Projeto 1 - Problema N-Rainhas

Autor: Marcos Angelo Cemim

(i) Operadores que poderão ser utilizados:

- Cruzamento:
  -- cxOnePoint()
  -- cxTwoPoint()
  -- cxUniform()

- Mutação:
  -- mutShuffleIndexes()
  -- mutUniformInt()

- Seleção:
  -- selTournament()
  -- selRoulette()

(ii) Formas de execução:

- Rodar pelo menos 2 (duas) combinações diferentes entre os operadores (Seleção, Cruzamento e Mutação)
- Apresentar as soluções obtidas para cada combinação.
- Desenvolver o Algoritmo Genético com o framework DEAP
- Executar as combinações para 10, 15 e 20 Rainhas

Encontrando todas as combinações possíveis dos parâmetros

In [ ]:
```
parameters_list = {
    'p1': ['OnePoint', 'TwoPoints', 'Uniform'],      # Cruzamentos
    'p2': ['Shuffleindexes', 'UniformInt'],          # Mutação
    'p3': ['Tournament', 'Roulette']                 # Seleção
}
```

In [ ]:
```
from itertools import product

def encontrar_combinacoes(dicionario):
    chaves = dicionario.keys()
    valores = [dicionario[chave] for chave in chaves]
    todas_combinacoes = list(product(*valores))
    return todas_combinacoes
```

In [ ]:
```
lista_parametros = encontrar_combinacoes(parameters_list)
```

Importação das bibliotecas

In [ ]:
```
import random
from deap import base, creator, tools, algorithms
import warnings
import numpy as np
warnings.filterwarnings("ignore")
import time
```

Implementação do algoritmo

Funçao de avaliação (fitness)

In [ ]:
```
# Defina a função de aptidão
def fitness_function(solucao):
        h = 0
        #Contagem de ataques na diagonal e vertical
        for i in range(0, len(solucao)):
            for j in range(0, len(solucao)):
                if j > i:
                    # Avalia a diferença entre as colunas e as posições
                    # das rainhas dentro da coluna
                    if abs(i - j) == abs(solucao[i] - solucao[j]):
                       #  print(f'{i} - {j} - {solucao[i]}-{solucao[j]}')
                       h += 1
                    # Ataques por linha (horizontal)
                    # Avalia apenas as posições das rainhas
                    if abs(solucao[i] - solucao[j]) == 0:
                       h += 1
        return h,

# Defina a função de aptidão "inversa"
# Por definição, a função de seleçao "selRoulette" nãqo pode ser utilizada para
# Portanto, vamos criar uma função "inversa" para avaliar
def fitness_function_inv(solucao):
        h = 0
        melhorFitness = (len(solucao) * (len(solucao) - 1) // 2)
        #Contagem de ataques na diagonal e vertical
        for i in range(0, len(solucao)):
            for j in range(0, len(solucao)):
                if j > i:
                    # Avalia a diferença entre as colunas e as posições
                    # das rainhas dentro da coluna
                    if abs(i - j) == abs(solucao[i] - solucao[j]):
                       #  print(f'{i} - {j} - {solucao[i]}-{solucao[j]}')
                       h += 1
                    # Ataques por linha (horizontal)
                    # Avalia apenas as posições das rainhas
                    if abs(solucao[i] - solucao[j]) == 0:
                       h += 1
        return melhorFitness - h,
```

Parâmetros gerais (dfefinidos arbitrariamente)

In [ ]:
```
pop_size = 100  # Tamanho da população
cxpb = 0.6      # Probabilidade de cruzamento
mutpb = 0.1     # Probabilidade de mutação
```

```
ngen = 2000      # Número de gerações
tourn_size = 3   # Indivíduos participantes do 'torunament'
indpb = 0.05     # Independent probability for each attribute to be exchanged to
```

Rodando todas as combinações em todos os cenários sugeridos para avaliar os melhores desempenhos

