UNIVERSIDADE FEDERAL DE ALAGOAS - UFAL
CAMPUS A. C. SIMÕES
CIÊNCIA DA COMPUTAÇÃO

LEANDRO MARTINS DE FREITAS

EASYGA: ALGORITMOS GENÉTICOS PARA INICIANTES

MACEIÓ
2019

Leandro Martins de Freitas

EasyGA: algoritmos genéticos para iniciantes

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal de Alagoas - UFAL, Campus A. C. Simões.

Orientador: Prof. Dr. Erick de Andrade Barboza

Maceió
2019

Leandro Martins de Freitas

EasyGA: algoritmos genéticos para iniciantes

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal de Alagoas - UFAL, Campus A. C. Simões.

Data de Aprovação: 06/09/2019

Banca Examinadora

Prof. Dr. Erick de Andrade Barboza
Universidade Federal de Alagoas
Campus A. C. Simões
Orientador

Prof. Dr. Rian Gabriel Santos Pinheiro
Universidade Federal de Alagoas
Campus A. C. Simões
Examinador

Prof. Dr. Evandro de Barros Costa
Universidade Federal de Alagoas
Campus A. C. Simões
Examinador

RESUMO

Inteligência artificial (IA) é uma área da ciência da computação em constante evolução e cuja aplicação se estende a diversas situações. O uso de técnicas de IA não deve ser limitado a profissionais de computação, pois a sua utilidade para solução de problemas, quer sejam corriqueiros ou de difícil compreensão, é inegável. Um problema comum no qual tais técnicas podem ser utilizadas é a otimização de funções matemáticas. Dentre os algoritmos de otimização possíveis, Algoritmos Genéticos se fazem interessantes pela sua simplicidade e eficiência, principalmente para problemas combinatoriais com muitas variáveis. É possível encontrar diversas ferramentas cujo propósito é automatizar a definição e utilização de Algoritmos Genéticos. Geralmente, tais ferramentas permitem a escolha dos parâmetros e definição da função objetivo facilitando o desenvolvimento do algoritmo para usuários que já possuem conhecimento e habilidade nessa técnica. A falta de ferramentas que dão ao usuário uma introdução ao tema de Algoritmos Genéticos pode ser um desincentivo ao seu aprendizado. Neste trabalho é apresentada a aplicação gráfica EasyGA, cujo objetivo é permitir que usuários não familiarizados com Algoritmos Genéticos possam compreender seu funcionamento, realizar simulações e entender as implicações das suas escolhas sem que sejam necessários conhecimentos prévios de programação ou IA. Na ferramenta proposta é possível escolher parâmetros tais como: tamanho da população, número de gerações e método de seleção; e definir múltiplas configurações com parâmetros diferentes, dando ao usuário a capacidade de comparar e analisar as estratégias definidas. A função objetivo, domínio de variáveis e demais aspectos do algoritmo são definidos na própria ferramenta, sem que nenhuma linha de código seja necessária. Registros de texto e gráficos contendo o histórico dos melhores indivíduos são disponibilizados ao final das simulações e podem ser salvos pelo usuário.

Palavras-chave: Inteligência Artificial. Otimização. Interface Gráfica.

ABSTRACT

Artificial intelligence (AI) is a constantly evolving area of computer science whose application extends to various situations. The use of AI techniques should not be limited to computer professionals, considering their usefulness in solving problems, whether ordinary or complex, is undeniable. A common problem in which such techniques can be used is the optimization of mathematical functions. Among the possible optimization algorithms, Genetic Algorithms are interesting for their simplicity and efficiency, especially for combinatorial problems with many variables. Several tools can be found whose purpose is to automate the definition and use of Genetic Algorithms. Usually, these tools allow the choice of parameters and definition of the objective function facilitating the development of the algorithm for users who already have knowledge and skill in this technique. The lack of tools that give the user an introduction to the theme of Genetic Algorithms can be a disincentive to their learning. This work presents the EasyGA graphical application, whose objective is to allow users unfamiliar with Genetic Algorithms to understand its operation, perform simulations and understand the implications of their choices without prior programming or AI knowledge. In the proposed tool it is possible to choose parameters such as: population size, number of generations and selection method; and define multiple configurations with different parameters, giving the user the ability to compare and analyze the defined strategies. The objective function, variable domain and other aspects of the algorithm are defined in the tool itself, without any line of code being required. Text records and charts containing the history of the best individuals are made available at the end of the simulations and can be saved by the user.

Keywords: Artificial Intelligence. Optimization. Graphical Interface.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Desde os primórdios da humanidade, temos a necessidade de facilitar a realização de tarefas, seja com a criação de novas ferramentas, métodos de trabalho ou mecanismos. Logo após a Segunda Guerra Mundial a humanidade engatinhava em direção ao desenvolvimento de uma ferramenta extremamente poderosa: o computador moderno. Inicialmente os computadores apenas realizavam operações de cálculo, o que já era de grande importância para grandes empresas e universidades, mas havia muito mais potencial naquelas máquinas.

Em 1946, no Laboratório Nacional de Física da Inglaterra, Alan Turing trabalhou no desenvolvimento da Automatic Computing Engine (ACE), uma das primeiras tentativas de construção de um computador digital. Turing, alguns anos depois, através do seu artigo Computing Machinery and Intelligence (TURING, 1950), começou a relacionar processos computacionais e a mente humana, ponderando se seria possível a construção de uma máquina que imitasse o comportamento humano.

Trabalhos como os de Alan Turing e diversos outros matemáticos e cientistas deram início ao estudo da Inteligência Artificial. Segundo Prateek Joshi (JOSHI, 2017), Inteligência Artificial é a área da computação responsável pelo estudo de teorias e técnicas que possibilitam o desenvolvimento de softwares dotados da capacidade de perceber o ambiente ao seu redor e reagir de forma semelhante a um ser humano.

Os pioneiros na área de Inteligência Artificial tomavam como base ideias estabelecidas na biologia, chegando a esquemas computacionais que simulavam o funcionamento do cérebro humano e a evolução biológica (MITCHELL, 1998). Dessas estratégias, a primeira transformou-se no estudo das Redes Neurais Artificiais e a segunda molda a Computação Evolucionária.

A Computação Evolucionária engloba vários tipos de algoritmos de otimização baseados na evolução biológica, dentre os quais estão os Algoritmos Genéticos - descritos rigorosamente por John Holland no artigo Adaptation in Natural and Artificial Systems (HOLLAND, 1975). Num Algoritmo Genético (AG), cada indivíduo de uma população representa uma possível solução para um problema e, através de operadores de seleção, cruzamento e mutação, esta população é modificada de modo que os indivíduos mais aptos (as soluções mais apropriadas) sobrevivam.

A otimização de uma função matemática consiste em determinar os valores extremos - mínimos e máximos - que a função assume em determinado intervalo. Dependendo dos valores que as variáveis da função podem assumir, a otimização é categorizada de duas formas:

otimização combinatória, caso as variáveis sejam discretas, e otimização numérica, caso as variáveis sejam contínuas. Algoritmos Genéticos são mais comumente utilizados na solução de problemas combinatórios, apesar de ser possível utilizá-los para otimização numérica.

Uma técnica de otimização possível é a programação linear (DANTZIG; THAPA, 2006), cujo resultado é exato, porém para problemas com espaços de busca extensos o custo computacional pode ser muito alto. Outras técnicas como hill climbing (SELMAN; GOMES, 2006) e simulated annealing (BERTSIMAS et al., 1993) são mais rápidas que a programação linear, porém são mais suscetíveis a ficar presas num ótimo local. Não há, ao se utilizar AGs, garantia de que a solução retornada será a melhor possível, porém a solução aproximada pode ser bastante satisfatória e os problemas de tempo e ótimos locais são minimizados.

1.1 JUSTIFICATIVA

Algoritmos Genéticos possuem diversas aplicações. Em (GLOBUS et al., 2006), eles foram utilizados para definir a forma de antenas utilizadas em missões espaciais, visando a otimização do ganho de transmissão. Em (FROWD et al., 2004), um conjunto de faces conhecidas evolui para gerar retratos falados mais fiéis às descrições fornecidas. Outro exemplo é apresentado em (GEIJTENBEEK et al., 2013), no qual simulações com criaturas bípedes 3D otimizam seu movimento, permitindo que aprendam a se locomover.

Há bastante espaço em qualquer área de conhecimento para o uso de AGs, porém a diversidade de detalhes e conceitos de um Algoritmo Genético pode desencorajar um estudo mais aprofundado por parte de programadores iniciantes ou pessoas leigas à computação. Algumas ferramentas, tais como DREAM (PAECHTER et al., 2002) e GAlib-ide (TEODORO et al., 2008), têm como objetivo fornecer uma plataforma intuitiva para definição de algoritmos genéticos. EasyGA - proposta deste trabalho - é uma ferramenta desse tipo, cujo diferencial é permitir que seu usuário defina simulações com diferentes parâmetros, possibilitando a análise e compreensão dessas escolhas.

1.2 OBJETIVO GERAL

Este trabalho tem como maior objetivo o desenvolvimento de uma aplicação gráfica que permita a definição, simulação e análise de problemas de otimização de funções matemáticas utilizando Algoritmos Genéticos sem que o usuário saiba, necessariamente, como implementar o algoritmo.

1.3 OBJETIVOS ESPECÍFICOS

- Facilitar e automatizar a utilização de Algoritmos Genéticos para otimização de funções matemáticas;
- Fornecer um ambiente amigável e eficiente para usuários com diferentes níveis de conhecimento sobre Algoritmos Genéticos;
- Permitir que o usuário defina os parâmetros do algoritmo e perceba as implicações de suas escolhas no resultado final.

1.4 ORGANIZAÇÃO DO DOCUMENTO

Fundamentos de Algoritmos Genéticos são abordados no Capítulo 2. O capítulo seguinte consiste de uma análise de trabalhos relacionados, apontando semelhanças e diferenças entre o trabalho proposto. No Capítulo 4 o desenvolvimento e funcionamento da ferramenta é relatado. E por fim, no Capítulo 5 são descritas as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Um Algoritmo Genético é um método de busca estocástico, geralmente utilizado em problemas de busca e otimização. A premissa básica de um AG é evoluir um conjunto de soluções de um problema, mantendo as características mais fortes - necessárias à sobrevivência - dos indivíduos, tal qual ocorre com organismos vivos no processo de evolução. Num AG, a "sobrevivência" e "reprodução" de um indivíduo dependem do quão bem ele se sai numa tarefa específica, ou seja, o quão apto ele é à otimização de uma função objetivo.

Além da otimização de funções matemáticas, AGs podem ser utilizados em uma infinidade de problemas. É importante lembrar que Algoritmos Genéticos, devido ao seu caráter aleatório, **não** garantem retornar a melhor solução, e sim soluções bastante próximas à solução ótima. Mesmo assim seu uso é bastante útil para problemas de otimização que não possuem algoritmos exatos ou cujo espaço de busca é muito grande (LINDEN, 2008).

As operações evolutivas num AG são realizadas sobre um conjunto de tamanho fixo de indivíduos (também chamados cromossomos) que representam possíveis soluções para o problema em questão. A esse conjunto damos o nome de população. Geralmente, cromossomos são representados como vetores, matrizes ou qualquer estrutura de dados cujas posições - chamadas de genes - armazenam suas características.

