UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA ELÉTRICA

GUSTAVO ALVES PACHECO

MAXIMIZAÇÃO DE UMA FUNÇÃO DE DUAS VARIÁVEIS

Uberlândia-MG

2019

GUSTAVO ALVES PACHECO

MAXIMIZAÇÃO DE UMA FUNÇÃO DE DUAS VARIÁVEIS

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

Anteriormente, havia sido implementado um algoritmo genético simples, para maximização de funções pré-determinadas. Tal algoritmo utilizava tanto o método da roleta quanto do torneio para seleção dos indivíduos da nova espécie. Também havia sido implementado o sistema de elitismo, que copiava os melhores indivíduos de uma geração para a próxima.

Tal versão do algoritmo suportava funções de uma variável real, otimizando a função no plano R^2. Entretanto, algumas aplicações possuem mais de uma entrada de dados, ou seja, necessitam de funções com mais de uma variável.

Diante desta problemática, novos desafios aparecem, como a nova técnica de crossover em dois pontos, e a mutação de todos os bits de cada indivíduo. O primeiro, sobre o crossover, pressupõe uma abordagem de representação de dados em n bits. Sendo que desses n bits, metade será destinada para representação da coordenada x do indivíduo, e a outra metade, a coordenada y.

Esse crossover, de dois pontos, visa gerar maior diversidade dos filhos em relação aos pais, já que o cromossomo original é divido em dois pontos (três partes), e a parte central dos dois pais é trocada. O processo de mutação de bits individuais é obtido percorrendo a *string* binária, e gerando uma probabilidade de mutação para cada posição. Caso essa probabilidade seja menor que a de entrada no algoritmo, aquele bit em específico é modificado.

Além disso, para que os novos requisitos sejam testados, a função a ser maximizada passa a ser a seguinte:

$$f(x, y) = 21{,}5 + xsen(4\pi x) + ysen(20\pi y)$$

2. OBJETIVOS

- Modificar o algoritmo genético implementado até o momento para que suporte maximizações de funções de duas variáveis
- Implementar crossover de dois pontos
- Implementar mutação bit a bit
- Criar método para plotagem dos gráficos em três dimensões
- Aprimorar o conhecimento obtido até o momento sobre algoritmos genéticos

Como objetivos extras, tem-se:

- Expandir o algoritmo para que se torne uma função de alta ordem, possibilitando a entrada de funções a serem otimizadas
- Otimizar o processo de plotagem da função em 3d e em 2d
- Permitir que a interface suporte todas as configurações possíveis, incluindo entrada de funções, modo de operação (2d/3d) e ajustes gerais

3. MATERIAIS E MÉTODOS

Os métodos utilizados foram semelhantes aos usados anteriormente, embora os materiais tenham sido diferentes.

Para que fosse possível a plotagem em 3d de forma eficiente, o código precisou ser reescrito em uma linguagem que permitisse tal tecnologia. A linguagem escolhida foi o Racket, que faz parte da família Lisp, não se distanciando da linguagem usada até o momento (Common Lisp). O *port* foi de certa forma trabalhoso, mas após a reescrita, o código ficou certamente mais rápido, eficiente e livre de erros. 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. 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.

Optou-se por modificar a forma de estruturamento de dados, para oferecer suporte às funções de duas variáveis. Tal decisão será abordada melhor na próxima seção.

4. RESULTADOS E DISCUSSÕES

Primeiramente, foi necessário reescrever completamente o código original, de Common Lisp para Racket. Durante essa etapa, alguns erros foram corrigidos, e várias funções foram otimizadas, optando por utilizar recursões de cauda, as quais são transformadas em iterações pelo próprio compilador da linguagem.

Além disso, o retorno da função principal mudou. Antes, a plotagem ocorria simultaneamente aos cálculos. Isso representa um erro grave em performance, já que o processo era repetidamente interrompido para a exibição gráfica. Na nova versão, todo o algoritmo é executado antes que qualquer gráfico seja gerado, e o retorno da função passa a ser uma lista contendo todas as n-gerações de populações. Tendo em mãos todos os indivíduos do processo, o processo de plotagem pode acontecer em sequência, junto com a exibição dos valores encontrados.

