UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA ELÉTRICA

GUSTAVO ALVES PACHECO

EVOLUÇÃO DIFERENCIAL

Uberlândia-MG

2019

GUSTAVO ALVES PACHECO

EVOLUÇÃO DIFERENCIAL

Trabalho apresentado ao curso de Engenharia de Computação da Universidade Federal de Uberlândia, como requisito parcial de avaliação da disciplina de Algoritmos Genéticos.

Prof.: Keiji Yamanaka

Uberlândia-MG

2019

SUMÁRIO

1. INTRODUÇÃO

Até o momento, foram utilizados algoritmos genéticos para otimização de funções. Esta estratégia é excelente para busca em conjuntos globais. Entretanto, o algoritmo genético falha em alguns pontos, principalmente na otimização de funções custo não-lineares e não diferenciáveis.

Uma alternativa para o algoritmo genético utilizado até agora, mas que não foge da estratégia de algoritmo evolutivo, é o da evolução diferencial. Um algoritmo mais simples e conciso, que apresenta resultados bem satisfatórios. Nessa abordagem, os cromossomos são tratados como vetores n-dimensionais, dependendo do número de variáveis da função de custo.

Assim como nos AG's, a população inicial é aleatoriamente escolhida (ou em alguns casos, distribuída uniformemente pelo espaço de busca). A partir dela, cada indivíduo é colocado em xeque, para compor a nova geração. Esse indivíduo, chamado de *target vector*, é comparado com um outro indivíduo, gerado através da combinação de três vetores aleatórios, e do próprio *target*, gerando o *trial vector*. O que possuir maior fitness, dentre os dois, é incorporado na nova geração. O processo de geração do *trial vector* é explicado com mais detalhes na seção 3.

Para aplicação da evolução diferencial, deve-se minimizar a função de Rosenbrock, para n variáveis.

$$f(x_1, x_2, \dots, x_n, x_{n+1}) = \sum_{i=1}^{n} [(a - x_i)^2 + b(x_{i+1} - {x_i}^2)^2]$$

Sendo a e b constantes, de valor 1 e 100 (geralmente), respectivamente.

2. OBJETIVOS

- Desenvolver um algoritmo de Evolução Diferencial
- Otimizar a função de Rosenbrock, utilizando Evolução Diferencial
- Implementar os novos operadores de Crossover, Mutação e Seleção
- Adequar a interface gráfica utilizada no projeto de 'Maximização de uma função de duas variáveis', de forma a abranger os novos operadores, além de oferecer suporte à função de Rosenbrock.
- Verificar impacto dos parâmetros da Evolução Diferencial, no processo de *tuning*

3. MATERIAIS E MÉTODOS

A linguagem utilizada foi o Racket, uma linguagem de uso geral, baseada em Scheme e Lisp. Como interface de desenvolvimento, foi utilizado o DrRacket.

Para confecção da interface gráfica, utilizou-se a linguagem *racket/gui*, um recurso nativo da própria linguagem Racket. O processo de plotagem foi realizado através das funções do pacote *plot*.

Como métodos, utilizou-se uma abordagem *top-down*, com auxílio de um framework SCRUM para desenvolvimento ágil. Em algumas situações, uma visão *bottom-up* foi adicionada ao projeto, na parte de incremento da própria linguagem, ao criar novos operadores e funcionalidades. O código produzido segue majoritariamente uma abordagem funcional, mas possui elementos procedurais, para facilitar a leitura de algumas funções.

Durante o desenvolvimento, a documentação de funções se mostrou fundamental. Principalmente, para que algumas estruturas de dados fossem melhor especificadas, e a execução do código fosse correta.

A geração do *trial vector* se dá da seguinte forma:

Primeiro, três vetores, x_1, x_2 *e* x_3, diferentes do *target* são escolhidos aleatoriamente da população atual. O *donor vector*, como é chamado, é criado a partir da relação:

$$v = x_1 + F(x_3 - x_2)$$

Sendo F uma constante real entre 0 e 2, definida pelo usuário. Esse processo é conhecido como mutação.

