ALGORITMOS GENÉTICOS APLICADOS À PROTOTIPAÇÃO ROBÓTICA: UM ESTUDO DE CASO

Cauã Anacleto[1], Rafael Trigo Azevedo Reis de Souza[2], Sérgio Muinhos Barroso Lima[3]

1Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais, Rio Pomba, Brasil(anacleto.for.work@gmail.com)
2 Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais, Rio Pomba, Brasil (rafael.trigo.azevedo@gmail.com)
3 Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais, Rio Pomba, Brasil (sergio.lima@ifsudestemg.edu.br)

Resumo: Este artigo apresenta o processo de desenvolvimento de um jogo da velha robótico desenvolvido em Raspberry Pi, assim como os problemas encontrados no caminho, possibilidades e outras potencialidades da aplicação de algoritmos genéticos em artefatos de prototipação robótica, seja para navegação autônoma, definição de comportamentos em jogos, ou na solução de outros problemas relacionados à natureza desses artefatos, com vistas à criação de protótipos de jogos, brinquedos, automação ou mesmo para uso educacional, principalmente no ensino da robótica e da inteligência artificial.

Palavras-chave: Algoritmos Genéticos; Inteligência Artificial; Raspberry Pi; Arduino.

INTRODUÇÃO

Este artigo apresenta o processo de desenvolvimento de um protótipo robótico para um jogo da velha com "inteligência" promovida por algoritmo genético programado em linguagem C e construído em Raspberry Pi. Esse artefato, bem como as outras iniciativas descritas neste artigo, foram desenvolvidas no contexto do projeto de iniciação científica "Algoritmos Genéticos Aplicados à Prototipação Robótica", financiado pelo CNPq.

Os algoritmos genéticos (AG) são oriundos dos estudos em Inteligência Artificial (IA), sendo classificados como algoritmos evolutivos, pois as possíveis soluções geradas se adaptam em um ciclo interativo de seleção natural, ou adaptabilidade, à natureza intrínseca e particularidades de um dado problema. Esse ciclo adaptativo envolve o cruzamento entre soluções candidatas e até mesmo mutações, simulando a teoria da evolução das espécies de Darwin (KOZA, 1992), (GOLDBERG, 1989), (NORVIG, RUSSEL, 2022) e (LINDEN, 2008).

Os AG não resultam em soluções ótimas para um determinado problema, mas, em geral, fornecem soluções aceitáveis, mesmo para problemas de grande complexidade. Apesar de a solução alcançada não ser necessariamente ótima, as grandes vantagens dos AG são a sua implementação simples, a sua boa performance na solução de problemas complexos, bem como na geração de soluções majoritariamente diferentes, o que traz uma sensação realística de inteligência, como, por exemplo, no caso do controle do comportamento de personagens autônomos em jogos, seja na definição de seu trajeto de navegação na tela ou mesmo na interação com o usuário ou com outros personagens autônomos.

Os AG vêm sendo utilizados geralmente para a definição de comportamentos de personagens autômatos em jogos eletrônicos (JUNIOR, LIMA, 2010), projetos de circuitos eletrônicos (VERTULO, 2023), no escalonamento de horários de professores (OLIVEIRA, MANZAN, 2022), na medicina, geologia, música, dentre outras diversas áreas do conhecimento.

A prototipação robótica ganhou grande impulso com o advento do Arduino, uma tecnologia de fácil utilização e com custo relativamente acessível para o desenvolvimento de artefatos robóticos (Arduino, 2024).

O campus Rio Pomba possui tradição nos estudos e desenvolvimento de artefatos robóticos, contando com o laboratório IF Maker como sua principal referência e suporte nessa área. O laboratório IF Maker disponibiliza peças e kits de desenvolvimento em Arduino Lego e Raspberry Pi, impressora 3D, além de ambiente de trabalho adequado.

Há poucas referências apontando para a utilização dos AG em artefatos robóticos, dentre elas a locomoção automática de um robô quadrúpede, onde os cromossomos, ou possíveis soluções, comandam de forma sequencial o acionamento dos diversos motores que movem as articulações desse artefato (SANTOS, DUARTE, 2023). Neste sentido, o projeto tem muito a contribuir com experimentos, inovações e avanços na área.