Um indivíduo consiste de um conjunto de dados estruturados que, quando aplicado à função objetivo, nos retorna sua aptidão. Consideraremos dois tipos de representação dos indivíduos: codificação binária e codificação real [1]. Na codificação binária cada variável da função objetivo é representada com um número fixo de bits (geralmente 32 para números de ponto flutuante), já na codificação real, cada gene do cromossomo armazena o valor real da variável. Adicionalmente, seja qual for a codificação escolhida, toda variável possui um limite inferior e superior que especificam, respectivamente, o valor mínimo e máximo que ela pode assumir. O conjunto desses limites define o domínio do problema;

[1] Um estudo comparativo sobre o uso de representação binária e representação de ponto flutuante pode ser visto em (JANIKOW; MICHALEWICZ, 1991)

Ao calcularmos a aptidão de um indivíduo através da função objetivo, temos como retorno uma “nota” que mostra de maneira explícita o quão eficiente foi aquela solução na tarefa de otimizar a função objetivo. Neste trabalho será considerado que a aptidão é armazenada na última posição de cada cromossomo. Os dois tipos de representação apresentados são exemplificados na Figura 1, onde as cores representam variáveis diferentes.

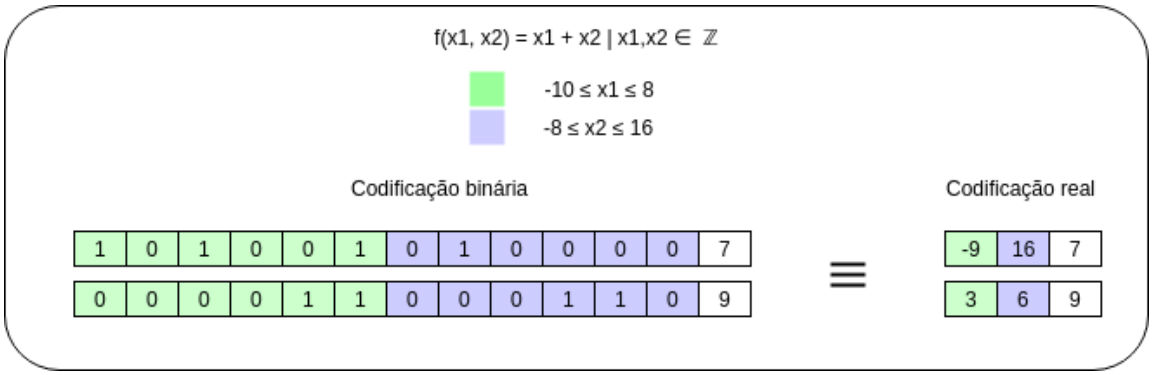

Figura 1 – Codificações binária e real.

Fonte: elaboração própria.

É comum que a população inicial seja composta de indivíduos totalmente aleatórios para que o algoritmo disponha de várias características distintas que possam ser exploradas. Outro aspecto importante é a definição do tamanho da população: a avaliação de uma população muito pequena leva pouco tempo, porém características importantes que levariam a uma solução ótima podem não ser cobertas por causa da baixa diversidade populacional; uma população muito grande, por outro lado, possui indivíduos diversos, mas necessita de mais tempo e recursos computacionais para ser avaliada (KIM; HAN, 2000).

2.1 PROCESSO EVOLUTIVO

Ao passarmos pelas etapas de seleção, cruzamento e mutação (abordadas nas seções posteriores), temos duas novas populações provisórias: uma constituída dos indivíduos da população anterior e outra contendo os novos indivíduos gerados. O fato da população ter um tamanho fixo implica que devemos selecionar quais indivíduos dentre as duas populações provisórias devemos escolher para fazer parte da população real. Podemos eliminar todos os indivíduos antigos e manter na nova população apenas os gerados, ou manter um número fixo dos melhores indivíduos da população anterior juntamente com os novos indivíduos - abordagem conhecida como elitismo (JONG, 1975), que garante que uma boa solução não se perca durante o processo evolutivo.

A cada iteração deste processo damos o nome de geração. Com o passar das gerações, a população é alterada e é importante que condições de parada sejam impostas para que a execução do algoritmo termine. Uma condição de parada comum é delimitar um número máximo de gerações que o algoritmo pode produzir. Outra estratégia é definir a quantidade máxima de gerações seguidas nas quais não há melhora significativa na aptidão (platô); neste caso diz-se que população convergiu. É possível também utilizar diferentes estratégias em conjunto: quando alguma das condições é satisfeita, a execução do algoritmo termina e nos é retornado como melhor solução o indivíduo mais adaptado da última geração. Na Figura 2 é apresentado um diagrama que descreve o processo evolutivo.

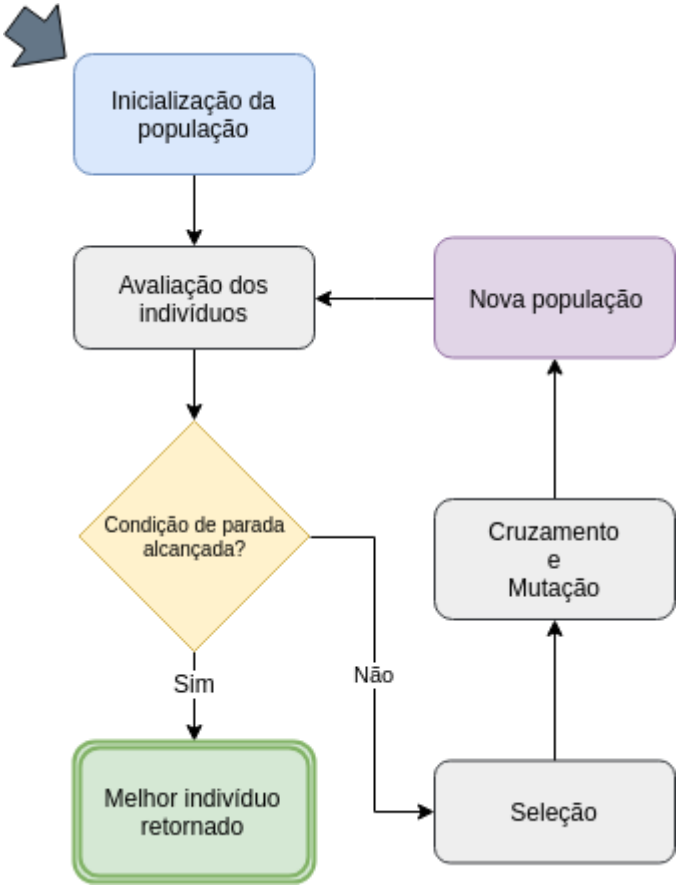

Figura 2 – Etapas do processo evolutivo.

Fonte: elaboração própria.

2.2 OPERADORES GENÉTICOS

O processo evolutivo depende das interações dos operadores genéticos com a população, pois através deles os indivíduos são modificados e evoluem. Nesta seção serão abordados os operadores genéticos mais comuns, sendo eles seleção, cruzamento e mutação. Algumas

estratégias serão apresentadas e suas vantagens e desvantagens expostas. É necessário, no entanto, analisar o problema para escolher os operadores adequados.

2.2.1 OPERADORES DE SELEÇÃO

Através do operador de seleção são escolhidos dentre os cromossomos aqueles mais aptos para que se reproduzam e levem seus genes para as gerações seguintes. É importante, porém, que os indivíduos com baixa aptidão também tenham chance de se reproduzir, pois podem possuir características que posteriormente - através da combinação com outros indivíduos - possam resultar numa solução ótima. Existem diversas estratégias de seleção. Exploraremos neste trabalho três delas: seleção por roleta, seleção por classificação e seleção por torneio.

No método de seleção por roleta, cada cromossomo tem uma probabilidade de escolha dada por

$$p(x_i) = fitness(x_i)/total\_fitness$$

, onde $fitness(x_i)$ é a aptidão do i-ésimo indivíduo e $total\_fitness$ a soma da aptidão de toda a população. Em algumas situações podemos encontrar na mesma população alguns indivíduos com aptidão positiva e outros negativa, então se faz necessário normalizar esses valores antes do cálculo da aptidão total. As probabilidades relativas podem ser vistas como fatias circulares em uma roleta: quanto maior a fatia, maior a chance do cromossomo ser escolhido, portanto indivíduos com boa aptidão têm maiores chances de ser escolhidos e, ainda assim, indivíduos pouco aptos não são ignorados. Apesar disso, indivíduos com aptidão muito maior que os demais podem dominar a roleta, reduzindo a chance dos demais a quase nulas. A título de exemplo, a roleta para uma população de quatro indivíduos com aptidões iguais a 1000, 6, 75 e 650 pode ser vista na Figura 3.

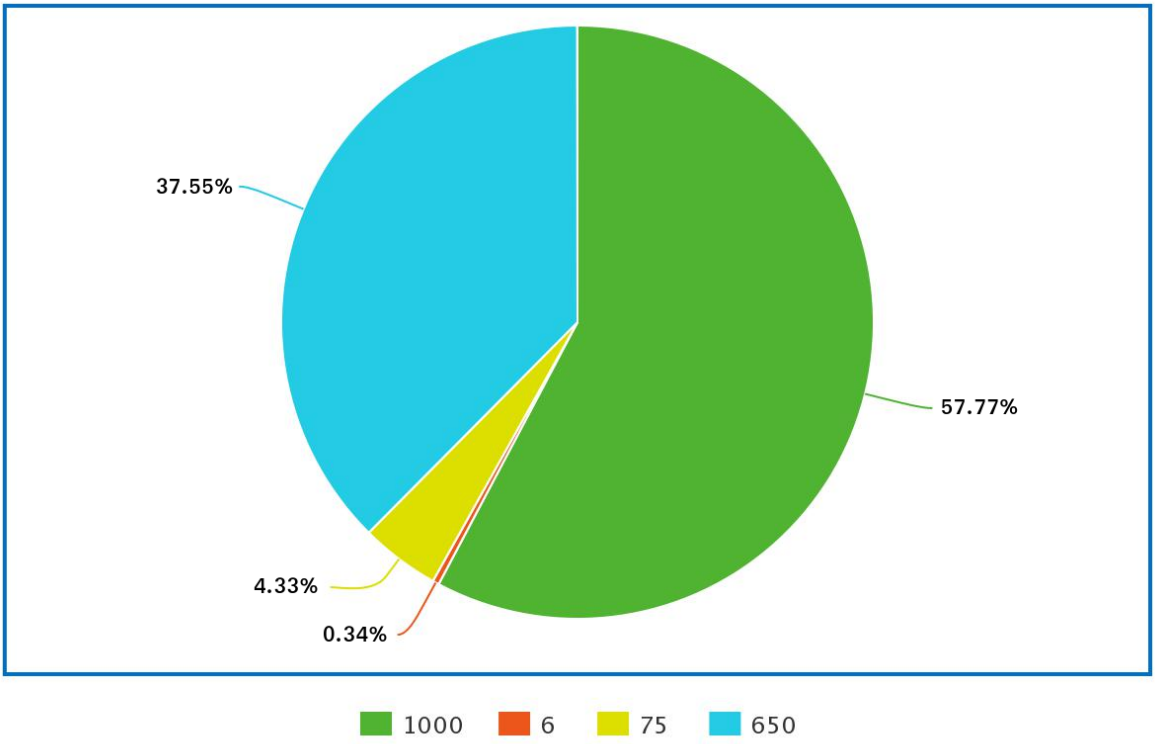

Figura 3 – Roleta de seleção para população desbalanceada.

