

## AEDS III - Trabalho Prático 4 - Heurística para um problema NP-Difícil

José Argemiro dos Reis Neto (2021.1.08.011)  
Otávio Augusto Marcelino Izidoro (2018.1.08.041)  
Pedro Augusto Mendes (2021.1.08.041)  
Rikson Pablo Gomes da Rocha (2022.2.08.007)  
Victor Ribeiro Gonzalez (2021.1.08.023)

29 de Maio 2024

# Introdução

O problema do caixeiro viajante é um dos problemas mais conhecidos e estudados na área de otimização combinatória. O problema consiste em descobrir o caminho mais curto que um vendedor tem de andar para ir exatamente uma vez a todas as cidades e a seguir para a cidade de onde partiu. Esses três algoritmos oferecem diferentes abordagens para resolver o problema do labirinto. Cada um tem suas vantagens e desvantagens, oferecendo opções para resolver o problema de forma eficaz

# Introdução

Para enfrentar esse desafio, muitas abordagens heurísticas e meta-heurísticas surgiram para produzir boas soluções para o TSP em um tempo limitado. Entre essas abordagens, os algoritmos evolutivos e de busca local são um dos mais populares quando se trata de problemas de otimização combinatória. Nesse contexto, este trabalho visa aplicar o algoritmo evolutivo de busca local ao problema do caixeiro viajante.

# Algoritmo

O algoritmo apresentado é um algoritmo de busca local para resolver o Problema do Caixeiro Viajante (PCV). Ele utiliza a heurística 2-opt para tentar melhorar iterativamente uma rota inicialmente gerada aleatoriamente.

# Algoritmo

```

algoritmo 2-Opt (  $s$  )
  para (  $i$  de 1 até  $n_r$  ) faça
     $j \leftarrow i + 2$ 
    enquanto (  $((j + 1) \bmod n_r) \neq i$  ) faça
      se (  $c_{i,i+1} + c_{j,j+1} - c_{i,j} - c_{i+1,j+1} > 0$  ) então
        { trocar as arestas  $(i,i+1)$  e  $(j,j+1)$  por  $(i,j)$  e  $(i+1,j+1)$  }
         $inicio \leftarrow (i + 1) \bmod n_r$ 
         $fim \leftarrow j$ 
        se (  $inicio > fim$  ) então
           $tam = n_r - inicio + fim + 1$ 
        senão  $tam = fim - inicio + 1$ 
         $p1 \leftarrow inicio$  e  $p2 \leftarrow fim$ 
        para (  $k$  de 1 até  $tam/2$  ) faça
          troque os vértices  $p1$  e  $p2$  de posição
           $p1 \leftarrow p1 + 1$ 
           $p2 \leftarrow p2 - 1$ 
        fim-para
         $f(s) \leftarrow f(s) + c_{i,i+1} + c_{j,j+1} - c_{i,j} - c_{i+1,j+1}$ 
      fim-se
    fim-enquanto
  fim-para
fim-algoritmo

```

# Algoritmo Genético

Os Algoritmos Genéticos são técnicas de otimização inspiradas no processo de seleção natural. Eles envolvem a geração de uma população inicial de soluções candidatas, a aplicação de operadores genéticos (crossover e mutação) para criar novas soluções e a avaliação dessas soluções de acordo com um critério de aptidão.

Heurísticas Desenvolvidas:

- Inicialização da População
- Avaliação da População
- Seleção de Pais
- Crossover
- Mutação

# Algoritmo Genético

Criar as populações Pop, MatingPool, Pais, Filhos;

Inicialização(Pop);

**para** *Cada geração até MaxGen faça*

    Avaliação(Pop);

    Penalidade(Pop);

    Aptidão(Pop);

    MatingPool  $\leftarrow$  Seleção(Pop);

    Pais  $\leftarrow$  Seleção(MatingPool);

    Filhos  $\leftarrow$  Cruzamento(Pais);

    Mutação(Filhos);

    TrocaLâmina(Filhos);

