Universidade Federal do Rio de Janeiro
Instituto de Computação
Inteligência Artificial - Trabalho 1

Problema das N-Rainhas

UFRJ

Alunos:
Felipe Melo (119093752) e Matheus Vargas (119039821)

Professor:
João Carlos Pereira da Silva

Introdução

O objetivo deste relatório é descrever o problema das N-Rainhas e discutir as soluções encontradas utilizando o algoritmo genético.

O problema trata de um tabuleiro de xadrez, de dimensão N x N, onde estão dispostas N rainhas, cada uma em uma coluna diferente, de modo que cada coluna tenha sua rainha correspondente. O objetivo do problema é encontrar uma configuração de tabuleiro em que não ocorra nenhum ataque entre as rainhas, respeitando as regras do xadrez.

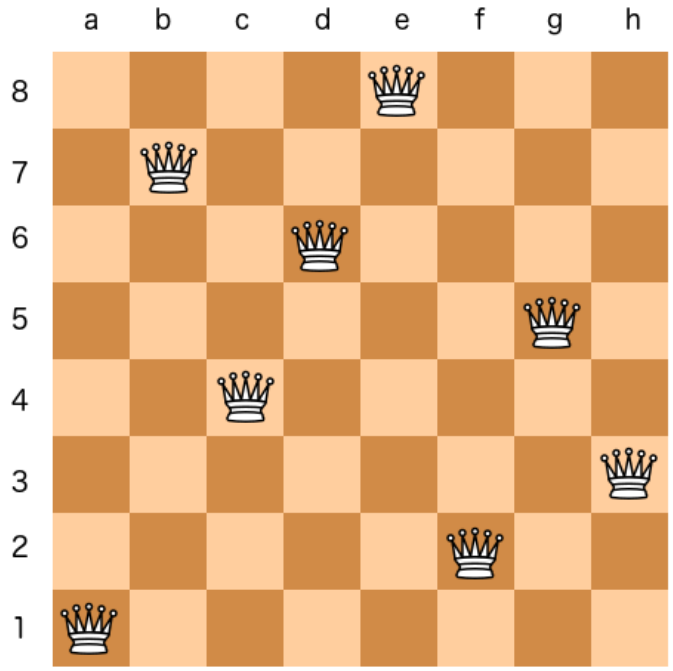

Figura 1: Solução para N = 8

Para isso, definimos um tabuleiro T, de dimensão N como uma lista de tamanho N, preenchida com valores de 0 a $N - 1$. Cada índice da lista representa uma coluna do tabuleiro e o valor dessa salvo nesse índice representa a linha ocupada pela rainha na coluna. Assim, o tabuleiro da figura 1 será representado como $[1, 7, 4, 6, 8, 2, 5, 3]$. Para fins de execução do algoritmo, consideramos a linha e coluna iniciais com índice 0 e finais com índice $N - 1$.

Definiremos também uma função de avaliação f que recebe um tabuleiro e retorna a quantidade de ataques entre suas rainhas (score do tabuleiro). Essa função é da forma

$$\frac{1}{1+A},$$

onde A é o número de ataques do tabuleiro. Teremos, portanto, uma solução para o tabuleiro de entrada quando $f(T) = 1$. O gráfico a seguir indica a relação entre os valores aproximados dos scores que a função de avaliação retornará (eixo Y) e a quantidade de ataques do tabuleiro (eixo X).

