# Computação Evolucionária
# Trabalho Prático 1

Victor São Paulo Ruela
Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
Email: victorspruela@gmail.com

***Resumo*—O objetivo deste trabalho é estudar a aplicação de algoritmos genéticos para minimização da função de Rastrigin e a solução do problema das N-Rainhas. Dessa forma, é possível ver como essa classe de algoritmos se comporta para dois tipos diferentes de problema de otimização, bem como os desafios de sua implementação e ajuste dos seus hiper-parâmetros.**

## I. Introdução

O problema das N-Rainhas [2] consiste em posicionar N rainhas num tabuleiro de xadrez regular (NxN), de forma que nenhuma rainha seja capaz de capturar outra rainha. Portanto, uma solução requer que não existam duas rainhas que compartilhem a mesma linha, coluna ou diagonal no tabuleiro. Para 8 rainhas, o problema já se torna computacionalmente custoso, uma vez que existem mais de 4 bilhões de organizações possíves para somente 92 soluções.

A função de Rastrigin [3] é uma função não-convexa cujo mínimo global é difícil de ser encontrado devido ao seu grande espaço de busca e quantidade de mínimos locais, sendo utilizada para avaliar o desempenho de algoritmos de otimização. Para duas variáveis, as curvas de nível são exibidas na Figura 1.

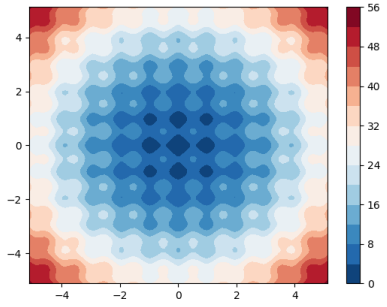


Figura 1. Curvas de níveis para função de Rastrigin em duas dimensões

## II. Metodologia

A primeira etapa consiste na implementação da funções objetivo de cada problema e dos algoritmos genéticos que serão utilizados para sua solução, o que será feito na linguagem Python (versão 3.8.5). A validação da implementação será feita sobre funções de teste simples presentes na literatura.

Será considerada inicialmente uma instância padrão dos algoritmos para cada problema e avaliados os principais hiper-parâmetros para entender o impacto de cada um sobre o desempenho do algoritmo. Baseado nos resultados obtidos, novos operadores ou estratégias para os algoritmos serão experimentadas com o objetivo de melhorar o desempenho da instância padrão, descritas a seguir.

### A. Problema das N-Rainhas

*1) Representação:* O genótipo será representado por permutação de inteiros, utilizando um vetor com o tamanho do número de colunas, onde cada valor será um inteiro de 1 a N, indicando em qual linha a rainha está posicionada. Além disso, não será permitido ter números repetidos, de forma a eliminar as soluções inválidas de rainhas na mesma linha, reduzindo o espaço de busca do algoritmo.

*2) Critérios de parada:* Será considerado como critério a convergência para uma solução ótima, ou um número máximo de 300 gerações.

*3) População inicial:* A população inicial será obtida aleatoriamente de acordo com o tamanho da população escolhida. Seu valor padrão será de 100 indivíduos.

*4) Função de escalonamento:* Dado que na representação adotada o número máximo de cheques em um tabuleiro é dado por $q_{max} = N(N-1)/2$, para $N = 8$ este número será 28. Como a solução do problema consiste em minimizar o número de cheques, logo a função de fitness utilizada seguirá a método de *shift*:

$$fitness = q_{max} - q(f) \tag{1}$$

*5) Operador de Cruzamento:* Foi utilizado o operador *Order Crossover*, o qual realiza os seguintes passos [1]:

1) Escolher dois pontos de cruzamento aleatoriamente e copiar o segmento entre eles do primeiro pai no primeiro filho
2) Iniciando do segundo ponto de cruzamento no segundo pai, copiar o restante dos números não utilizadas no primeiro filho na ordem que eles aparecem
3) Criar o segundo filho analogamente, porém invertendo a ordem dos pais.

