitness da população: [84868, 84977, 181943, 211158, 185314, 172376, 165305, 247151, 131112, 165590, 165169, 165305, 165305]
  - Geração: 8 - Fitness da população: [84868, 84977, 247151, 165305, 172376, 247151, 247151, 247151, 172376, 247151, 181943, 172376, 247151]
  - Geração: 9 - Fitness da população: [84868, 84977, 247151, 247151, 172376, 247151, 247151, 172376, 247151, 247151, 247151, 247151, 247151]
  - Geração: 10 - Fitness da população: [84868, 84977, 247151, 172376, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 172376]
  - Geração: 11 - Fitness da população: [84868, 84977, 247151, 172376, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 247151, 247151]

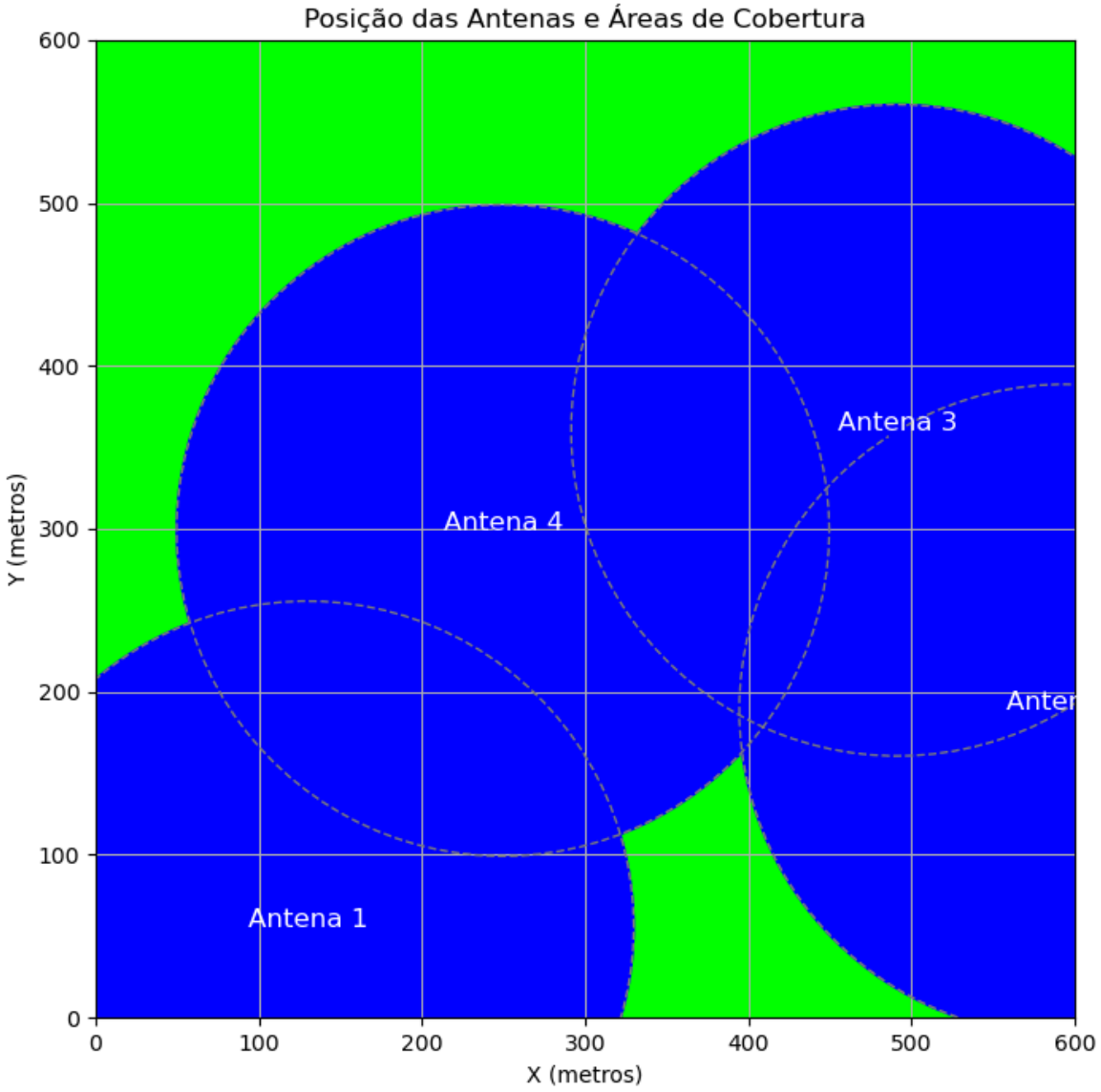
- Geração: 12 - Fitness da população: [84868, 84977, 247151, 172376, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 172376, 247151]
- Geração: 13 - Fitness da população: [84868, 84977, 247151, 247151, 172376, 247151, 247151, 247151, 247151, 247151, 172376, 247151, 247151]
- Geração: 14 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 247151, 247151, 247151]
- Geração: 15 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 172376, 247151]
- Geração: 16 - Fitness da população: [84868, 84977, 172376, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 172376, 247151, 247151, 247151]
- Geração: 17 - Fitness da população: [84868, 84977, 247151, 247151, 172376, 247151, 247151, 247151, 172376, 172376, 247151, 247151, 172376]
- Geração: 18 - Fitness da população: [84868, 84977, 247151, 247151, 172376, 172376, 247151, 247151, 247151, 172376, 172376, 172376, 172376]
- Geração: 19 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 172376, 172376, 247151, 247151, 172376, 172376, 247151, 247151]
- Geração: 20 - Fitness da população: [84868, 84977, 247151, 172376, 247151, 172376, 172376, 247151, 172376, 247151, 247151, 172376, 172376]
- Geração: 21 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 172376, 172376, 172376, 172376]
- Geração: 22 - Fitness da população: [84868, 84977, 247151, 172376, 172376, 172376, 172376, 247151, 247151, 247151, 172376, 247151, 172376]
- Geração: 23 - Fitness da população: [84868, 84977, 172376, 172376, 247151, 247151, 247151, 247151, 247151, 172376, 247151, 172376, 172376]
- Geração: 24 - Fitness da população: [84868, 84977, 172376, 247151, 247151, 247151, 247151, 172376, 172376, 247151, 247151, 172376, 247151]