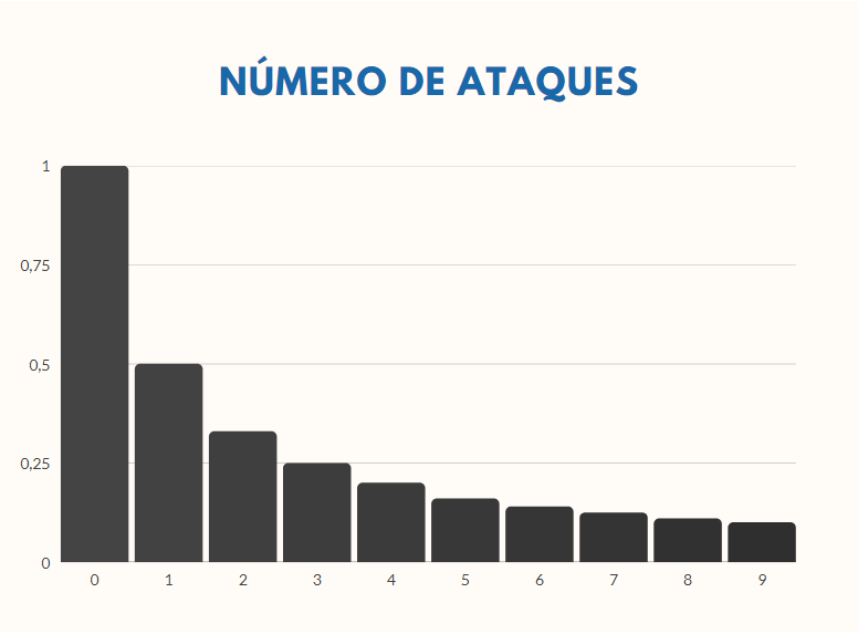

Figura 2: Gráfico de Scores

Entrada e saída do programa

O algoritmo genético que desenvolvemos tem como entrada os seguintes parâmetros:

1. A dimensão N do tabuleiro (que é também o número de rainhas);
2. O tamanho da população (quantidade de tabuleiros gerados aleatoriamente);
3. Ordem de grandeza do limite superior do número de gerações. Um valor igual a x simboliza que serão produzidas no máximo 10^x gerações;
4. A probabilidade de crossover;
5. A probabilidade de mutação;
6. O percentual de indivíduos que farão parte da elite.

A saída do programa é o melhor tabuleiro encontrado, que pode ou não ser um dos ideais (com 0 ataques). Ademais, o tempo de execução e o número de gerações são mostrados, além de um gráfico contendo os scores do melhor indivíduo, da média dos indivíduos, e do pior indivíduo no eixo y, e as gerações no eixo x.

Resultados

Os resultados obtidos com tabuleiros de dimensão N = 4, foram irrelevantes, devido ao baixo número de configurações de tabuleiro (4^4 = 256). Isso faz com que até uma busca aleatória, com probabilidade de mutação igual a 100% seja eficaz e, desse modo, optamos por realizar os testes em tabuleiros de N = 8, 16 e 32.

Testamos o programa com os seguintes parâmetros:

População	Máx. Gerações	Crossover	Mutação
20	10^5	0,75	0,02

Além desses parâmetros, testamos casos com e sem elitismo. A seguir estão os resumos das execuções:

- Tabuleiro 8x8 sem elitismo

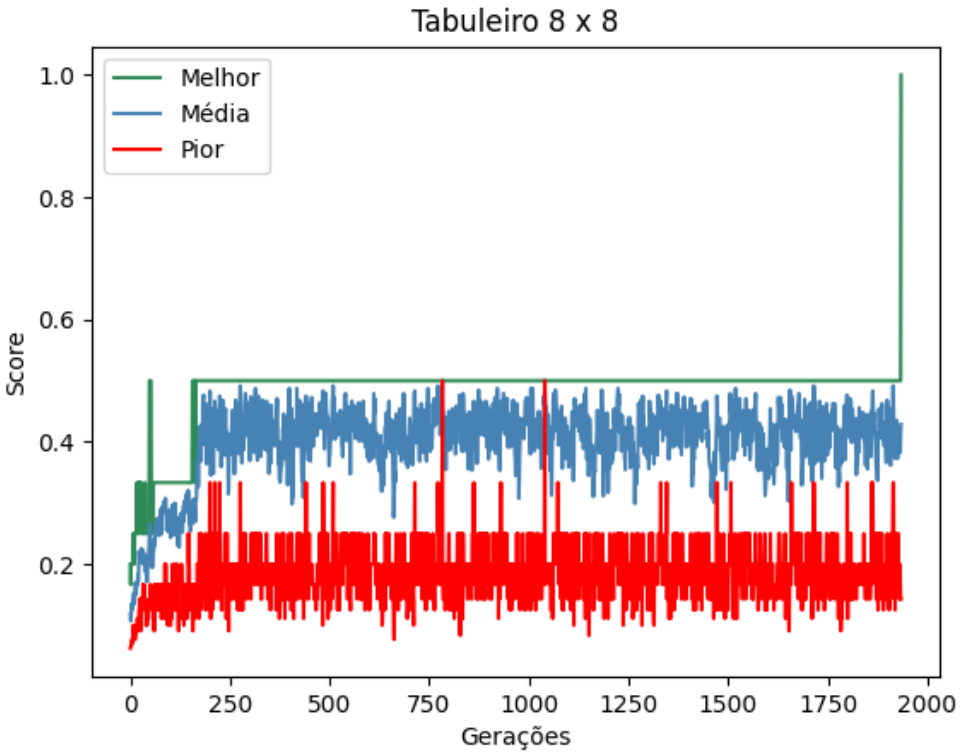

Figura 3: N = 8 sem elitismo

O resultado foi obtido quase instantaneamente, com 1 segundo de execução. Além disso, podemos ver que entre as gerações 0 e 250 o score do melhor indivíduo caiu algumas vezes. Isso pode ocorrer tanto por crossover quanto por mutação, e teria sido evitado se houvesse pelo menos um indivíduo pertencente à elite. Por fim, vemos que em alguns momentos, as linhas de melhor e pior indivíduos se tocam. Isso ocorre pois a população se torna totalmente homogênea nesses momentos.

- Tabuleiro 8x8 com elite de 10%

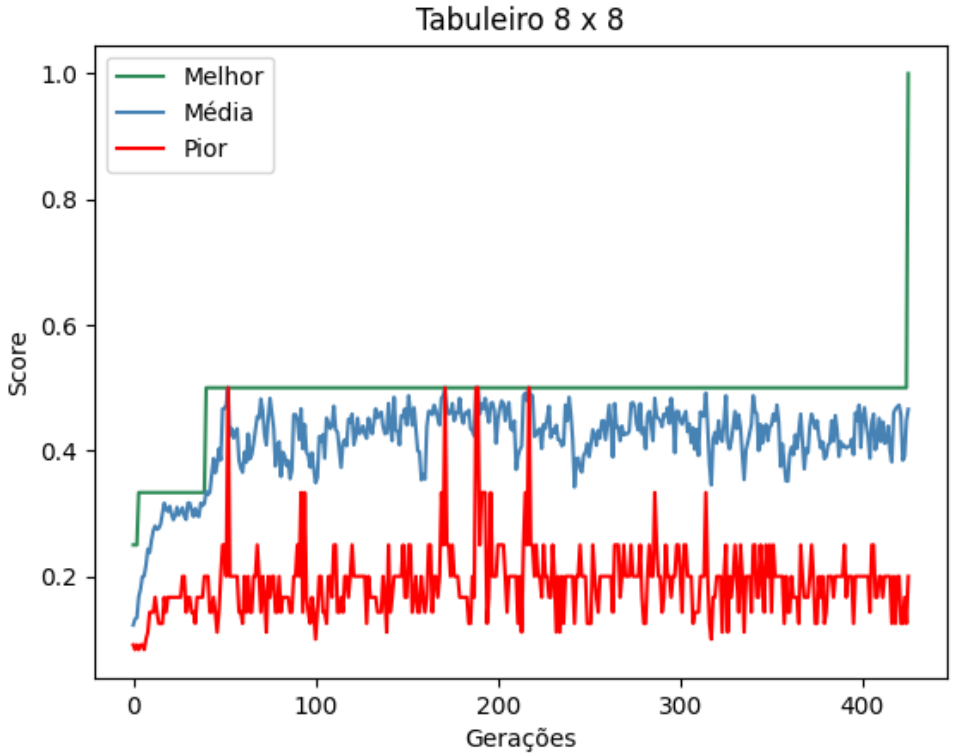

Figura 4: N = 8 com elitismo de 10%

Nessa execução, podemos ver que a linha verde, que representa os melhores indivíduos, é consistente, ou seja, não tem seu valor de score variando para baixo. A razão para isso é a utilização do elitismo. Com elitismo de 10%, os 2 melhores indivíduos de uma geração passam sem alteração para a próxima, refletindo na estabilidade da linha verde.