Dois indivíduos são escolhidos aleatoriamente da população para o cruzamento com uma determinada probabilidade, cujo valor padrão é de 0.6.

*6) Operador de Mutação:* Será utilizado o operador *Swap Mutation*. Nele, duas variáveis de um indivíduo são escolhidas aleatoriamente e trocadas de posição. Em cada geração, a mutação poderá ocorrer para cada indivíduo com uma determinada probabilidade, cujo valor padrão é de 0.1.

*7) Seleção dos pais:* Dois indivíduos são escolhidos utilizando a técnica de amostragem estocástica universal. Ela é uma extensão do algoritmo da roleta, com a diferença de que ao invés de girar a roleta $\lambda$ vezes, esta é divida em $\lambda$ espaços iguais e girada somente uma vez. É mais indicada quando queremos amostrar mais de um indivíduo de uma mesma população [1].

*8) Seleção dos sobreviventes:* Os dois filhos gerados são incluídos na população e são eliminados os dois indivíduos com o pior fitness.

### B. Função de Rastrigin

*1) Representação:* O genótipo será representado usando código de gray com $L = 20$ bits por variável de decisão. Como o problema possui muitos mínimos locais, o fenômeno de *Hamming Cliffs* pode afetar o desempenho do algoritmo, o que motivou o uso da codificação de gray em relação à binária.

*2) Critérios de parada:* Será considerado como critério de parada 10000 avaliações da função objetivo.

*3) População inicial:* A população inicial será obtida aleatoriamente de acordo com o tamanho da população escolhido. O tamanho padrão será de 100 indivíduos.

*4) Função de escalonamento:* Dado que este é um problema de minimização e o mínimo da função de rastrigin possui valor 0, a técnica de inversão será utilizada para definir a função de fitness da seguinte forma:

$$fitness = \frac{1}{f(x) + 1} \tag{2}$$

*5) Operador de Cruzamento:* Será utilizado o operador de 1 ponto de corte por variável. Pares de indivíduos são escolhidos aleatoriamente na população, sem repetição. O cruzamento é realizado para cada par com uma determinada probabilidade, cujos valores serão adaptados linearmente a cada geração. A faixa padrão na qual os valores variam é de 0.6 a 0.9.

*6) Operador de Mutação:* Será utilizado o operador bit fip. A mutação poderá ocorrer por bit e para cada variável, com uma probabilidade variando linearmente de acordo com a geração. A faixa padrão na qual os valores variam é de 0.05 a 0.01.

*7) Seleção dos pais:* Em cada geração, o operador de seleção poderá ser o da roleta ou torneio. É sorteado um número no intervalo de 0 a 1, e se o valor é menor que 0.5, a seleção será por roleta, caso contrário será por torneio. Serão considerados 10 candidatos aleatórios da população para a seleção por torneio.

*8) Seleção dos sobreviventes:* Será utilizada uma abordagem geracional, logo esta etapa não será necessária.

## III. Resultados

### A. Problema das N-Rainhas

*1) Efeito do tamanho da população:* Para avaliar o efeito do tamanho da população, executou-se o algoritmo para populações de 10 e 100 indivíduos. A distribuição dos resultados para 30 execuções de cada instância podem ser vistos na Figura 2. É possível notar que para a população menor, a distribuição da geração de convergência concentra-se no valor máximo de gerações, ou seja, o algoritmo não convergiu na maioria das execuções. De fato, 18 execuções desta instância convergiram, contra 22 da com 100 indivíduos. Logo, pode-se concluir que populações maiores favorecem a convergência do algoritmo.

Um resultado inesperado que ocorreu para a instância com população de 100 indivíviduos está no fato de que em 13 vezes uma solução ótima era encontrada na população inicial. Isso pode ter sido o resultado de uma boa escolha do genótipo para representar o problema, ou simplesmente sorte.

