Primeiro Relatório de OCEV sobre Desenvolvimento de Ferramenta Baseada em Algoritmo Genético

Rafael Granza de Mello[1]

[1]Universidade Estadual do Estado de Santa Catarina (UDESC)
Caixa Postal 631 – 89.219-710 – Joinville – SC – Brazil

rafaelgranzademello@gmail.com

Resumo. *O presente relatório tem como objetivo a aplicação de conceitos de algoritmos genéticos para utilizar e avaliar uma ferramenta de otimização baseada em protocolos web. Por ser um software ainda em desenvolvimento foram escolhidos e analisados um problema famosos e mais complexo de otimização, as N-Rainhas valoradas. Os resultados obtidos ainda são inconclusivos para se validar a aplicação como um todo, porém os problemas apresentados foram resolvidos satisfatoriamente.*

1. Introdução

O setor de desenvolvimento web tem ganho muito espaço nas duas últimas décadas, por esse motivo muitas ferramentas e serviços foram criadas para auxiliar o trabalho dos programadores dessa área. Dentro deste mesmo contexto, é comum que seja necessário resolver diversos problemas de otimização no *back-end* de algumas aplicações como a melhor rota, menor valor de uma lista de itens ou a recomendação que gere mais cliques. Esse relatório se propõe a avaliar a eficiência de uma ferramentas simples e genérica desenvolvida [1] para resolver problemas de otimização com algoritmos genéticos se valendo dos paradigmas web.

Por ser uma ferramenta elaborada no padrão *API Rest* e ainda estar em fase inicial de criação, nossos testes e resultados serão simbólicos e terão um caráter mais voltado à auxiliar na calibragem na ferramenta. Valendo-se disso foi escolhido um problema clássico porém de difícl otimização:

- ***N*-rainhas Valoradas**: Dados valores para cada célula de um tabuleiro, deve-se posicionar N rainhas em um tabuleiro de xadrez $N \times N$ de tal forma que nenhuma delas possa se atacar e a soma das células escolhidas seja máxima possível.

Este artigo está organizado da seguinte forma: a seção 2 está dividida em três etapas a primeira explicará um pouco sobre o funcionamento da ferramenta, a segunda vai ilustrar a codificação do problema, e a outra dirá quais foram as variáveis e funções escolhidas; a seção 3 apresenta os resultados obtidos das simulações e experimentações; a seção 4 analisará os valores obtidos na seção anterior e as considerações finais são apresentadas na seção 5.

2. Metodologia de Desenvolvimento

Antes de explicar a codificação para cada problema, faz-se necessário explicar como nos comunicamos com a ferramenta avaliada.

O paradigma é baseado em cliente-servidor de tal forma que toda a computação é feita em um servidor e este apenas responde com os melhores indivíduos e seus valores de *fitness* de cada *run* do algoritmo genético. A comunicação se dá por meio do URL com as informações compiladas em um objeto do tipo JSON, onde o desenvolvedor indica uma função de *fitness* e um protótipo de indivíduo, além disso ele pode passar alguns parâmetros comuns aos algoritmos genéticos. A figura 1 tem o esquema de como funcionaria essa API.

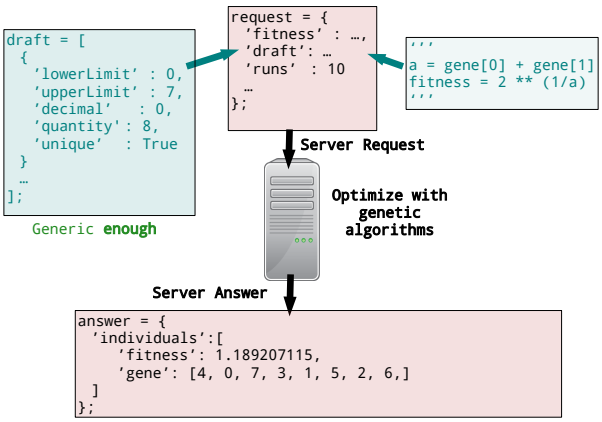


Figura 1. Esquema da API

A lista dos possíveis parâmetros que podem ser passados para a API e seus significados:

Parâmetro	Tipo	Significado
fitness	Função descrita via String	Computa o fitness de cada indivíduo
draft	Lista de dicionários	Define o modelo de cada indivíduo
title	String	Nome da aplicação
runs	Int	Quantidade de simulações (Paralelismo Ativo)
generations	Int	Quantidade de gerações da simulação
penaltyFactor	Float	Fator de penalidade
mutationProb	Float [0, 1]	Probabilidade de Mutação
crossoverProb	Float [0, 1]	Probabilidade de Crossover
elitism	Boolean	Define se o melhor indivíduo será preservado em cada iteração
crossoverOperator	String (somente pré-definidos)	Operador de Crossover
mutationOperator	String (somente pré-definidos)	Operador de Mutação

Tabela 1. Parâmetros, tipos e significados

Dentro do *fitness* é possível definir algumas variáveis que serão reconhecidas pela API e serão utilizadas. Na tabela 2 é possível visualizar como e quais delas serão acessadas, onde 'Leitura' significa que a variável será selecionada por quem envia requisição

da API enquanto 'Escrita' se refere àquelas variáveis cujo o valor é calculado dentro da função de *fitness*, mas a API as usa como informação importante para o algorítmo de otimização.

Variável	Leitura	Escrita
fitness		✓
penalty		✓
penaltyFactor	✓	

Tabela 2. Variáveis do Fitness

2.1. N-Rainhas com Tabuleiro Valorado

Como já explicado antes, esse otimização consiste em posicionar *N* rainhas em um tabuleiro de xadrez $N \times N$ de tal sorte que nenhuma delas possa se atacar. Para essa versão pontual desse problema, iremos testar os valores de N apenas como as potências de 2, partindo do 8 e seguindo até 128 (incluso).

Para dar mais agilidade e eficiência ao código, o protótipo escolhido era baseado no fato de que para nenhuma rainha pode estar na mesma linha de outra, sabendo disso é determinado uma permutação com números de 0 à $N - 1$, sendo que cada valor indicaria a linha onde a rainha respectiva ficaria.

Mesmo dessa forma, ainda é possível que hajam colisões e por esse motivo foi definido um valor de penalidade P para auxiliar na formação do cálculo da *fitness* como podemos ver nas fórmulas abaixo:

$$Col_{factor} = \frac{Col}{Col_{max}}$$

$$P = min(\frac{V}{V_{max}}, penaltyFactor \times Col_{factor})$$

$$F = \frac{V}{V_{max}} - P$$

Onde Col_{factor} é o valor do fator de colisão, Col é o número de colisão desse indivíduo, Col_{max} é o número máximo de colisões que podem existir, V é a valoração desse indivíduo, V_{max} é uma valoração melhor ou igual a ótima, $penaltyFactor$ é um fator de penalidade definido na chamada da API e F é o valor final de *fitness*. De acordo com a função definida nem sempre será possível que o indivíduo ótimo tenha o valor de *fitness* igual a 1.

A valoração de cada célula do tabuleiro foi definida pela seguinte fórmula:

$$v_{i,j} = n \times i + j + 1$$

$$C_{i,j} = \begin{cases} \sqrt{v_{i,j}} & \text{se } i \text{ é par} \\ \log_{10}(v_{i,j}) & \text{se } i \text{ é ímpar} \end{cases}$$

Onde n é a dimensão do tabuleiro, i o índice da linha, j o índice da coluna e $C_{i,j}$ a valoração da célula na linha i e coluna j.

Os outros parâmetros utilizados na ferramenta estão indicados na tabela 3.

Parâmetro	Valor
runs	10
generations	500_8, 1000_{16}, 2000_{32}, 4000_{64}, 16000_{128}
elitism	True
mutationProb	0.05
crossoverProb	0.85
populationSize	30
penaltyFactor	8
crossoverOperator	cycleCrossover
mutationOperator	swapMutation

Tabela 3. Parâmetros das N-Rainhas com Tabuleiro Valorado

Na linha *generations* são dados uma lista de valores com um número em *underline* que representa o caso de teste em que esse valor foi escolhido.

3. Resultados

Nessa seção serão mostrados alguns exemplos de retorno da ferramenta com o melhor indivíduo encontrado, é importante ressaltar que apesar da ferramenta retornar os melhores indivíduos encontrados durante as simulações será mostrado apenas o melhor para não consumir muito espaço. Além disso serão mostradas a curva de convergência e os gráficos *BoxPlot* do problemas das rainhas valoradas com 8, 16, 32, 64 e 128 dimensões.

