Computação Evolutiva na Resolução dos Problemas das N-Rainhas e Fábrica de Rádios

Bruno M. Pires[1]

[1]Departamento de Ciências da Computação
Universidade do Estado de Santa Catarina (UDESC)
Caixa Postal 15.064 – 88.035-901 – Joinville – SC – Brasil

bruno.pires@edu.udesc.br

Abstract. *This report presents an effective solution to the N-Queens and Radio Factory problems using evolutionary computation. The approach employs genetic algorithms to find optimal placements of queens on an NxN chessboard, preventing mutual attacks. Additionally, genetic algorithm techniques are utilized to pinpoint an optimal production point for luxury radios. Promising results indicate the feasibility of this technique in addressing these highly classical optimization challenges within the field of Intelligence.*

Resumo. *Este relatório apresenta uma solução eficaz para o problema das N-rainhas e Fábrica de Rádios por meio de computação evolutiva. A abordagem utiliza algoritmos genéticos para encontrar disposições ideais das rainhas em um tabuleiro NxN, evitando ataques mútuos. E utilizam também técnicas de algoritmos genéticos para encontrar um ponto ótimo na produção de rádios de luxo. Resultados promissores indicam a viabilidade dessa técnica em resolver esses desafios de otimização muito clássicos dentro dessa área de Inteligência*

1. O problema das N-Rainhas

Proposto em 1848 por ”Max Bezzel” o problema das N-Rainhas consiste classicamente em um tabuleiro com 8 linhas e 8 colunas onde devem ser dispostas 8 rainhas de forma que elas não se ataquem mutuamente, isso significa que nenhuma rainha pode permanecer em uma posição onde esteja ameaçada ou que ameaçe outra rainha. Extrapolando já para lógica, cada linha, coluna e diagonal só pode ter uma rainha.

Percebemos com a descrição, que se trata de um problema NP, cujo o tempo para se providenciar uma solução cresce exponencialmente, até um ponto em que não vai existir poder computacional no mundo capaz de calcular uma solução. E no caso, o tempo cresce de acordo com o tamanho do tabuleiro, um tabuleiro pequeno em segundos já se acha uma solução, mas isso não é verdade para grandes tabuleiros.

Existem diversas formas de se resolver este problema, como por exemplo com algoritmos de Força Bruta testando todas as possibilidade de posições das rainhas e verificando a viabilidade. Mas o foco deste relatório está em resolver utilizando computação evolutiva.

2. Utilizando Computação Evolutiva no problema das N-Rainhas

Sabendo do que se trata o problema, como podemos pensar em uma solução que utilize os conceitos de AG’s (Algoritmos genéticos) para resolvê-lo?

É importante ter em mente como funcionará o algoritmo, quais seus parâmetros e saída.A Figura 1 apresenta a tela inicial da ferramenta com todas as configurações possíveis para solucionar o problema.

- Parâmetros: Na ferramenta desenvolvida a maioria dos parâmetros são personalizados de acordo com a necessidade, temos então ”Dimensão dos Indivíduos”, ”Tamanho da População”, ”Quantidade de Gerações”, ”Tipo de técnica de seleção”, ”Algoritmo é Elistista?”, ”Quantidade de Execuções”
- Saída: Como saída temos diversas aproximações de soluções válidas, que como característica dos AG’s convergem para uma solução ótima com o passar das gerações.

Tendo em mente os parâmetros necessários, é preciso pensar nas características do problema e como pode ser feita uma codificação do mesmo. A lógica utilizada nesta implementação leva em consideração Indivíduos com cromossomos de tamanho N, considerando um tabuleiro de tamanho NxN. Isso significa que cada indíviduo terá um cromossomo (Lista) de N genes onde cada gene representa uma posição de uma rainha do tabuleiro, por exemplo, em um tabuleiro 8x8, um indivíduo possui um cromossomo de tamanho 8 onde cada posição do cromossomo diz respeito á uma coluna e o valor que está na estrutura de dados diz respeito á linha em que a rainha está posicionada naquela coluna. Outro conceito importante, é o tipo de codificação destes cromossomos, os valores que um gene pode assumir. Na solução, foi utilizada o tipo de codificação de ”Inteiro Permutado”, ou seja, analisando o cromossomo como um todo, os valores contidos nos genes nunca se repetem. A figura 2 ilustra bem estes conceitos, apresentando um cromossomo que representa uma solução válida para um tabuleiro 8x8.

Figure 1. Tela Inicial Ferramenta de aplicação de Algoritmo Genético

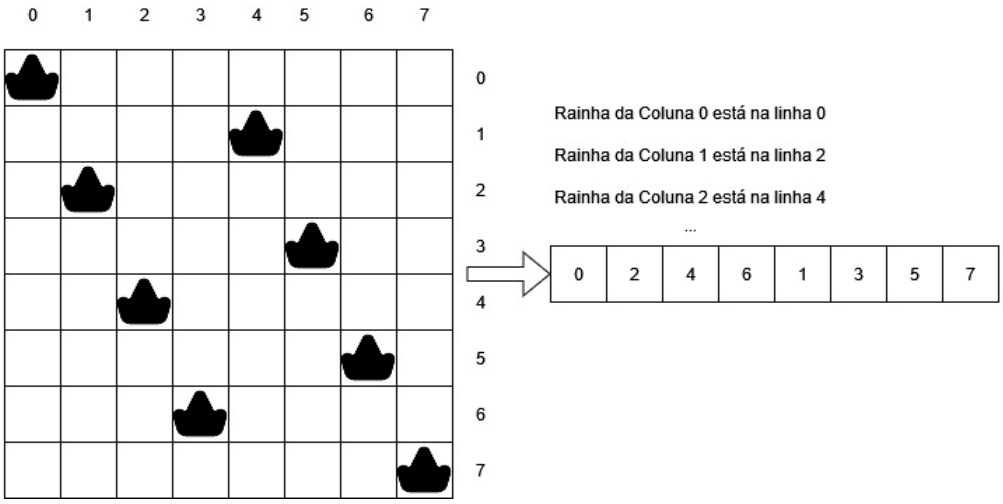

Figure 2. Exemplo de Cromossomo Inteiro Permutado

3. Resultados Obtidos no problema das N-Rainhas

Com finalidade de comparação de resultados, vale a pena deixar claro quais configurações foram utilizadas:

- Quantidade de Execuções: 10
- Codificação: Inteiro Permutado
- Tipo de Seleção: Torneio Estocástico
- Tipo de Crossover: PMX
- Tipo de Mutação: SWAP
- Elitismo: Verdadeiro
- Dimensão dos Indivíduos: 8, 16, 32, 64, 128
- Tamanho da População Inicial: 20
- Quantidade de Gerações: 3500

Os resultados serão separados por tamanho de tabuleiro

3.1. Tabuleiro 8x8

Com um tabuleiro 8x8, o problema ainda é bastante simples, e um solução válida sempre é encontrada dentro das 3500 gerações, na verdade, para um tabuleiro tão pequeno, geralmente poucas gerações já resolvem o problema. Isso torna o gráfico de convergência bastante simples, algo interessante de se notar, é que cada execução (10 foram realizadas) chegou em uma solução diferente para o problema, em poucas gerações. O box plot para este caso não é tão interessante, visto que a média das execuções sempre será o valor ótimo e não teremos variações nas execuções para um caso tão simples, mas está representado na 4. A figura 3 apresenta o gráfico de convergência de todas as 10 execuções, a ferramenta possibilita analisar individualmente, mas para o relatório todas as execuções estão plotadas no mesmo gráfico, juntamente com uma linha em vermelho que representa a média dos valores. Nota-se que uma das execuções, representada pela cor verde demorou mais gerações para convergir para o valor ótimo, algo perfeitamente normal, visto a alta estocastiscidade da computação evolutiva.

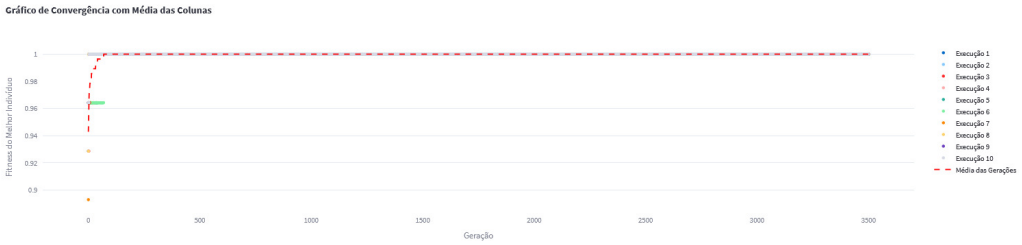

Figure 3. Gráfico de convergência para tabuleiro 8x8

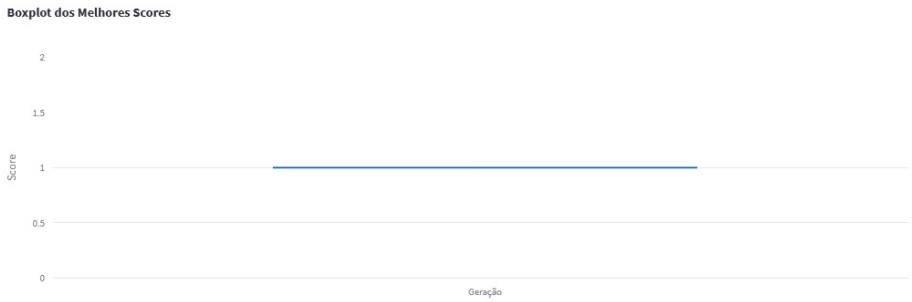

Figure 4. Gráfico Boxplot para tabuleiro 8x8

A função objetivo, sendo aqui o número de colisões terminou sendo 0 em todas as execuções, visto que o algoritmo convergiu para soluções ótimas neste caso. Resultados na tabela a seguir.

Execução	Função Objetivo	Solução
1	0	3, 7, 0, 4, 6, 1, 5, 2
2	0	5, 3, 6, 0, 7, 1, 4, 2
3	0	3, 5, 0, 4, 1, 7, 2, 6
4	0	5, 2, 0, 7, 3, 1, 6, 4
5	0	3, 1, 6, 2, 5, 7, 0, 4
6	0	5, 2, 6, 1, 7, 4, 0, 3
7	0	3, 1, 6, 2, 5, 7, 4, 0
8	0	5, 2, 0, 6, 4, 7, 1, 3
9	0	3, 5, 0, 4, 1, 7, 2, 6
10	0	1, 4, 6, 3, 0, 7, 5, 2

Table 1. Tabela de resultados tabuleiro 8x8

3.2. Tabuleiro 16x16

Com um tabuleiro 16x16, o problema ainda é bastante simples, e um solução válida sempre é encontrada dentro das 3500 gerações, na verdade, para um tabuleiro tão pequeno, a solução ótima é encontrada aproximadamente entre 30 e 60 gerações. Isso torna o gráfico de convergência bastante simples, algo interessante de se notar assim como no caso anterior, é que cada execução (10 foram realizadas) chegou em uma solução diferente para o problema, em poucas gerações. O box plot para este caso não é tão interessante, visto que a média das execuções sempre será o valor ótimo e não teremos variações no fitness do melhor indivíduo nas execuções para um caso tão simples.