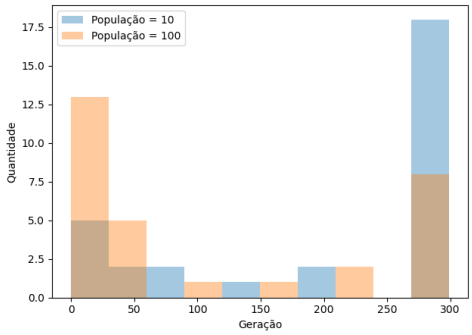


Figura 2. Histograma da geração de convergência para teste de variação do tamanho da população

*2) Efeito da probabilidade de crossover:* Para estudar o efeito de crossover na solução do algoritmo, foram avaliados os valores de probabilidade 0.3 e 0.9. A mutação foi desativada para as duas instâncias. Os resultados para 30 execuções de cada instância podem ser vistos na Figura 3. É possível ver também que o crossover é importante para a convergência do algoritmo, uma vez que todas as execuções para a instância de probabilidade de 0.9 convergem, entretanto para a instância com probabilidade de 0.3, 14 não conseguem chegar a uma solução ótima com menos de 300 gerações.

*3) Efeito da probabilidade de mutação:* Para estudar o efeito da mutação na solução do algoritmo, foram avaliados os valores de probabilidade 0.001 e 0.2. Os resultados para 30 execuções de cada instância podem ser vistos na Figura 4. Note que a probabilidade de mutação é muito importante para a convergência do algoritmo, uma vez que para a instância com probabilidade de 0.001 somente 6 instâncias convergem, equanto para a instância com probabildiade 0.2 isso ocorre para 16 execuções.

*4) Mutação por Inversion Mutation:* Conforme visto anteriormente, diferentes probabilidades de mutação tem grande impacto no desempenho do algoritmo. Portanto, foi proposto

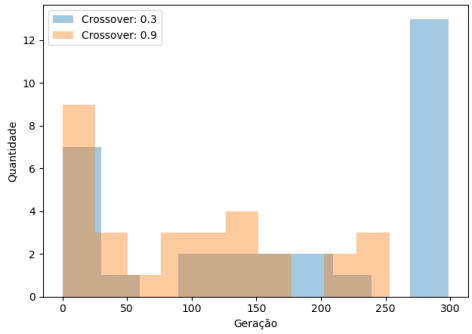


Figura 3. Histograma da geração de convergência para teste de variação da probabilidade de crossover

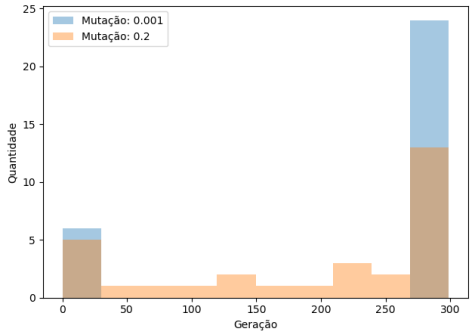


Figura 4. Histograma da geração de convergência para teste da variação da probabilidade de mutação

como melhoria o uso do operador *Inversion Mutation*, o qual funciona da seguinte forma:

1) Ecolher dois índices aleatoriamente
2) Inverter a ordem dos elementos entre eles

Os resultados da comparação com o operador de *Swap Mutation* são exibidos na Figura 5. Para 30 execuções do algoritmo, ele convergiu 24 vezes para *Swap Mutation* e 25 para *Inversion Mutation*. Desconsiderando as convergências com menos de 25 gerações, às quais estão bastante relacionadas com a obtenção de uma população inicial que já contém a solução ótima ou uma bem próxima, que é um processo aleatório, o novo operador proposto aparenta possuir melhor desempenho, uma vez que possui maior quantidade de execuções que convergiram abaixo de 175 gerações.

### *B. Função de Rastrigin*

Os gráficos referentes aos testes sobre essa função apresentam somente os dados das 50 gerações iniciais, uma vez que o algoritmo convergia muito rápido para um valor próximo de zero, dificultando a comparação dos resultados para todas as gerações disponíveis.

*1) Efeito do número de bits da representação:* Para avaliar o efeito de bits utilizados na representação, executou-se o algoritmo para os valores de 10 e 30 bits, mantendo os demais hiper-parâmetros fixos nos valores padrão. Os resultados para 30 execuções de cada instância podem ser vistos na Figura 6.