In [ ]:

```
for N in [10, 15, 20]:
    best_fit = 10000
    best_sol = ''
    best_param = ''
    print(f"{'-'*50} {N} RAINHAS {'-'*50}")
    for param in lista_parametros:
        inicio = time.time()
        print(f'{str(param):45s}', end= ' | ')
        # Configurar a estrutura DEAP para minimizar

        # Gera o objetivo toolbox responsável por registrar as configurações do
        toolbox = base.Toolbox()


        if param[2] == 'Roulette':
            # Cria o tipo de função fitness e indivíduo
            creator.create("maximizar", base.Fitness, weights=(1.0,))
            creator.create("individual", list, fitness=creator.maximizar)

        else:
            # Cria o tipo de função fitness e indivíduo
            creator.create("minimizar", base.Fitness, weights=(-1.0,))
            creator.create("individual", list, fitness=creator.minimizar)

        # Registra os nomes e os tipos de individuo, fitness e população
        toolbox.register("atributo", random.randint, 0, 1)
        toolbox.register("solucaoFinal", tools.initRepeat, creator.individual, t
        toolbox.register("Populacao", tools.initRepeat, list, toolbox.solucaoFin


        # Define a criação de indivíduos e populações
        toolbox = base.Toolbox()
        toolbox.register("permutation", random.sample, range(N), N)
        toolbox.register("individual", tools.initIterate, creator.individual, tc
        toolbox.register("population", tools.initRepeat, list, toolbox.individua

        # Definir operadores genéticos
        ## Cruzamento
        if param[0] == 'OnePoint':
            toolbox.register("mate", tools.cxOnePoint)
        elif param[0] == 'TwoPoints':
            toolbox.register("mate", tools.cxTwoPoint)
        elif param[0] == 'Uniform':
            toolbox.register("mate", tools.cxUniform, indpb=indpb)

        ## Mutação
        if param[1] == 'Shuffleindexes':
            toolbox.register("mutate", tools.mutShuffleIndexes, indpb=indpb)
        elif param[1] == 'UniformInt':
            toolbox.register("mutate", tools.mutUniformInt, low=0, up=N, indpb=i

        ## Seleção
```

```
        if param[2] == 'Tournament':
            toolbox.register("select", tools.selTournament, tournsize=tourn_size
            toolbox.register('evaluate', fitness_function)
        elif param[2] == 'Roulette':
            toolbox.register("select", tools.selRoulette)
            toolbox.register('evaluate', fitness_function_inv)

        pop = toolbox.population(n=pop_size)
        hof = tools.HallOfFame(10)
        stat = tools.Statistics(lambda ind: ind.fitness.values)
        stat.register("Melhor Solucao", np.min)
        stat.register("Media da Populacao", np.mean)
        finalPop, log = algorithms.eaSimple(pop, toolbox, cxpb, mutpb, ngen, sta
        best_solution = tools.selBest(finalPop, 1)[0]
        new_fit = fitness_function(best_solution)[0]
        best_sol = best_solution if new_fit < best_fit else best_sol
        best_param = str(param) if new_fit < best_fit else best_param
        best_fit = new_fit if new_fit < best_fit else best_fit
        tempo = time.time()-inicio
        print(f"{tempo:10.2f} s | Melhor solução: {best_solution} @ {new_fit}")