Fonte: elaboração própria.

Para eliminar o problema de disparidade de aptidão, é possível utilizar a seleção por classificação. Neste método, a população é ordenada de forma decrescente de acordo com a aptidão, e cada indivíduo x_i recebe uma aptidão temporária $temp\_fitness(x_i) = i$ de ser escolhido, portanto o pior indivíduo recebe aptidão 1, o segundo pior aptidão 2 e assim por diante. A partir daí a seleção se dá da mesma forma que o método da roleta. Dessa maneira a diferença da probabilidade de escolha do indivíduo i e do indivíduo $i + 1$ é sempre a mesma, não importando o quão distintas sejam suas aptidões. Utilizando esta estratégia o cenário anterior é descrito no gráfico da Figura 4.

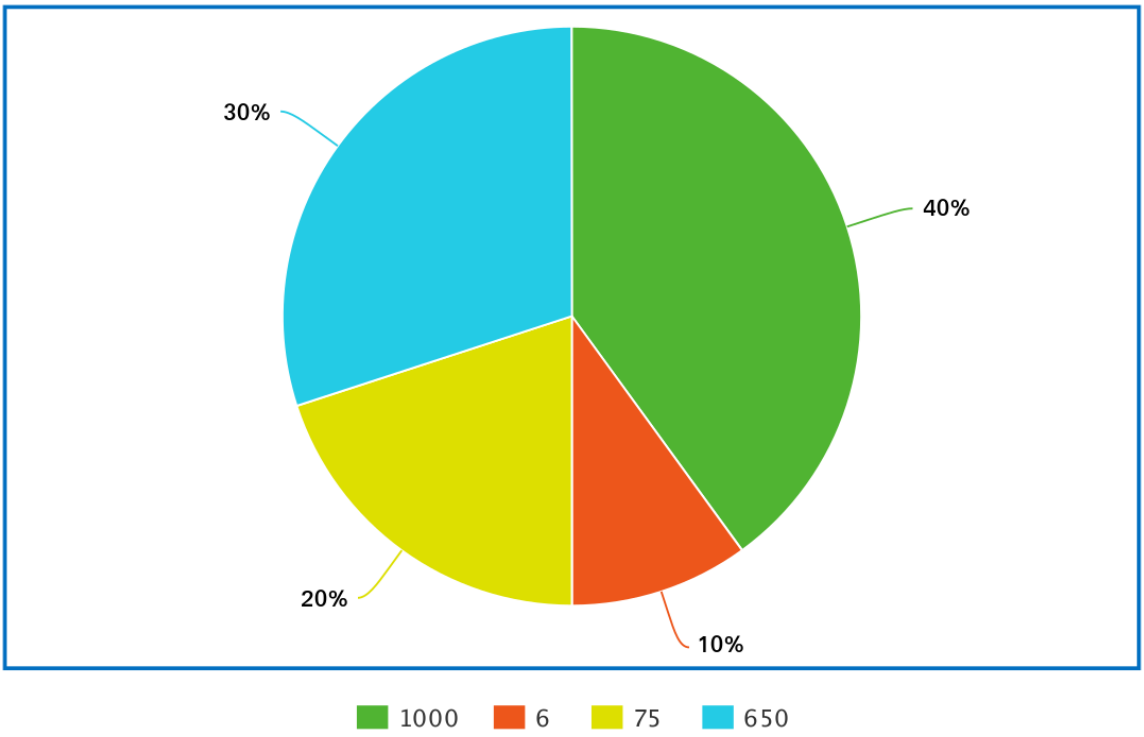

Figura 4 – Roleta para população desbalanceada utilizando classificação.

Fonte: elaboração própria.

Outra estratégia comum é a seleção por torneio, na qual n indivíduos são escolhidos randomicamente para competirem entre si. Neste trabalho consideraremos $n = 2$. Após escolhidos os "participantes" do torneio, o indivíduo com melhor aptidão entre eles é escolhido, dessa forma os indivíduos mais aptos são priorizados, mas os demais também têm chance de se reproduzir.

O método de seleção por roleta, apesar de simples, pode levar à convergência prematura, pois diferenças de aptidão muito gritantes podem anular a chance de indivíduos pouco aptos se reproduzirem, reduzindo a diversidade da população. Na seleção por classificação as chances de convergência prematura são bastante reduzidas, porém a convergência se dá mais lentamente devido à similaridade das probabilidades de escolha. A seleção por torneio é a mais simples de ser implementada pois não precisa de normalização dos valores de aptidão ou de ordenação dos indivíduos, porém a velocidade de convergência é um pouco prejudicada. Um estudo comparativo entre alguns métodos de seleção é apresentado em (SAINI, 2017).

2.2.2 OPERADORES DE CRUZAMENTO

Após escolhidos os indivíduos na etapa anterior, a passagem de características é realizada através do operador de cruzamento. A frequência em que os indivíduos "pais" trocam informações é chamada de probabilidade de cruzamento. Caso haja cruzamento, os genes dos dois indivíduos

pais são combinados, permitindo que troquem informações gerando dois novos indivíduos com suas próprias aptidões. Caso contrário, duas cópias idênticas dos pais são inseridos na população. É necessário atentar para que os indivíduos gerados sejam válidos, ou seja, estejam dentro do domínio do problema. Caso o indivíduo seja inválido, ele pode ser eliminado ou ter seus genes modificados para se ajustar no domínio do problema.

Uma estratégia de cruzamento básica é o cruzamento de ponto único. Primeiramente um ponto de corte aleatório é escolhido e, após isso, os genes desde o começo do indivíduo até o ponto de corte são copiados do primeiro pai e o resto copiado do outro pai. Esta estratégia é exemplificada na Figura 5, onde o traço vermelho é o ponto de corte.

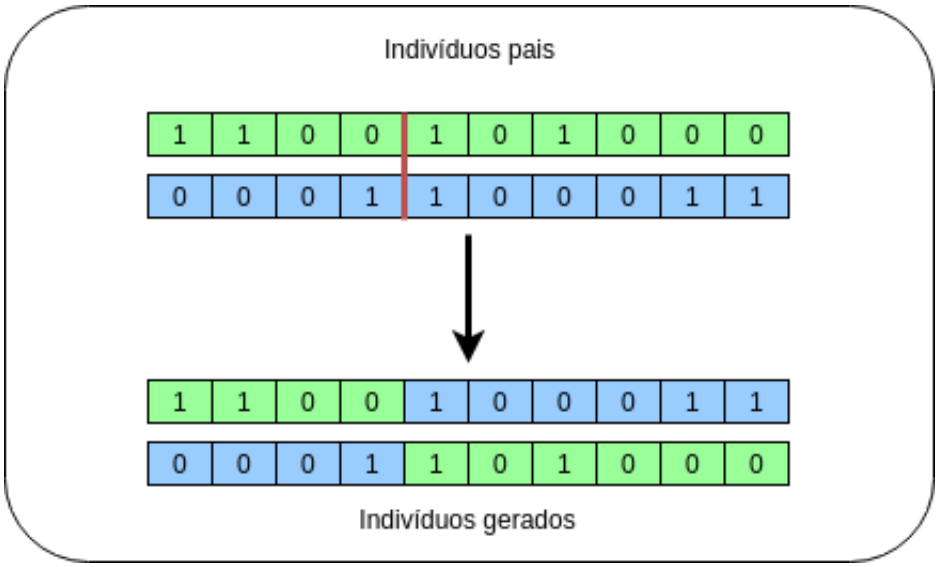

Figura 5 – Cruzamento de ponto único.

Fonte: elaboração própria.

O método de cruzamento de dois pontos funciona de forma semelhante ao anterior, porém no lugar de um único ponto, selecionamos aleatoriamente dois pontos de corte e então os genes entre eles são copiados de um cromossomo pai para o outro. A Figura 6 mostra uma exemplo do método de cruzamento de dois pontos.

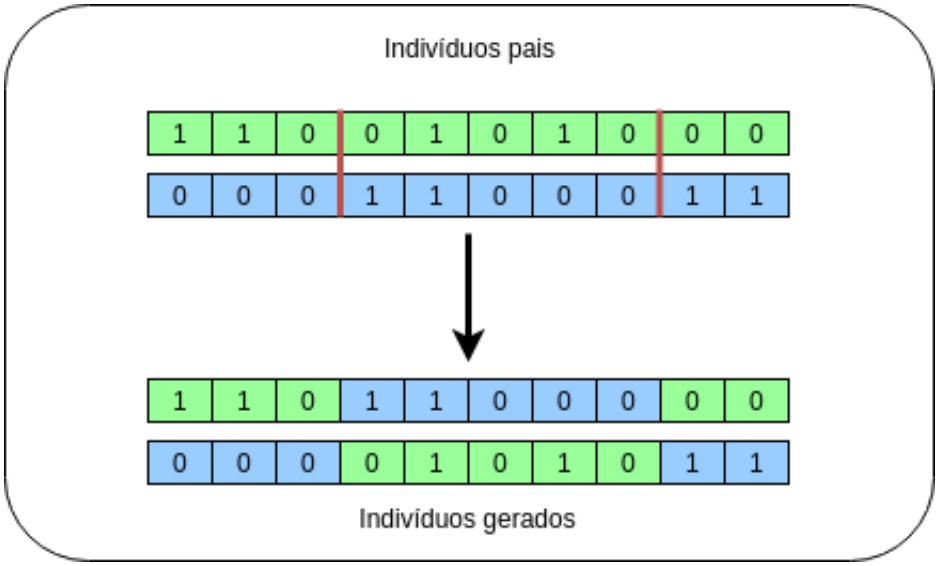

Figura 6 – Cruzamento de dois pontos.

Fonte: elaboração própria.

O último método de cruzamento abordado neste trabalho será o de cruzamento uniforme. Nesta estratégia, os genes são copiados aleatoriamente de um ou outro pai, portanto se o filho 1 recebe o gene k do pai 2, o filho 2 recebe o gene k do pai 1. A Figura 7 ilustra o funcionamento do método.