A figura 3 apresenta o gráfico de convergência de todas as 10 execuções, a ferramenta possibilita analisar individualmente, mas para o relatório todas as execuções estão plotadas no mesmo gráfico, juntamente com uma linha em vermelho que representa a média dos valores. Nota-se que o mesmo comportamento do caso anterior ocorreu, e uma das execuções demorou mais gerações para atingir um valor ótimo.

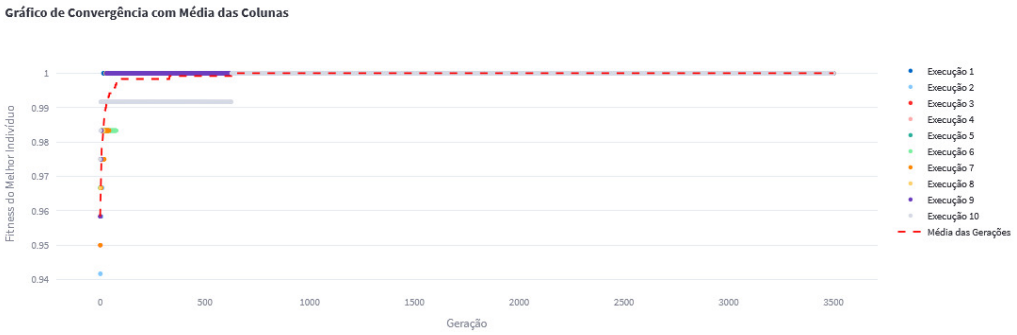

Figure 5. Gráfico de convergência para tabuleiro 16x16

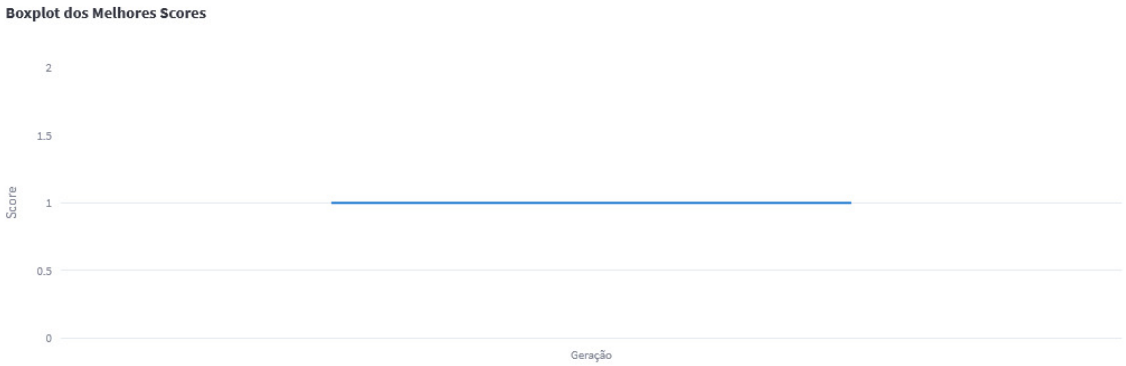

Figure 6. Gráfico Boxplot para tabuleiro 16x16

A função objetivo segue o mesmo comportamento do caso anterior, em todos os casos chegamos em diferentes soluções ótimas sem colisões. Como o comportamento é o mesmo, para poupar espaço, apenas um exemplo de resultado logo da primeira execução é o seguinte indivíduo: 12, 2, 7, 10, 4, 1, 15, 9, 3, 13, 8, 14, 11, 6, 0, 5

3.3. Tabuleiro 32x32

Com um tabuleiro de tamanho 32x32 as coisas já ficam mais interessantes. Ainda conseguimos alcançar o valor ótimo com 3500 gerações, mas algumas execuções não convergiram para o ponto ótimo, e o gráfico de convergência deixa bastante claro a grande diferença entre cada execução. Umas convergindo muito rapidamente como é o caso da cor verde (Exec 6), outras demorando algumas gerações a mais, e outras nem chegando no ponto ótimo, como é o caso da cor amarela (Exec 8).

O gráfico boxplot deste conjunto de execuções, ja nos trás mais informações, a figura 8 representa as 10 execuções. Temos um valor minímo referente a uma execução que não conseguiu atingir o ponto ótimo, e percebemos que o restante das execuções ficaram dentro de uma boa margem entre um fitness de 0.995 e 1.0.