- Geração: 25 - Fitness da população: [84868, 84977, 247151, 247151, 172376, 247151, 247151, 172376, 172376, 172376, 247151, 247151, 247151]
- Geração: 26 - Fitness da população: [84868, 84977, 172376, 247151, 247151, 247151, 247151, 247151, 247151, 172376, 247151, 247151, 172376]
- Geração: 27 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 247151]
- Geração: 28 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 172376, 172376, 172376, 247151, 247151, 247151, 172376, 247151]
- Geração: 29 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 172376, 172376, 172376, 247151, 247151, 172376, 247151, 247151]
- Geração: 30 - Fitness da população: [84868, 84977, 172376, 247151, 247151, 247151, 172376, 247151, 247151, 172376, 247151, 247151, 172376]
- Geração: 31 - Fitness da população: [84868, 84977, 172376, 247151, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 247151, 247151]
- Geração: 32 - Fitness da população: [84868, 84977, 172376, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151]
- Geração: 33 - Fitness da população: [84868, 84977, 247151, 172376, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 247151, 172376]
- Geração: 34 - Fitness da população: [84868, 84977, 172376, 247151, 247151, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 247151]
- Geração: 35 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 247151, 247151, 247151]
- Geração: 36 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 172376, 247151, 247151]
- Geração: 37 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151]

- Geração: 38 - Fitness da população: [84868, 84977, 172376, 247151, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 247151, 247151]
- Geração: 39 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 172376, 247151, 247151, 247151]
- Geração: 40 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 247151]
- Geração: 41 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151]
- Geração: 42 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 172376, 247151, 247151, 247151, 247151, 172376, 247151, 247151]
- Geração: 43 - Fitness da população: [84868, 84977, 247151, 172376, 172376, 172376, 247151, 247151, 172376, 247151, 247151, 172376, 247151]
- Geração: 44 - Fitness da população: [84868, 84977, 247151, 247151, 247151, 172376, 172376, 247151, 172376, 247151, 247151, 247151, 247151]
- Geração: 45 - Fitness da população: [84868, 84977, 247151, 172376, 247151, 172376, 172376, 247151, 247151, 247151, 247151, 172376, 247151]
- Geração: 46 - Fitness da população: [84868, 84977, 172376, 247151, 247151, 172376, 247151, 247151, 172376, 247151, 247151, 172376, 247151]
- Geração: 47 - Fitness da população: [84868, 84977, 172376, 247151, 172376, 247151, 247151, 172376, 172376, 247151, 247151, 247151, 247151]
- Geração: 48 - Fitness da população: [84868, 84977, 247151, 172376, 172376, 172376, 247151, 247151, 247151, 172376, 247151, 247151, 247151]
- Geração: 49 - Fitness da população: [84868, 84977, 172376, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 247151, 172376, 247151, 172376]
- Geração: 50 - Fitness da população: [84868, 84977, 247151, 247151, 172376, 247151, 172376, 172376, 172376, 247151, 247151, 247151, 247151]
- Melhor fitness calculado: 84868
- Melhor solução encontrada:

- Antena 1: (129.80, 55.67)
- Antena 2: (594.43, 188.84)
- Antena 3: (491.37, 360.55)
- Antena 4: (249.59, 299.05)

- **Gráfico resultante:**



- **TESTE 3:**
  - **Interpretação:**
    - Já na primeira geração foi encontrada a melhor área de cobertura. Dada a informação de que a localização das antenas é criada de forma aleatória e o número relativamente baixo da população, mesmo com a mutação alta, essa realmente deve ser a melhor posição das antenas deste teste.

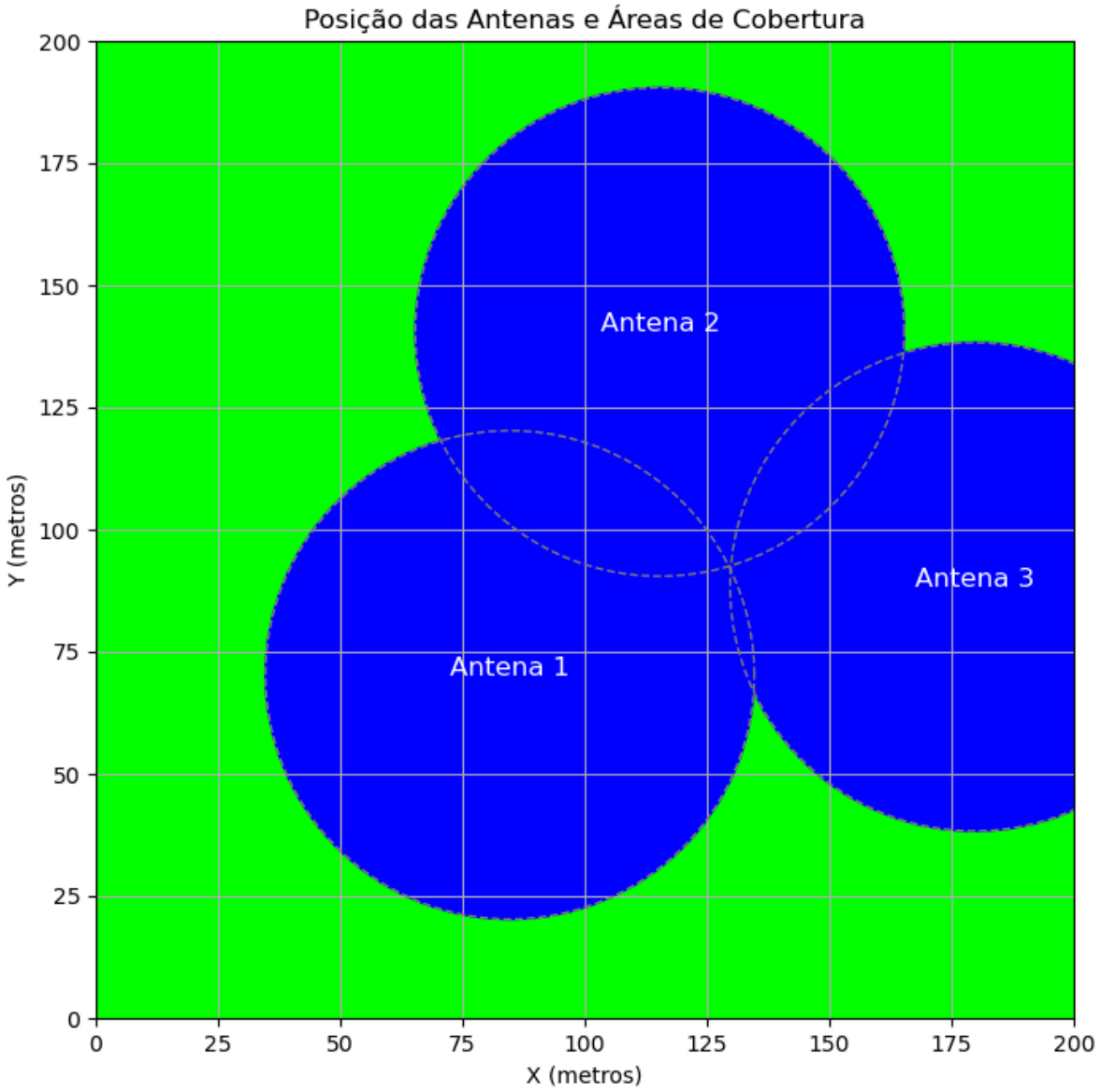
- **Constantes utilizadas:**
  - NUM_ANTENAS = 3
  - NUM_POPULACAO = 8
  - NUM_GERACOES = 50
  - NUM_ELITISMO = 2
  - AREA_LADO = 200
  - RAIO_ALCANCE = 50
  - DISTANCIA_MIN = 50
  - DISTANCIA_MAX = 100
  - PROB_MUTACAO = 0.7