O projeto de iniciação científica em questão possui alto grau de ineditismo. Em meados de 2023, foi realizada uma busca no sítio do INPI (Instituto Nacional da Propriedade Industrial), na área de programas de computador, utilizando-se as palavras chaves “algoritmos genéticos”, e “robótica”, não sendo encontrado nenhum registro no sítio eletrônico do INPI com as palavras chaves do projeto descrito neste artigo. Apenas no Google acadêmico encontramos uma referência sobre a utilização de AG em artefatos Arduino (SANTOS, DUARTE, 2023).

MATERIAL E MÉTODOS

Inicialmente, o projeto envolveu o levantamento bibliográfico e estudos em AG e prototipação robótica. A seguir foram analisados os kits e peças Arduino e outras plataformas de prototipação robótica, como o Raspberry Pi, disponíveis no laboratórios IF Maker e no Laboratório de Montagem e Manutenção do campus Rio Pomba com vistas à prospecção de potenciais formas de aplicação.

Uma vez definidas as potenciais aplicações, os artefatos foram construídos, assim como o AG apropriado para cada um, incluindo a sua instalação no respectivo artefato robótico e testes.

Para apropriação da tecnologia em robótica pelos estudantes envolvidos no projeto, foi construído primeiramente um carrinho triciclo robótico em Arduino (figura 1). O carrinho anda para frente até encontrar um obstáculo, quando então adota outro sentido de forma aleatória, não empregando, portanto, nenhuma técnica de IA.

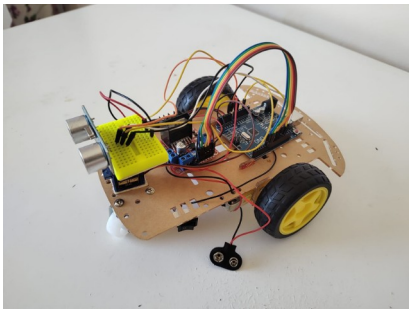

Figura 1. Triciclo robótico em Arduino.

Já para a prática dos conceitos em AG, os estudantes desenvolveram navegação autônoma para um NPC (personagem não jogável) em uma matriz de duas dimensões (figura 2). Conforme o descrito nas conclusões deste artigo, este experimento será a base para a instalação desse AG de navegação autônoma adaptado ao carrinho exibido na figura 1, com vistas à criação de um artefato robótico similar aos robôs de limpeza doméstica. A figura 2 exibe o trajeto de um NPC em uma matriz 10x10, percorrendo (tracejados) da posição inicial 0,0 até a extremidade oposta, na posição 9,9, desviando automaticamente dos obstáculos (asteriscos). Apesar de a trajetória nem sempre ser a ótima, ou seja, a menor distância possível, ela é sempre diversa em cada execução do AG, o que tornaria imprevisível a movimentação do NPC pelo cenário de um jogo, por exemplo, trazendo uma impressão mais realística para a movimentação do NPC.

```
[-][-][-][-][-][o][-][o][o][o]
[o][o][o][o][o][-][-][o][o][o]
[o][*][*][*][o][-][-][o][o][o]
[o][o][*][o][-][o][o][o][o][o]
[o][*][*][-][o][o][*][*][*][o]
[*][o][*][*][-][o][o][o][*][o]
[o][o][*][o][*][-][o][o][*][o]
[o][o][o][o][*][-][*][*][*][*]
[o][o][o][o][*][o][-][-][-][o]
[o][o][o][o][o][o][o][o][o][@]
```

Figura 2. Tela final da navegação de um NPC em uma malha 2D.

Resumidamente, a metodologia definida para o desenvolvimento dos artefatos robóticos controlados pelos AG foi assim definida:

- levantamento bibliográfico e estudos em AG, Arduino, Raspberry Pi e casos correlatos;

- levantamento da disponibilidade de kits, artefatos e peças eletrônicas, inclusive oriundos de lixo eletrônico;
- definição da aplicação (de acordo com a disponibilidade de material, vista no passo anterior), implementação e testes dos AG em ambiente virtual;
- triagem das peças e montagem dos artefatos robóticos para aplicação dos AG desenvolvidos no passo anterior;
- implantação dos AG nos artefatos desenvilvidos, testes e *tuning*;
- análise, reflexão e avaliação sobre os resultados obtidos; e
- documentação, registro e divulgação dos protótipos construídos.