Em sequência, o crossover é realizado. Para isso, o *trial vector* é criado, escolhendo aleatoriamente posições ou do *target vector* ou do *donor vector*, segundo a relação:

$$u_i = \begin{cases} v_i, & se\ r_i \leq CR\ ou\ i = I \\ x_i, & se\ r_i > CR\ e\ i \neq I \end{cases}$$

Sendo x o *target vector*, r_i um número real aleatório entre 0 e 1, gerado para cada posição do vetor, CR uma constante entre 0 e 1, definida pelo usuário, i o número da posição

atual e I um número aleatório gerado apenas uma vez, entre 1 e D, sendo D o número de dimensões do vetor. Isso é feito para que pelo menos uma componente de v esteja em u.

Ao final, deve-se verificar se o mesmo se encontra dentro dos intervalos especificados.

4. RESULTADOS E DISCUSSÕES

Inicialmente, adaptou-se a interface antiga para que passasse a trabalhar com algoritmos de evolução diferencial. Os campos ajustáveis podem ser vistos na Figura 1, abaixo.

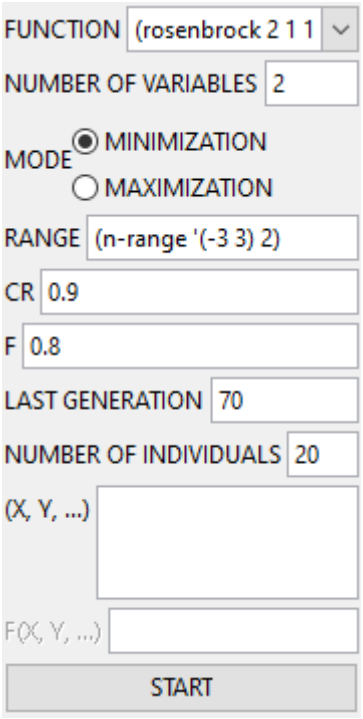

Figura 1: Configurações na interface gráfica

Em seguida, foi implementada a função de Rosenbrock. Um certo tipo de macro que gera sua expansão em uma função de n variáveis, assim como feito com a função de Rastrigin, anteriormente. Para duas variáveis, utilizando a=100 e b=100, no intervalo entre -3 e 3, para todas as dimensões, a função de Rosenbrock é da forma (Figura 2):

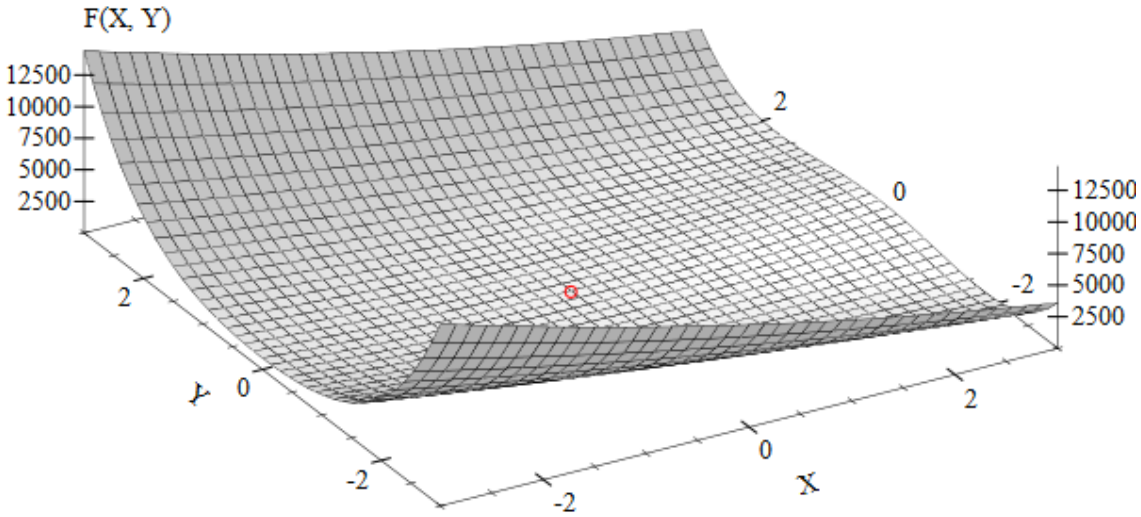

Figura 2: Função de Rosenbrock para duas variáveis

A partir daí, o algoritmo de evolução diferencial foi implementado. Comparado ao algoritmo genético, o desenvolvimento foi extremamente rápido, e o código ficou consideravelmente menor. Várias funções puderam ser reutilizadas, em especial as de retorno, que plotavam os gráficos de performance e da função.

