Trabalho de ICC1 - Algoritmo Genético

Enzo N. Sedenho e Pedro Augusto M. Delgado

BCC 022 - Junho 2022

1 Resumo

O presente trabalho consiste em um algoritmo genético simplificado, escrito em C, cujo objetivo é encontrar uma raiz inteira exata no intervalo [-255, 255] para uma equação de até 5º grau. A equação é informada pelo usuário, seguindo o modelo abaixo.

$$y = ax^5 + bx^4 + cx^3 + dx^2 + ex + f$$

Os coeficientes a, b, c, d, e, f são informados pelo usuário, assim como o tamanho das populações, o número máximo de gerações e o coeficiente de elitismo. Caso o programa encontre uma raiz inteira exata para a equação, o seu funcionamento é encerrado logo após a raiz ser retornada ao usuário. Caso contrário, o programa retorna a melhor resposta (mais próxima de uma raiz) encontrada.

1.1 Coeficiente de elitismo

O *coeficiente de elitismo* é um parâmetro criado para controlar a elitização das populações geradas pelo algoritmo. Esse parâmetro é um número inteiro positivo que indica o limite de populações elitizadas (com indivíduos muito próximos entre si, mas todos diferentes de uma raiz da equação analisada) a serem geradas pelo algoritmo. Quando a quantidade de populações elitizadas geradas pelo programa igualar-se ao coeficiente de elitismo, uma nova população será gerada randomicamente.

1.2 Interface inicial

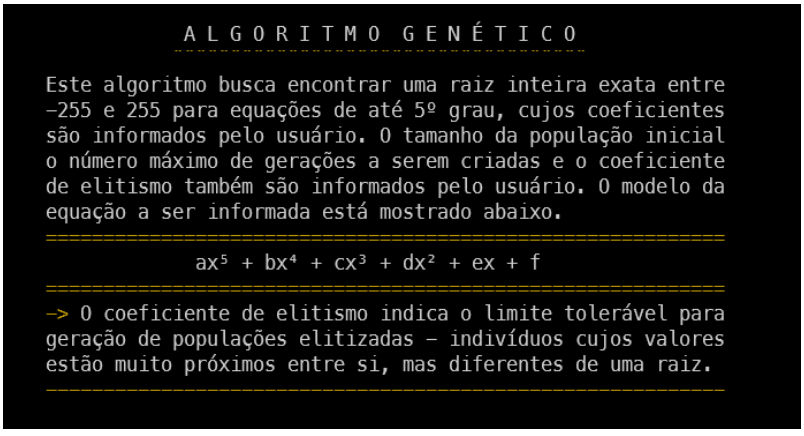

Figura 0: Interface inicial do programa em terminal do Linux, Ubuntu 20.0.04. [Fonte: Autores]

2 Funcionamento do programa

2.1 Variáveis e constantes

- *'tam'* , *'gen_max'* e *'elitismo_max'* são parâmetros informados pelo usuário. *'tam'* indica o tamanho da população, *'gen_max* indica o número máximo de gerações, limitando o funcionamento do programa e *'elitismo_max'* é o coeficiente de elitismo;
- *'a', 'b', 'c', 'd', 'e', 'f'* são coeficientes da equação a ser analisada;
- Os vetores *'gena'* e *'genb'* são os *vetores-população*, que armazenam os indivíduos de cada geração. O vetor *'genb'* é utilizado como vetor auxiliar durante a seleção e o cruzamento;
- O vetor *'prob'* armazena as probabilidades (ou scores) dos indivíduos serem selecionados;
- Os vetores *'melhores_x'* e *'melhores_y'* armazenam os melhores indivíduos de cada geração e suas equações, respectivamente;
- *'prob_mutacao'* indica a chance do bit analisado sofrer mutação;
- *'masc'* é utilizada como máscara para realizar operações bit-a-bit nas etapas de cruzamento e mutação;
- *'i', 'j','gen'* e *'elitismo'* são contadores. *'gen'* é utilizado para demarcar a que geração cada população pertence e *'elitismo'* indica a quantidade de populações elitizadas;
- *'melhor_x'* e *'melhor_y'* armazenam o melhor indivíduo da geração e sua equação, respectivamente;
- *'mais_proximo_x'*, *'mais_proximo_y'* e *'mais_proximo_gen'* armazenam o melhor indivíduo de toda a execução do programa, sua equação e sua geração, respectivamente;
- *'media'* é utilizada para calcular a média aritmética da população.

