APLICANDO ALGORITMOS GENÉTICOS AO PROBLEMA DAS N RAINHAS

Victor Hugo Zeschau
Instituto Federal Catarinense
hugozeschau@hotmail.com

1. Introdução

O **problema das n rainhas** é um desafio matemático intrigante e complexo que envolve o posicionamento estratégico de n rainhas em um tabuleiro de xadrez n x n. O objetivo deste desafio é arranjar as rainhas de tal maneira que nenhuma delas esteja em posição de atacar qualquer outra. No jogo de xadrez, uma rainha pode atacar qualquer peça que esteja na mesma linha, coluna ou diagonal. Este problema intrigante foi proposto pela primeira vez em 1850 pelo enxadrista alemão Max Bezzel, e desde então, tem fascinado e desafiado matemáticos e cientistas da computação ao redor do mundo.

Uma das soluções mais interessantes e eficazes para o problema das N rainhas é o uso de **algoritmos genéticos**. Estes algoritmos, que foram inspirados pela evolução biológica, foram introduzidos pela primeira vez por John Holland em 1975, e posteriormente popularizados por seu aluno, David Goldberg, em 1989. Algoritmos genéticos operam seguindo o princípio da seleção natural e da sobrevivência dos mais aptos, conforme descrito por Charles Darwin em 1859. Eles são particularmente úteis para buscar soluções eficientes em um vasto espaço de busca, onde uma busca direta provavelmente seria muito custosa e demorada.

O algoritmo genético aplicado ao problema das N rainhas começa com uma população inicial de soluções geradas aleatoriamente. Cada solução é representada por um vetor de n posições, onde o índice do vetor representa a coordenada x no tabuleiro de xadrez, e o valor do índice representa a coordenada y. O algoritmo então avalia a aptidão de cada solução, ou seja, quantas rainhas estão seguras e não estão em posição de serem atacadas. As soluções mais aptas são selecionadas para reprodução, gerando uma nova geração de soluções. Este processo é repetido, geração após geração, até que uma solução satisfatória seja encontrada, onde todas as rainhas estão seguras.

2. Metodologia e testes

O xadrez, uma jóia da cultura humana com milênios de história, é um jogo de informação perfeita. Esse conceito indica que, em qualquer momento da

partida, todos os jogadores têm acesso às mesmas informações, tornando todas as peças e movimentos visíveis e transparentes para ambos os jogadores. Esta característica de transparência e igualdade de informação torna o xadrez um jogo excepcionalmente justo, reduzindo a possibilidade de vantagens injustas e surpresas inesperadas.

No entanto, esta natureza de informação perfeita do xadrez pode também levar a partidas que são previsíveis até certo ponto. Esta previsibilidade surge do princípio do determinismo, que é intrínseco ao jogo de xadrez. O determinismo é a ideia de que, se as condições iniciais são as mesmas, o resultado de uma partida de xadrez será inevitavelmente o mesmo.

Este determinismo pode apresentar um desafio na medida em que pode resultar em jogos que, para alguns, podem parecer repetitivos e monótonos. Isso implica que, se dois jogadores altamente qualificados e experientes jogarem a mesma partida várias vezes, repetindo as mesmas jogadas e estratégias, o resultado final não irá variar. Este aspecto do xadrez, embora possa ser um desafio para alguns, é também um testemunho da sua profundidade estratégica e complexidade.

2.1 Representação Cromossômica:

Esta abordagem tem como base a ideia de determinismo, especificamente aplicada ao problema das N rainhas. Foi a partir desta concepção que foi desenvolvido este algoritmo. O objetivo deste algoritmo é permitir a criação de um tabuleiro de xadrez com n linhas e n colunas, onde cada linha e coluna terão uma rainha posicionada. No entanto, a disposição das rainhas no tabuleiro não é aleatória ou arbitrária, mas é determinada por uma série de regras e condições que precisam ser cumpridas.

Uma vez que o tabuleiro é criado e as rainhas são posicionadas, o próximo passo é resolver o tabuleiro. Para isso, recorremos ao uso de algoritmos genéticos. Através da aplicação desses algoritmos, somos capazes de encontrar a solução para o problema, mantendo as regras do jogo de xadrez e garantindo que nenhuma rainha esteja em posição de atacar outra.