Finalmente, testes foram feitos, para minimização da função de Rosenbrock. Além dela, outras funções foram otimizadas, utilizando o mesmo algoritmo, já que a evolução diferencial implementada é uma função de alta ordem, recebendo outras funções como argumento. Os testes realizados estão dispostos nas imagens abaixo.

Para duas variáveis, o menor valor encontrado foi de 0.99010, quando x=-0.00990 e y=0.00000. Os melhores resultados foram obtidos utilizando CR=0.9 e F=0.8.

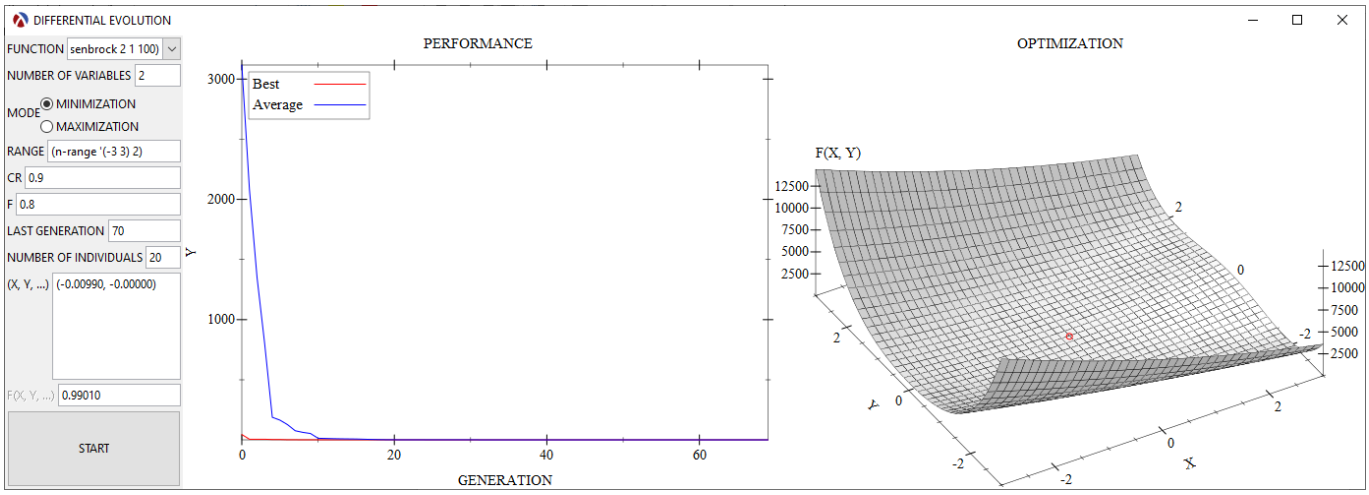

Figura 3: Minimização da função de Rosenbrock para duas variáveis

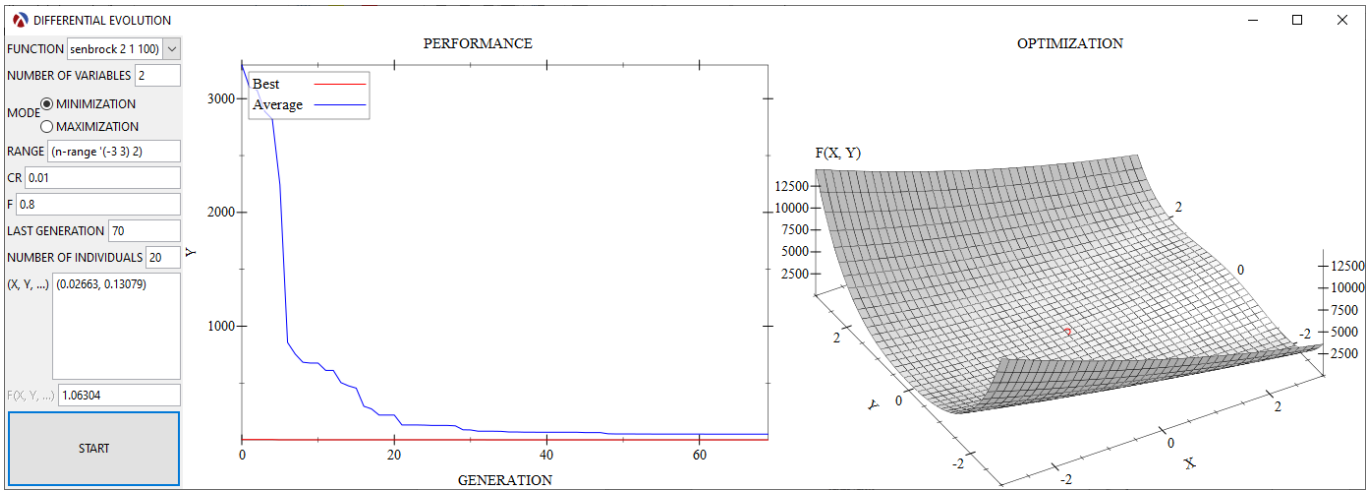

Figura 4: Diminuindo o valor de CR, percebe-se uma convergência mais lenta

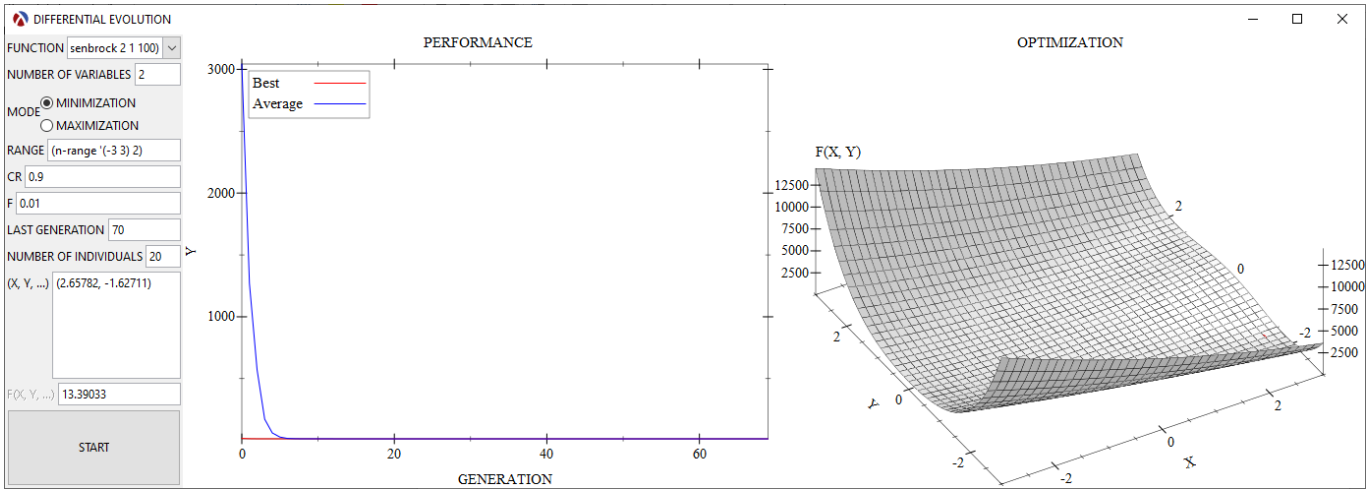

Figura 5: Diminuindo o valor de F, percebe-se uma convergência para mínimos locais equivocados