- **Resultados Obtidos:**
  - Geração: 1 - Fitness da população: [174555, 207742, 187301, 241039, 185745, 249601, 208808, 218455]
  - Geração: 2 - Fitness da população: [174555, 185745, 194652, 219725, 213309, 193433, 194652, 243603]
  - Geração: 3 - Fitness da população: [174555, 185745, 219725, 224956, 213309, 213309, 224016, 215450]
  - Geração: 4 - Fitness da população: [174555, 185745, 243603, 219725, 243603, 224956, 224956, 220744]
  - Geração: 5 - Fitness da população: [174555, 185745, 243603, 232533, 266395, 250316, 243603, 243603]
  - Geração: 6 - Fitness da população: [174555, 185745, 250316, 219279, 266395, 266395, 266395, 232533]
  - Geração: 7 - Fitness da população: [174555, 185745, 232533, 266395, 266395, 266395, 243603, 250316]
  - Geração: 8 - Fitness da população: [174555, 185745, 266395, 232533, 232533, 232533, 266395, 224956]
  - Geração: 9 - Fitness da população: [174555, 185745, 232533, 266395, 232533, 266395, 232533, 266395]
  - Geração: 10 - Fitness da população: [174555, 185745, 266395, 266395, 196334, 266395, 266395, 232533]
  - Geração: 11 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 232533, 266395, 266395]
  - Geração: 12 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 232533]
  - Geração: 13 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 266395]
  - Geração: 14 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 266395]

- Geração: 15 - Fitness da população: [174555, 185745, 232533, 266395, 266395, 266395, 266395, 266395]
- Geração: 16 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 266395]
- Geração: 17 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 266395]
- Geração: 18 - Fitness da população: [174555, 185745, 232533, 266395, 266395, 266395, 266395, 266395]
- Geração: 19 - Fitness da população: [174555, 185745, 266395, 232533, 266395, 266395, 266395, 266395]
- Geração: 20 - Fitness da população: [174555, 185745, 266395, 232533, 266395, 266395, 266395, 266395]
- Geração: 21 - Fitness da população: [174555, 185745, 266395, 232533, 266395, 266395, 232533, 266395]
- Geração: 22 - Fitness da população: [174555, 185745, 232533, 232533, 266395, 266395, 266395, 266395]
- Geração: 23 - Fitness da população: [174555, 185745, 266395, 232533, 266395, 266395, 232533, 232533]
- Geração: 24 - Fitness da população: [174555, 185745, 266395, 232533, 232533, 232533, 266395, 266395]
- Geração: 25 - Fitness da população: [174555, 185745, 232533, 232533, 232533, 266395, 232533, 266395]
- Geração: 26 - Fitness da população: [174555, 185745, 232533, 266395, 232533, 232533, 266395, 266395]
- Geração: 27 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 232533]
- Geração: 28 - Fitness da população: [174555, 185745, 266395, 266395, 232533, 266395, 232533, 266395]
- Geração: 29 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 232533]
- Geração: 30 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 266395]
- Geração: 31 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 266395]
- Geração: 32 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 266395]
- Geração: 33 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 232533, 266395, 266395]
- Geração: 34 - Fitness da população: [174555, 185745, 266395, 266395, 232533, 266395, 266395, 266395]