Usando a biblioteca DEAP em Python foi criadas as classes FitnessMin e Individual, o FitnessMin é usada para minimizar a função de aptidão, que é usada para definir a aptidão do indivíduo (gene), o parâmetro weights é uma tupla que define os pesos para cada objetivo da função de aptidão é minimizada, o atributo fitness é um objeto na classe representa a aptidão do indivíduo.

A função individual cria um indivíduo para um algoritmo genético. Ela cria uma nova classe chamada Individual usando a função 'create' do módulo

'creator' do Deap. A classe Individual é uma subclasse da classe 'list' e possui um atributo 'fitness' que é um objeto da classe FitnessMin.

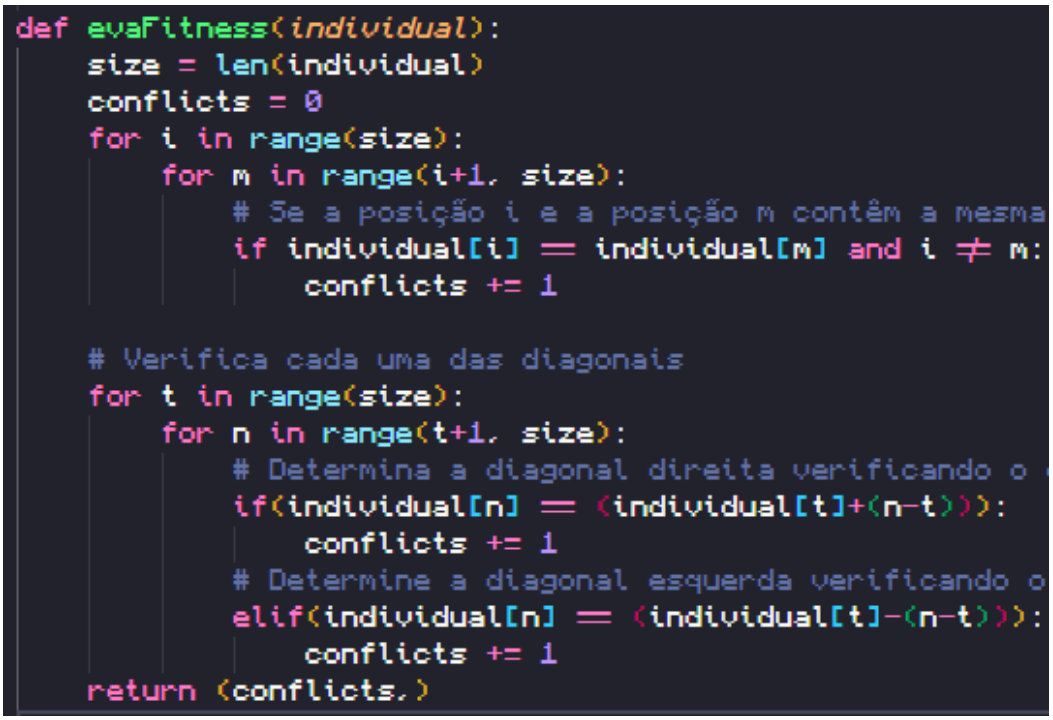
O método 'initRepeat' é utilizado para criar uma lista de tamanho 'n' com valores gerados pela função 'attr_int'. A função 'attr_int' é registrada na toolbox e retorna um número inteiro aleatório entre 0 e o tamanho do tabuleiro - 1. O método 'toolbox.individual()' é usado para criar um indivíduo com valores gerados pela função individual. Isso significa que ele utiliza o método 'initRepeat' para gerar uma lista de tamanho 'n' com valores produzidos pela função 'attr_int'.

```
# A classe FitnessMin é usada quando o objetivo é minimizar a função
# O parâmetro weights é uma tupla que define os pesos para cada objeti
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))

# O atributo fitness é um objeto da classe FitnessMin que representa
creator.create("Individual", list, fitness=creator.FitnessMin)

# Criando e registrando funções na toolbox.
toolbox = base.Toolbox()

# O attr_int é uma função que retorna um número inteiro aleatório ent
toolbox.register("attr_int", random.randint, 0, board_size - 1)

# O individual é uma função que cria um indivíduo para um algoritmo g
# O método initRepeat é usado para criar uma lista de tamanho n com v
toolbox.register("individual", tools.initRepeat, creator.Individual,
                 toolbox.attr_int, board_size)

# O método toolbox.individual() é usado para criar um indivíduo com v
toolbox.individual()
```

Figura 1 - Objetos da biblioteca DEAP utilizados.