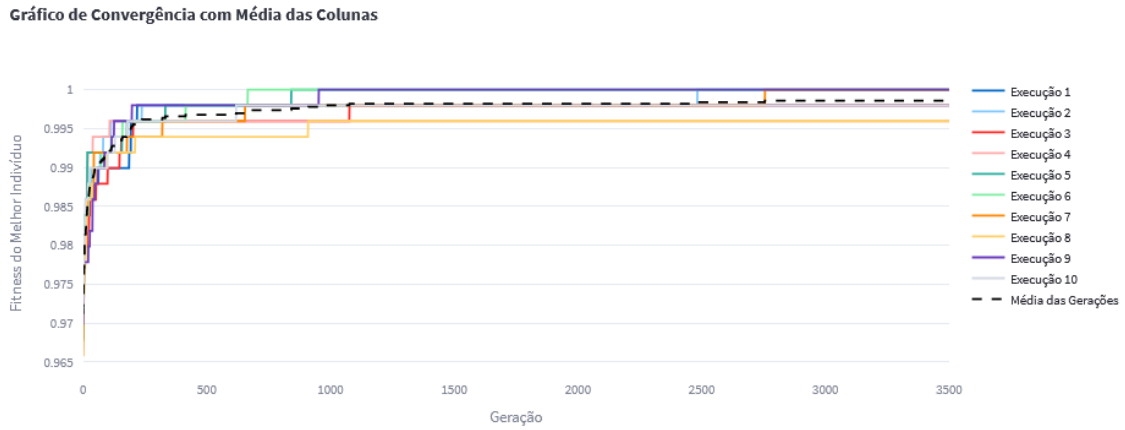

Figure 7. Gráfico de convergência para tabuleiro 32x32 com solução ótima

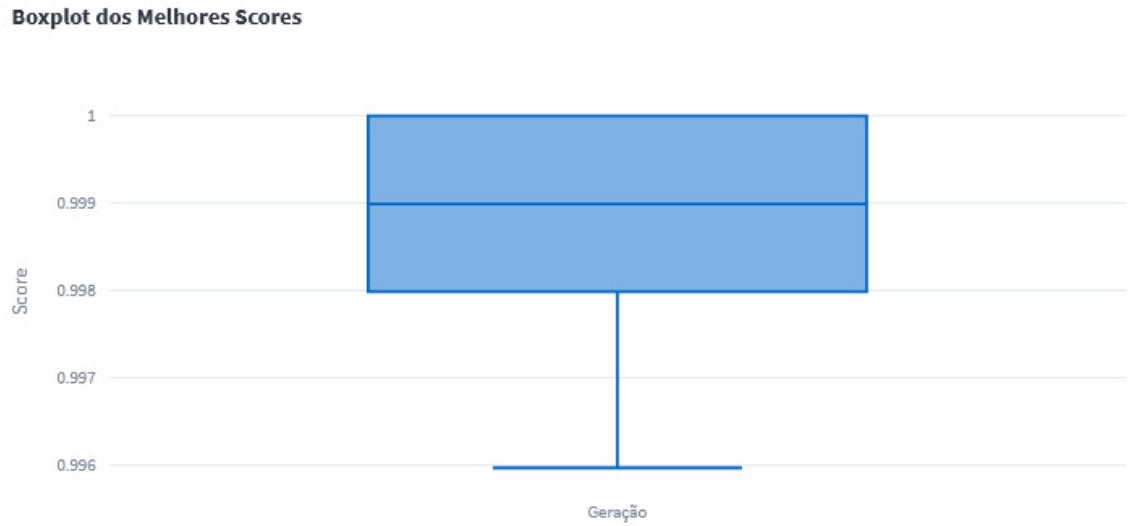

Figure 8. Gráfico Boxplot para tabuleiro 32x32 com solução ótima

A tabela abaixo mostra a relação da função objetivo (colisões) de cada execução.

Execução	Função Objetivo	Fitness
1, 3, 10	1	0.9979
2, 5, 6, 7, 9	0	1.0
4, 8	2	0.995

Table 2. Tabela de resultados tabuleiro 32x32

3.4. Tabuleiro 64x64

Como vimos no início do relatório, quanto maior o tabuleiro, mais a complexidade do problema aumenta, e no caso de 64x64, começamos a perceber isto. No conjunto das 10 execuções, absolutamente nenhuma conseguiu alcançar o valor ótimo em 3500 gerações, muitas chegaram bem próximo, mas sem sucesso. A figura 9 e a figura 10 refletem todas as situações. Novamente percebe-se um comportamento bem diferente em cada execução,

umas convergindo mais rapidamente que outras. Algo interessante de se notar, é que depois de certo tempo, quando um avanço na fitness é feito, o algoritmo fica por muito tempo ”parado” neste mesmo valor de fitness, até sofrer alguma mutação ou crossover que gere um indivíduo melhor, no gráfico de convergência esse comportamento fica bem claro pelas longas linhas em um mesmo valor de fitness.

