Tech Challenge – Fase 2

Integrantes:

• Breno Robert de Oliveira Ribeiro

• Daniel Correia dos Reis

• Felipe Moura Monteiro

• João Alberto Dutra da Silveira Duarte

• Romulo Figueiredo Romano

Link do vídeo: https://www.youtube.com/watch?v=OxLNvJxT4DA

Link do Github: https://github.com/FelipeMouraMonteiro/TechChallenge_Fase2

Definição do Problema

Uma empresa de logística precisa otimizar a localização de seus Centros de Distribuição para atender um número X de cidades de forma eficiente. Essa eficiência está diretamente relacionada à distância entre os Centros de Distribuição e as cidades que eles atendem.

Os objetivos

Determinar a melhor localização para um número fixo de centros de distribuição dentro de uma área geográfica definida.

Minimizar a distância total percorrida para conectar cada cidade ao centro de distribuição mais próximo.

Os critérios de sucesso

Encontrar uma configuração de Centros de Distribuição que resulte no menor custo total de transporte (distância mínima).

Respeitar a restrição de que cada centro de distribuição pode atender apenas a um número X de cidades.

Implementação do Algoritmo

Bibliotecas Utilizadas

- **Pygame**: Usada para criar a interface gráfica e gerenciar eventos de usuário, como fechar a janela ou interagir com a simulação.
- **Random**: Utilizada para a geração de números aleatórios, essencial para operações de mutação e crossover no algoritmo genético.
- **Math**: Fornece funções matemáticas necessárias, como a função de raiz quadrada usada no cálculo de distâncias euclidianas.
- **Matplotlib**: Usada para criar gráficos que mostram a evolução da função de fitness ao longo das gerações.
- **Collections (defaultdict)**: Facilita a manipulação de dicionários ao proporcionar valores padrão para chaves que ainda não existem.

Configuração Inicial

As variáveis configuradas inicialmente incluem dimensões da tela para a simulação, número de cidades e centros de distribuição, o tamanho da população do algoritmo genético, o número de gerações a serem simuladas e a taxa de mutação.

Funcionalidades do Código

- Geração de Indivíduos
    - Cada indivíduo na população representa um possível conjunto de localizações para os centros de distribuição. A inicialização e as mutações são feitas com base em coordenadas aleatórias dentro dos limites definidos pela tela.

- Cálculo de Fitness
    - O fitness de cada indivíduo é calculado como a negação da soma das distâncias das cidades aos seus centros de distribuição mais próximos. A abordagem visa minimizar a distância total, portanto, melhores soluções têm fitness mais alto (menos negativo).

- Seleção e Reprodução
    - Utiliza-se um método de seleção baseado no fitness para escolher quais indivíduos passarão seus genes para a próxima geração. Crossovers e mutações são aplicadas para gerar diversidade genética.