- Geração: 35 - Fitness da população: [174555, 185745, 266395, 232533, 232533, 266395, 266395, 266395]
- Geração: 36 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 232533]
- Geração: 37 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 232533]
- Geração: 38 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 232533, 266395, 232533]
- Geração: 39 - Fitness da população: [174555, 185745, 266395, 232533, 266395, 266395, 232533, 266395]
- Geração: 40 - Fitness da população: [174555, 185745, 266395, 232533, 232533, 232533, 266395, 266395]
- Geração: 41 - Fitness da população: [174555, 185745, 266395, 266395, 232533, 266395, 266395, 266395]
- Geração: 42 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 232533, 232533, 266395]
- Geração: 43 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 232533]
- Geração: 44 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 232533, 232533]
- Geração: 45 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 232533]
- Geração: 46 - Fitness da população: [174555, 185745, 266395, 266395, 232533, 266395, 232533, 232533]
- Geração: 47 - Fitness da população: [174555, 185745, 232533, 232533, 266395, 266395, 266395, 266395]
- Geração: 48 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 232533, 266395, 266395]
- Geração: 49 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 266395]
- Geração: 50 - Fitness da população: [174555, 185745, 266395, 266395, 266395, 266395, 266395, 266395]
- Melhor fitness calculado: 174555
- Melhor solução encontrada:
- Antena 1: (84.69, 70.27)
- Antena 2: (115.33, 140.45)
- Antena 3: (179.70, 88.30)

- **Gráfico resultante:**



- **TESTE 4:**
  - **Interpretação:**
    - Este é o teste com a maior população, logo, com maior quantidade de distribuição de antenas no terreno, e também com uma taxa alta de mutação, que gera alterações na combinação entre os indivíduos. Dada a diferença do melhor fitness e a repetição quase padronizada entre eles, os valores provavelmente são muito similares pela distribuição geométrica no gráfico.
  - **Constantes utilizadas:**
    - `NUM_ANTENAS = 4`
    - `NUM_POPULACAO = 15`
    - `NUM_GERACOES = 30`
    - `NUM_ELITISMO = 1`