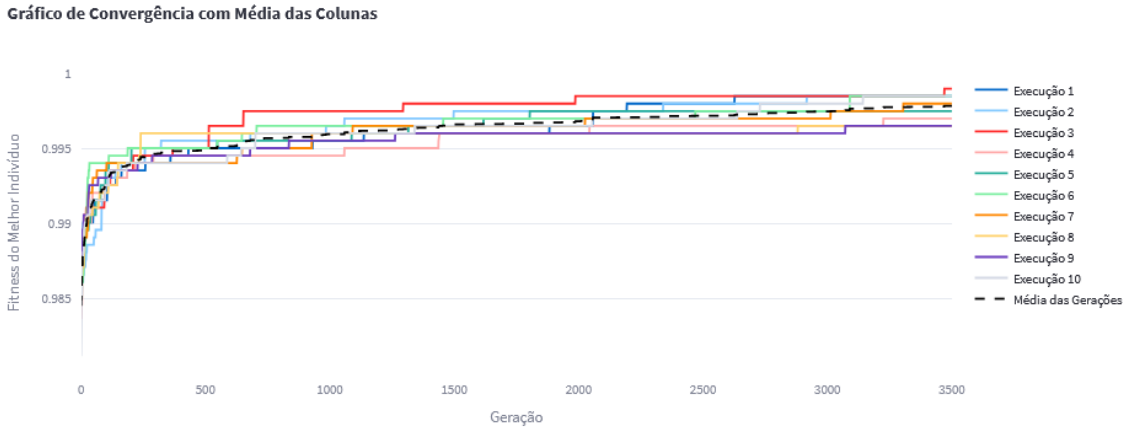

Figure 9. Gráfico de convergência para tabuleiro 64x64

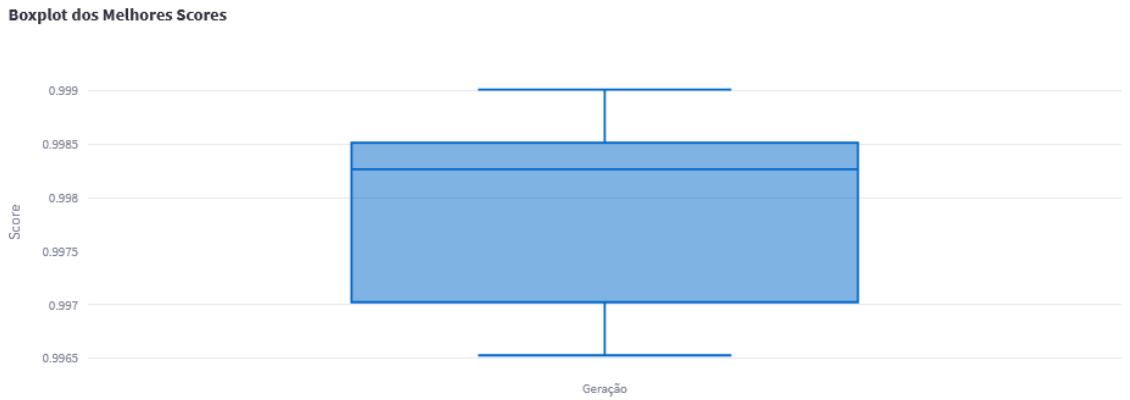

Figure 10. Gráfico Boxplot para tabuleiro 64x64

A tabela 3 mostra os resultados obtidos para função objetivo e consequente fitness em todas as execuções. Percebe-se que por mais que a quantidade de colisões aumente, dentro das possíveis colisões ainda temos sempre boas aproximações que com certeza seriam ainda melhores com o passar das gerações. Neste caso, não conseguimos nenhuma solução ótima.

3.5. Tabuleiro 128x128

Se com um tabuleiro 64x64 não conseguimos chegar em um ponto ótimo com 3500 gerações, o esperado ao se aumentar ainda mais o tamanho do tabuleiro era que também não convergisse para o valor ótimo. E foi o que realmente aconteceu, pelos gráficos, percebemos que chegamos em aproximações mas que de longe ainda não resolvem o problema. Para solucionar o problema nestes casos, o correto seria aumentar o número de

Execução	Função Objetivo	Fitness
1, 2, 6, 10	3	0.9985
3	2	0.999
4	6	0.9970
5	5	0.9975
7	4	0.9980
8, 9	7	0.996

Table 3. Tabela de resultados tabuleiro 64x64

gerações e observar o comportamento. Seria possível também, como conhecemos muito bem o problema, calcular um critério de parada, onde o algoritmo rode até que o problema seja resolvido, ou seja, até que o ponto ótimo seja alcançado.

O gráfico de convergência das execuções é interessante pois mostra a execução 9 desenpenhando e convergindo muito mais rapidamente que as demais, mas ainda devido ao número de iterações utilizadas, não atingiu uma solução ótima. Este caso também está representado pelo outlier superior, que está bem longe dos demais pontos.