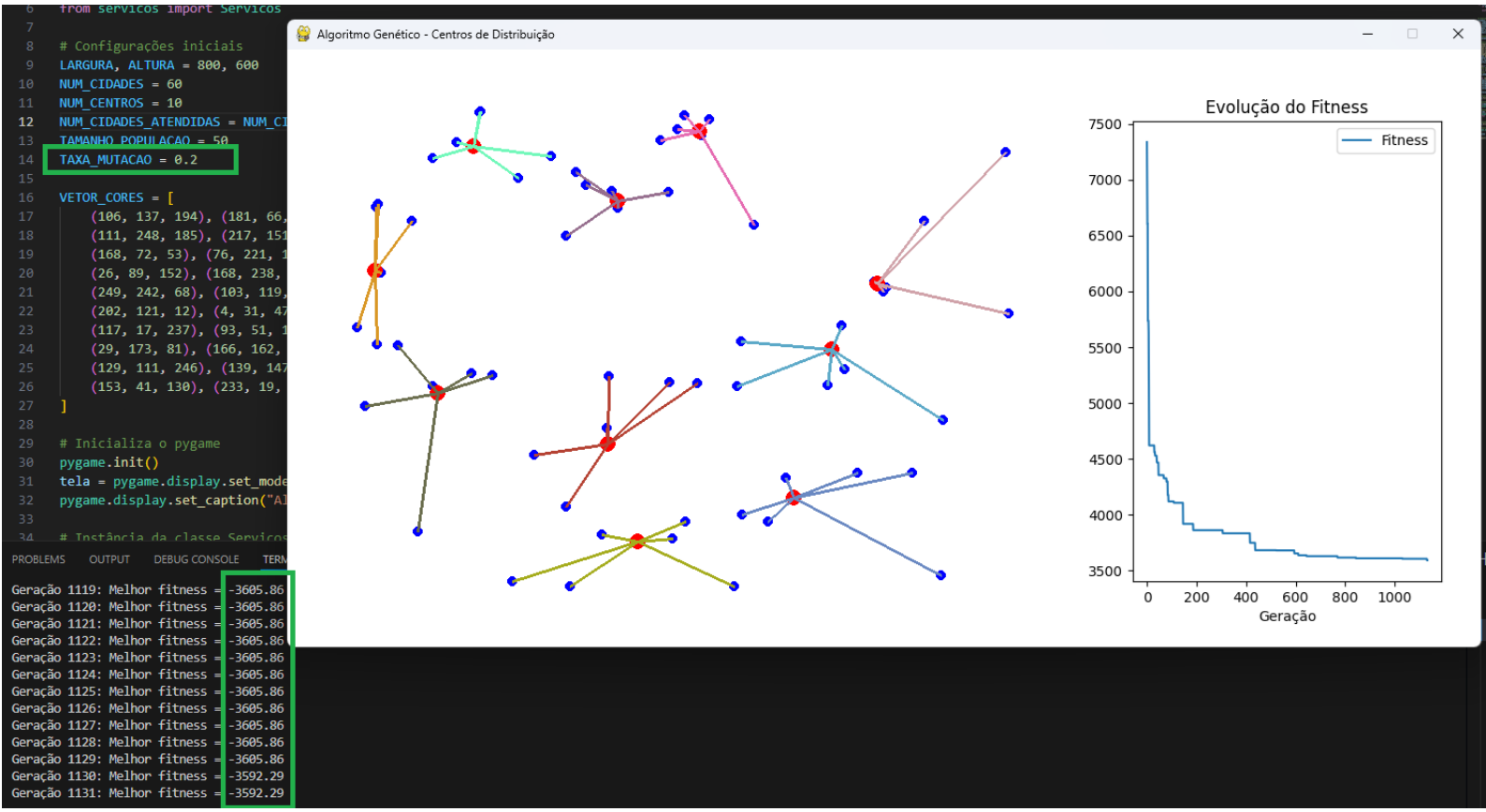
- Visualização
    - As posições das cidades e dos centros são representadas na tela com diferentes cores. Linhas são desenhadas conectando cada cidade ao seu centro de distribuição designado.

- Evolução do Fitness
    - A evolução do fitness é plotada em um gráfico ao lado da visualização principal, mostrando como o melhor fitness evolui com cada geração, oferecendo insights sobre a performance do algoritmo.