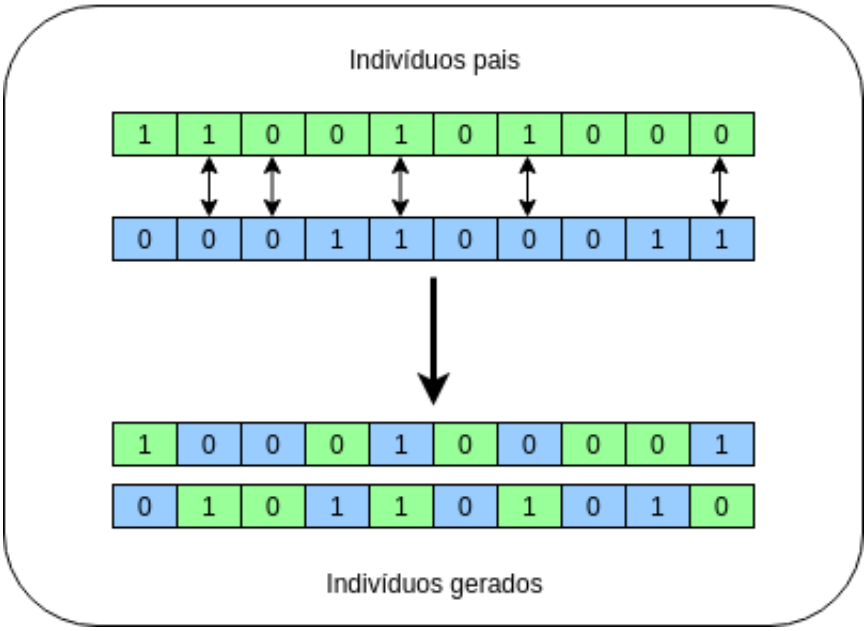

Figura 7 – Cruzamento uniforme.

Fonte: elaboração própria.

O operador de ponto único possui uma limitação chamada viés posicional (ESHELMAN, 1989), onde longas sequências de genes que seriam importantes para se chegar a uma solução

ótima são quebradas, retardando a velocidade de convergência. Utilizando o operador de dois pontos, sequências longas têm menos chance de serem quebradas, porém, assim como o operador de ponto único, algumas configurações não podem ser formadas (Não é possível combinar 1111, 0000 e formar 1010, por exemplo). No cruzamento uniforme qualquer combinação pode ser realizada, entretanto sequências importantes podem ser completamente destruídas.

2.2.3 OPERADORES DE MUTAÇÃO

Posteriormente à etapa de cruzamento, os cromossomos gerados passam por um novo operador genético cujo propósito é impedir que a busca fique estagnada num máximo (ou mínimo) local. O operador de mutação amplia o espaço de busca modificando cada gene do novo indivíduo de acordo com uma probabilidade de mutação p_m, ou seja, cada gene tem a probabilidade p_m de ser alterado. Assim como no cruzamento, é importante garantir que a mutação não gere indivíduos inválidos, ou seja, que fujam do domínio do problema. Os operadores de mutação abordados neste trabalho serão: mutação por inversão de bit, mutação uniforme e mutação não-uniforme.

A mutação por inversão de bit é utilizada em cromossomos com codificação binária e consiste na simples inversão de bits (o bit 0 torna-se 1 e vice-versa). Para codificação inteira e de ponto flutuante serão consideradas as mutações uniforme e não-uniforme.

Na mutação uniforme o novo valor v'_n da variável v_n é obtido através da equação

$$v'_n = random(LB(v_n), UB(v_n))$$

, onde $LB(v_n)$ e $UB(v_n)$ retornam, respectivamente, os limites inferior e superior de v_n e a função $random(a, b)$ retorna um número real aleatório dentro do intervalo [a, b].

A mutação não-uniforme, por sua vez, permite que as alterações nos indivíduos sejam mais agressivas nas gerações iniciais e mais suaves nas últimas. O novo valor de v_n é dado por

$$v'_n = \begin{cases} v_n + \Delta(t, UB(v_n) - v_n) & \text{, se k} \leqslant 0.5 \\ v_n - \Delta(t, v_n - LB(v_n)) & \text{, se k} > 0.5. \end{cases}$$

O parâmetro k é um número aleatório no intervalo [0,1] e t representa a geração atual do processo evolutivo. A função Δ, proposta por (MICHALEWICZ, 1996), é definida como

$$\Delta(t, y) = y * (1 - random(0, 1)^{(\frac{1-t}{T})^b})$$

, onde T é o número máximo de gerações e b é um parâmetro de dependência que determina o quão rápido Δ se aproxima de zero. Quanto maior é t, ou seja, quanto mais avançada a geração,

menores serão as alterações dos indivíduos, permitindo que o operador de mutação não-uniforme explore o espaço globalmente nas gerações iniciais e localmente em gerações avançadas.

Os operadores de mutação uniforme e não-uniforme funcionam de maneira semelhante, porém o segundo permite maior exploração do espaço de busca no início do processo evolutivo e um refinamento das soluções nas gerações avançadas.

3 TRABALHOS RELACIONADOS

Neste capítulo serão analisadas algumas ferramentas com propostas semelhantes a este trabalho e as vantagens e desvantagens de seu uso serão destacadas. Algumas das propostas abrangem outras estratégias de computação evolutiva, porém as comparações serão realizadas no escopo de Algoritmos Genéticos.

GAlib-IDE (TEODORO et al., 2008) permite a escolha de parâmetros para execução do AG através da interface gráfica. A função de aptidão é definida utilizando-se uma linguagem de linkagem dinâmica (C++, por exemplo), sendo necessário que o usuário tenha conhecimentos de programação. Outro ponto negativo é que os registros de saída são puramente textuais, o que não é muito atraente para usuários iniciantes.

EASEA (Université de Strasbourg, 2014) é uma plataforma para algoritmos evolucionários que possui interface gráfica apenas para visualização de resultados. A sua maior desvantagem é que para definição e configuração do algoritmo, uma linguagem especial - utilizada somente no EASEA - é utilizada.

ECJ (LUKE, 2012) é um framework de computação evolucionária escrito em Java. Possui várias configurações e opções, porém há apenas uma interface gráfica para plotagem de gráficos. A definição do problema é feita através da modificação de diversos arquivos de configuração, o que traz a necessidade de conhecer seus detalhes, tornando sua utilização pouco intuitiva.

A plataforma DREAM (PAECHTER et al., 2002) permite a implementação de algoritmos evolucionários utilizando Java. A ferramenta possui opções diversas e interface gráfica para definição e monitoramento de resultados, porém, ao contrário da ferramenta proposta, não há possibilidade de definir configurações de parâmetros diferentes.

Uma comparação mais simplificada entre a ferramenta proposta neste trabalho - EasyGA - e as demais apresentadas pode ser vista na tabela a seguir:

Característica	**EasyGA**	**GAlib-IDE**	**EASEA**	**ECJ**	**DREAM**
Linguagem de programação	Python	C++	C++	Java	Java
Interface para definição de problemas	Sim	Sim	Não	Não	Sim
Resultados gráficos	Sim	Não	Sim	Sim	Sim
Definição da função de aptidão através de programação	Não	Sim	Sim	Sim	Sim
Definição de configurações de parâmetros	Sim	Não	Não	Não	Não

Tabela 1 – Estudo comparativo das ferramentas.

Uma diferença entre as ferramentas analisadas e a proposta é o fato de não ser necessário conhecimento em programação para definição da função objetivo. Outra vantagem é a possibili-

dade de definir configurações de parâmetros - melhor explicadas no capítulo seguinte -, o que se faz interessante para iniciantes, pois permite a comparação das estratégias adotadas.

4 A FERRAMENTA

A aplicação proposta tem como objetivo dar liberdade ao usuário para que defina conjuntos diferentes de parâmetros do AG. Os componentes da aplicação estão na língua inglesa, visando maior divulgação e alcance da ferramenta. O código da ferramenta é aberto e instruções de instalação são disponibilizadas no apêndice A. Abordaremos na Seção 4.1 questões de desenvolvimento da ferramenta e na Seção 4.2 suas funcionalidades serão descritas.

4.1 TECNOLOGIAS UTILIZADAS

A linguagem utilizada no desenvolvimento da ferramenta foi Python 3.7, tanto pela familiaridade do autor, como pela sua extensa documentação. Python dispõe de diversas bibliotecas para implementação de Algoritmos Genéticos, tais como DEAP (Université Laval, 2009), Pyevolve (PERONE, 2009), Inspyred (GARRETT, 2012), entre outras. Apesar disso, todas as estratégias de AG foram implementadas do zero.

Para implementação da interface gráfica, foi utilizada a biblioteca de código aberto Kivy (VIRBEL et al., 2011). Para plotagem e visualização dos gráficos, a biblioteca escolhida foi a Matplotlib (HUNTER, 2007) devido à facilidade de uso e variedade de opções para construção dos gráficos.

4.2 MÓDULOS DA FERRAMENTA

Para tornar a utilização da aplicação mais intuitiva e organizada, sua apresentação é dada em quatro módulos diferentes, cada um deles representado numa aba da aplicação. Os módulos disponíveis são: *"Individual"*, *"Fitness function"*, *"Algorithm"* e *"Simulation and results"*. Os três primeiros módulos são utilizados para definição do problema, enquanto que no último o algoritmo é executado e os resultados são apresentados.

Nas seções seguintes cada um dos módulos será apresentado em detalhes. Para facilitar o entendimento, as telas possuem marcações especificando os componentes disponíveis.

4.2.1 Individual

Neste módulo são definidas as características e estrutura dos cromossomos utilizados na execução do algoritmo. Na visualização padrão - apresentada na Figura 8 - o cromossomo possui representação binária e apenas uma variável cujo domínio é o intervalo [-10, 10].

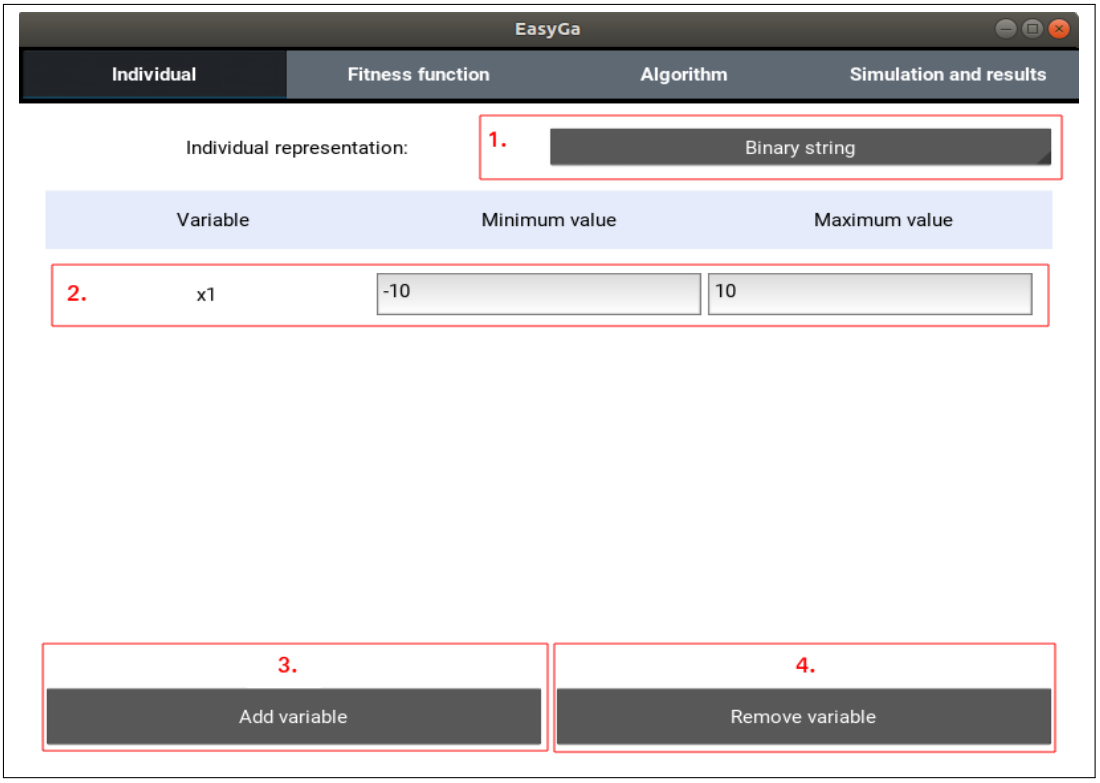

Figura 8 – Visualização padrão da aba “Individual”.