3.1. 8-Rainhas Valoradas

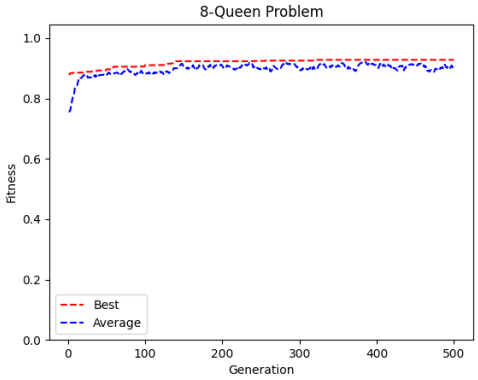
O melhor indivíduo encontrado nessa busca não teve nenhuma penalidade como podemos ver abaixo:

```
{
    "fitness": 0.9510099004479453,
    "penalty": 0,
    "value": 26.823263695511123,
    "genes": [6, 2, 7, 1, 4, 0, 5, 3]
}
```

Algoritmo 1. Melhor indivíduo encontrado para o problema das 8-Rainhas Valoradas

Os gráficos de convergência abaixo mostram a média do melhor indivíduo em cada *run* além da média das médias durante a evoluções das gerações. Podemos notar que com 500 gerações foi possível alcançar a convergência.

(a) Geral



(b) Zoom

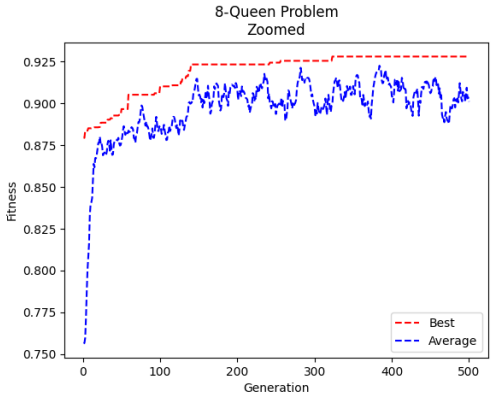
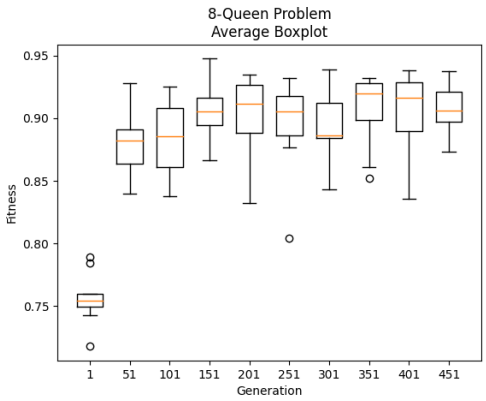


Tabela 4. Gráfico de convergência das 8-Rainhas Valoradas

Os gráficos abaixo são Boxplots da médias(a) e dos melhores(b) indivíduos durante as *runs*, nele é possível perceber que mesmo considerando o desvio padrão é possível indicar que o algoritmo estava convergindo com o passar das gerações.

(a) Médias



(b) Melhores

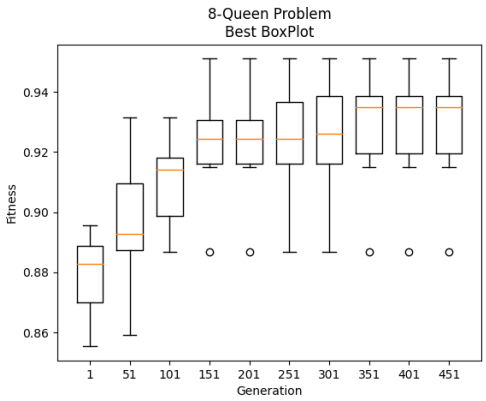


Tabela 5. BoxPlot das 8-Rainhas Valoradas

3.2. 16-Rainhas Valoradas

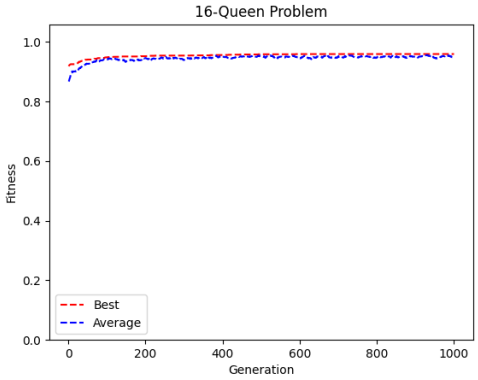
O melhor indivíduo encontrado nessa busca não teve nenhuma penalidade como podemos ver abaixo:

```
{
    "fitness": 0.9634212605289554,
    "penalty": 0,
    "value": 99.00432419807952,
    "genes": [10, 6, 15, 12, 4, 9, 3, 5, 14, 11, 1, 7, 13, 0, 2, 8]
}
```