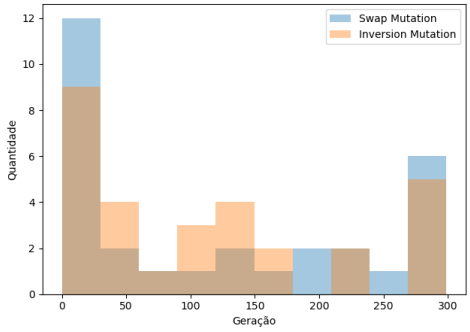


Figura 5. Histograma da geração de convergência para operador de *Inversion Mutation* em relação ao *Swap Mutation*

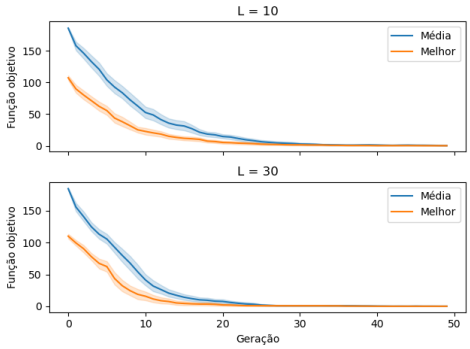


Figura 6. Perfil de convergência das execuções para a variação do número de bits

Para 10 bits, o função objetivo em média converge para 0.05, enquanto para 30 bits este valor é de aproximadamente $1.42\text{x}10^{-14}$. Logo, é possível concluir que o número de bits afeta bastante a melhor solução que o algoritmo consegue obter.

*2) Efeito do tamanho da população:* Para avaliar o efeito do tamanho da população, executou-se o algoritmo para populações de 10 e 100 indivíduos, utilizando os demais parâmetro da instância padrão. Os resultados para 30 execuções de cada instância podem ser vistos na Figura 7. É interessante notar que para os dois tamanhos de população, a convergência para um valor próximo de zero é bem rápida, entretanto para uma população maior a função objetivo apresenta um decréscimo mais estável.

*3) Efeito da probabilidade de crossover:* Para estudar o efeito de crossover na solução do algoritmo, foram avaliadas duas instâncias com intervalos $[0.6, 0.8]$ e $[0.8, 1.0]$, dado que está sendo utilizada a abordagem por adaptação linear dos parâmetros a cada geração. A probabilidade de mutação foi desativada para os testes. Os resultados para 30 execuções de cada instância podem ser vistos na Figura 8. É possível notar que os resultados são bem similares para as duas instâncias avaliadas.

*4) Efeito da probabilidade de mutação:* Para estudar o efeito de crossover na solução do algoritmo, foram avaliadas duas instâncias com os intervalos $[0.6, 0.8]$ e $[0.8, 1.0]$, dado

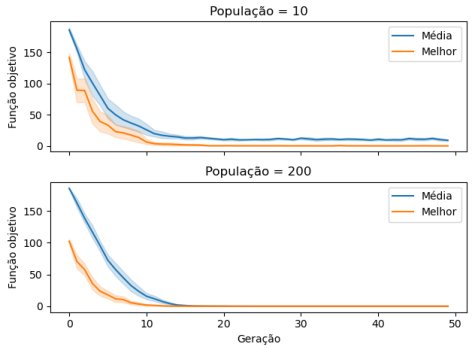


Figura 7. Perfil de convergência das execuções para a variação do tamanho da população

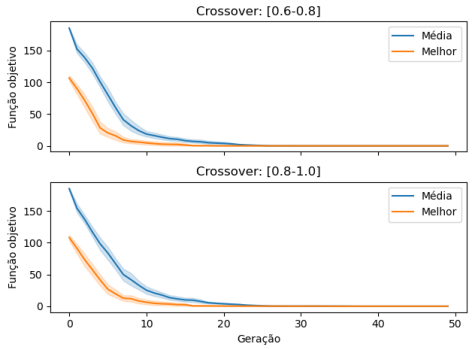


Figura 8. Perfil de convergência das execuções para variação das faixas de probabilidade de crossover

que está sendo utilizada a abordagem por adaptação linear dos parâmetros a cada geração. A probabilidade de crossover foi desativada para os testes. Os resultados para 30 execuções de cada instância podem ser vistos na Figura 9. É possível notar que os resultados são bem similares para as duas instâncias avaliadas, não havendo como diferenciar o desempenho entre elas.