Após a construção dos artefatos robóticos equipados com AG, estes foram testados na prática. Os testes envolvem a avaliação de comportamento, *tuning* ou ajustes dos parâmetros dos AG com vistas à sua otimização em performance, tais como o número de iterações necessário para se chegar a uma solução com qualidade, o tempo de processamento, uso de memória, os critérios de qualidade da potencial solução e/ou da população como um todo, número de indivíduos da população, a quantidade de passos de cada possível solução ou indivíduo (quantidade de alelos em cada cromossomo), percentual de descarte de indivíduos que não atingem qualidade mínima (extinção), metodologia de cruzamento, ou *crossover,* como, por exemplo, o cruzamento aleatório ou entre vizinhos, ou seja, indivíduos com pontuação semelhante, percentual de mutação, dentre inúmeras outras variáveis envolvidas na execução de um AG.

Aqueles artefatos que se mostrarem aptos para uso dos AG, com perspectivas de uso educacional, ou para a solução de problemas da comunidade, serão alvo de documentação, publicações, registro, no caso do software, ou patente, no caso do artefato como um todo.

Após a fase da definição da metodologia, estudos, desenvolvimentos e testes iniciais, partiu-se para o desenvolvimento do jogo da velha, conforme a metodologia descrita anteriormente.

RESULTADOS E DISCUSSÃO

Após o desenvolvimento do AG para navegação autônoma, ainda sem o uso de artefatos robóticos, desenvolveu-se um triciclo robótico em Arduino, sem o uso dos AG. Apenas um simples algoritmo de detecção colisão, via sensor, e redirecionamento aleatório.

Após esses testes, uniram-se as duas tecnologias, robótica e AG, para o desenvolvimento do jogo da velha em Arduino, utilizando-se, além da placa Arduino Uno, *leds* verdes e vermelhos para indicar a marcação das jogadas e um *display* numérico 3x3 para que o usuário humano indique a posição de sua jogada.

Primeiramente, conforme a metodologia definida para o projeto, o jogo da velha foi implementado e testado no mundo virtual. Assim, foi desenvolvido um programa em Java (figura 3) para a realização dos testes de calibração (*tuning*) dos parâmetros do AG.

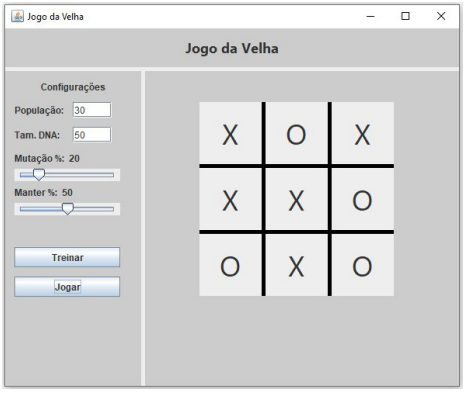

Figura 3. Protótipo do jogo da velha em Java para testes e definição dos parâmetros do AG.

O protótipo implementado em Java permite a definição do tamanho da população, ou seja, o número de indivíduos, ou soluções candidatas, o tamanho do cromossomo (número de jogadas por indivíduo), o percentual de mutação da população e o percentual e descarte de indivíduos menos aptos, ou seja, de qualidade inferior, conforme ilustra a figura 3.

Após os testes em java, o protótipo robótico em Arduino foi montado. Para o AG instalado nesse artefato, foram definidos os seguintes parâmetros (oriundos dos testes em java):

- tamanho da população: 50 indivíduos;
- tamanho do cromossomo: 30 alelos;
- percentual de remoção: 50%;

- metodologia de pontuação por qualidade (*fitness*): jogos entre dois indivíduos aleatórios, onde vitória soma 3 pontos, empate 1 ponto e derrota não soma pontos;
- metodologia de *crossover*: indivíduos consecutivos (vizinhos) na população ordenada por *fitness*;
- metodologia de mutação: 10% dos indivíduos da população sofrem mutação de forma aleatória, com alteração também aleatória em 10% de seus alelos;
- critério de parada da seleção natural: quando o "melhor" indivíduo, ou seja, o indivíduo com o maior *fitness* atingir 70% da pontuação máxima possível (3 vezes o número de embates);
- critério de parada ajustado: o critério anterior acrescido a um máximo de 20 *rounds* ou jogadas de cada indivíduo.