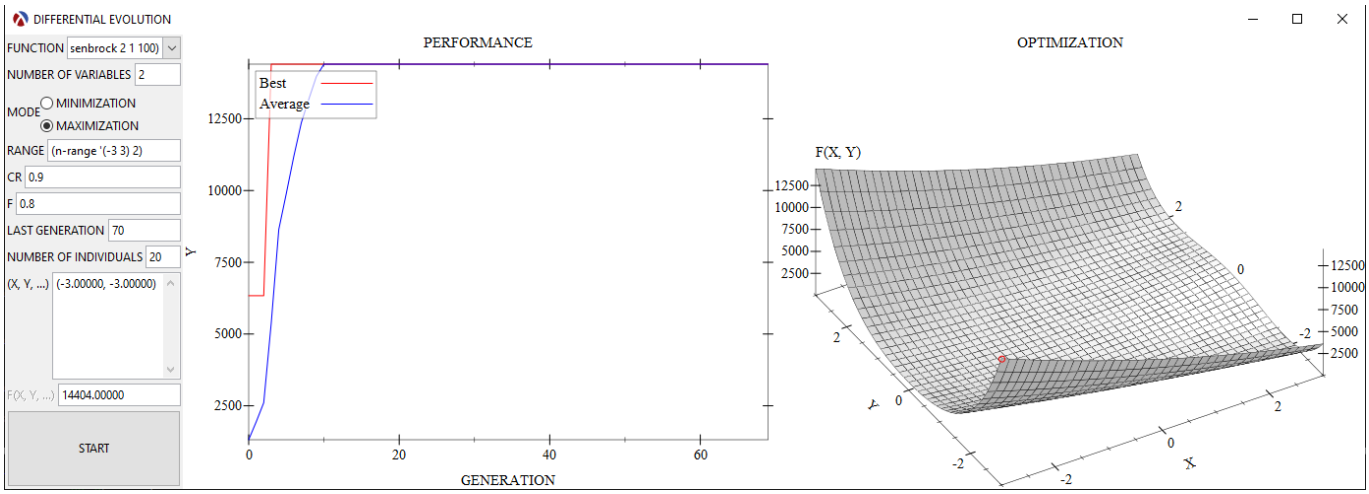

Figura 6: Maximização da função de Rosenbrock, no intervalo -3 e 3

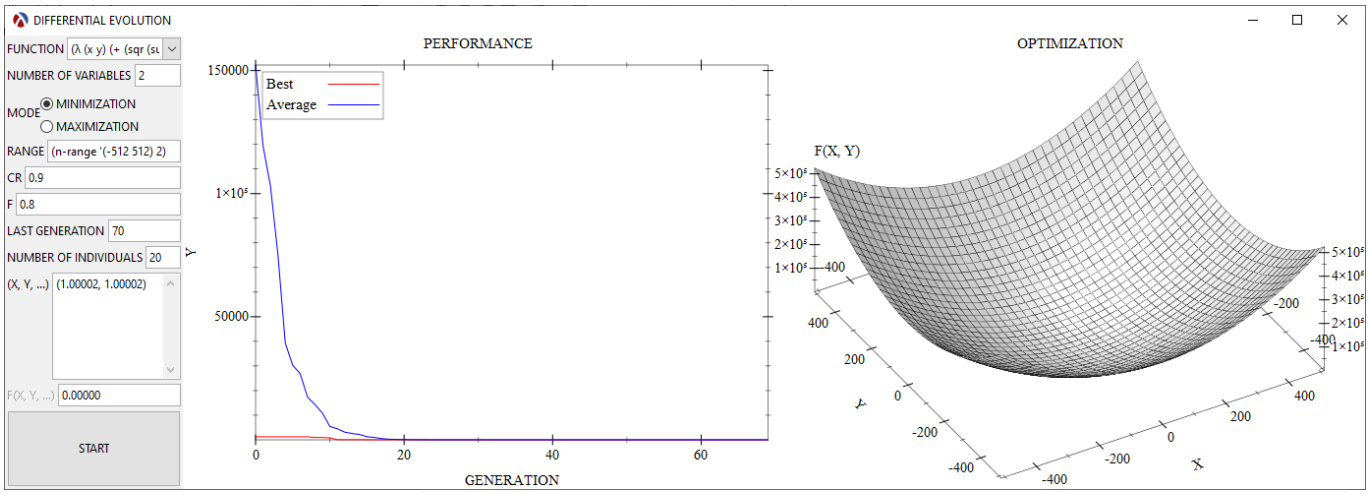

Figura 7: Minimização da função $f\ (x, y) = \ (x-1)^2 + (y-1)^2$