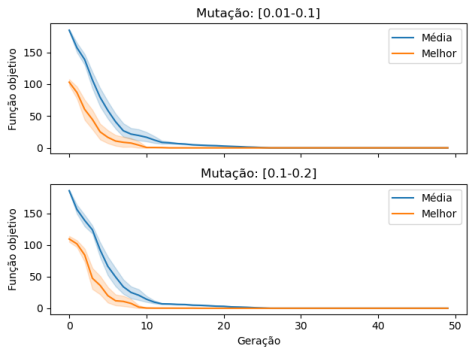


Figura 9. Perfil de convergência das execuções para variação das faixas de probabilidade de mutação

*5) Aplicação de escala linear ao fitness:* O objetivo desta possível melhoria é fazer com que a função de fitness represente melhor a função objetivo próximo de seu mínimo global, uma vez que a função de escala utilizada não consegue diferenciar muito bem valores próximos de zero. Utilizando o fator de escala $K = 1.5$, os resultados para 30 execuções das instâncias com e sem escala linear podem ser vistos na Figura 10.

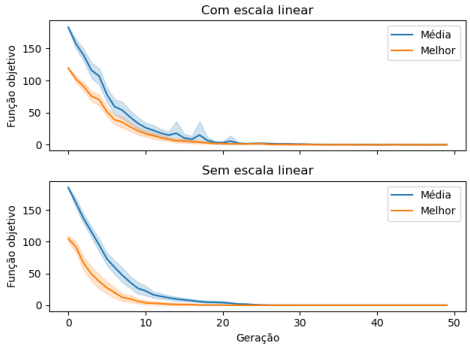


Figura 10. Resultados do uso de escala linear

É interessante notar que em média, o algoritmo possui uma convergência mais rápida e estável quando não é usada escala linear. Um resultado indesejado durante a execução, porém esperado, foi que alguns valores abaixo de zero eram obtidos após a aplicação da escala linear em algumas gerações. Isso não afeta o funcionamento do algoritmo, porém poderiam causar resultados indesejados na geração dos gráficos.

## IV. Conclusão

Neste trabalho foi feito o estudo da aplicação de algoritmos genéticos para o problema das N-Rainhas e minimização da função de reastrigin. Foi possível implementar, executar e comparar diversas instâncias de algoritmos genéticos na solução dos dois problemas, de forma a entender como os principais hiper-parâmetros afetam o desempenho do otimizador e os desafios na sua implementação.

Uma grande dificuldade enfrentada durante o trabalho foi como desenhar os experimentos a serem realizados para se avaliar deferentes configurações. É necessário tomar muito cuidado pois pequenos ajustes em alguma cofiguração e inconsistências na definição de hiperparâmetros podem gerar resultados inespreados, necessitando que eles sejam executados novamente. De fato, um ponto de melhora pessoal é estudar mais a literatura de computação evolucionária e entender melhor a metodologia geralmente utilizada pelos autores. Além disso, o uso de controle e ajuste automático de hiper-parâmetros se mostra uma opção muito melhor para ser utilizada nos próximos trabalhos, já que não foram muito exploradas neste primeiro.

## Referências

[1] Eiben, A.E. and Smith, J.E., 2015. Introduction to evolutionary computing. springer.

[2] Bell, J. and Stevens, B., 2009. A survey of known results and research areas for n-queens. Discrete Mathematics, 309(1), pp.1-31.

[3] Hoffmeister, F. and Bäck, T., 1990, October. Genetic algorithms and evolution strategies: Similarities and differences. In International Conference on Parallel Problem Solving from Nature (pp. 455-469). Springer, Berlin, Heidelberg.