- Tabuleiro 16x16 sem elitismo

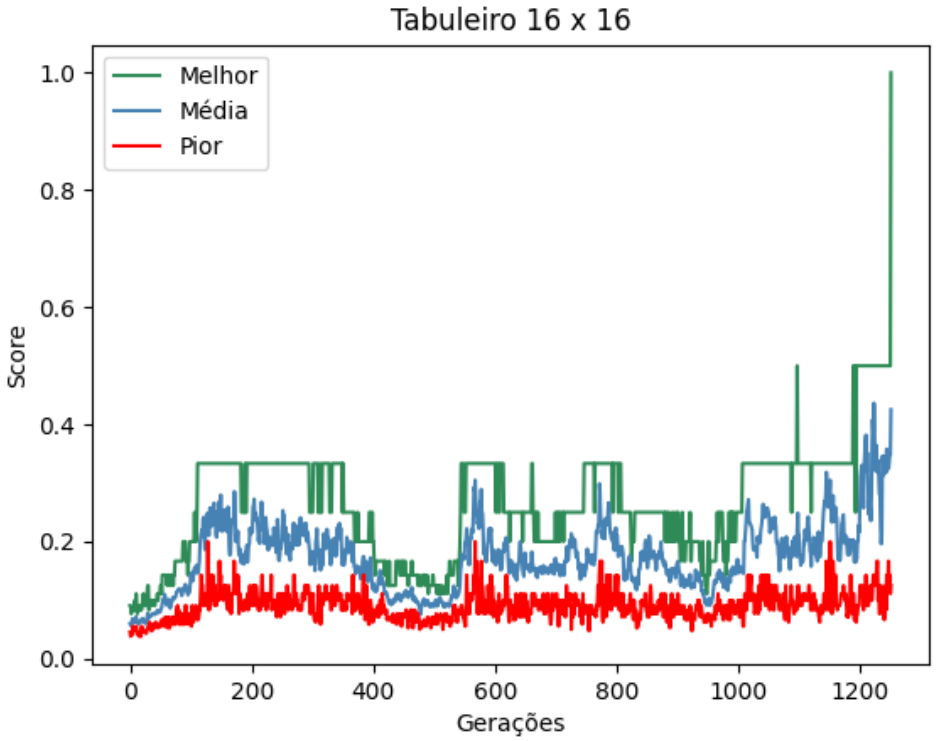

Figura 5: N = 16 sem elitismo

A execução levou 1 segundo e teve 1.251 gerações (aproximadamente 101.000 gerações por minuto).