Detalhadamente, de acordo com a teoria dos AG, uma população é um conjunto de indivíduos. Cada indivíduo representa uma possível solução ao problema. No caso em pauta, chegou-se a um número de equilíbrio em 50 indivíduos. Esse número leva em conta a quantidade de iterações para o AG apurar um indivíduo com qualidade satisfatória em um tempo de execução aceitável, ou seja, confortável para um usuário humano.

Cada indivíduo da população é uma tentativa de solução do problema. No caso do jogo da velha, o problema é ganhar do opositor, logo cada indivíduo é uma sequência de possíveis jogadas no tabuleiro 3x3 do jogo da velha. No início do AG, cada jogada é definida aleatoriamente, dentre as nove jogadas possíveis no tabuleiro. Foram definidas no máximo 30 jogadas por indivíduo, pois nem sempre uma determinada jogada é possível, pois podem haver posições repetidas nos alelos, ou já preenchidas anteriormente pelo oponente, que pode ser outro NPC, na fase de treinamento, ou um oponente humano no embate "real". Por questões de performance não foram descartadas jogadas repetidas nos alelos dos indivíduos, pois tal descarte necessitaria de busca no vetor de alelos, o que causaria deterioração da performance.

Assim que os 50 indivíduos são gerados, ou seja, cada uma de suas 30 jogadas são definidas aleatoriamente, um laço de apuração de qualidade é executado, quando todos os indivíduos jogam entre si, cada um acumulando sua pontuação de qualidade ou *fitness.*

Após os embates ou *rounds*, os indivíduos são ordenados por critério de qualidade, e os 50% piores são eliminados (remoção). Os restantes cruzam entre si (*crossover*), restaurando os 50% eliminados no passo anterior. O cruzamento se dá entre vizinhos, ou seja, entre dois indivíduos em posições consecutivas no vetor de indivíduos ordenado por qualidade. Optou-se por essa metodologia para uma convergência mais rápida do processo evolutivo, ou seja, a criação de um indivíduo apto com um tempo de execução aceitável. Dois vizinhos têm pontuação semelhante, logo, a tendência é que produzam filhos parecidos. Caso o cruzamento fosse entre indivíduos de posição aleatória, a tendência seria a geração de filhos mais diversos, o que poderia atrasar a apuração em qualidade, como, de fato, foi observado nos testes em Java.

Após o cruzamento, 10% dos indivíduos sofrem mutação, ou seja, indivíduos no vetor são selecionados aleatoriamente. Os indivíduos assim selecionados sofrem alterações de valor em 10% de seus alelos de forma aleatória tanto em posição, quanto em conteúdo. De acordo com os testes realizados em java, percentuais muito maiores que 10% causavam atraso na convergência.

O processo iniciado no laço de apuração, ou seleção natural, persiste até que os critérios de parada sejam satisfeitos: indivíduo com pontuação maior ou igual a 70% da pontuação máxima possível, ou que 20 *rounds* sejam atingidos. O limite de 20 embates foi uma trava de segurança colocada à posteriori, depois que os testes mostraram que, em determinados casos, o AG não conseguia convergir para o critério de qualidade de 70% e entrava em *looping* infinito.

Resumidamente, o AG do jogo da velha foi assim definido:

- geração dos 50 indivíduos aleatoriamente;
- repita os passos seguintes:
- todos os indivíduos jogam entre si e, assim, acumulam pontos (*fitness*);
- ordenar os indivíduos em ordem de qualidade (*fitness*);
- remover os 50% piores;

- cruzar (crossover) os indivíduos, restaurando o número de indivíduos original;
- promover mutação em 10% dos indivíduos;
- enquanto o critério de parada não for alcançado (critério de qualidade mais a trava de segurança).

Esse AG do jogo da velha, conforme mencionado anteriormente, foi inicialmente implementado em Java e testado em computadores desktop e notebook com performance aceitável.

Porém, quando foi reescrito em C++, linguagem na qual o Arduino dá suporte, começaram a surgir problemas.