Fonte: elaboração própria.

No menu *“Individual representation”* (**1**) é possível escolher o tipo de representação (codificação) dos cromossomos. Ao clicar no botão, um menu de seleção apresenta os três tipos de representação possíveis: cadeia binária, cadeia de inteiros e cadeia de ponto flutuante.

A área **2** possui o contexto da variável *“x1”*. O primeiro campo representa o nome da variável, o segundo o valor mínimo que a variável pode assumir e o terceiro o valor máximo. O usuário pode alterar o segundo e o terceiro campo a vontade. O nome da variável é padronizado e novas variáveis seguem essa sequência.

O botão *“Add variable”* (**3**) acrescenta uma nova variável ao final da lista de variáveis. Todas as variáveis, assim como a primeira, podem ter seus domínios alterados. O botão *“Remove variable”* (**4**), por sua vez, remove a última variável da lista quando pressionado, com exceção da variável “x1” que não permite exclusão. Na Figura 8 temos três variáveis definidas.

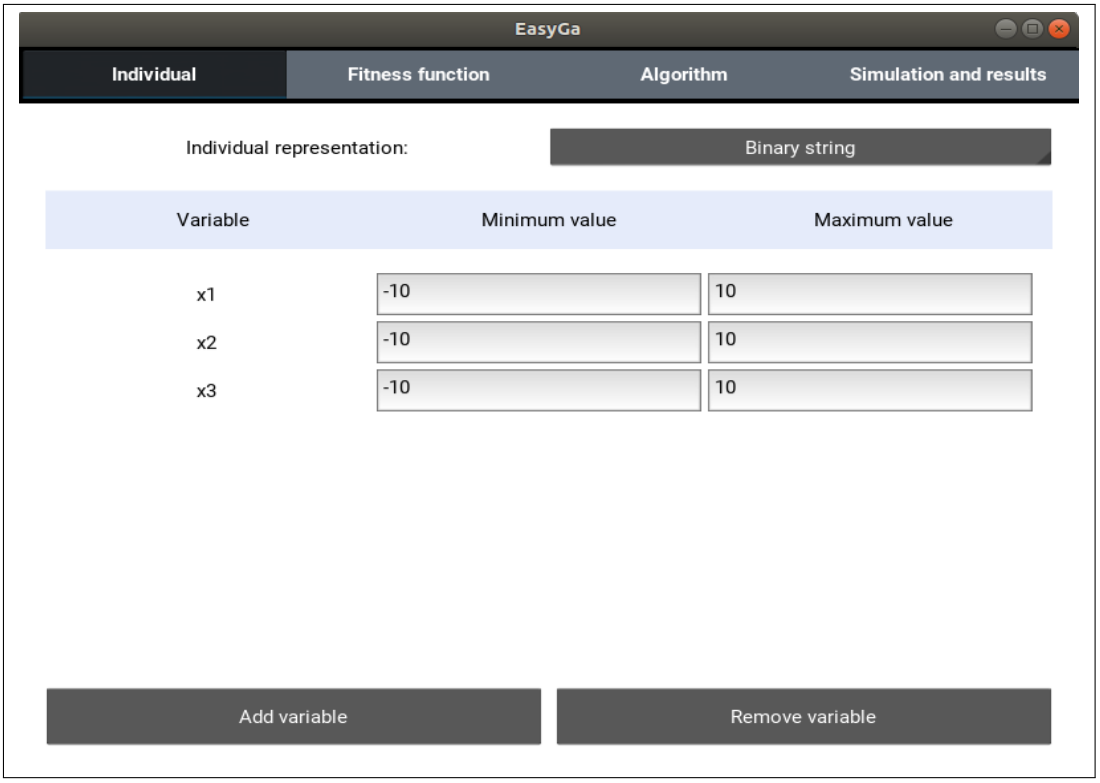

Figura 9 – Visualização da aba "Individual"com três variáveis adicionadas.

Fonte: elaboração própria.

4.2.2 Fitness function

Aqui a função de aptidão é definida e o objetivo da otimização é escolhido. A função padrão é a função identidade, podendo ser modificada pelo usuário. Abaixo a visualização padrão do módulo é apresentada:

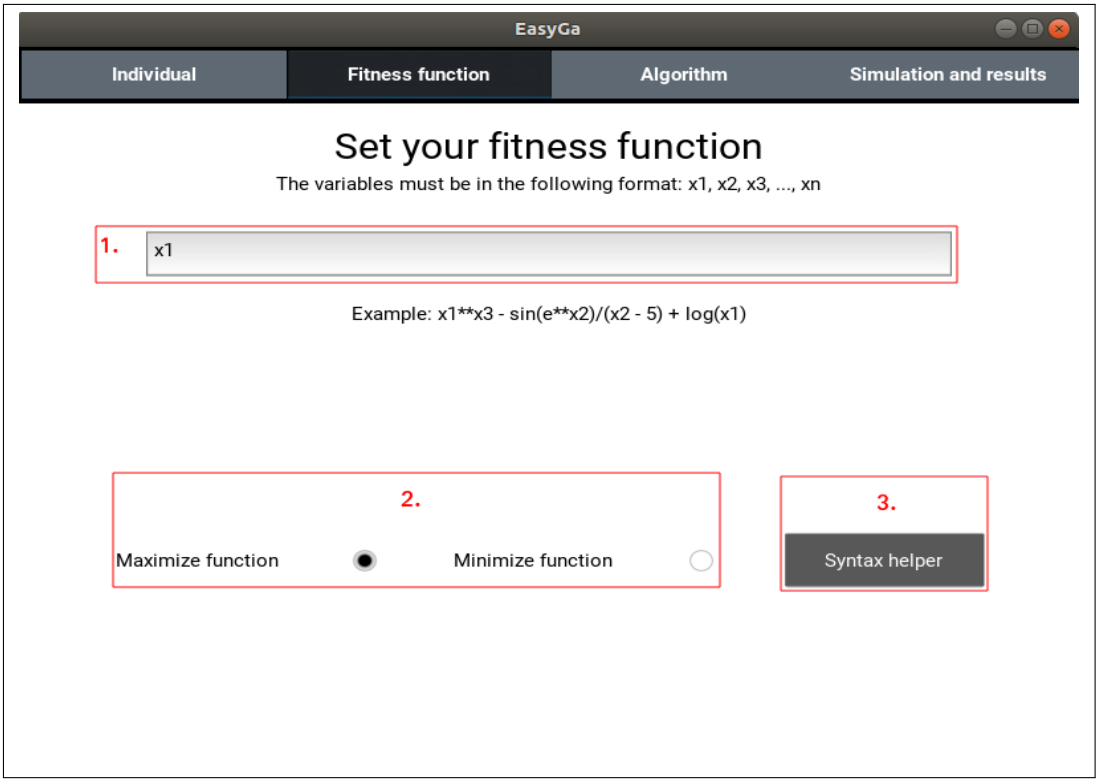

Figura 10 – Visualização padrão da aba “Fitness function”.

Fonte: elaboração própria.

Na entrada de texto (**1**) o usuário define a função de aptidão. Caso algum erro de sintaxe ou variável não declarada seja encontrado na função, o usuário é devidamente notificado com um pop-up de aviso quando o algoritmo é executado na aba *“Simulation and results”*.

O componente de seleção (**2**) permite a escolha do objetivo da otimização. Já o botão *“Syntax helper”* (**3**) mostra uma tela de ajuda, exemplificando as funcionalidades e demonstrando a sintaxe que deve ser utilizada na definição da função de aptidão. A tela de ajuda é apresentada na Figura 11.

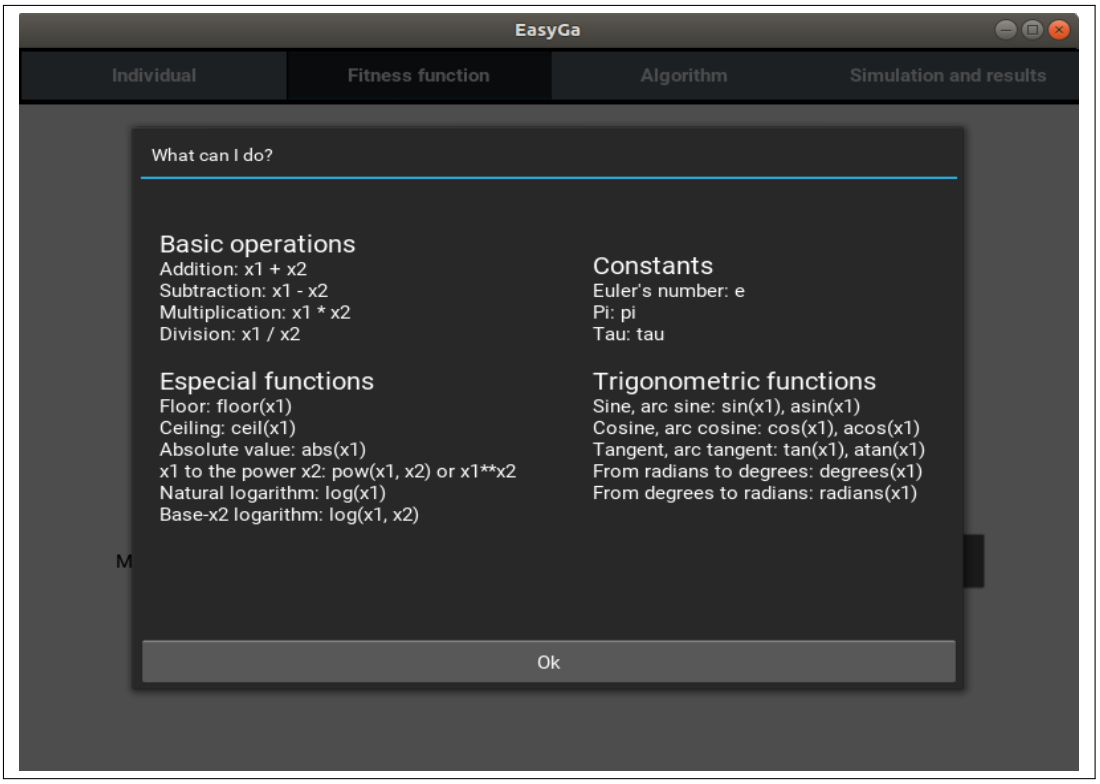

Figura 11 – Tela de ajuda para função de aptidão.

Fonte: elaboração própria.

4.2.3 Algorithm

Aqui os parâmetros para execução do AG são definidos e as configurações gerenciadas. Cada configuração possui uma combinação de parâmetros definida pelo usuário, chamaremos essa combinação de **contexto**. Como pode ser visto na Figura 12 a seguir, o módulo é dividido em duas áreas.