Estes dois elementos contribuíram bastante para a melhoria de eficiência do algoritmo, permitindo cálculos de maiores populações, com maiores resoluções, em menos tempo.

Depois dessa reescrita, vários testes foram feitos para garantir que o funcionamento ainda estava correto, para as funções já testadas. Nesse momento, optou-se por expandir a função principal, tornando-a de alta ordem, permitindo que outras funções fossem passadas como argumentos para a primeira. Vale lembrar que este não é um recurso trivial e comum em linguagens de programação, sendo que apenas linguagens com certa familiaridade com (ou com elementos importados do) Lisp o possuem.

A partir daí, foi feita a adaptação do código para duas variáveis, ainda permitindo que as mesmas funções pudessem operar com uma variável. Houveram problemas a respeito do intervalo de cada variável. Devido à representação binária, deve-se adotar um procedimento próprio para números com sinal. Geralmente, a estratégia utilizada é o complemento de dois, que usa o bit mais significativo para indicar o sinal do número. Entretanto, os valores máximos que passam a ser representados com a mesma resolução de bits diminuem pela metade.

Para simplificar esse processo, visto que a própria função de conversão decimal para binário fora implementada anteriormente, optou-se por deslocar a faixa de valores das variáveis por um valor constante, para que ambos os intervalos sejam positivos (após ser feito o cálculo do fitness, visando garantir a integridade da aptidão). A conversão em binário ocorre e, em sequência, as operações genéticas acontecem. Ao término das mesmas, as faixas de valores são novamente deslocadas para o original, novamente se preocupando com a confiabilidade dos dados na próxima geração.

O crossover de dois pontos foi implementado, mas decidiu-se não representar as duas coordenadas do indivíduo em uma mesma *string* de n caracteres. Ao invés, a *string* passa a ter um tamanho de 2n, sendo que cada dimensão é representada em n bits. Assim, a precisão se mantém alta, com um custo não tão grande de processamento extra.

Em relação ao processo de plotagem, inicialmente foi utilizada a estratégia de *refresh* a cada geração. Entretanto, havia *flickering* constante na tela e, em três dimensões, era impossível observar os pontos se movimentando pelo gráfico. Logo, uma nova estratégia foi desenvolvida, que partia da utilização de um *canvas* com background invisível, sendo que a cada geração, uma imagem da função pré-renderizada era colocada como fundo desse *canvas*. Em cima disso, os pontos eram animados. Precisou-se pensar

em uma maneira de fixar o eixo z do gráfico, pegando o melhor e o pior indivíduo de todos os encontrados.

Com o *software* em mãos, vários testes foram realizados. Alguns deles estão ilustrados abaixo. Os parâmetros utilizados em cada teste podem ser observados na própria interface, à esquerda.