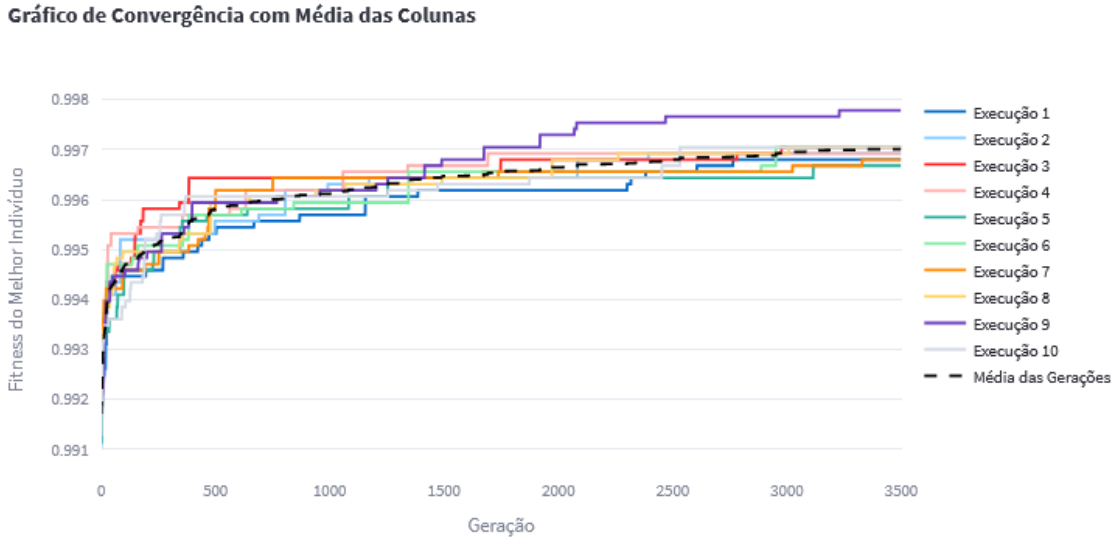

Figure 11. Gráfico de convergência para tabuleiro 128x128

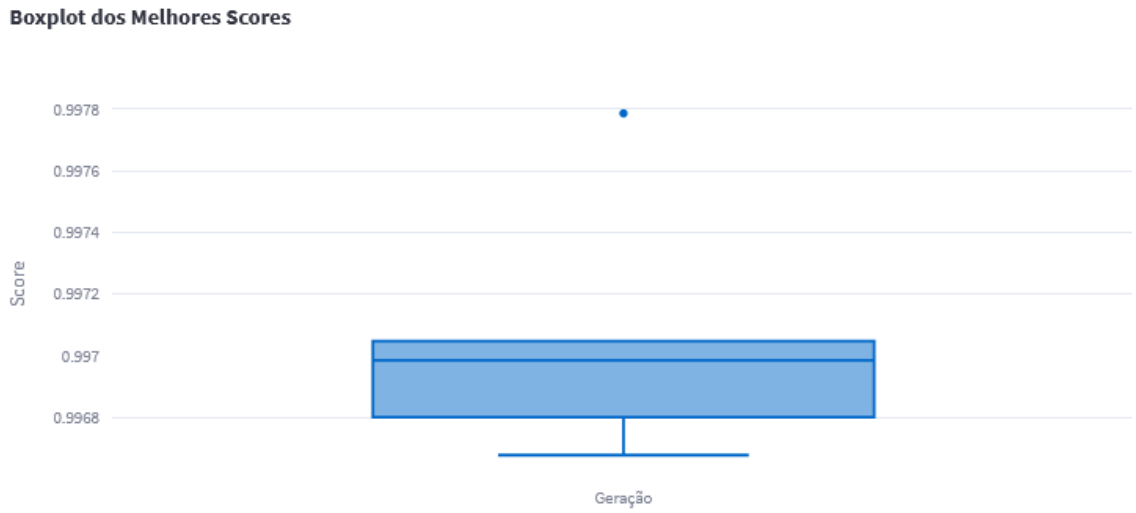

Figure 12. Gráfico Boxplot para tabuleiro 128x128

A tabela 4 apresenta os resultados obtidos nas 10 execuções, percebemos que a primeira execução teve uma tendência de convergência maior que as demais, e é graças a estocasticidade que casos como este ocorrem. Mas no geral, todas tiveram um comportamento semelhante.

Execução	Função Objetivo	Fitness
9	18	0.997785
3, 6, 8 e 10	24	0.997047
2 e 4	25	0.996924
1 e 7	26	0.996801
5	27	0.996678

Table 4. Tabela de resultados tabuleiro 128x128

4. Resultados Obtidos no problema das N-Rainhas Valorado

Como uma extensão do problema atacado nas seções anteriores, agora teremos um board valorado, onde cada posição do board possui um peso. Isso adiciona mais complexidade ao problema, visto que antes possuíamos diversas soluções válidas, e agora mesmo existindo diversas soluções válidas, conseguimos distinguí-las pela soma do peso das casas onde existem rainhas. A definição do peso de uma posição é feita da seguinte forma, em linhas com índice par é calculado o logaritmo base 10 do índice daquela posição e em linhas com índice ímpar, é calculada a raiz quadrada do índice daquela posição. O objetivo aqui é maximizar, então quanto maior for a soma das posições ocupadas por rainhas, melhor a solução. Atacamos aqui somente tabuleiros de tamanho 8x8 e 16x16.

Para realidade de 8x8, a melhor solução, com melhor pontuação, alcançou 0.963, enquanto a pior alcançou 0.946. Ambas resolvem o problema, mas com board valorado, uma é melhor que outra. A tabela 5 mostra os resultados obtidos da função objetivo, definida pelo valor da soma das posições com rainhas e da função fitness. As Execuções 7 e 8 foram as mais bem sucedidas e obtiveram o melhor posicionamento entre as demais, ficando claro pelo maior valor da função objetivo.

Execução	Função Objetivo	Solução	Fitness
1	26.2637	5, 1, 6, 0, 2, 4, 7, 3	0.9547
2	26.2637	5, 1, 6, 0, 2, 4, 7, 3	0.9547
3	26.2377	7, 3, 0, 2, 5, 1, 6, 4	0.9538
4	26.2573	4, 1, 5, 0, 6, 3, 7, 2	0.9545
5	25.9339	7, 2, 0, 5, 1, 4, 6, 3	0.9427
6	26.0498	7, 1, 4, 2, 0, 6, 3, 5	0.9469
7	26.5093	5, 3, 6, 0, 7, 1, 4, 2	0.9636
8	26.5093	5, 3, 6, 0, 7, 1, 4, 2	0.9636
9	25.7916	4, 0, 7, 5, 2, 6, 1, 3	0.9375
10	26.3197	7, 1, 3, 0, 6, 4, 2, 5	0.9567

Table 5. Tabela de resultados tabuleiro 8x8 valorado

Para realidade de 16x16, a melhor solução, com melhor pontuação, alcançou 0.982, enquanto a pior alcançou 0.9750. Não é possível comparar as pontuações dos dois casos (8x8 e 16x16) pois com um board maior, temos mais posições valoradas, mudando a lógica. Todas as soluções resolvem o problema em termos de colisão, mas dentro das execuções, o melhor caso ficou com a execução 4. A tabela 6 apresenta os dados retirados das execuções.

Execução	Função Objetivo	Solução	Fitness
1	99.1441	14, 3, 13, 0, 7, 12, 10, 6, 2, 15, 11, 1, 8, 5, 9, 4	0.9750
2	99.6502	15, 4, 12, 9, 6, 1, 13, 2, 5, 8, 11, 0, 10, 7, 14, 3	0.9800
3	99.5174	13, 6, 14, 3, 7, 9, 12, 2, 15, 1, 11, 8, 0, 4, 10, 5	0.9787
4	99.8957	14, 4, 15, 5, 12, 2, 13, 3, 7, 9, 11, 1, 10, 0, 6, 8	0.9824
5	99.6463	15, 5, 14, 2, 10, 6, 13, 1, 4, 0, 3, 9, 12, 8, 11, 7	0.9799
6	99.4318	14, 4, 7, 0, 11, 1, 15, 5, 10, 13, 9, 2, 12, 3, 6, 8	0.9778
7	99.5035	10, 8, 13, 3, 12, 2, 15, 1, 9, 5, 14, 0, 11, 6, 4, 7	0.9785
8	99.2418	15, 7, 9, 0, 12, 5, 8, 2, 11, 14, 3, 1, 6, 4, 10, 13	0.9759
9	99.2767	10, 13, 7, 4, 12, 3, 15, 6, 11, 2, 14, 1, 8, 5, 9, 0	0.9763
10	99.7141	14, 3, 15, 7, 9, 6, 13, 1, 4, 0, 8, 10, 12, 2, 11, 5	0.9806

Table 6. Tabela de resultados tabuleiro 16x16 valorado

Os gráficos de convergência deixam bastante claro que diferentes execuções chegam em diferentes conclusões de soluções ótimas em 3500 gerações. Algo muito interessante ao analisar o gráfico da figura 15 é que a execução que chegou no melhor resultado, representada pela execução 4 de linha cor de rosa claro, não necessariamente foi a execução que mantinha uma tendência de convergência alta, conseguimos ver que perto das ultimas gerações ela evolui além das demais, graças a estocasticidade.