Algoritmo 2. Melhor indivíduo encontrado para o problema das 16-Rainhas Valoradas

Os gráficos de convergência abaixo mostram a média do melhor indivíduo em cada *run* além da média das médias durante a evoluções das gerações. Podemos notar que com 1000 gerações foi possível alcançar a convergência.

(a) Geral



(b) Zoom

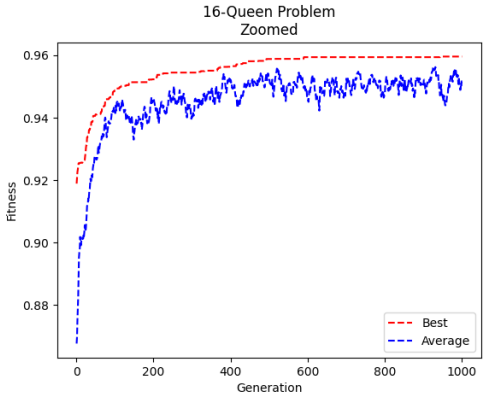
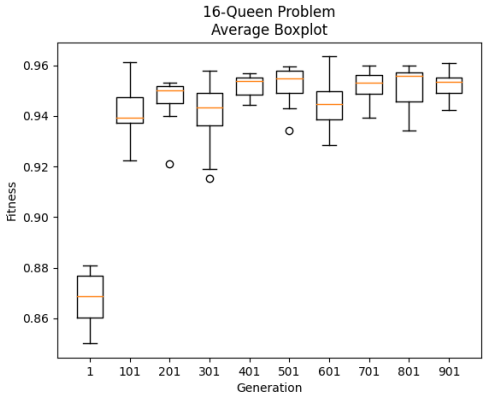


Tabela 6. Gráfico de convergência das 16-Rainhas Valoradas

Os gráficos abaixo são Boxplots da médias(a) e dos melhores(b) indivíduos durante as *runs*, nele é possível perceber que mesmo considerando o desvio padrão é possível indicar que o algoritmo estava convergindo com o passar das gerações.

(a) Médias



(b) Melhores

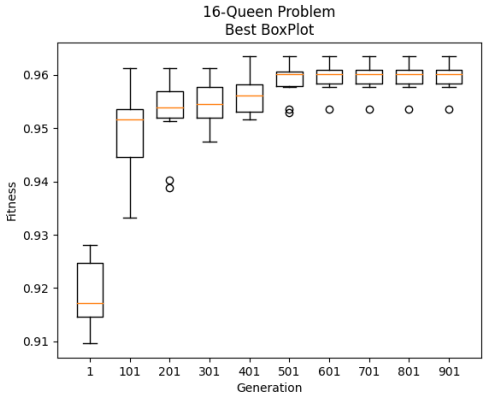


Tabela 7. BoxPlot das 16-Rainhas Valoradas

3.3. 32-Rainhas Valoradas

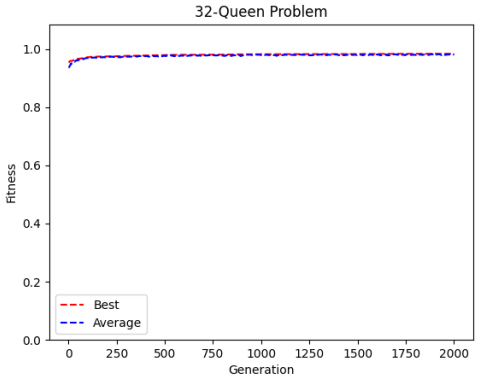
O melhor indivíduo encontrado nessa busca não teve nenhuma penalidade como podemos ver abaixo:

```
{
    "fitness": 0.9864983003724732,
    "penalty": 0,
    "value": 379.2911519242103,
    "genes": [30, 13, 15, 24, 22, 14, 25, 8, 31, 2, 26, 1, 9, 7, 28, 10, 27, 21, 5, 3,
        12, 19, 29, 23, 20, 16, 11, 6, 17, 0, 18, 4]
}
```

Algoritmo 3. Melhor indivíduo encontrado para o problema das 32-Rainhas Valoradas

Os gráficos de convergência abaixo mostram a média do melhor indivíduo em cada *run* além da média das médias durante a evoluções das gerações. Podemos notar que com 2000 gerações foi possível alcançar a convergência.