Figura 1: Interface refeita, exibida com os valores padrões do aplicativo

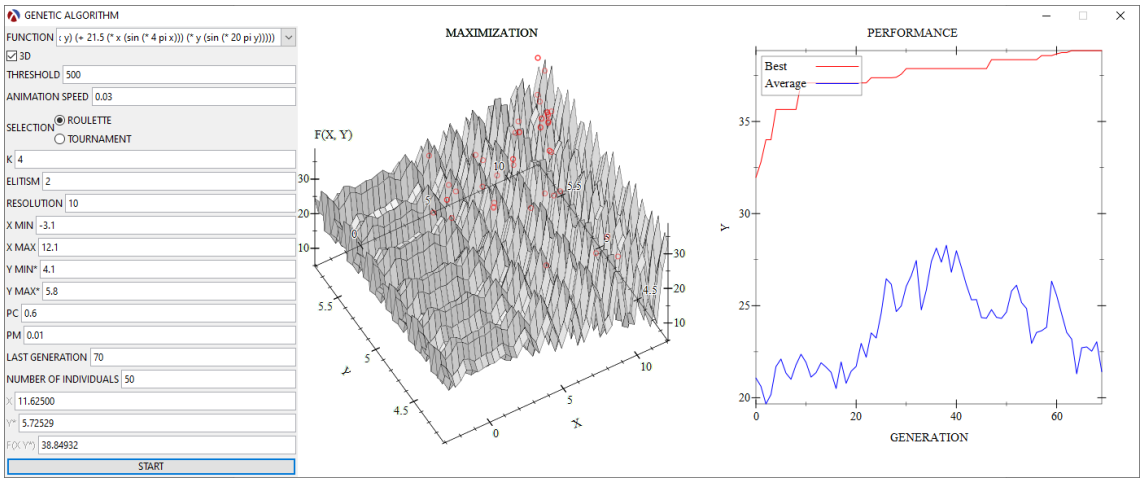

Figura 2: Maximização da função desejada, utilizando roleta

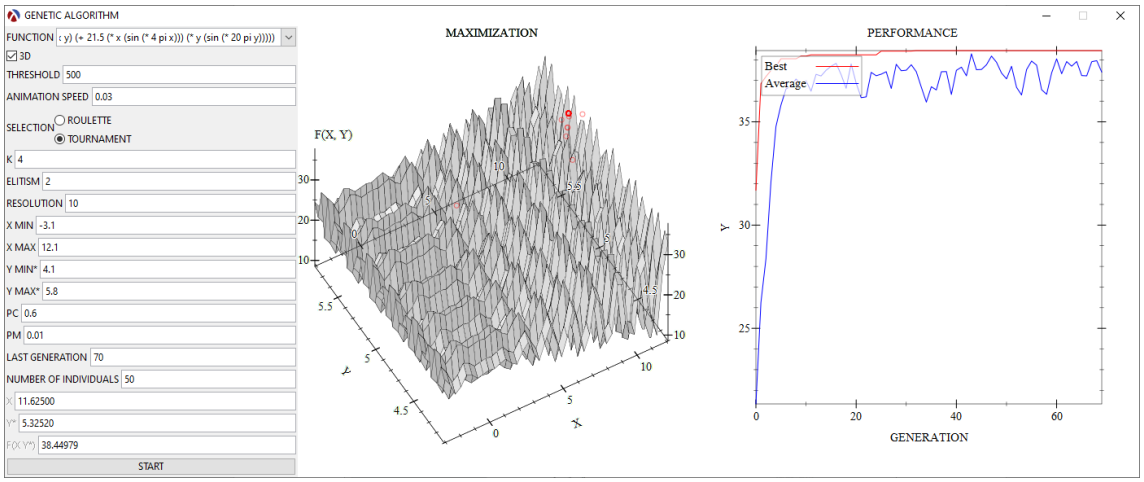

Figura 3: Maximização da função desejada, utilizando torneio, nota-se a melhoria na média geral da população e a rápida convergência dos valores