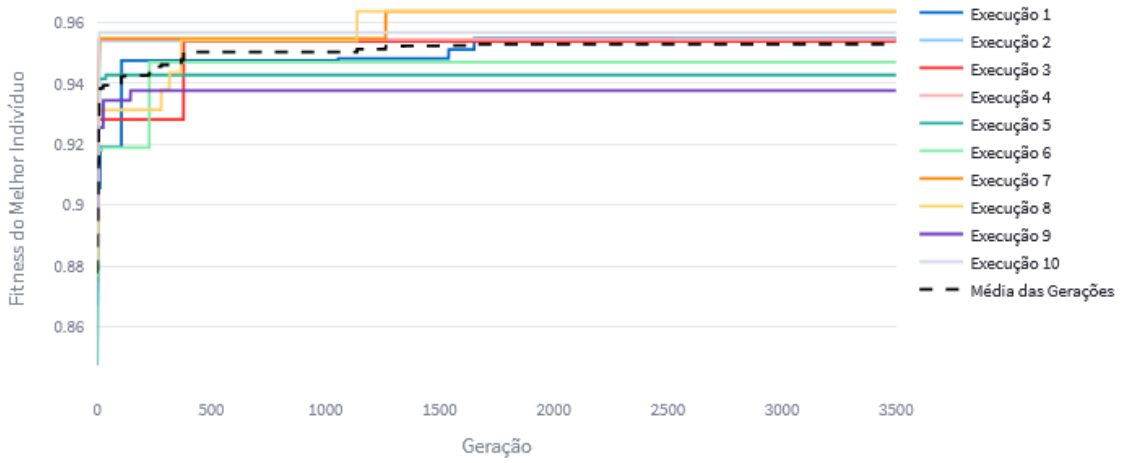

Figure 13. Gráfico de convergência para tabuleiro 8x8 valorado

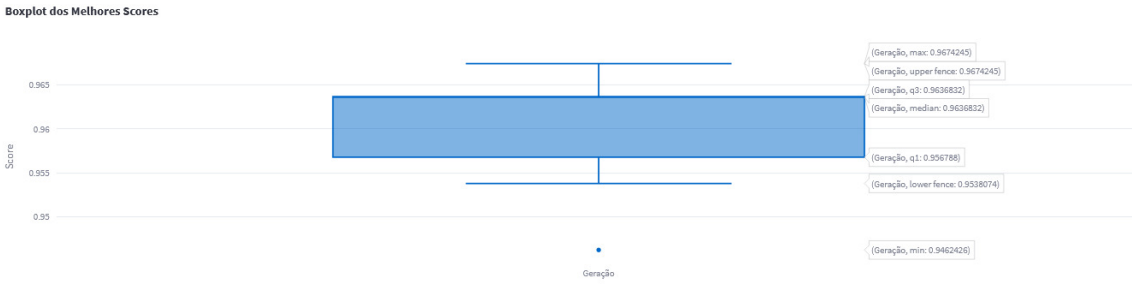

Figure 14. Gráfico Boxplot para tabuleiro 8x8 valorado

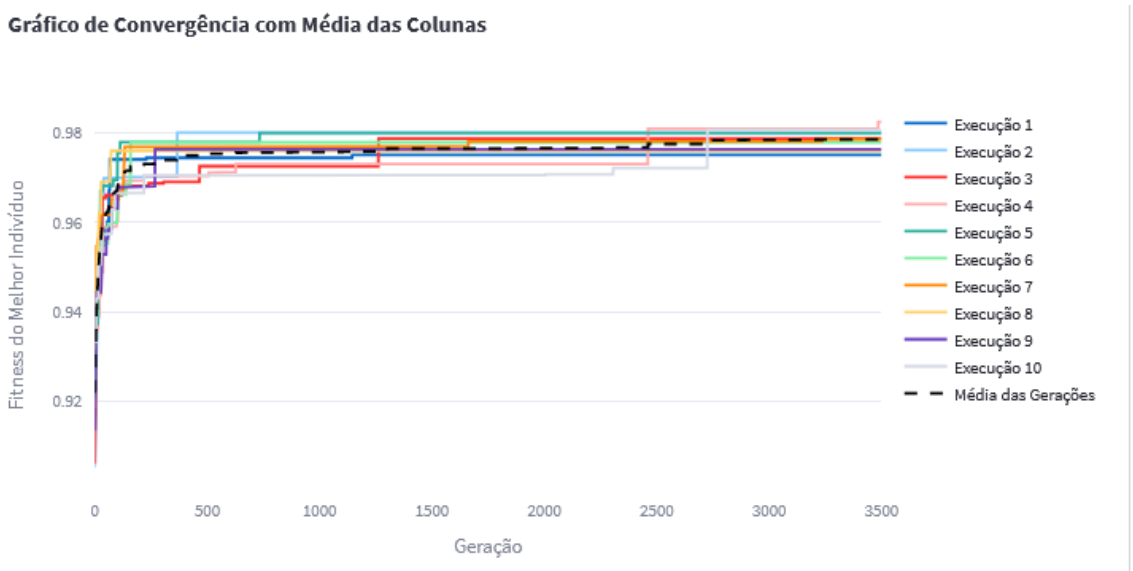

Figure 15. Gráfico de convergência para tabuleiro 16x16 valorado

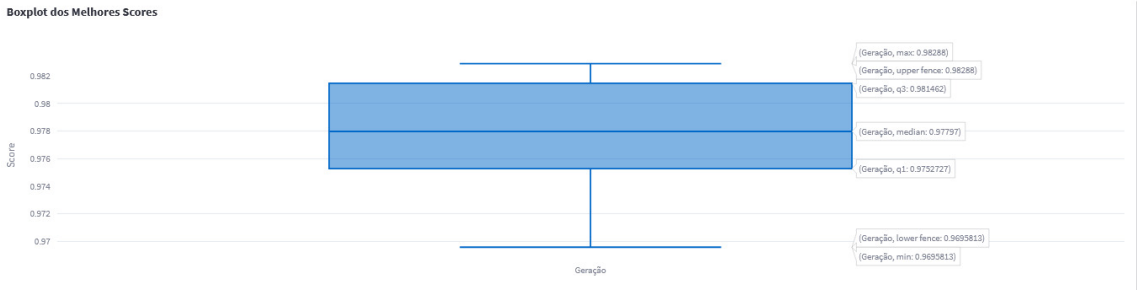

Figure 16. Gráfico Boxplot para tabuleiro 16x16 valorado

5. O Problema dos Rádios

Relativamente mais simples que os problemas atacados anteriormente, o problema dos rádios considera uma fábrica com duas linhas de produção de rádios Standard e de Luxo, com diferentes informações de produção cada. Para standard, a linha de produção comporta no máximo 24 funcionários, cada rádio consome 1 homem/dia e fornece 30 reais de lucro. Para Luxo, a linha de produção comporta no máximo 32 funcionários, cada rádio consome 2 homem/dia e fornece 40 reais de lucro, porém a fábrica possui um total de 40 funcionários a serem alocados. O objetivo prático, é maximizar o lucro diário.

A codificação utilizada levou em consideração indivíduos do tipo binário de 10 bits, sendo 5 bits para representar os rádios Standard e 5 bits para os rádios de luxo, dessa forma, o cálculo realizado da função objetivo inclui ajustes de escala para trabalhar com os valores em binário. Outra opção seria realizar a codificação em tipo real, dessa maneira não seriam necessários ajustes de escala.

As configurações utilizadas foram as seguintes:

- Quantidade de Execuções: 10
- Codificação: Binário
- Tipo de Seleção: Torneio Estocástico
- Tipo de Crossover: crossover de 1 ponto
- Tipo de Mutação: Bit-flip
- Elitismo: Verdadeiro
- Dimensão dos Indivíduos: 10
- Tamanho da População Inicial: 20
- Quantidade de Gerações: 100

O problema é bastante simples, no final o maior desafio foi trabalhar com codificação binária. A figura 17 deixa bem claro a rápida convergência de alguma execuções e mais lenta em outras.