- Tabuleiro 16x16 com elite de 10%

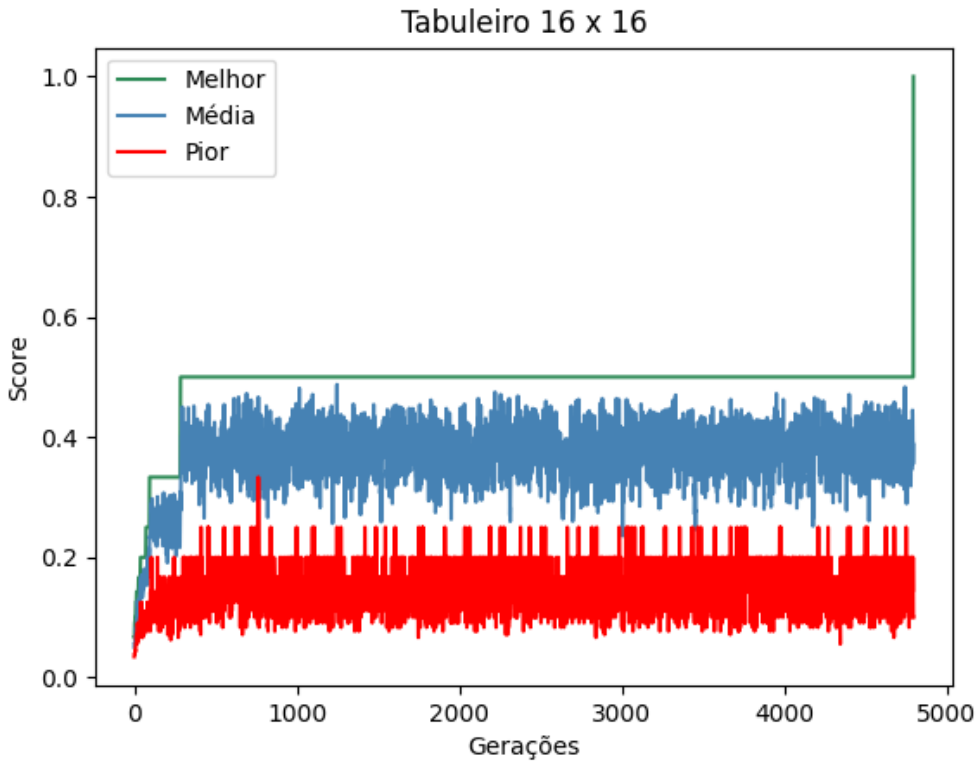

Figura 6: N = 16 com elitismo de 10%

A execução levou 3 segundos e teve 4.794 gerações (aproximadamente 103.000 gerações por minuto).

- Tabuleiro 32x32 sem elitismo

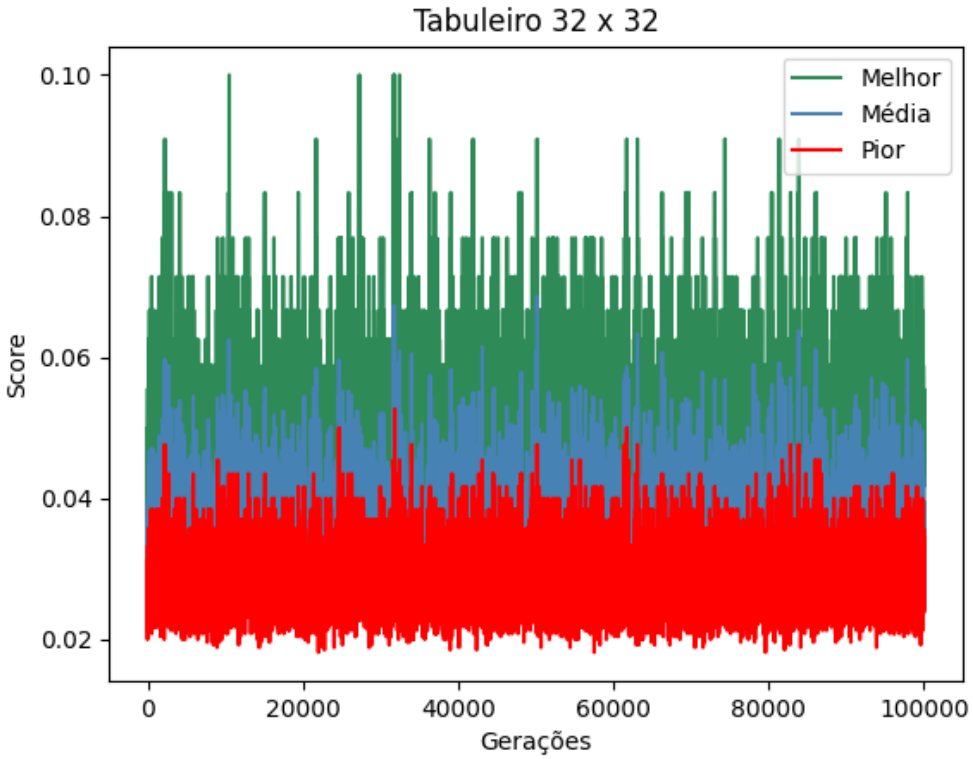

Figura 7: N = 32 sem elitismo

Nesse caso, o algoritmo atingiu o limite máximo de gerações e sequer chegou perto de resolver o problema. Na próxima seção, analisaremos quais mudanças de parâmetros podem mudar essa situação.