    print("-"*180)
    print(f"Melhor solução: {best_param} com {best_fit} ataques")
```

```
------------------------------------------------ 10 RAINHAS ------------------
-------------------------------
('OnePoint', 'Shuffleindexes', 'Tournament')  |       4.70 s | Melhor solução:
[2, 0, 9, 6, 4, 9, 1, 8, 5, 7] @ 1
('OnePoint', 'Shuffleindexes', 'Roulette')    |       8.28 s | Melhor solução:
[5, 8, 0, 7, 7, 2, 8, 3, 9, 3] @ 4
('OnePoint', 'UniformInt', 'Tournament')      |       4.69 s | Melhor solução:
[9, 6, 8, 1, 4, 7, 0, 10, 5, 2] @ 0
('OnePoint', 'UniformInt', 'Roulette')        |       8.42 s | Melhor solução:
[0, 9, 9, 6, 10, 2, 5, 1, 4, 7] @ 2
('TwoPoints', 'Shuffleindexes', 'Tournament') |       4.79 s | Melhor solução:
[5, 9, 4, 1, 8, 6, 2, 0, 7, 9] @ 1
('TwoPoints', 'Shuffleindexes', 'Roulette')   |       8.40 s | Melhor solução:
[7, 9, 0, 8, 1, 3, 8, 6, 1, 3] @ 4
('TwoPoints', 'UniformInt', 'Tournament')     |       4.74 s | Melhor solução:
[6, 4, 2, 0, 3, 10, 8, 5, 9, 1] @ 0
('TwoPoints', 'UniformInt', 'Roulette')       |       8.44 s | Melhor solução:
[7, 2, 8, 6, 9, 7, 10, 1, 3, 5] @ 1
('Uniform', 'Shuffleindexes', 'Tournament')   |       4.50 s | Melhor solução:
[5, 7, 4, 1, 3, 9, 6, 8, 2, 0] @ 0
('Uniform', 'Shuffleindexes', 'Roulette')     |       8.38 s | Melhor solução:
[8, 4, 8, 0, 9, 4, 8, 5, 9, 2] @ 5
('Uniform', 'UniformInt', 'Tournament')       |       4.71 s | Melhor solução:
[7, 5, 3, 0, 9, 4, 2, 10, 6, 1] @ 0
('Uniform', 'UniformInt', 'Roulette')         |       8.34 s | Melhor solução:
[0, 7, 3, 6, 9, 2, 0, 5, 10, 8] @ 1
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------
Melhor solução: ('OnePoint', 'UniformInt', 'Tournament') com 0 ataques
------------------------------------------------ 15 RAINHAS ------------------
-------------------------------
('OnePoint', 'Shuffleindexes', 'Tournament')  |       7.19 s | Melhor solução: [1
0, 3, 13, 2, 8, 6, 1, 1, 6, 0, 7, 14, 12, 9, 4] @ 2
('OnePoint', 'Shuffleindexes', 'Roulette')    |      11.03 s | Melhor solução:
[0, 9, 0, 9, 1, 13, 10, 2, 9, 11, 1, 2, 0, 9, 13] @ 15
('OnePoint', 'UniformInt', 'Tournament')      |       7.08 s | Melhor solução:
[2, 8, 14, 11, 1, 5, 9, 12, 0, 13, 3, 10, 6, 4, 15] @ 0
('OnePoint', 'UniformInt', 'Roulette')        |      11.07 s | Melhor solução: [1
5, 13, 7, 3, 12, 8, 14, 5, 3, 10, 2, 0, 6, 4, 13] @ 5
('TwoPoints', 'Shuffleindexes', 'Tournament') |       7.30 s | Melhor solução: [1
1, 2, 8, 1, 9, 13, 6, 14, 0, 5, 12, 4, 7, 10, 3] @ 0
('TwoPoints', 'Shuffleindexes', 'Roulette')   |      10.89 s | Melhor solução:
[4, 7, 12, 0, 2, 14, 13, 12, 14, 14, 0, 4, 13, 0, 2] @ 15
('TwoPoints', 'UniformInt', 'Tournament')     |       7.15 s | Melhor solução:
[8, 13, 2, 9, 7, 14, 0, 11, 15, 1, 3, 6, 10, 12, 4] @ 1
('TwoPoints', 'UniformInt', 'Roulette')       |      10.80 s | Melhor solução:
[4, 8, 11, 5, 3, 9, 2, 10, 13, 6, 0, 0, 14, 5, 1] @ 6
('Uniform', 'Shuffleindexes', 'Tournament')   |       6.99 s | Melhor solução:
[6, 3, 0, 11, 13, 5, 1, 14, 12, 2, 9, 7, 4, 10, 8] @ 0
('Uniform', 'Shuffleindexes', 'Roulette')     |      10.90 s | Melhor solução: [1