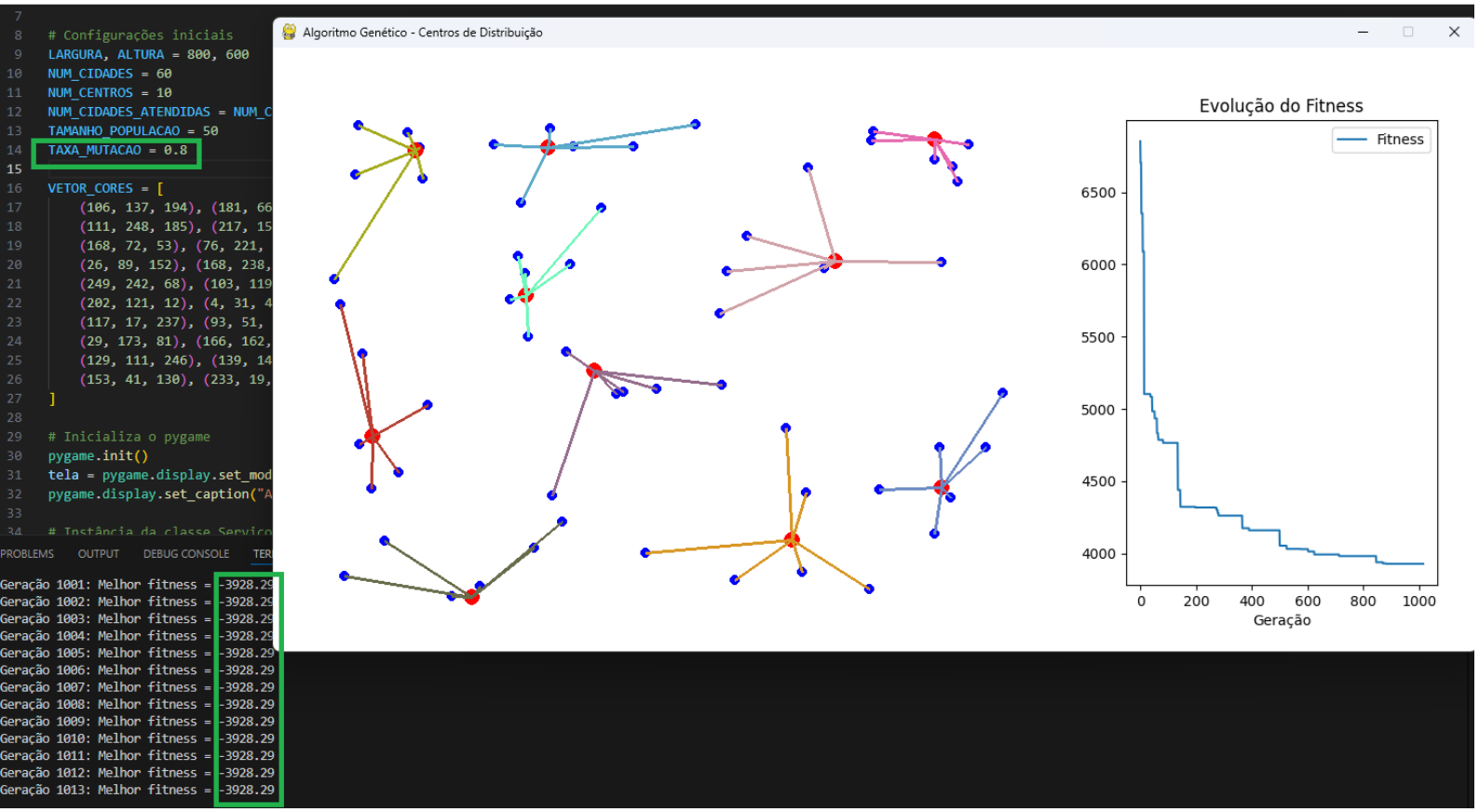
- Encerramento do Programa
    - O loop principal do algoritmo continua até que o usuário feche a janela do Pygame ou interrompa o processo através de uma entrada de teclado específica.