    AdiçãoLâmina(Filhos);

    EliminaçãoLâmina(Filhos);

    Pop  $\leftarrow$  Filhos

**fim**


# Descrição das instâncias


- Definição do Número de Cidades
- Atribuição de Coordenadas
- Distribuição Aleatória
- Finalidade das Instâncias



# Descrição das instâncias

Exemplo de uma instância

 tsp10.txt

1	10	20
2	30	50
3	50	30
4	70	60
5	20	70
6	30	40
7	60	80
8	90	20
9	80	50
10	 10	10
11		

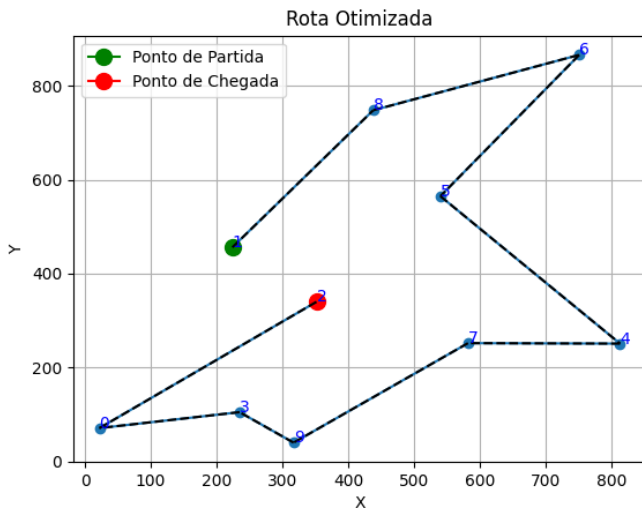
# Resultados

## Busca Local

```
PS C:\Users\jdosr\Downloads\trabalho4> .\tsp_busca_local.exe .\tsp10.txt
Iteracao 1:
Rota inicial: 3 2 9 0 6 8 7 4 5 1
Distancia inicial: 4125.78
Rota otimizada: 1 8 6 5 4 7 9 3 0 2
Distancia otimizada: 2970.87
Tempo de execucao: 0.0000 segundos

Rota otimizada salva em 'rota_otimizada.txt' (Iteracao 1).
Iteracao 2:
Rota inicial: 3 2 1 5 6 9 4 8 7 0
Distancia inicial: 4553.93
Rota otimizada: 5 2 1 0 3 9 7 4 6 8
Tempo de execucao: 0.0010 segundos
Iteracao 3:
Rota inicial: 2 6 5 9 1 0 4 7 8 3
Distancia inicial: 4954.68
Rota otimizada: 3 0 1 8 6 5 4 7 2 9
Distancia otimizada: 3015.14
Tempo de execucao: 0.0000 segundos
```