2.2 Função de Avaliação ou de Fitness:

A função evaFitness é uma função de avaliação que calcula a soma dos conflitos horizontais, verticais e diagonais entre as rainhas no tabuleiro, o objetivo é minimizar esses conflitos, ou seja, encontrar uma solução onde nenhuma rainha ataque outra. Esta função recebe como entrada um indivíduo, que representa uma possível solução para o problema das N rainhas. O indivíduo é um vetor de tamanho N, onde cada posição representa uma coluna do tabuleiro e o valor dessa posição indica a linha onde a rainha está posicionada. A função evaFitness retorna uma tupla com o número de conflitos encontrados, quanto menor o número de conflitos, melhor a solução.



```python
def evaFitness(individual):
    size = len(individual)
    conflicts = 0
    for i in range(size):
        for m in range(i+1, size):
            # Se a posição i e a posição m contêm a mesma
            if individual[i] == individual[m] and i != m:
                conflicts += 1

    # Verifica cada uma das diagonais
    for t in range(size):
        for n in range(t+1, size):
            # Determina a diagonal direita verificando o
            if(individual[n] == (individual[t]+(n-t))):
                conflicts += 1
            # Determine a diagonal esquerda verificando o
            elif(individual[n] == (individual[t]-(n-t))):
                conflicts += 1
    return (conflicts,)
```

Figura 2 - Função que avalia cada geração do algoritmo.

2.3 Operadores Genéticos:

Seleção: O método de seleção por torneio foi utilizado. Este método consiste em selecionar aleatoriamente um número fixo de indivíduos da população e escolher o melhor entre eles. O tamanho do torneio foi definido como 3.

Cruzamento: O cruzamento de dois pontos foi utilizado. Este método consiste em escolher dois pontos aleatórios no vetor que representa o indivíduo e intercambiar os valores entre esses pontos nos dois indivíduos pais. Este operador foi adaptado ao problema das N rainhas, considerando o vetor que representa o indivíduo como uma possível solução para o problema.

Mutação: A mutação uniforme foi utilizada. Este método consiste em escolher um gene aleatório do indivíduo e substituí-lo por um valor inteiro escolhido uniformemente dentro de um intervalo definido. Este operador foi também adaptado ao problema das N rainhas, utilizando o vetor que representa o indivíduo como uma possível solução para o problema.

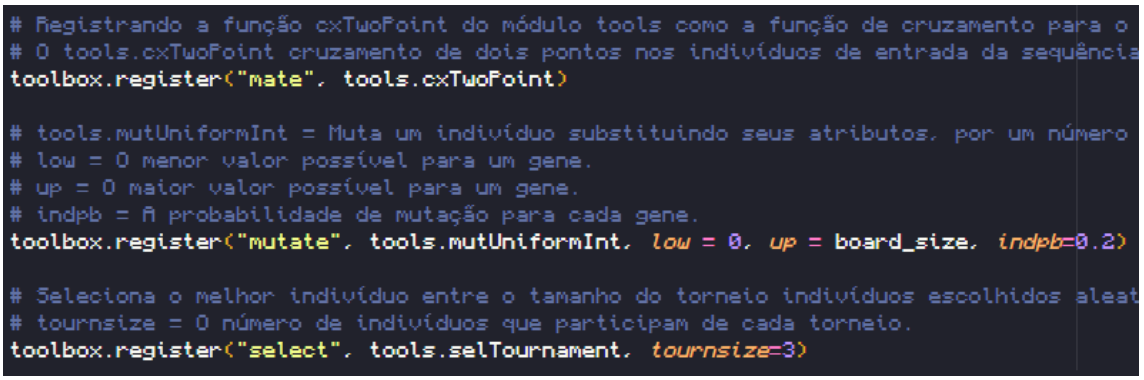


Figura 3 - Operadores Genéticos Utilizados no Código

3. Testes:

Para realizar os testes do algoritmo em discussão, tomamos a decisão de implementar a utilização de certos parâmetros específicos. Inicialmente, estabelecemos um tabuleiro com dimensões de 20x20, que comportaria 20 rainhas, com uma única rainha ocupando cada linha ou coluna. A escolha desta configuração foi feita com o objetivo de estabelecer um cenário ideal para o teste do algoritmo, proporcionando as condições necessárias para uma análise abrangente e pouco exaustiva.