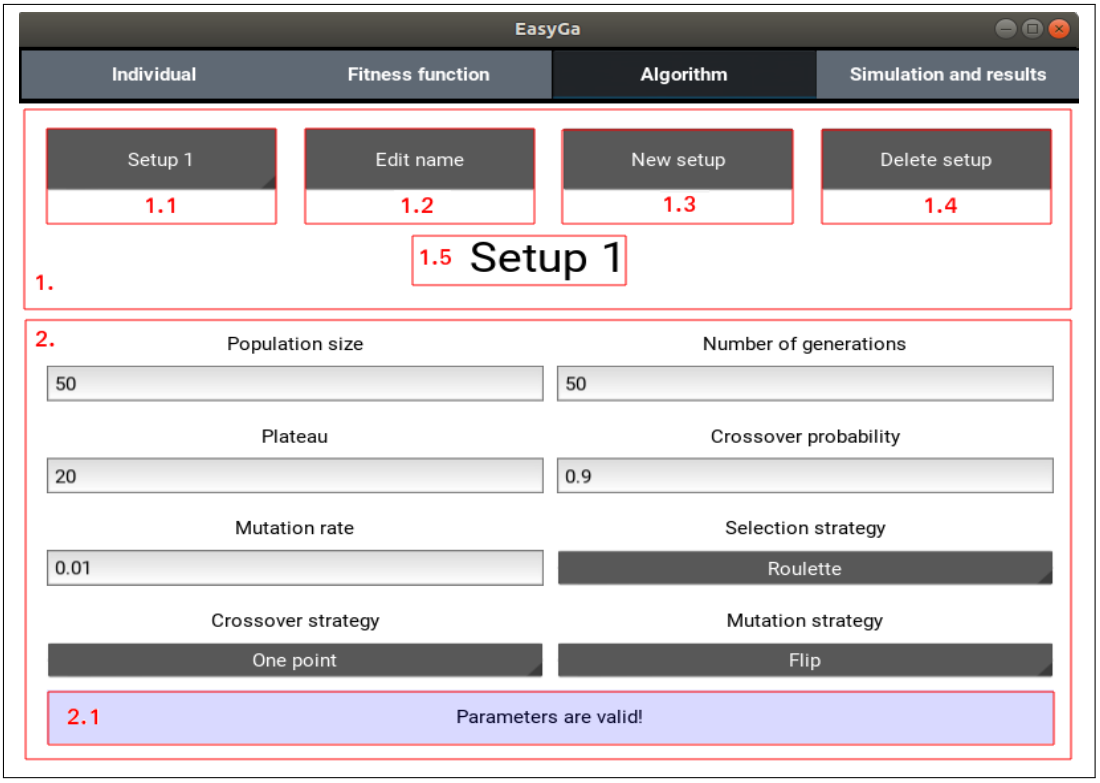

Figura 12 – Visualização padrão da aba “Algorithm”.

Fonte: elaboração própria.

Na área **1** estão presentes as opções de configuração. Quando pressionado, o botão **1.1** mostra a lista das configurações existentes (apenas “Setup 1” na inicialização da ferramenta), permitindo a troca de contexto. O título **1.5** indica o nome da configuração atual.

O botão *“Edit name”* (**1.2**) permite a edição do nome da configuração. Ao pressionar o botão *“Delete setup”* (**1.4**) a configuração atual é excluída, com exceção dos casos onde apenas uma configuração existe, nos quais a operação é ignorada. Uma nova configuração pode ser criada utilizando o botão *“New setup”* (**1.3**): um novo contexto será criado e pode ser modificado de forma independente aos contextos já existentes.

Na área **2**, os parâmetros referentes à configuração escolhida são definidos, sendo eles: tamanho da população, número de gerações, platô, probabilidade de cruzamento, taxa de mutação, estratégia de seleção, estratégia de cruzamento e estratégia de mutação. As estratégias de seleção disponíveis são: roleta, torneio e ranking. As de cruzamento são: ponto único, dois pontos e uniforme. As estratégias de mutação, porém, dependem do tipo de codificação escolhida pelo usuário: caso a representação dos indivíduos seja cadeia binária, a única estratégia disponível é a inversão de bits; caso a representação seja inteira ou de ponto flutuante, as estratégias possíveis são mutação uniforme e não-uniforme.

A área **2.1** mostra o estado do contexto. Caso algum parâmetro seja inválido (um ponto

flutuante no campo de tamanho da população, por exemplo) uma mensagem de erro apropriada será apresentada e os demais campos serão desabilitados até que o campo inválido seja corrigido. Na Figura 13 este exemplo é apresentado.

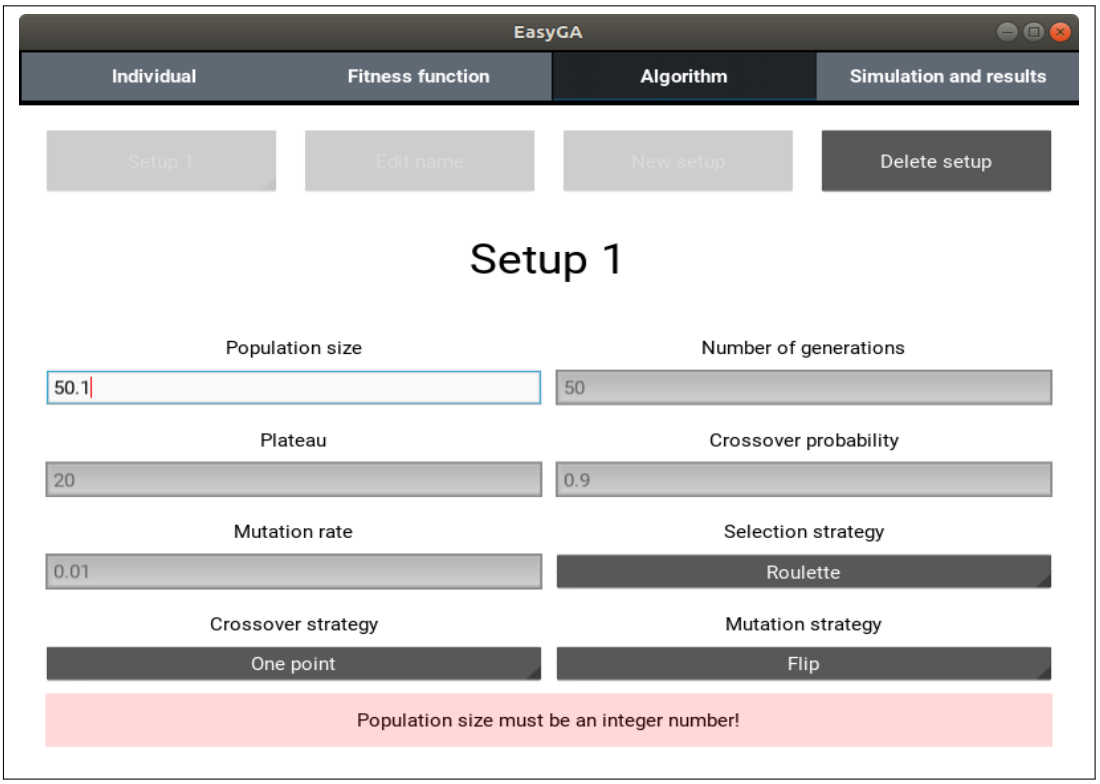

Figura 13 – Parâmetro de configuração inválido.

Fonte: elaboração própria.

4.2.4 Simulation and results

O módulo “Simulation and results” é responsável pela execução do Algoritmo Genético e disponibilização dos resultados. A visualização padrão do módulo é apresentada na figura a seguir.

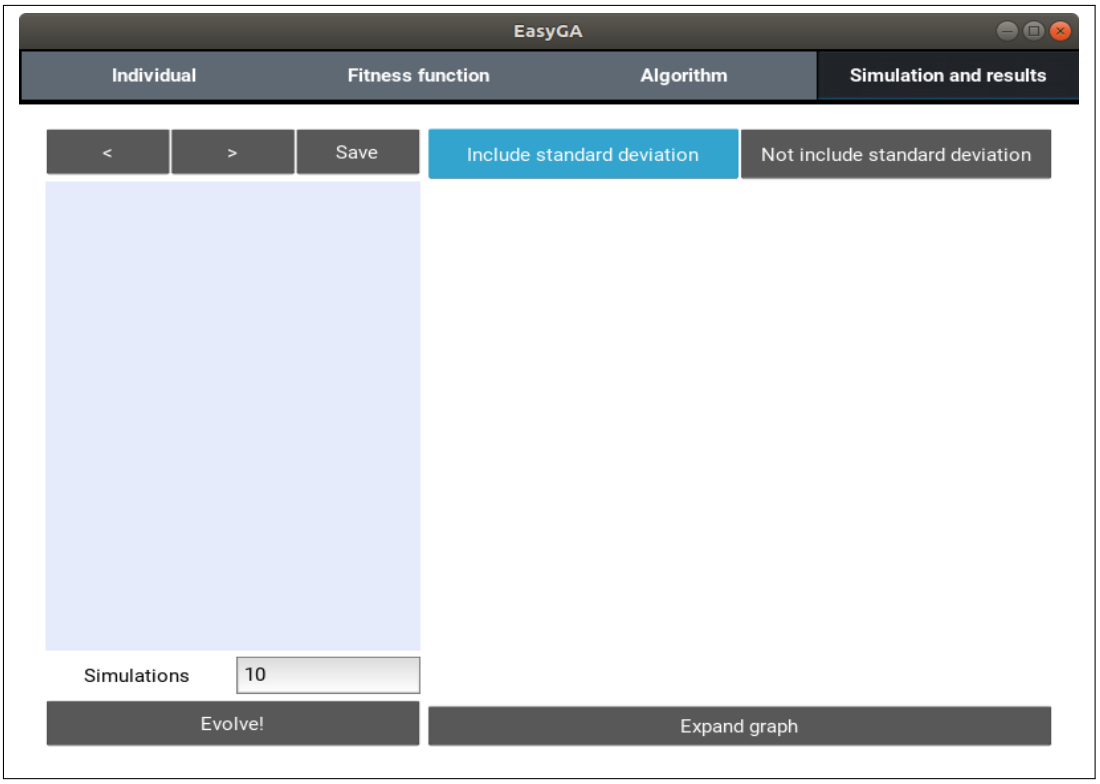

Figura 14 – Visualização padrão da aba “Simulation and results”.

Fonte: elaboração própria.

Ao pressionarmos o botão *“Evolve!”* - encontrado no canto inferior da coluna esquerda - os dados fornecidos pelo usuário são recolhidos e o AG executado. Devido ao caráter aleatório dos AGs, é interessante executar o AG múltiplas vezes para os mesmos parâmetros. No campo *"Simulations"* é definido quantas vezes o AG é executado para cada configuração definida. Cada execução é chamada de simulação. Após o processamento, a coluna da esquerda apresenta os registros de texto e a da direita os gráficos.