(a) Geral



(b) Zoom

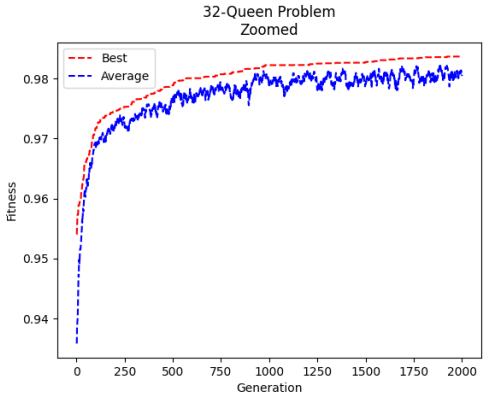
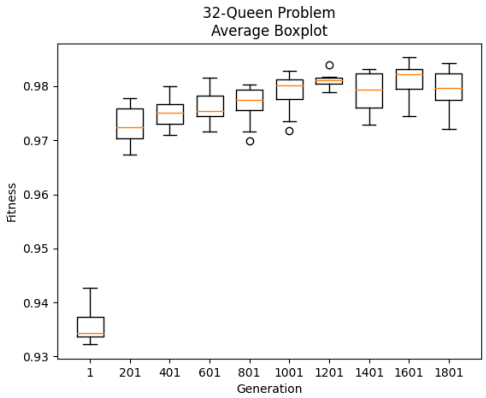


Tabela 8. Gráfico de convergência das 32-Rainhas Valoradas

Os gráficos abaixo são Boxplots da médias(a) e dos melhores(b) indivíduos durante as *runs*, nele é possível perceber que mesmo considerando o desvio padrão é possível indicar que o algoritmo estava convergindo com o passar das gerações.

(a) Médias



(b) Melhores

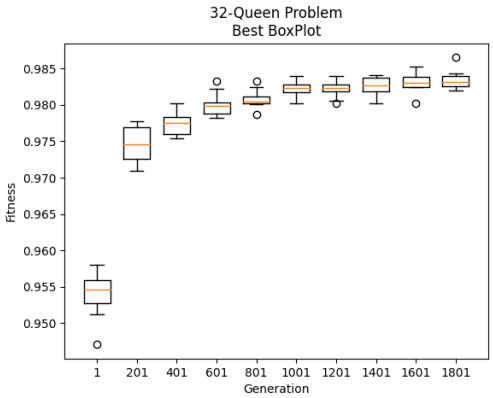


Tabela 9. BoxPlot das 32-Rainhas Valoradas

3.4. 64-Rainhas Valoradas

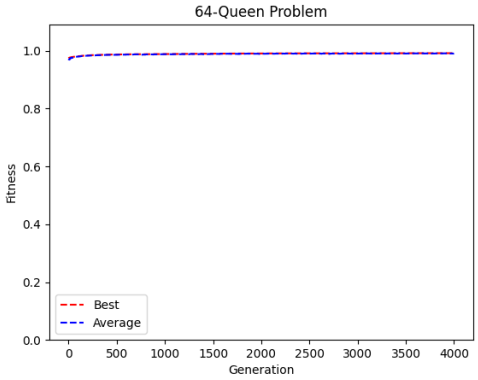
O melhor indivíduo encontrado nessa busca não teve nenhuma penalidade como podemos ver abaixo:


```
{
    "fitness": 0.992796407173767,
    "penalty": 0,
    "value": 1458.7337048049258,
    "genes": [56, 24, 32, 12, 44, 25, 58, 10, 50, 60, 51, 39, 34, 11, 46, 5, 41, 30, 61,
        53, 57, 6, 29, 16, 21, 3, 26, 15, 45, 9, 38, 18, 33, 52, 48, 8, 19, 55, 49, 35,
        22, 0, 63, 1, 59, 14, 40, 7, 47, 4, 17, 37, 62, 27, 43, 36, 20, 13, 42, 54, 23,
        2, 28, 31]
}
```


Algoritmo 4. Melhor indivíduo encontrado para o problema das 64-Rainhas Valoradas

Os gráficos de convergência abaixo mostram a média do melhor indivíduo em cada *run* além da média das médias durante a evoluções das gerações. Podemos notar que com 4000 gerações foi possível alcançar a convergência.