Baseando-se nestes parâmetros, concluímos que seriam necessárias aproximadamente 1000 gerações para a resolução efetiva do problema proposto. Este número foi determinado após uma cuidadosa deliberação, levando em consideração tanto a complexidade do problema quanto a capacidade computacional.

Com esses parâmetros em mente, passamos a apresentar uma tabela meticulosamente elaborada, esta tabela oferece um resumo estatístico contendo informações extremamente relevantes para cada gene do algoritmo genético. Nela, você encontrará informações sobre as gerações que foram criadas (gen), o número de avaliações realizadas para cada geração (nevals), o valor médio de erro em relação aos conflitos observados (avg) e, finalmente, os conflitos mínimos detectados durante o meticuloso processo de análise (min).

...	gen	nevals	avg	min
	0	100	21.55	13
	1	62	18.86	11
	2	71	17.82	10
	3	75	16.88	10
	4	75	16.02	9
	5	74	15.62	8
	6	80	15.21	8
	7	66	14.83	9
	8	74	14.27	7
	9	71	14.66	7
	10	77	14.01	7
	11	72	13.59	7
	12	74	13.65	6
	13	61	12.19	6
	14	77	12.09	6
	15	80	12.71	7
	16	78	12.76	7
	17	84	12.48	6
	18	80	12.64	6
	19	64	11.6	6
	20	80	11.44	6
	21	69	12.18	6
	22	79	11.82	6
	23	72	10.92	7
	...			
	997	83	5.44	0
	998	82	5.88	0
	999	75	5.97	0
	1000	73	5.36	0

Figura 4 - Tabela estatística a partir dos dados de cada geração

Foi realizado um trabalho mais aprofundado com os dados obtidos através da tabela de estatísticas. Decidiu-se criar uma representação visual para uma melhor compreensão dos mesmos. Para isso, utilizou-se a biblioteca do matplotlib, uma ferramenta muito útil para a plotagem de gráficos. Dessa forma, foi possível plotar um gráfico que deu vida aos dados da tabela, tornando a informação mais acessível e fácil de compreender.