Primeiramente, o programa não coube na memória *flash* ROM do Arduino Uno. Esse problema foi resolvido reescrevendo o programa em C++ orientado a objetos para um programa em C estruturado, menos verboso, o que configura uma perda da qualidade do software inicialmente implementado, dado o decaimento do paradigma de programação.

O segundo problema foi o tempo proibitivo de execução. No Arduino Uno, o AG demorou, em média, 55 segundos, ou quase um minuto, para encontrar uma solução com a qualidade desejada, ou seja, um tempo demasiado alto para um oponente humano esperar.

Para resolver o problema, buscou-se, antes de mais nada, a realização de um *benchmarking* entre o computador utilizado pelos estudantes, o Arduino Uno e o Raspberry Pi versão 3, Model B. Esse *benchmarking* envolveu o levantamento do hardware subjacente a cada um desses aparelhos, bem como a execução de 1000 vezes o AG do jogo da velha, acumulando-se o tempo final dessas 1000 execuções, usando-se funções de temporização do C e Java.

O Arduino Uno é equipado com uma CPU ATmega386P, de 8 bits e 16MHz de frequência, com 32 registradores de 8 bits cada um, contando ainda com uma memória principal SRAM de 2KB e uma memória secundária de 32KB em *flash* ROM para o armazenamento do programa (ATMEL, 2024). O tempo de execução de 1000 AG do jogo da velha, na linguagem C, nessa arquitetura foi de 55.342.312 milissegundos, ou cerca de 922 minutos, ou ainda 15 horas e meia, com média de 55 segundos por vez.

Já o Raspberry Pi versão 3, Model B, conta com uma CPU Quad Core 1.2GHz Broadcom BCM2837 de 64 bits. O tempo de execução de 1000 AG, em linguagem C, nessa arquitetura foi de 6.025.213 milissegundos, ou cerca de 6.000 segundos, ou 6 segundos por vez. Aproximadamente 10 vezes mais rápido do que o Arduino Uno. Esses 10 segundos se mostraram toleráveis para o treinamento dos cromossomos, uma vez que, durante o processo de apuração, os leds piscam mostrando alguns indivíduos, selecionados no começo e no fim da apuração, jogando entre si.

O computador de desenvolvimento dos AG foi um notebook com uma CPU Intel Core i3-9100F, 64 bits, Quad-Core. O tempo de execução de 1000 AG em Java nessa arquitetura foi de 8.321 milissegundos, ou cerca de 8 segundos, ou seja, cerca de 6600 vezes mais rápido do que o Arduino Uno.

Dados os tempos de execução obtidos e as diferentes configurações de hardware, optou-se então, pela migração do jogo da velha para a plataforma Raspeberry Pi (figura 4). Tal migração não foi problemática, dada a compatibilidade do Raspberry Pi com os elementos eletrônicos utilizados no Arduino Uno, bem como o suporte à linguagem C, ou seja, não foram necessárias alterações no programa nem nos componentes eletrônicos utilizados.

O artefato final produzido em Raspberry Pi, o jogo da velha, ao ser executado, mostra o processo de apuração dos indivíduos através do acendimento dos leds. Após o término de uma partida, o AG é treinado novamente para se adaptar ao jogador humano, aumentando gradativamente a qualidade de suas jogadas e instigando a curiosidade dos alunos.

O protótipo ilustrado na figura 4 será encapsulado em um acabamento feito com MDF, para torná-lo mais atrativo esteticamente, além de contar com um *display* numérico para o jogador humano indicar a posição da sua jogada.

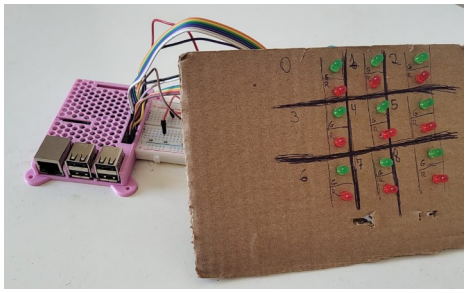

Figura 4. protótipo robótico do jogo da velha na plataforma Raspberry Pi em fase de testes.

CONCLUSÃO

Apesar de o Arduino Uno ser uma plataforma que vem ganhando popularidade devido a sua simplicidade, disponibilidade e facilidade no uso, ela não foi capaz de executar um AG para jogo da velha em tempo hábil para produzir um jogador autônomo com a qualidade desejada, o que causou a migração para a plataforma concorrente: o Raspberry Pi.