- AREA_LADO = 800
- RAIO_ALCANCE = 250
- DISTANCIA_MIN = 100
- DISTANCIA_MAX = 300
- PROB_MUTACAO = 0.8

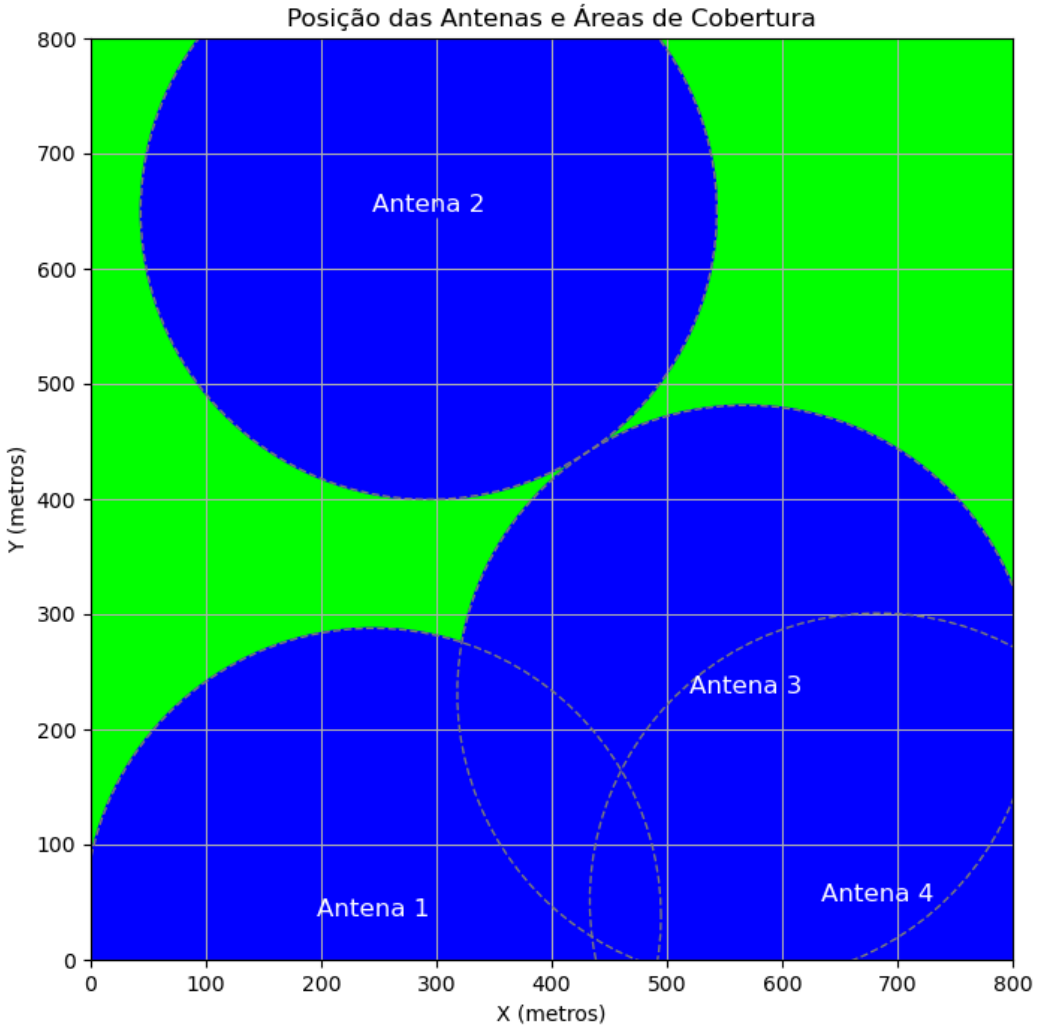
- **Resultados Obtidos:**
  - Geração: 1 - Fitness da população: [216098, 145772, 160547, 239397, 196697, 173573, 214567, 213843, 162007, 182195, 233673, 180103, 213864, 234348, 230655]
  - Geração: 2 - Fitness da população: [145772, 242553, 191067, 239397, 237765, 177315, 187508, 177801, 191342, 98015, 183737, 144681, 240945, 184730, 194359]
  - Geração: 3 - Fitness da população: [98015, 242553, 240945, 213094, 223411, 240945, 242553, 178942, 228825, 253970, 245409, 217159, 239397, 224838, 237765]
  - Geração: 4 - Fitness da população: [98015, 237476, 267541, 240945, 224838, 240945, 239397, 253970, 253970, 248290, 267541, 253970, 245409, 268474, 266842]
  - Geração: 5 - Fitness da população: [98015, 255283, 266842, 267541, 253970, 267541, 268474, 267541, 255283, 267541, 253970, 266842, 267541, 267541, 267541]
  - Geração: 6 - Fitness da população: [98015, 266842, 266842, 267541, 267541, 267541, 267541, 267541, 268474, 267541, 267541, 267541, 267541, 266842, 267541]
  - Geração: 7 - Fitness da população: [98015, 267541, 267541, 266842, 279433, 267541, 267541, 266842, 267541, 267541, 266842, 267541, 267541, 267541, 267541]
  - Geração: 8 - Fitness da população: [98015, 267541, 266842, 267541, 267541, 267541, 267541, 267541, 279433, 267541, 267541, 267541, 267541, 279433, 267541]

- Geração: 9 - Fitness da população: [98015, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541]
- Geração: 10 - Fitness da população: [98015, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 279433]
- Geração: 11 - Fitness da população: [98015, 267541, 267541, 267541, 279433, 279433, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541]
- Geração: 12 - Fitness da população: [98015, 267541, 279433, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 279433]
- Geração: 13 - Fitness da população: [98015, 267541, 279433, 267541, 267541, 267541, 267541, 267541, 279433, 279433, 267541, 279433, 267541, 267541, 279433]
- Geração: 14 - Fitness da população: [98015, 267541, 279433, 279433, 267541, 267541, 267541, 267541, 267541, 267541, 267541, 279433, 279433, 267541, 267541]
- Geração: 15 - Fitness da população: [98015, 279433, 279433, 279433, 267541, 267541, 267541, 267541, 279433, 267541, 267541, 279433, 267541, 267541, 267541]
- Geração: 16 - Fitness da população: [98015, 267541, 279433, 279433, 279433, 279433, 279433, 279433, 267541, 279433, 279433, 279433, 279433, 267541, 267541]
- Geração: 17 - Fitness da população: [98015, 279433, 267541, 279433, 279433, 267541, 279433, 267541, 267541, 267541, 267541, 279433, 267541, 267541, 267541]
- Geração: 18 - Fitness da população: [98015, 267541, 279433, 267541, 279433, 267541, 279433, 279433, 279433, 267541, 279433, 279433, 279433, 279433, 267541]