- Tabuleiro 32x32 com elite de 10%

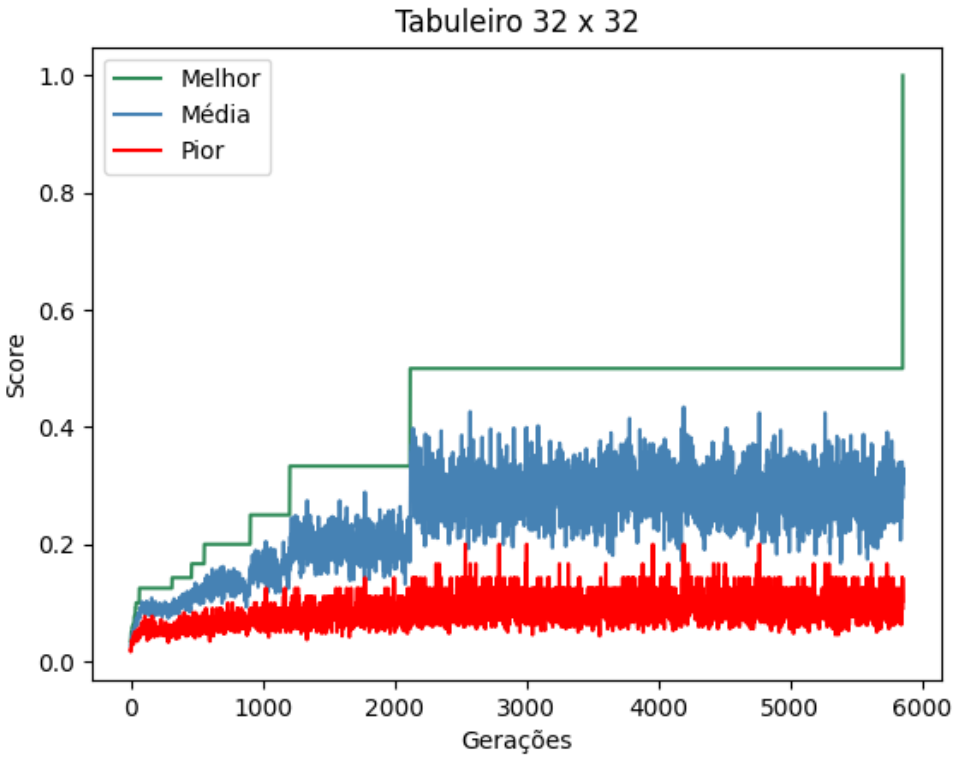

Figura 8: N = 32 com elitismo de 10%

Com o elitismo habilitado, o resultado foi muito diferente, obtendo um resultado em 6 segundos e 5.853 gerações (aproximadamente 58.000 gerações por minuto).

Alterando os Parâmetros

Nesta seção, consideramos tabuleiros de N = 32, a fim de facilitar a detecção de mudanças a cada execução. No caso do exemplo da figura 7, a correção mais natural obviamente é a adição de elitismo, como visto no exemplo da figura 8.

Vamos também tentar alterar outros parâmetros, mantendo o elitismo desabilitado e avaliar os resultados, começando por aumentar a população, que antes era de 20 indivíduos.

Parâmetros:

População	Máx. Gerações	Crossover	Mutação	Elitismo
200	10^5	0,75	0,02	0,0

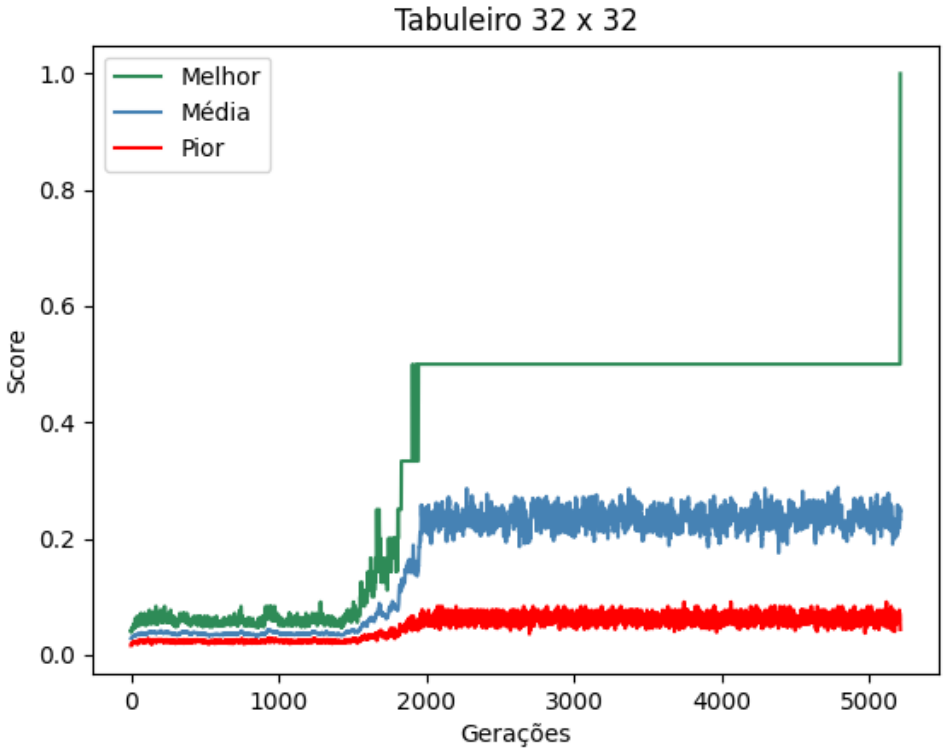

Figura 9: População 10x maior

É possível encontrar uma solução com população maior provavelmente porque, sem o elitismo, o melhor indivíduo frequentemente se perde em uma população pequena. Todavia, com uma população maior isso é mais difícil de ocorrer, o que torna o algoritmo mais estável.

Com o tamanho da população de volta a 20, tentamos alterar os percentuais de crossover e de mutação da população intermediária de maneira individual. Entretanto, mudanças para mais e para menos não chegaram a uma das soluções ótimas.

Conclusões

Como vimos, o elitismo é extremamente benéfico para o algoritmo, já que ele impede que bons indivíduos se percam, o que possibilita utilizar uma população menor, aumentando o número de gerações por minuto.

Além disso, o tamanho da população inicial dita a velocidade de execução do algoritmo genético. Em populações com tamanho igual a 20 indivíduos, temos um valor consideravelmente mais elevado de gerações por minuto que em populações maiores. Em geral, isso ocorre pois a função de avaliação é custosa e quanto mais tabuleiros para verificar, maior o tempo de execução.

Outro ponto que vale ser destacado é que o valor do percentual de mutação deve ser pequeno (em torno de 5%), para evitar que a busca pelo melhor indivíduo seja aleatória. Em tabuleiros de dimensão $N \leq 8$, entretanto, a busca aleatória funciona, devido ao número de configurações não ser tão alto, mas em tabuleiros maiores isso se torna inviável.

Ademais, o tamanho da elite deve ser proporcional à mutação, já que o elitismo garante maior estabilidade genética, e a mutação faz justamente o contrário.

Por fim, a probabilidade de crossover tem um grande intervalo de valores viáveis. Em tabuleiros 32x32, encontrar uma solução é rápido com valores de 10 a 90% (e elitismo habilitado). Com elitismo desabilitado, entretanto, valores pequenos de crossover (menores que 40%) se tornam impraticáveis.

Testando os Limites do Algoritmo

A seguir, mostramos exemplos de execução para N = 128:

Parâmetros:

População	Máx. Gerações	Crossover	Mutação	Elitismo
10	10^6	0,7	0,02	0,2

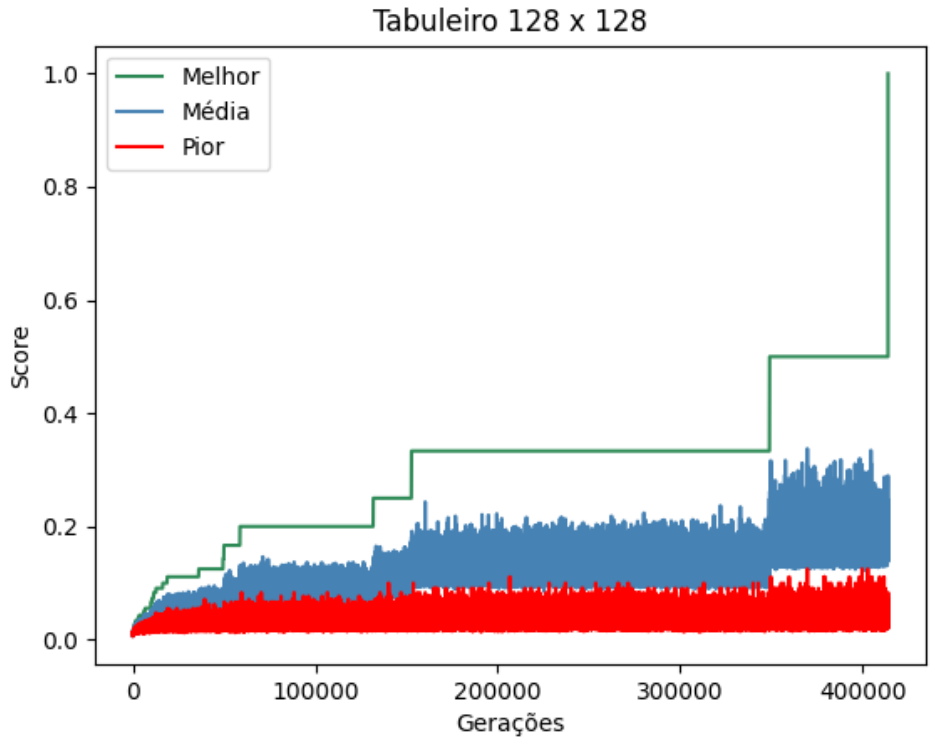

Figura 10: N = 128 com população de 10 indivíduos

Esse resultado foi obtido em 8 minutos e 55 segundos, com 413.840 gerações e taxa de 46.424 gerações por minuto.

Parâmetros:

População	Máx. Gerações	Crossover	Mutação	Elitismo
20	10^6	0,75	0,02	0,1

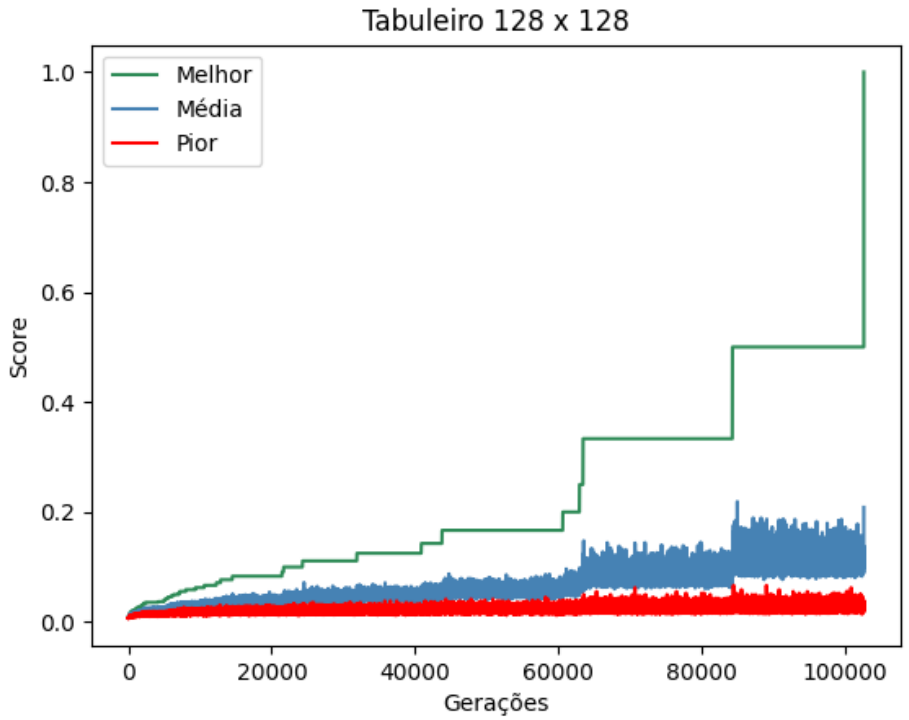

Figura 10: N = 128 com população de 20 indivíduos

O resultado foi gerado em 7 minutos e 13 segundos, com 102.602 gerações e uma taxa de aproximadamente 14.200 gerações por minuto.