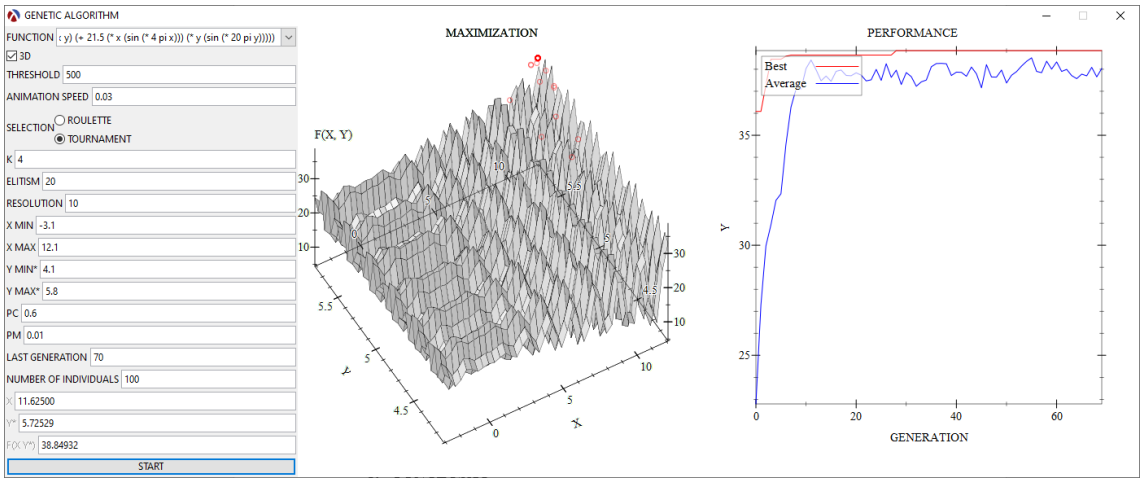

Figura 4: Melhor situação encontrada, utilizando torneio, um grande número de indivíduos e elitismo elevado