0, 10, 0, 14, 5, 4, 2, 14, 10, 2, 2, 5, 14, 0, 0] @ 16
('Uniform', 'UniformInt', 'Tournament')       |       6.71 s | Melhor solução:
[7, 4, 8, 5, 15, 10, 0, 2, 9, 13, 6, 3, 1, 11, 14] @ 0
('Uniform', 'UniformInt', 'Roulette')         |      10.28 s | Melhor solução:
[1, 14, 2, 15, 15, 0, 10, 6, 13, 6, 0, 5, 13, 9, 12] @ 7
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------
Melhor solução: ('OnePoint', 'UniformInt', 'Tournament') com 0 ataques
```

```
------------------------------------------------ 20 RAINHAS ------------------
------------------------------
('OnePoint', 'Shuffleindexes', 'Tournament')  |       9.72 s | Melhor solução:
[6, 14, 5, 10, 13, 0, 8, 4, 19, 17, 11, 7, 16, 12, 2, 9, 3, 15, 18, 1] @ 0
('OnePoint', 'Shuffleindexes', 'Roulette')    |      13.42 s | Melhor solução:
[7, 16, 16, 18, 15, 18, 16, 2, 7, 18, 2, 7, 18, 15, 19, 3, 16, 7, 19, 3] @ 27
('OnePoint', 'UniformInt', 'Tournament')      |       9.70 s | Melhor solução: [1
2, 16, 5, 11, 15, 19, 2, 4, 18, 9, 0, 20, 13, 8, 1, 7, 14, 3, 17, 10] @ 0
('OnePoint', 'UniformInt', 'Roulette')        |      13.83 s | Melhor solução: [1
1, 13, 3, 12, 9, 13, 1, 6, 15, 3, 0, 8, 2, 19, 0, 6, 14, 16, 4, 7] @ 11
('TwoPoints', 'Shuffleindexes', 'Tournament') |      10.04 s | Melhor solução: [1
6, 8, 10, 17, 4, 2, 9, 3, 14, 14, 1, 10, 5, 15, 12, 19, 11, 18, 0, 6] @ 2
('TwoPoints', 'Shuffleindexes', 'Roulette')   |      13.44 s | Melhor solução:
[1, 6, 16, 7, 12, 1, 7, 15, 3, 15, 16, 10, 3, 7, 3, 10, 10, 12, 1, 3] @ 27
('TwoPoints', 'UniformInt', 'Tournament')     |      10.05 s | Melhor solução:
[5, 15, 19, 4, 11, 13, 6, 20, 12, 8, 3, 0, 2, 9, 16, 10, 7, 14, 1, 17] @ 0
('TwoPoints', 'UniformInt', 'Roulette')       |      13.52 s | Melhor solução:
[0, 16, 19, 2, 13, 17, 5, 18, 15, 4, 16, 9, 13, 18, 20, 3, 20, 11, 11, 6] @ 11
('Uniform', 'Shuffleindexes', 'Tournament')   |       9.97 s | Melhor solução:
[3, 13, 10, 18, 1, 19, 2, 12, 17, 11, 8, 0, 4, 14, 9, 15, 6, 16, 5, 7] @ 1
('Uniform', 'Shuffleindexes', 'Roulette')     |      13.36 s | Melhor solução: [1
0, 17, 11, 3, 19, 16, 19, 3, 10, 2, 16, 2, 19, 17, 2, 11, 16, 16, 17, 17] @ 28
('Uniform', 'UniformInt', 'Tournament')       |       9.84 s | Melhor solução:
[2, 15, 8, 10, 3, 18, 6, 4, 20, 17, 19, 14, 2, 5, 7, 9, 1, 13, 16, 0] @ 1
('Uniform', 'UniformInt', 'Roulette')         |      13.28 s | Melhor solução:
[5, 16, 5, 9, 3, 16, 19, 14, 8, 5, 3, 8, 20, 2, 15, 19, 7, 11, 2, 17] @ 10
------------------------------------------------------------------------------
------------------------------------------------------------------------------
-----------------
Melhor solução: ('OnePoint', 'Shuffleindexes', 'Tournament') com 0 ataques
```

CONCLUSÃO

Para cada um dos cenários solicitados (N = 10, 15 e 20), as configuração otimizadas são ligeiramente diferentes.

Além disso, para cenários com menor complexidade e tamanho dos vetores de resposta, seria possível otimizar os parâmetros de tamanho de população e numero de gerações.

A conclusão final é de que para cada cenário e aplicação, uma nova configuração é indicada para otimizar o algoritmo.

Para cenários parecidos, os ganhos com diferentes configurações não ficam tão expressivos.