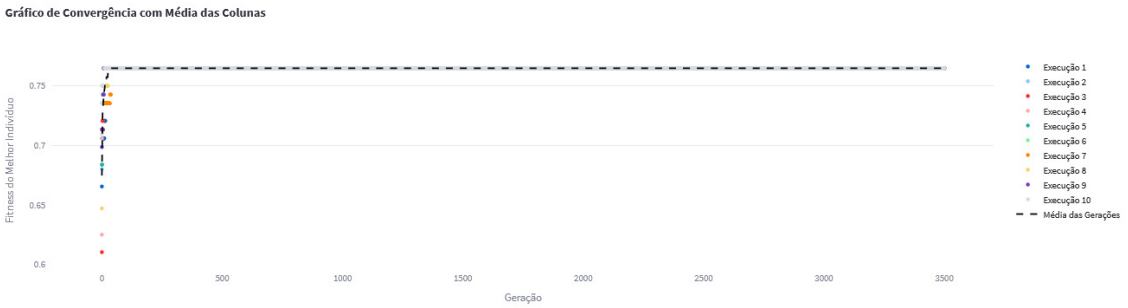

Figure 17. Gráfico de convergência para problema dos rádios

A tabela a seguir demonstra os valores que compuseram a função objetivo e por consequência, o cálculo da fitness, a maioria das execuções convergiu para o valor de 24 rádios standard e 8 rádios de luxo. As demais soluções provavelmente convergiriam para a mesma solução no decorrer de mais gerações.

Execução	Rádios Standard	Rádios de Luxo	Fitness
1	24	7	0.735
2,3,4,5,6,7,8 e 10	24	8	0.764
9	22	9	0.75

Table 7. Tabela dos valores das melhores soluções encontradas em cada execução

6. Conclusão e Trabalhos Futuros

A implementação desta ferramenta contribuiu para consolidação de todos os conceitos básicos de computação evolutiva, vistos de forma teórica durante as aulas ministradas e artigos lidos. Permitiu perceber o quanto é util a utilização de aplicações de computação evolutiva principalmente na resolução de problemas complexos. Um exemplo é o problema das rainhas, talvez em um algoritmo de força bruta, o problema se torne inviável de se resolver muito mais facilmente, mas como não foi testado outro algoritmo, fica como ideia para trabalhos futuros.

Um ponto que foi bastante crítico e que atacou diretamente a qualidade dos testes foi o paralelismo, a ferramenta não foi desenvolvida de forma paralela ao aplicar a função fitness em todos os casos, causando um alto custo computacional e uma certa lentidão ao realizar os testes. Este foi o principal motivo da escolha de 3500 iterações, visto que se realizasse 10 execuções até encontrar um valor ótimo principalmente em tabuleiros 128x128, levaria muito tempo. Então como trabalhos futuros seria bastante interessante implementação de técnicas de paralelismo pra otimização da complexidade de tempo do código.