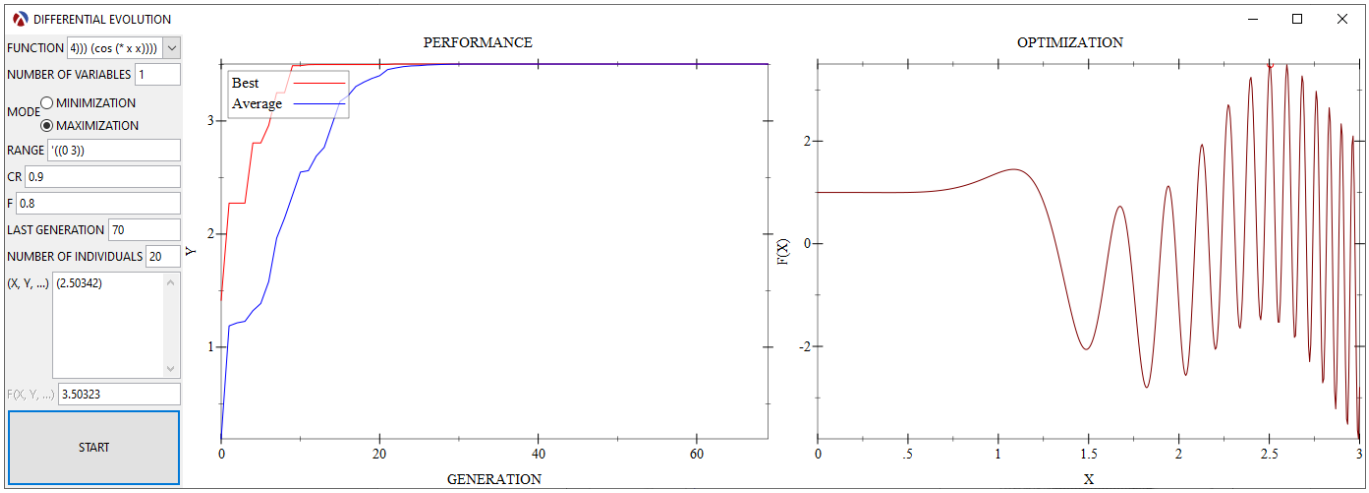

Figura 8: Maximização da função $f\ (x) = \mathrm{x} * \sin x^4 + \cos x^2$

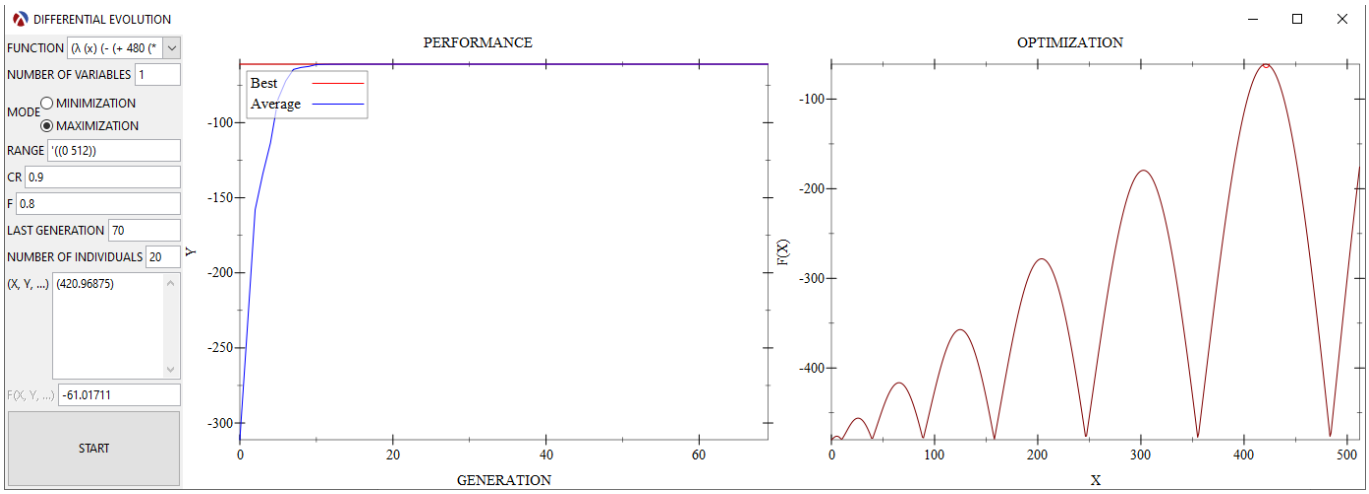

Figura 9: Maximização da função $f\ (x) = \ -480 + \left|x * \sin\sqrt{|x|}\right|$