# Resultados



# Resultados

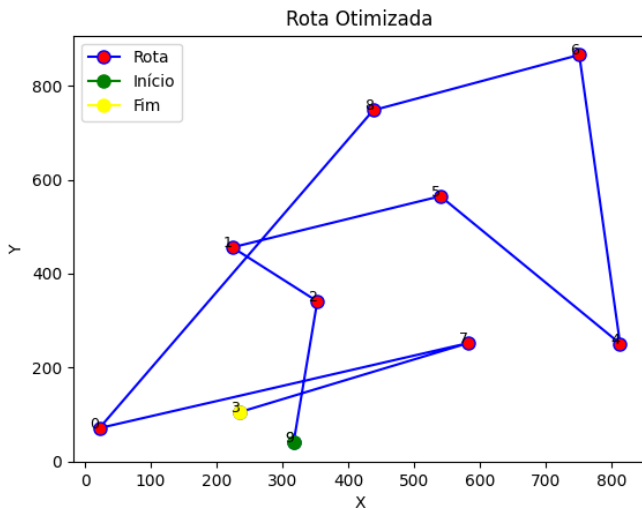
Evolutivo: O código captura a instância informada e executa três vezes para fornecer uma noção do estado em cada iteração.

```
P5 C:\Users\jdosr\Downloads\trabalho4> .\tsp_evolutivo.exe .\tsp10.txt
Rota inicial:
22 71
224 456
353 341
235 105
813 251
540 565
751 866
582 252
438 748
317 40
Iteracao 1:
Melhor solucao encontrada:
Rota otimizada:
9 2 1 5 4 6 8 0 7 3
Distancia otimizada: 4043.55
Tempo de execucao: 0.1850 segundos
Tempo de execucao: 54.9410 segundos

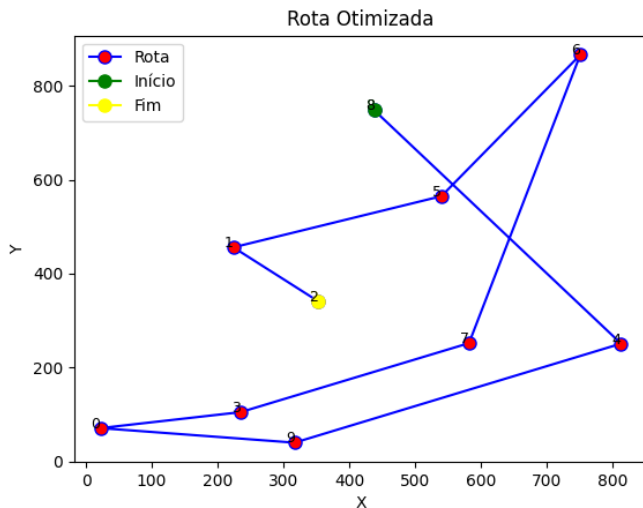
Iteracao 2:
Melhor solucao encontrada:
Rota otimizada:
8 4 9 0 3 7 6 5 1 2
Distancia otimizada: 3978.08
Tempo de execucao: 0.1870 segundos
Tempo de execucao: 11.3150 segundos

Iteracao 3:
Melhor solucao encontrada:
Rota otimizada:
4 7 9 3 0 2 1 8 5 6
Distancia otimizada: 3047.91
Tempo de execucao: 0.2260 segundos
Tempo de execucao: 12.8560 segundos
```