Para demonstrar o funcionamento deste módulo, o seguinte cenário será considerado: iremos minimizar a função de Himmelblau, definida por $f(x1, x2) = (x1^2 + x2 - 11)^2 + (x1 + x2^2 - 7)^2 \mid x1, x2 \in \mathbb{R}$, cujo valor mínimo é 0 e as variáveis assumem valores entre -5 e 5. Definiremos as configurações Setup 1 e Setup 2, apresentadas na tabela a seguir:

Parameter	Setup 1	Setup 2
Population size	100	50
Number of generations	50	50
Plateau	20	20
Crossover probability	0.9	0.9
Mutation rate	0.01	0.2
Selection strategy	Roulette	Tournament
Crossover strategy	One point	One point
Mutation strategy	Uniform	Non-uniform

Tabela 2 – Contexto das configurações Setup 1 e Setup 2.

Fonte: elaboração própria.

Realizaremos 30 simulações para cada configuração. Os resultados das simulações podem ser conferidos na Figura abaixo e serão analisados nas seções 4.2.4.1 e 4.2.4.2.

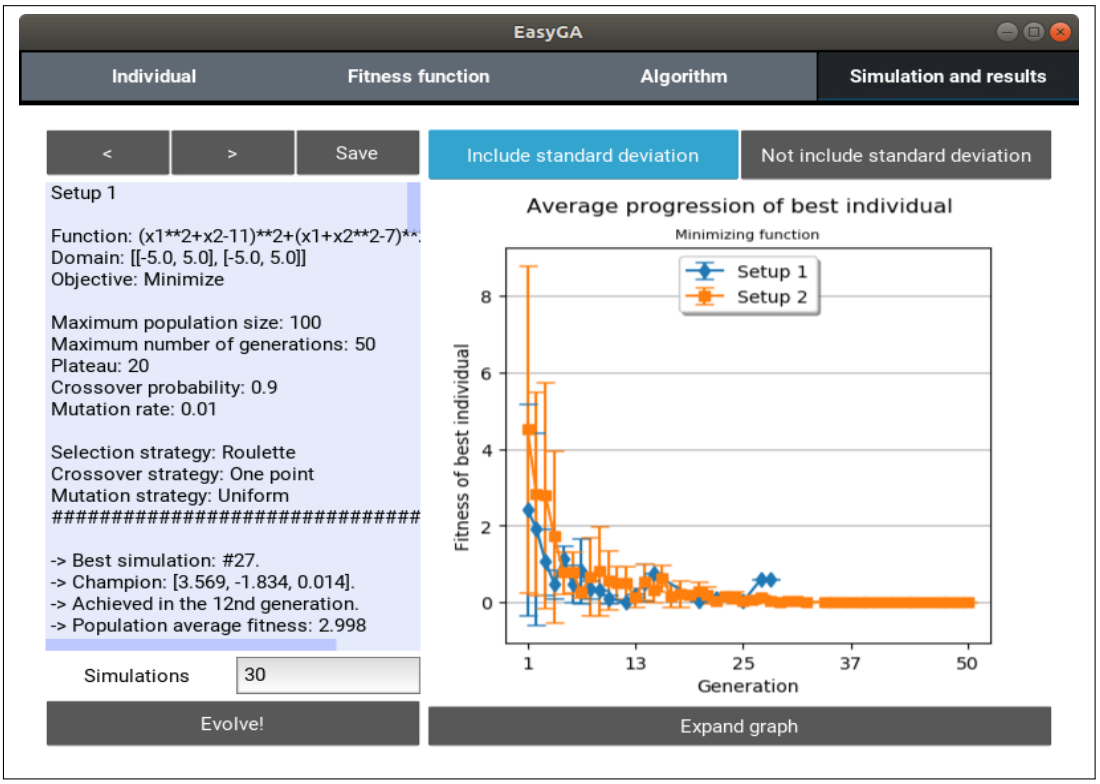

Figura 15 – Aba “Simulation and results” após execução do AG.

Fonte: elaboração própria.

4.2.4.1 Registros de texto

Na coluna esquerda da Figura 15 estão descritos os registros de texto. Apenas uma configuração por vez tem seus resultados mostrados na coluna, começando pela primeira configuração definida. Na barra superior estão dispostos três botões. Os botões “<” e “>”, mostram o registros

da configuração anterior e posterior, respectivamente. Enquanto que o botão *"Save"* salva o registro da configuração atual num arquivo de texto na pasta *"Output"*, criada no diretório no qual o script está sendo executado.

Abaixo da barra superior há uma área de texto. Esta área possui barras de rolagens vertical e horizontal para facilitar a leitura do usuário. Aqui podemos visualizar o contexto da configuração e os resultados finais, como visto na Figura a seguir.

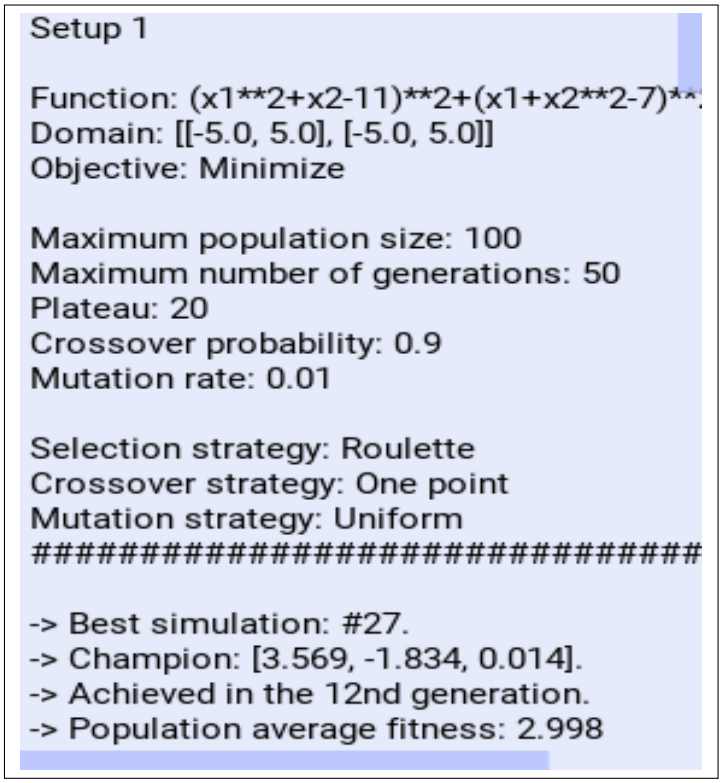

Figura 16 – Resultados da configuração 1.

Fonte: elaboração própria.

Os resultados finais são: melhor simulação (aquela que gerou o campeão e cuja população tem a melhor aptidão média), campeão (o indivíduo mais apto), geração na qual o campeão foi originado e a aptidão média dos indivíduos desta simulação (característica importante para análise de convergência). Além disso, caso o usuário role a tela para baixo, os dados de progressão do melhor indivíduo para cada simulação são detalhados, tal qual a Figura abaixo.

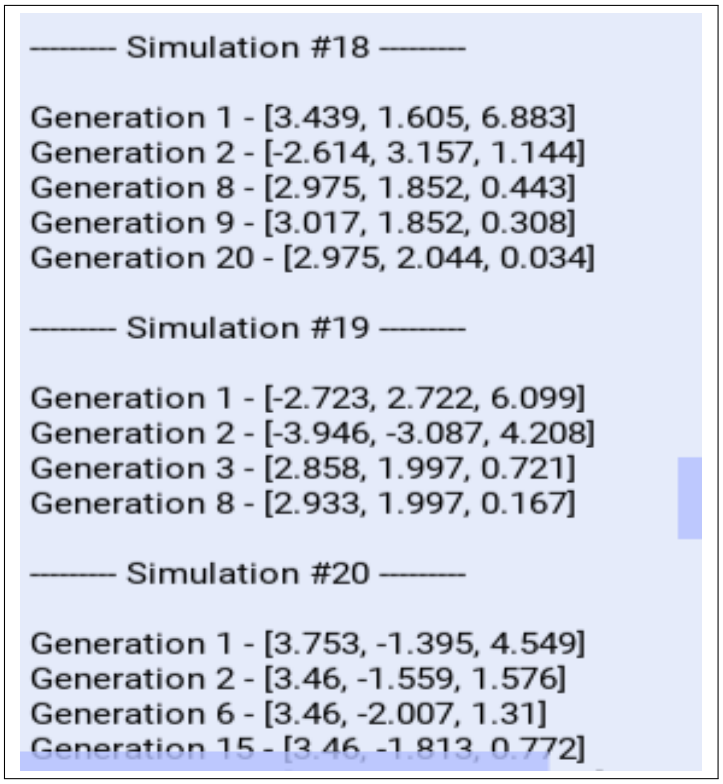

Figura 17 – Registro de progressão da configuração 1.

Fonte: elaboração própria.

4.2.4.2 Registros gráficos

Na coluna direita visualizamos os registros gráficos. Na barra superior estão dispostos dois botões de seleção. O primeiro deles, o botão *"Include standard deviation"*, mostra o gráfico de progressão média do melhor indivíduo - visto na Figura 18 -, no qual cada marcação representa a aptidão média naquela geração e as barras verticais o desvio padrão.

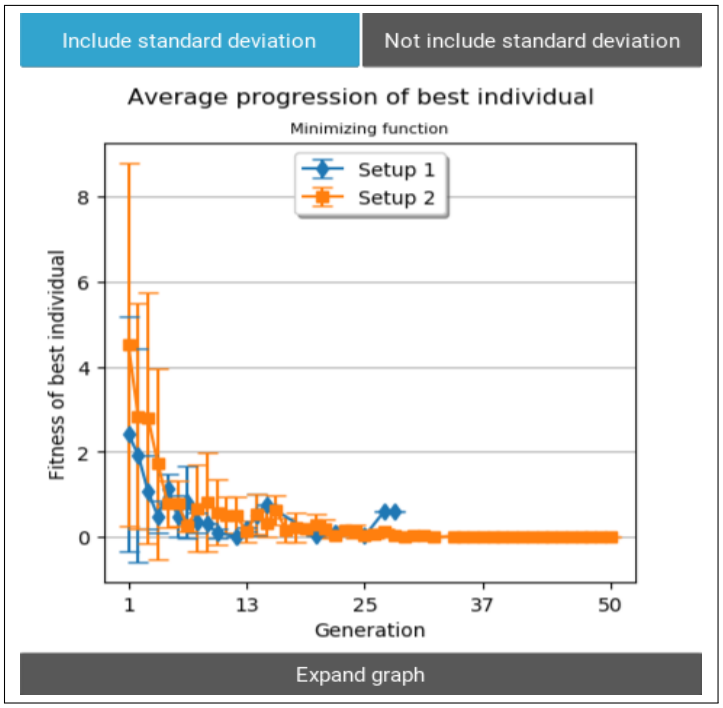

Figura 18 – Gráfico de progressão média do melhor indivíduo.

Fonte: elaboração própria.

Ao selecionarmos o botão *"Not include standard deviation"* habilitamos a visualização do mesmo gráfico, porém sem as barras de desvio padrão, permitindo melhor visualização dos dados. Este gráfico é exemplificado a seguir.