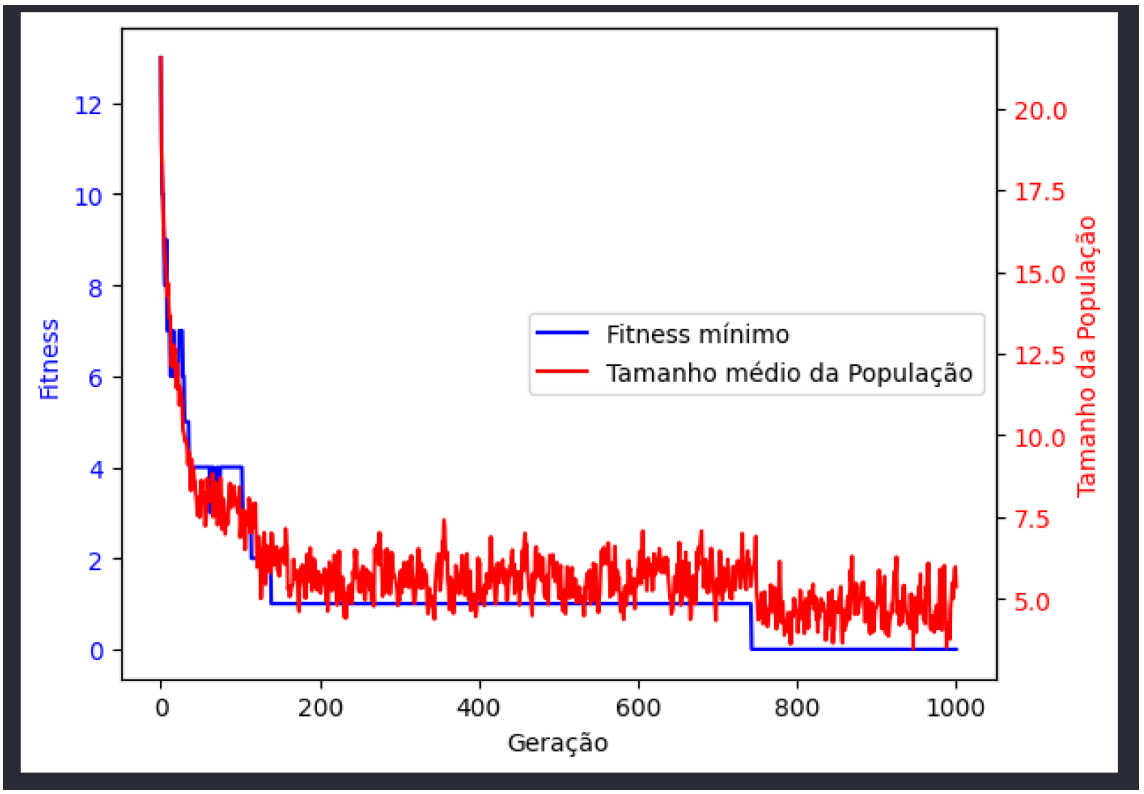


Figura 5 - Representação visual de dados usando um gráfico de linhas.

Ao concluirmos a execução do nosso código, o que acontece é que o valor da matriz que mais se aproximou do valor esperado de avaliação é retornado. Este valor retornado é então processado por uma função de impressão dedicada. O papel desta função é exibir o resultado final do tabuleiro de xadrez. Ela toma o estado final do jogo e o traduz em uma representação visual compreensível que é finalmente exibida no terminal. Dessa forma, o usuário pode visualizar o estado final do tabuleiro de xadrez e entender o resultado do jogo.

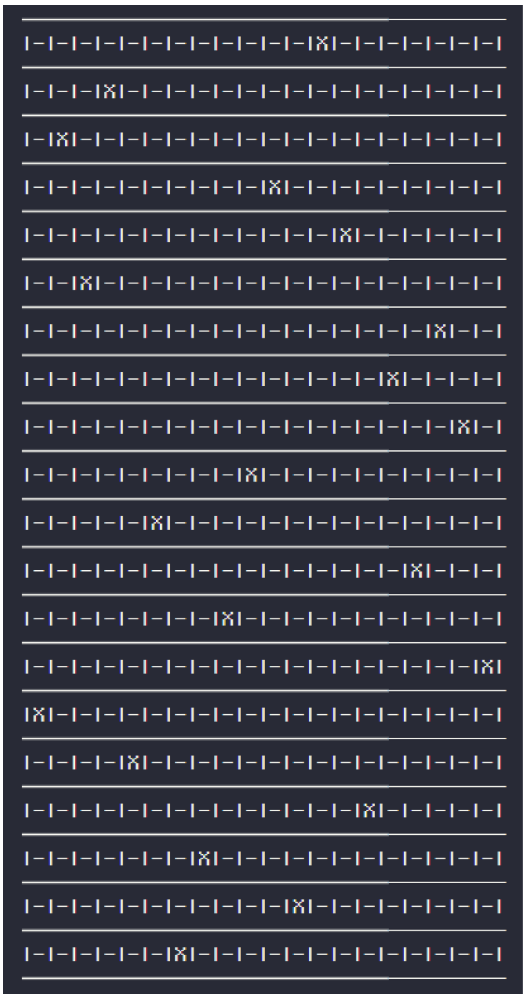


Figura 6 - Saída de dados representando o tabuleiro no terminal.

No intuito de proporcionar uma melhor compreensão e facilitar a visualização do gene que conseguiu alcançar a cobiçada primeira posição no nosso "hall da fama", decidi criar uma representação gráfica ilustrativa. Esta imagem foi concebida com o propósito de demonstrar, de forma clara e intuitiva, como seria o resultado deste gene se o mesmo fosse aplicado em uma situação prática, mais especificamente, a um tabuleiro de xadrez:

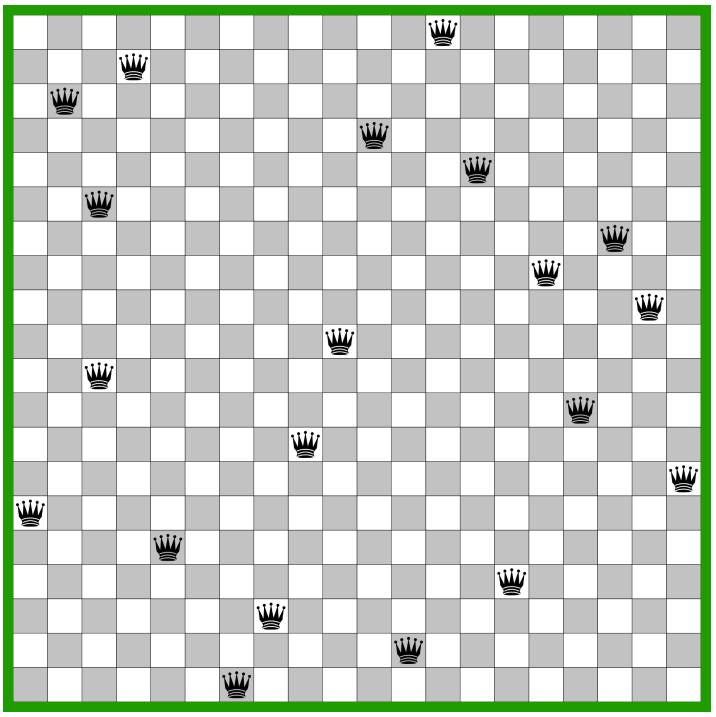


Figura 7 - Representação das rainhas no tabuleiro, de forma que não se ataquem.

4. Conclusão

O problema das N rainhas, apesar de sua complexidade e desafio intelectual, não é necessariamente um problema prático que encontramos diariamente em nossa rotina. Por causa dessa característica, reconheço que talvez tenha me desviado ligeiramente do objetivo principal deste trabalho. O propósito era, afinal, gerar uma solução eficaz e útil a partir de um problema real, algo que pudesse ser aplicado em um cenário comum e cotidiano. Por isso, compreendo completamente se o professor sentir a necessidade de desconsiderar algum aspecto do meu trabalho ou tomar qualquer outra medida necessária em relação a isso.

Dito isso, devo destacar que este trabalho me proporcionou minha primeira experiência com o desenvolvimento de um algoritmo genético. Foi uma oportunidade incrível de aprendizado e descoberta. Fiquei fascinado pela aplicabilidade deste tipo de algoritmo nas mais diversas situações e problemas, demonstrando sua versatilidade e potencial como uma ferramenta de resolução de problemas.

Com essa reflexão em mente, a primeira ideia que me veio foi a possibilidade de aplicar algoritmos sofisticados no campo que mais me fascina: o desenvolvimento de jogos eletrônicos. Acredito que essa aplicação poderia revolucionar a criação de jogos e proporcionar uma experiência mais rica e envolvente para os jogadores.

A inteligência artificial, especialmente os algoritmos genéticos, têm o potencial de causar uma revolução significativa na indústria de desenvolvimento de jogos. Esses algoritmos podem ser usados para criar personagens e ambientes mais realistas que aprendem e se adaptam ao estilo do jogador.

Além disso, a IA poderia ser usada para testar jogos durante o desenvolvimento. Em vez de depender exclusivamente de testadores humanos, que podem não conseguir explorar todas as combinações de jogo possíveis, poderíamos usar IA para simular diferentes estilos de jogo e identificar erros ou problemas potenciais.

5. Referências

Problema do Xadrez de 152 anos é finalmente solucionado. Disponível em: <https://www.hypeness.com.br/2022/02/problema-do-xadrez-de-152-anos-e-finalmente-solucionado-por-matematico-de-harvard/>.
Acessado em: 14/11/2023.

Algoritmo genético. Disponível em: <https://pt.wikipedia.org/wiki/Algoritmo_gen%C3%A9tico/>.
Acessado em: 14/11/2023.

Problema das oito damas. Disponível em: <https://pt.wikipedia.org/wiki/Problema_das_oito_damas/>.
Acessado em: 15/11/2023.

Matemático de Harvard soluciona problema de 153 anos do xadrez. Disponível em: <https://revistagalileu.globo.com/Ciencia/noticia/2022/01/matematico-de-harvard-soluciona-problema-de-153-anos-do-xadrez.html/>.
Acessado em: 15/11/2023.

Um Algoritmo Genético Especializado para o Problema das N Rainhas. Disponível em: <https://www.researchgate.net/publication/368591172_Um_Algoritmo_Genetico_Especializado_para_o_Problema_das_N_Rainhas/>.
Acessado em: 15/11/2023.

Resolvendo N-Rainhas com Algoritmo Genético. Disponível em: <https://github.com/leonamtv/FGA-N-Rainhas/>.
Acessado em: 16/11/2023.