(a) Geral



(b) Zoom

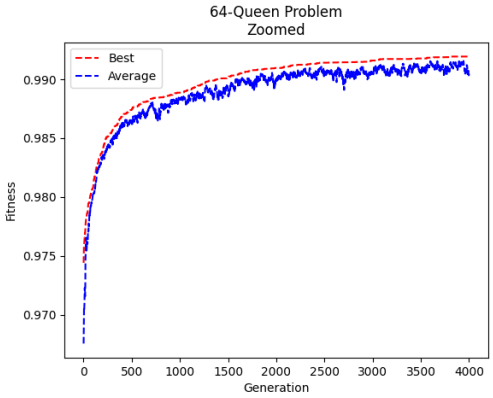
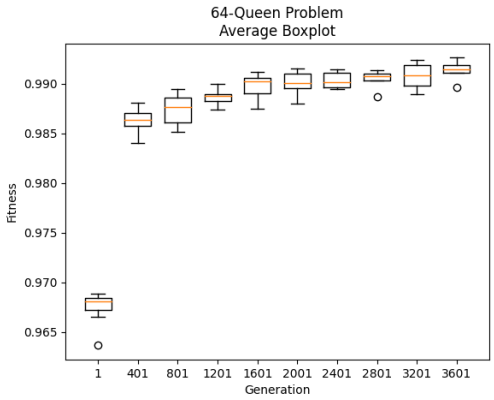


Tabela 10. Gráfico de convergência das 64-Rainhas Valoradas

Os gráficos abaixo são Boxplots da médias(a) e dos melhores(b) indivíduos durante as *runs*, nele é possível perceber que mesmo considerando o desvio padrão é possível indicar que o algoritmo estava convergindo com o passar das gerações.

(a) Médias



(b) Melhores

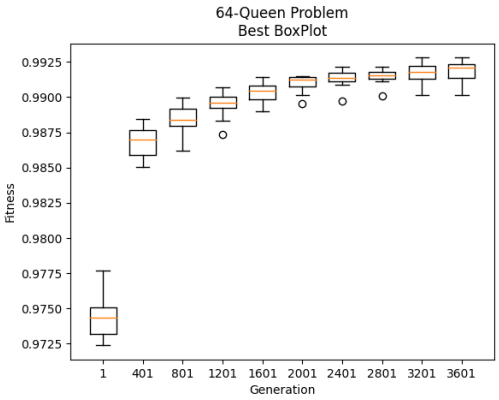


Tabela 11. BoxPlot das 64-Rainhas Valoradas

3.5. 128-Rainhas Valoradas

O melhor indivíduo encontrado após mais de **12 horas** de processamento nessa busca não teve nenhuma penalidade como podemos ver abaixo:

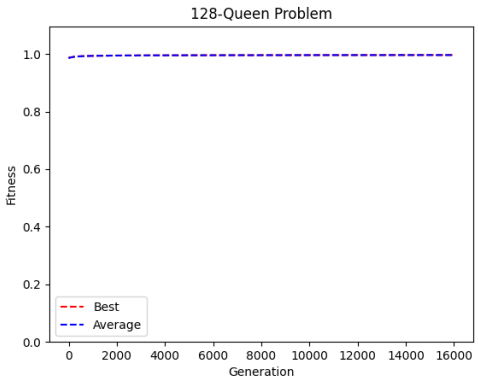

```
{
    "fitness": 0.9967976103704793,
    "penalty": 0,
    "value": 5687.712870578774,
    "genes": [126, 29, 98, 105, 120, 24, 81, 48, 88, 27, 93, 57, 109, 61, 79, 84, 115,
        64, 58, 39, 96, 46, 127, 102, 124, 47, 118, 8, 89, 66, 85, 44, 110, 6, 101, 74,
        125, 3, 91, 22, 113, 25, 116, 69, 121, 17, 77, 10, 111, 36, 1, 19, 40, 60, 83,
        16, 107, 123, 82, 51, 122, 2, 114, 78, 34, 32, 41, 38, 23, 21, 119, 18, 14, 4,
```

```
        95, 13, 103, 55, 108, 59, 71, 117, 86, 0, 73, 45, 50, 104, 99, 75, 87, 15, 92,
        67, 33, 7, 54, 76, 30, 20, 112, 100, 97, 43, 63, 42, 68, 26, 106, 53, 56, 11,
        94, 5, 90, 37, 72, 28, 52, 62, 65, 35, 70, 80, 31, 9, 49, 12]
}
```

Algoritmo 5. Melhor indivíduo encontrado para o problema das 128-Rainhas Valoradas

Os gráficos de convergência abaixo mostram a média do melhor indivíduo em cada *run* além da média das médias durante a evoluções das gerações. Podemos notar que com 16000 gerações foi possível alcançar a convergência.