Testes

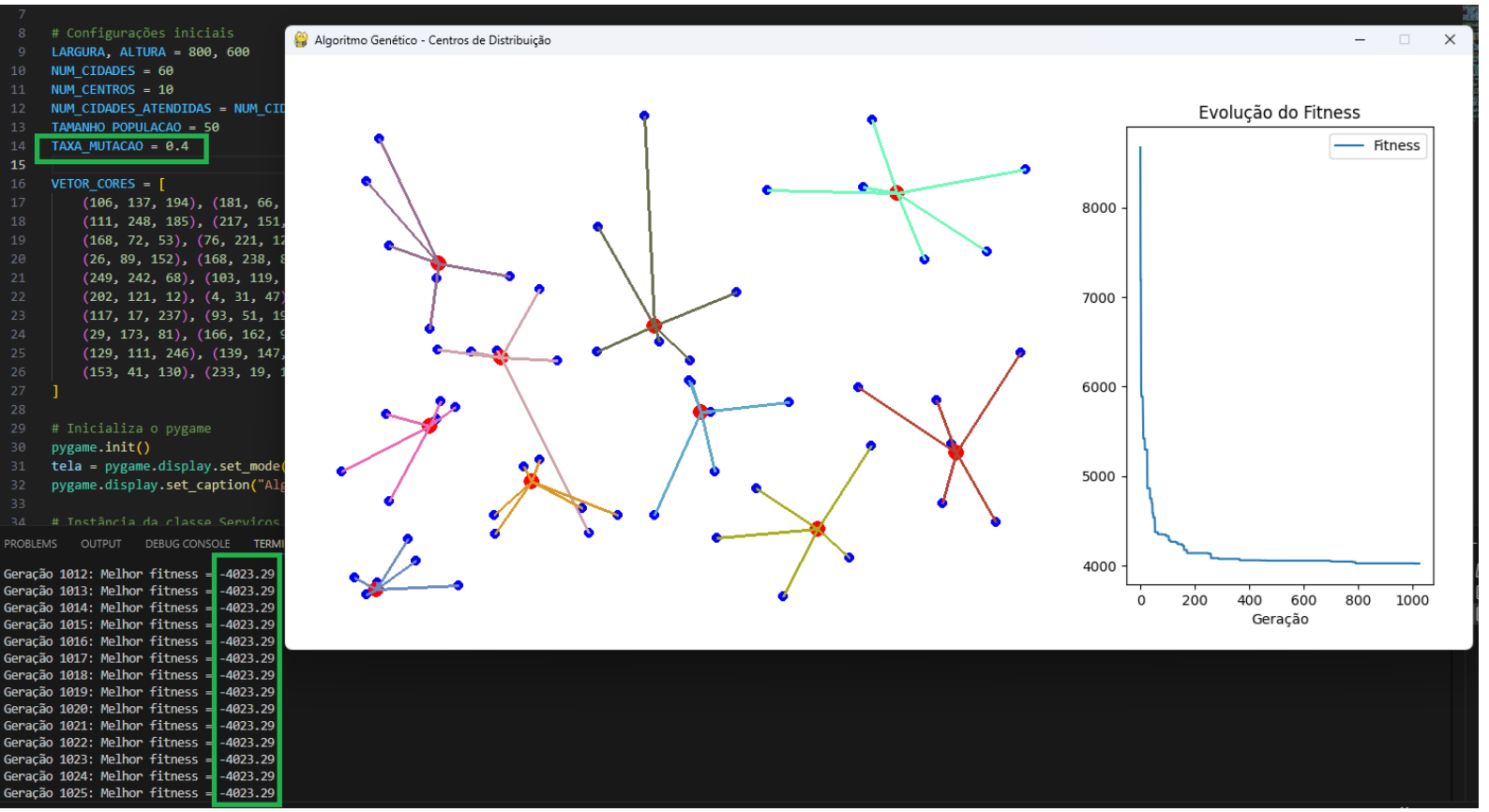
- Teste 1
  - Realizando o teste com os seguintes parâmetros:
    - Quantidade de Cidades Atendidas: **60**
    - Quantidade de Centros de Distribuição: **10**
    - Tamanho da População: **50**
    - Mutação: **0.2**
  - Analisando o teste podemos constatar que após Mil gerações, o algoritmo localizou um fitness de **3592,29**.



- Teste 2
  - Realizando o teste com os seguintes parâmetros:
    - Quantidade de Cidades Atendidas: **60**
    - Quantidade de Centros de Distribuição: **10**
    - Tamanho da População: **50**
    - Mutação: **0.8**
  - Analisando o teste podemos constatar que após Mil gerações, o algoritmo localizou um fitness de **3928,29**.



- Teste 3
  - Realizando o teste com os seguintes parâmetros:
    - Quantidade de Cidades Atendidas: **60**
    - Quantidade de Centros de Distribuição: **10**
    - Tamanho da População: **50**
    - Mutação: **0.4**
  - Analisando o teste podemos constatar que após Mil gerações, o algoritmo localizou um fitness de **4023,29**.



Conclusão

Com base nos testes realizados, foi possível constatar que a taxa de mutação tem um impacto significativo no desempenho do algoritmo genético. O Teste 1, com taxa de mutação de 0.2, apresentou o melhor fitness após 1000 gerações, demonstrando que um nível controlado de mutação é eficaz para evitar convergência prematura e garantir soluções otimizadas.