O Raspberry Pi, não tão popular quanto o Arduino, conta com um processador mais potente, que foi capaz de executar a tarefa de seleção para o jogo da velha em tempo confortável para o oponente humano.

Como a migração de uma plataforma para a outra foi facilitada devido à compatibilidade de linguagens de programação e equipamentos eletrônicos, não houve atraso substancial no desenvolvimento do projeto.

Os problemas encontrados, entretanto, subsidiaram uma maior apropriação das tecnologias, em termos de entendimento e testes de performance sobre o hardware subjacente às plataformas, evidenciando que o Arduino Uno não tem capacidade de processamento ideal para a execução do AG proposto e, provavelmente, para outros algoritmos oriundos da inteligência artificial.

O artefato produzido, o jogo da velha, se mostrou com potenciais de instigar a curiosidade dos alunos e despertar o interesse tanto em robótica quanto na inteligência artificial, tornando-se ferramenta prática, lúdica, didática e divertida para o ensino dessas disciplinas, abrindo espaço para a gamificação aplicada ao ensino.

Como extensões, estão previstos o desenvolvimento de um robô com navegação autônoma em plano 2D, e o desenvolvimento do processo de andar para um robô quadrúpede, similar ao visto em (SANTOS, DUARTE, 2023), além do controle de robôs de competição, como o futebol cibernético, por exemplo. Além disso, outras técnicas e algoritmos da IA serão explorados, tais como os algoritmos de enxame, navegação autônoma específica para robôs domésticos e aprendizado de máquina.

Para o desenvolvimento do robô quadrúpede, primeiramente um modelo 3D será criado e inserido na *game engine* Unity, quando então será testado. Uma vez funcionando adequadamente no mundo virtual, o modelo será materializado em uma impressora 3D, disponível no laboratório IF Maker, para então ser desenvolvido roboticamente em Raspberry Pi.

AGRADECIMENTOS

Direção de Pesquisa do campus Rio Pomba do Instituto Federal do Sudeste de Minas Gerais.

Laboratório IF Maker do campus Rio Pomba, do Instituto Federal do Sudeste de Minas Gerais.

CNPq.

REFERÊNCIAS

Arduino. **Explore our learning solutions**. Disponível em: https://www.arduino.cc/education. Acessado em 01 de março de 2024.

ATMEL. **ATmega328P Datasheet**. Disponível em: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. Acessado em 10 de março de 2024.

GOLDBERG, David E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. EUA: Addison-Wesley. 1989.

JUNIOR, Gilberto Timótheo; LIMA, Sérgio Muinhos Barroso. **Algoritmos Genéticos Aplicados a Jogos Eletrônicos**. Revista Eletrônica da Faculdade Metodista Granbery. 2010.

KOZA, J.R. **Genetic Programming. On the Programming of Computers by Means of Natural Selection**. [S.l.]: MIT Press. 1992.

LINDEN, Ricardo. **Algoritmos Genéticos - uma importante ferramenta da inteligência computacional**. Brasport. 2008.

NORVIG, Peter, RUSSEL, Stuart. **Inteligência Artificial - Uma Abordagem Moderna**. LTC. 2022.

OLIVEIRA, A. L. de; MANZAN, J. R. G.. **Implementação de algoritmos genéticos para geração de grade horária de aula: caso IFTM – campus avançado Uberaba parque tecnológico**. Brazilian Journal of Health Review. 2022.

SANTOS, Pedro Matheus; DUARTE, Vitor Alves. **Aprendizado de Marchas para um Robô Quadrúpede Utilizando Algoritmo Genético Multiobjetivo**. Trabalho de Conclusão do Curso de Engenharia Mecatrônica da Universidade de Brasília. Disponível em: https://bdm.unb.br/handle/10483/21326. Acesso em 11 de maio de 2023.

VERTULO, Rodrigo Cesar. **Projeto de Circuitos Eletrônicos com Algoritmo Genético**. Disponível em: <http://labdeeletronica.com.br/noticias/projeto-de-circuitos-eletronicos-com-algoritmo-genetico/>. Acessado em: 11/05/2023.