(a) Geral



(b) Zoom

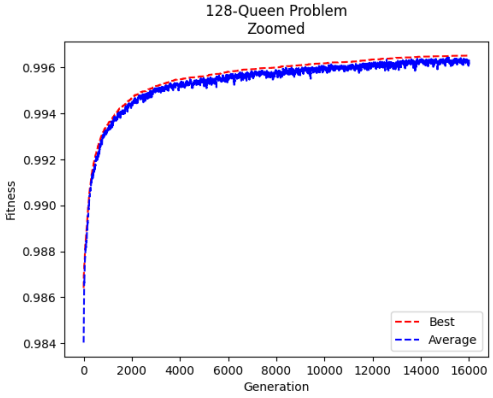
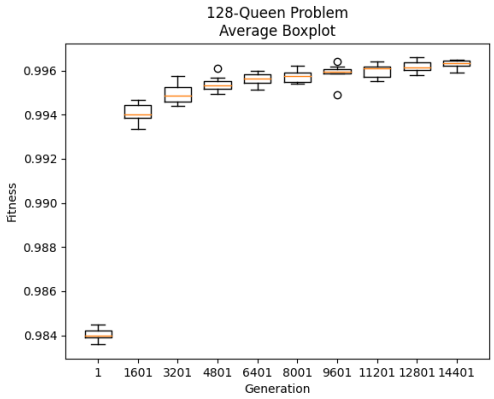


Tabela 12. Gráfico de convergência das 128-Rainhas Valoradas

Os gráficos abaixo são Boxplots da médias(a) e dos melhores(b) indivíduos durante as *runs*, nele é possível perceber que mesmo considerando o desvio padrão é possível indicar que o algoritmo estava convergindo com o passar das gerações.

(a) Médias



(b) Melhores

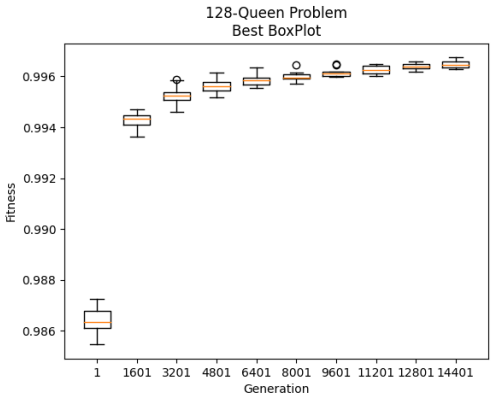


Tabela 13. BoxPlot das 128-Rainhas Valoradas

4. Análise dos Resultados

Apesar de terem diferetens dimensões de tabuleiros, os resultados obtidos pelos problemas do tipo N-Rainhas tiveram finais muito similares, alcançando indivíduos com *fitness*

muito próximos do valor ótimo e mantendo uma melhoria constante durante todas as gerações. É notório que quanto maior a dimensão mais gerações foram necessárias para alcançar um resultado satisfatório, porém isso ja era esperado.

5. Conclusões

De forma geral, apesar de ter se aproximado ao *fitness* ótimo em todos os exemplos analisados, é evidente que muitas melhorias ainda podem ser feitas. Podem ser apontados dificuldades como a necessidade de muitas gerações em alguns problemas e a falta de convergência das variáveis médias em alguns casos que , podem indicar que algumas variáveis devem ser ajustadas e que algumas operações genéticas podem estar mal aplicadas.

Por fim sobre a ferramenta utilizada, apesar do trabalho ter esclarecido alguns pontos pendentes, ainda são necessários alguns testes com problemas muito mais complexos para saber realmente se esse tipo de arquitetura pode se encaixar em categorias diferentes de questões.

Referências

[1] Granza R., *Github:GA_Rest*, https://github.com/RafaelGranza/GA_Rest, 2021.