- Geração: 19 - Fitness da população: [98015, 279433, 279433, 267541, 279433, 267541, 267541, 279433, 267541, 279433, 279433, 267541, 267541, 267541, 267541]
- Geração: 20 - Fitness da população: [98015, 279433, 279433, 279433, 279433, 267541, 267541, 267541, 267541, 267541, 267541, 279433, 279433, 279433, 279433]
- Geração: 21 - Fitness da população: [98015, 267541, 267541, 267541, 267541, 279433, 279433, 279433, 279433, 279433, 279433, 267541, 267541, 279433, 267541]
- Geração: 22 - Fitness da população: [98015, 279433, 279433, 279433, 279433, 279433, 267541, 279433, 267541, 267541, 267541, 279433, 267541, 279433, 279433]
- Geração: 23 - Fitness da população: [98015, 267541, 279433, 279433, 279433, 279433, 279433, 279433, 267541, 279433, 279433, 279433, 279433, 267541, 279433]
- Geração: 24 - Fitness da população: [98015, 279433, 279433, 279433, 267541, 267541, 267541, 279433, 279433, 267541, 279433, 267541, 279433, 267541, 267541]
- Geração: 25 - Fitness da população: [98015, 279433, 267541, 279433, 279433, 279433, 279433, 267541, 279433, 267541, 279433, 279433, 279433, 279433, 267541]
- Geração: 26 - Fitness da população: [98015, 279433, 279433, 279433, 279433, 267541, 279433, 279433, 279433, 267541, 279433, 279433, 279433, 279433, 267541]
- Geração: 27 - Fitness da população: [98015, 279433, 279433, 279433, 279433, 267541, 279433, 279433, 279433, 279433, 279433, 279433, 267541, 267541, 267541]
- Geração: 28 - Fitness da população: [98015, 267541, 279433, 267541, 279433, 279433, 267541, 279433, 267541, 279433, 267541, 279433, 267541, 267541, 279433]

- Geração: 29 - Fitness da população: [98015, 279433, 279433, 279433, 279433, 267541, 267541, 279433, 267541, 267541, 267541, 267541, 267541, 279433, 267541]
- Geração: 30 - Fitness da população: [98015, 279433, 279433, 279433, 279433, 279433, 279433, 267541, 279433, 279433, 279433, 279433, 279433, 267541, 267541]
- Melhor fitness calculado: 98015
- Melhor solução encontrada:
- Antena 1: (244.88, 37.84)
- Antena 2: (293.31, 649.55)
- Antena 3: (568.02, 231.45)
- Antena 4: (683.01, 50.98)

- **Gráfico resultante:**



5. Conclusões

- **Resumo:**
  - Algoritmo genético é uma abordagem viável para otimização
  - Resultados promissores indicam potencial para aplicação em cenários reais.

"O uso de Algoritmos Genéticos mostrou-se eficaz na busca por uma configuração otimizada de posições de antenas, respeitando as restrições de distância mínima e máxima. A abordagem de inicialização aleatória seguida por seleção, crossover e mutação permite uma exploração ampla do espaço de soluções, encontrando uma configuração que maximiza a área de cobertura. A metodologia pode ser aplicada a problemas semelhantes de otimização espacial com restrições específicas".

6. Anexos

- **Código Fonte:**
  - Disponível no repositório GitHub: https://github.com/chlbnd/fiap_ia_para_devs/tree/master/fase_2

- **Dados:**
    - O conjunto de dados utilizado é gerado de forma aleatória.

7. Adições em relação ao vídeo

- As constantes podem ser alteradas para gerar outras áreas e quantidades de antena (obviamente atentando ao consumo de processamento) — A escolha da gravação foi feita pensando em simplificar a explicação;
- O elitismo não está aleatório, ele ordena o `fitness_pop` de forma crescente e pega os N indivíduos com o menor número, sendo esse N definido pela constante;
- O fitness, na verdade, calcula a área sem cobertura. Por isso, quanto menor o fitness, melhor;