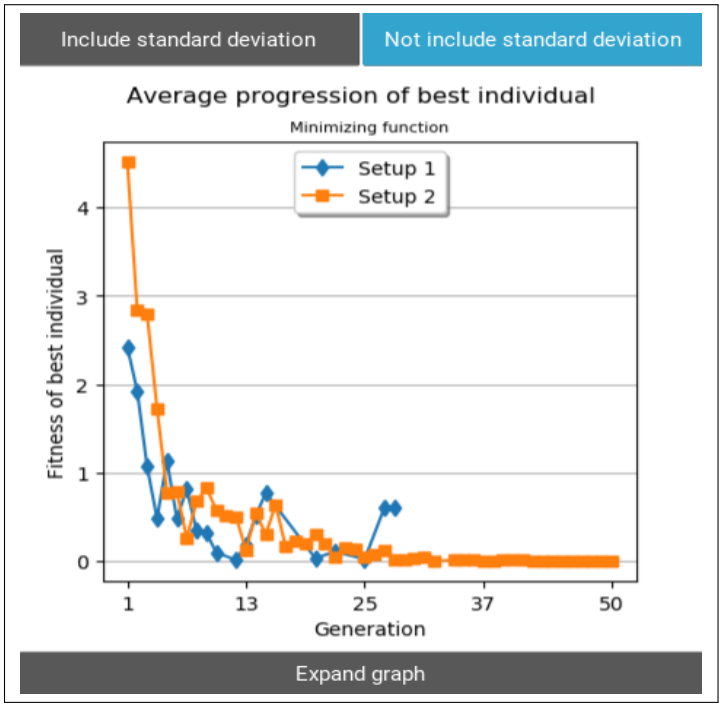

Figura 19 – Gráfico de progressão média do melhor indivíduo sem desvio padrão.

Fonte: elaboração própria.

Por fim, o botão *"Expand graph"* expande o gráfico visualizado em uma nova janela [1], permitindo melhor observação de detalhes e também que o usuário salve o gráfico em formato de imagem.

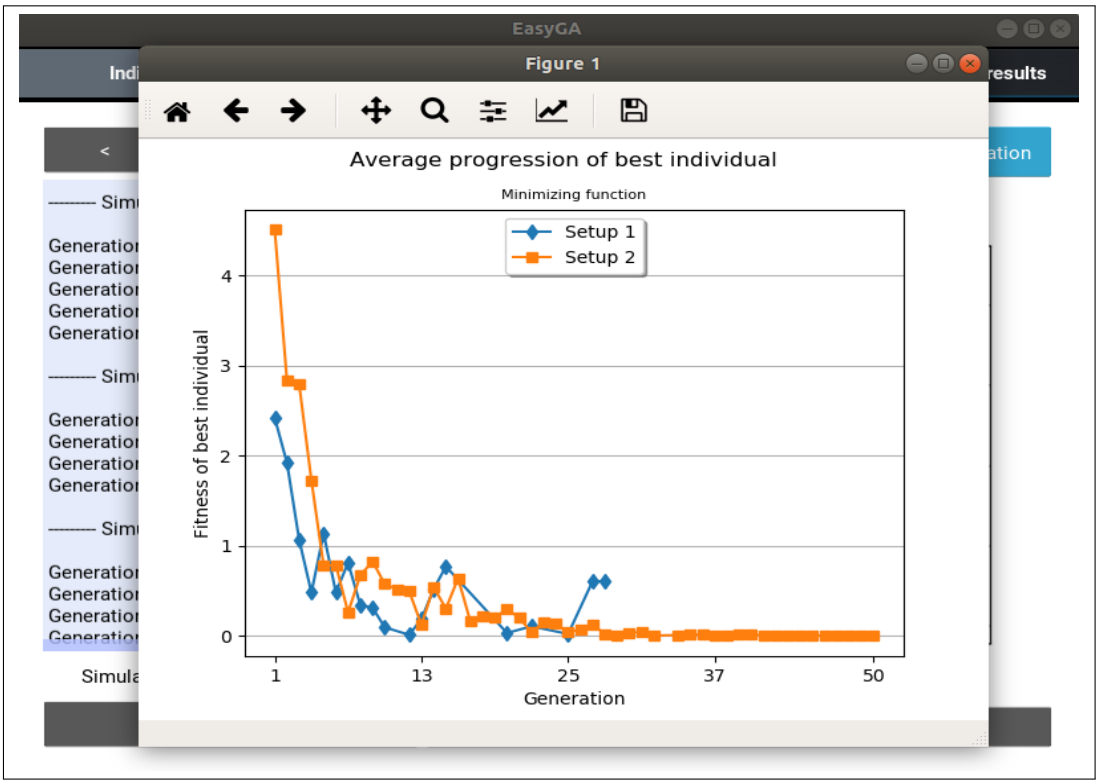

Figura 20 – Gráfico expandido.

Fonte: elaboração própria.

[1] Para realizar tal expansão foi utilizada a visualização padrão da biblioteca matplotlib (HUNTER, 2007).

5 CONCLUSÃO

Ao longo deste trabalho, a implementação, bem como as bases teóricas da aplicação EasyGA foram descritas. A ferramenta permite a definição e simulação de AGs para solução de problemas de otimização matemática de forma simples e intuitiva, possibilitando que usuários com diferentes níveis de conhecimento compreendam o funcionamento básico de um Algoritmo Genético. Essas características tornam essa uma boa ferramenta auxiliar para disciplinas que contenham AG em seu conteúdo programático.

Dentre as opções de parâmetros estão: tipo de codificação dos indivíduos; estratégias de operadores genéticos; tamanho da população e número máximo de gerações. A possibilidade de criar diferentes configurações de parâmetros dá ao usuário embasamento prático para escolher estratégias para atacar problemas diferentes. Os registros de saída podem ser salvos para estudo posterior e permitem uma análise mais detalhada ao disponibilizar a progressão de aptidão dos indivíduos e a aptidão média da melhor simulação.

Por ser de código aberto, a ferramenta favorece a evolução e melhoria das funcionalidades já existentes ou adição de novas funcionalidades não contempladas em sua primeira versão. Algumas sugestões de trabalhos futuros seriam: otimização multi-objetiva, adição de novos operadores e codificações, ou a ampliação do escopo, de forma a permitir a modelagem de outros problemas mais próximos ao mundo real, tais como o problema do agendamento ou do caixeiro-viajante.

REFERÊNCIAS

BERTSIMAS, D.; TSITSIKLIS, J. et al. Simulated annealing. **Statistical science**, Institute of Mathematical Statistics, v. 8, n. 1, p. 10–15, 1993.

DANTZIG, G. B.; THAPA, M. N. **Linear programming 1: introduction**. Stanford: Springer Science & Business Media, 2006.

ESHELMAN, L. J. Biases in the crossover landscape. **ICGA'89**, p. 10–19, 1989.

FROWD, C. D.; HANCOCK, P. J.; CARSON, D. Evofit: A holistic, evolutionary facial imaging technique for creating composites. **ACM Transactions on applied perception (TAP)**, ACM, v. 1, n. 1, p. 19–39, 2004.

GARRETT, A. **Inspyred: Bio-inspired Algorithms in Python**. 2012. <https://pythonhosted.org/inspyred/>, Acessado por último em 2019-08-17.

GEIJTENBEEK, T.; PANNE, M. V. D.; STAPPEN, A. F. V. D. Flexible muscle-based locomotion for bipedal creatures. **ACM Transactions on Graphics (TOG)**, ACM, v. 32, n. 6, p. 206, 2013.

GLOBUS, A.; HORNBY, G.; LINDEN, D.; LOHN, J. Automated antenna design with evolutionary algorithms. Citeseer, 2006.

HOLLAND, J. H. Adaptation in natural and artificial systems. **Ann Arbor: University of Michigan Press**, 1975.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

JANIKOW, C. Z.; MICHALEWICZ, Z. An experimental comparison of binary and floating point representations in genetic algorithms. p. 31–36, 1991.

JONG, K. A. D. Analysis of the behavior of a class of genetic adaptive systems. 1975.

JOSHI, P. **Artificial Intelligence with Python**. Birmingham: Packt Publishing, 2017.

KIM, K.-j.; HAN, I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. **Expert systems with Applications**, Elsevier, v. 19, n. 2, p. 125–132, 2000.

LINDEN, R. **Algoritmos genéticos (2a ediçao)**. Rio de Janeiro: Brasport, 2008.

LUKE, S. **ECJ Project**. 2012. <http://cs.gmu.edu/~eclab/projects/ecj/">, Acessado por último em 2019-08-25.

MICHALEWICZ, Z. **Genetic Algorithms + Data Structures = Evolution Programs**. New York: Springer Science & Business Media, 1996.

MITCHELL, M. **An introduction to genetic algorithms**. Cambridge: MIT press, 1998.

PAECHTER, B.; BACK, T.; SCHOENAUER, M.; SEBAG, M.; EIBEN, A.; MERELO, J.; FOGARTY, T. **Distributed Resource Evolutionary Algorithms Machine**. 2002. <"http://www.soc.napier.ac.uk/~benp/dream/dream.htm">, Acessado por último em 2019-08-25.

PERONE, C. S. **Pyevolve Documentation, Release 0.5**. 2009.

SAINI, N. Review of selection methods in genetic algorithms. **International Journal Of Engineering And Computer Science**, v. 6, n. 12, p. 22261–22263, 2017.

SELMAN, B.; GOMES, C. P. Hill-climbing search. **Encyclopedia of Cognitive Science**, Wiley Online Library, 2006.

TEODORO, F. R.; PARPINELLI, R. S.; SÁ, C. C. de. Galib-ide: um framework para experimentos com algoritmos genéticos. **Anais SULCOMP**, v. 4, 2008.

TURING, A. M. Computer machinery and intelligence. **Mind**, v. 59, p. 433–460, 1950.

Université de Strasbourg. **EAsy Specification of Evolutionary Algorithms**. 2014. <"http://easea.unistra.fr/index.php/EASEA_platform/">, Acessado por último em 2019-08-25.

Université Laval. **Distributed Evolutionary Algorithms in Python**. 2009. <https://deap.readthedocs.io/en/master/>, Acessado por último em 2019-08-17.

VIRBEL, M. et al. **Kivy: Cross-platform Python Framework for NUI Development**. 2011. <"https://kivy.org/">, Acessado por último em 2019-08-25.

APÊNDICE A – Instalação da ferramenta

A aplicação EasyGA foi feita para plataformas Linux. Os passos de instalação e execução são descritos abaixo:

1. Instale git e pip. Utilize o instalador de pacotes adequando para sua distribuição Linux (apt para o Ubuntu)

   ```
   $ sudo apt install git
   $ sudo apt install python3-pip
   ```

2. Instale o virtualenv

   ```
   $ pip install virtualenv
   ```

3. Crie um ambiente virtual (isso criará uma pasta chamada "EgaEnv"na pasta raíz)

   ```
   $ virtualenv EgaEnv
   ```

4. Ative o ambiente virtual

   $ **source** EgaEnv/bin/activate

5. Instale as bibliotecas kivy, matplotlib e six

   ```
   $ pip install kivy matplotlib six
   ```

6. Instale kivy.garden.matplotlib

   ```
   $ garden install matplotlib
   ```

7. Clone o repositório EasyGA

   ```
   $ git clone https://github.com/Leandro97/EasyGA.git
   ```

8. Entre no repositório clonado e execute a ferramenta

   ```
   $ python easyga.py
   ```

9. Para sair do ambiente virtual execute

   ```
   $ deactivate
   ```