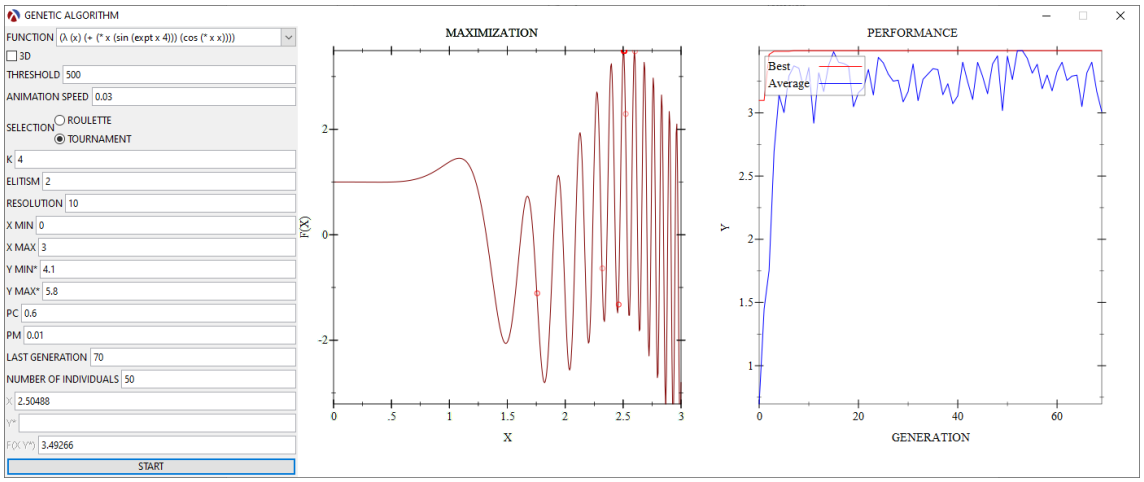

Figura 5: Otimização de funções de uma variável, utilizando a mesma interface

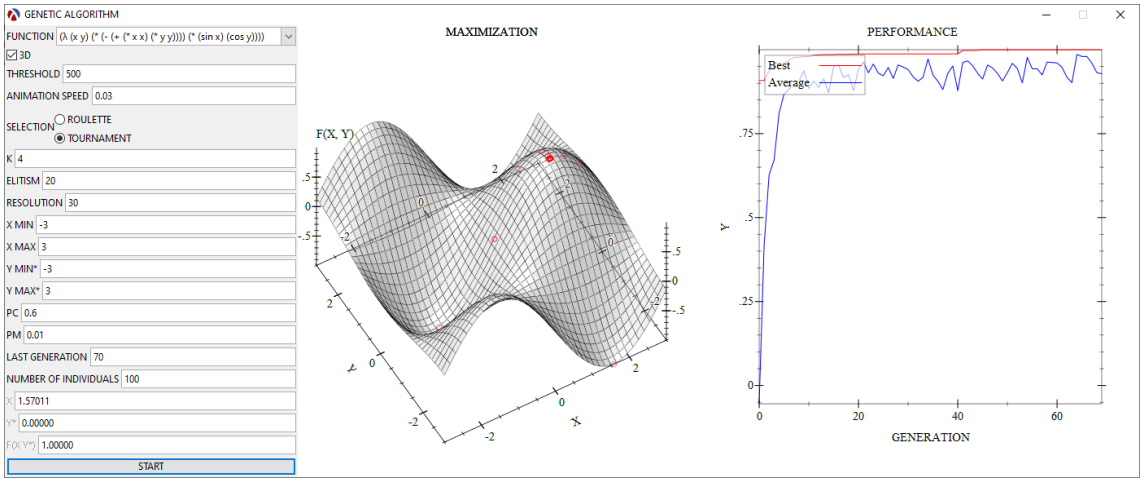

Figura 6: Otimização de funções de duas variáveis. Nota-se que a função foi passada como parâmetro para o algoritmo

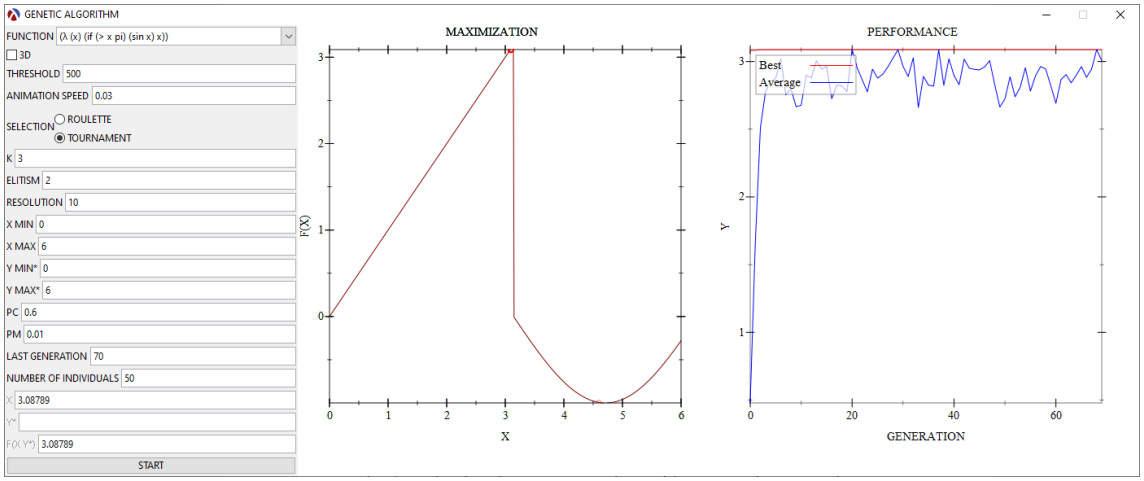

Figura 7: Função otimizada contendo operadores condicionais

5. CONCLUSÃO

Como observado pelos testes, existe uma certa problemática no que diz respeito ao ajuste dos parâmetros para que a melhor situação seja encontrada. Certamente, um dos sucessos para a execução correta do algoritmo é essa configuração.

Embora apresente uma complexidade elevada (como observado nos gráficos de superfície), o algoritmo foi preciso e exato na determinação do maior valor, em determinadas situações de ajustes.

Novamente, surpreende-se com o poder dos algoritmos genéticos, com as técnicas copiadas e inspiradas da natureza, espelhando os processos que ocorrem naturalmente. É interessante ver a gama de aplicações que começam a surgir, que possam utilizar o maquinário fornecido pelos AG's e que, certamente, resolverão problemas complexos e, aparentemente impossíveis, no mundo tecnológico que está por vir.

6. REFERÊNCIAS BIBLIOGRÁFICAS

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*.