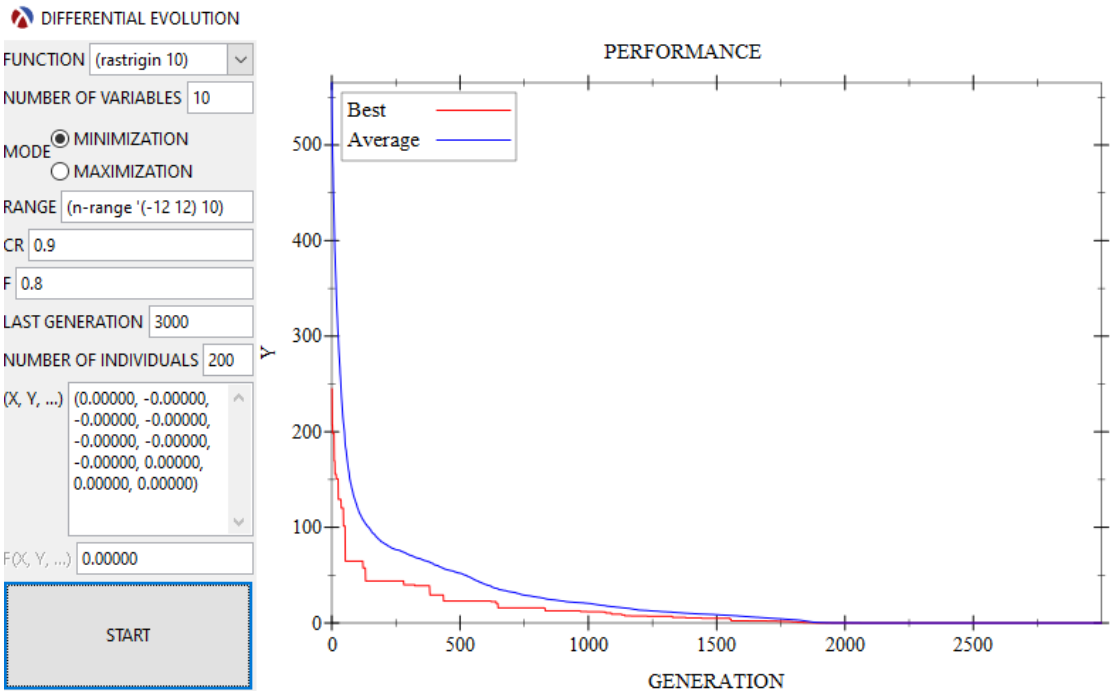

Figura 10: Minimização da função de Rastrigin para 10 variáveis

5. CONCLUSÃO

O algoritmo de Evolução Diferencial se mostrou extremamente eficiente, aplicado a diversos casos já trabalhados até o momento. A resposta foi confiável, precisa e exata. A execução do algoritmo foi rápida, chegando a superar em velocidade os AG's implementados.

Além disso, o tempo de implementação foi reduzido, e o código ficou menor que os dos AG's. O ED também foi capaz de encontrar o mínimo da função de Rastrigin para 10 variáveis, em 3000 gerações, com 200 indivíduos, com precisão e exatidão melhores que do AG, em tempo menor.

Com certeza, a Evolução Diferencial se mostra um forte concorrente aos AG's, principalmente em casos onde os algoritmos genéticos apresentam alguma fraqueza.

6. REFERÊNCIAS BIBLIOGRÁFICAS

Al-Gharaibeh, J., Qawagneh, Z., & Al-Zahawi, H. (29 de Outubro de 2015). Genetic Algorithms with Heuristic. *Research Gate*.

Boccato, L., Attux, R. R., & Von Zuben, F. J. (2009). *Evolução Diferencial: Introdução e Conceitos Básicos.* DCA - Unicamp.

Mallawaarachchi, V. (7 de Julho de 2017). *Introduction to Genetic Algorithms*. Fonte: Towards Data Science: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

Montesanti, J. d. (s.d.). *Seleção natural*. Fonte: InfoEscola: https://www.infoescola.com/evolucao/selecao-natural/

Tomassini, M. (s.d.). A Survey of Genetic Algorithms. *Annual Reviews of Computational Physics, Volume III*.

Yamanaka, P. K. (Agosto de 2017). *ALGORITMOS GENÉTICOS: Fundamentos e Aplicações*.

Zini, É. d., Neto, A. B., & Garbelini, E. (18 de Novembro de 2014). ALGORITMO MULTIOBJETIVO PARA OTIMIZAÇÃO DE PROBLEMAS RESTRITOS APLICADOS A INDÚSTRIA. *Congresso Nacional de Matemática Aplicada à Indústria*.