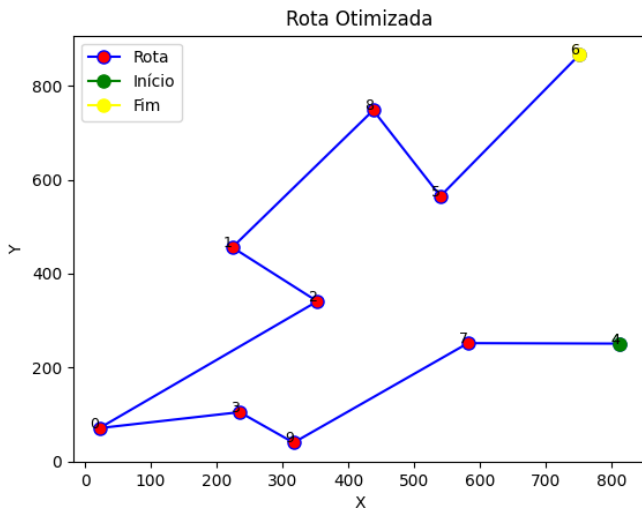
# Resultados



# Resultados



# Resultados



# Resultados

## Execução com 50 cidades

```
PS C:\Users\jdosr\Downloads\trabalho4> .\tsp_busca_local.exe .\tsp50.txt
```

```
Iteracao 1:
```

```
Rota inicial: 5 14 19 4 23 32 30 48 36 29 10 26 34 18 0 38 22 43 9 35 17 21 12 45 44 20 39 42 25 2 8 11 37 27 40 33 24 28 1 46 6 15 7 49 47 16 41 13 3 31
```

```
Distancia inicial: 2363.46
```

```
Rota otimizada: 35 25 15 49 39 29 19 9 0 10 20 30 40 5 1 11 21 31 41 4 14 24 34 44 6 16 26 36 46 43 33 23 13 3 8 18 28 38 48 47 37 27 17 7 42 32 22 12 2 45
```

```
Distancia otimizada: 362.40
```

```
Tempo de execucao: 0.0780 segundos
```

```
Rota otimizada salva em 'rota_otimizada.txt' (Iteracao 1).
```

```
Iteracao 2:
```

```
Rota inicial: 1 5 35 36 6 41 23 26 11 32 16 31 24 15 3 14 27 49 20 48 9 17 33 39 46 38 37 2 21 13 43 0 25 4 30 42 7 47 40 12 29 8 19 10 22 34 44 18 45 28
```

```
Distancia inicial: 2199.55
```

```
Rota otimizada: 23 33 43 46 36 26 16 6 44 34 24 14 4 1 11 21 31 41 45 35 25 15 5 40 30 20 10 0 9 19 29 39 49 2 12 22 32 42 7 17 27 37 47 48 38 28 18 8 3 13
```

```
Distancia otimizada: 361.56
```

```
Tempo de execucao: 0.1360 segundos
```

```
Iteracao 3:
```

```
Rota inicial: 20 17 15 21 7 3 28 27 41 11 35 5 34 19 18 43 48 10 24 2 42 47 37 32 49 31 38 29 46 22 33 13 23 26 12 36 44 0 40 6 45 16 9 1 14 25 8 4 39 30
```

```
Distancia inicial: 2057.86
```

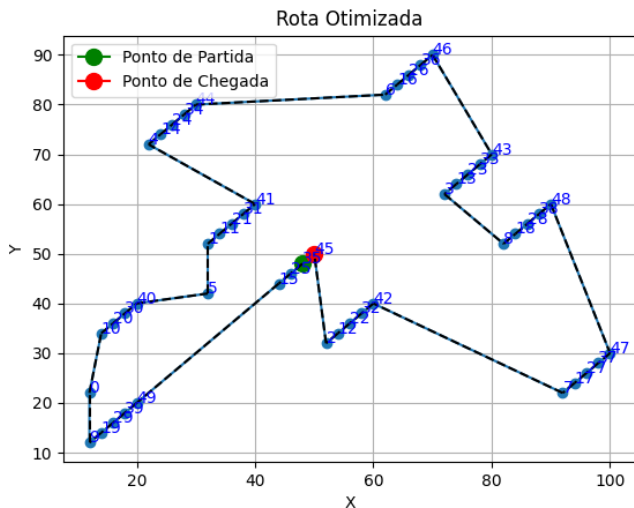
```
Rota otimizada: 39 29 19 9 0 10 20 30 40 4 14 24 34 44 6 16 26 36 46 43 33 23 13 3 8 18 28 38 48 47 37 27 17 7 42 32 22 12 2 25 35 45 41 31 21 11 1 15 5 49
```

```
Distancia otimizada: 376.80
```

```
Tempo de execucao: 0.1400 segundos
```



# Resultados



# Referências

- Problema do caixeiro-viajante  
[https://pt.wikipedia.org/wiki/Problema\\_do\\_caixeiro-viajante](https://pt.wikipedia.org/wiki/Problema_do_caixeiro-viajante)", Acesso em 29 de maio de 2024.
- Pseudocodigo-da-heuristica-2-Opt  
[https://www.researchgate.net/figure/Figura-56-Pseudocodigo-da-heuristica-2-Opt\\_fig11\\_41804451](https://www.researchgate.net/figure/Figura-56-Pseudocodigo-da-heuristica-2-Opt_fig11_41804451), Acesso em 29 de maio de 2024.
- Pseudo-codigo-do-Algoritmo-Genetico  
[https://www.researchgate.net/figure/Figura-86-Pseudo-codigo-do-Algoritmo-Genetico\\_fig33\\_331829005](https://www.researchgate.net/figure/Figura-86-Pseudo-codigo-do-Algoritmo-Genetico_fig33_331829005), Acesso em 29 de maio de 2024.