2.2 Passo-a-passo

Aqui, apresenta-se o passo-a-passo detalhado do programa para melhor compreensão da lógica do algoritmo. A referência das linhas citadas pode ser encontrada no arquivo "AG.c"anexado ao documento.

Linhas 1 a 92 – INICIANDO O PROGRAMA
O programa inicia-se com uma interface explicativa sobre o objetivo do algoritmo e solicita ao usuário os valores das constantes que limitam o funcionamento do programa. O algoritmo detecta valores inválidos para os parâmetros em questão, solicitando novamente a inserção dos dados. Além disso, inicializa as variáveis descritas no tópico anterior.

Linhas 93 a 102 – GERANDO A POPULAÇÃO INICIAL
Por meio de um [for], preencheu-se todo o vetor *'gena'* com números aleatórios entre 0 e 255 (8 bits). Logo após isso, utilizou-se um [if] para definir se o número será positivo ou negativo e, por meio da geração de números aleatórios, o sinal é atribuído aos indivíduos.

Linhas 103 e 104 – DEFININDO UM LAÇO PRINCIPAL
Por meio de um [while], define-se um limite para o funcionamento do programa, cuja variável de parada é informada pelo usuário (*'gen_max'*). Excepcionalmente, o programa é encerrado ao encontrar uma raiz inteira exata para a equação informada.

Linhas 106 a 116 – INICIALIZANDO OS MELHORES INDIVÍDUOS
Inicializam-se *'melhor_x'*, *'melhor_y'*, *'mais_proximo_x' e 'mais_proximo_y'* com os dados relativos ao primeiro indivíduo da geração. Essas informações são modificadas se o algoritmo encontrar um indivíduo melhor.

Linhas 117 a 141 – ENCONTRANDO OS MELHORES INDIVÍDUOS
Em um [for], calculam-se as equações geradas por cada indivíduo do vetor *'gena'*. Caso o módulo da equação

seja mais próximo de 0 do que o módulo de 'melhor_y', a equação está mais próxima de 0 e, consequentemente, o indivíduo está mais próximo de ser raiz. Portanto, caso isso ocorra, atribui-se a *'melhor_x'* o indivíduo analisado, e a *'melhor_y'* a equação analisada. Logo após isso, verifica-se se o módulo de *'melhor_y'* é menor que o módulo de *'mais_proximo_y'*. Caso isso seja ocorra, o melhor indivíduo da geração chegou mais perto de ser raiz da equação que o melhor indivíduo global. Dessa forma, atribui-se as informações do indivíduo analisado às informações relativas ao melhor indivíduo global.

Linhas 142 a 157 – CALCULANDO AS PROBABILIDADES
O melhor indivíduo recebe, arbitrariamente, 99% de chance de ser selecionado. As demais probabilidades (ou scores) são calculadas por meio de uma regra de 3 (a razão entre *'melhor_y'* e o *y* analisado), multiplicada por 990 (a escala auxilia a manipulação com inteiros). Amortizaram-se as probabilidades por meio de uma raiz quarta (do numerador e do denominador) com o objetivo de reduzir a discrepância entre a maior probabilidade e as demais. Caso a probabilidade de um indivíduo seja menor do que 1, ela recebe 1, para que todos os indivíduos tenham alguma chance de serem selecionados.

Linhas 158 a 172 – CALCULANDO O ELITISMO
Calcula-se a média da população analisada e verifica-se se o módulo da diferença entre a média e o melhor indivíduo é menor ou igual a 1. Caso isso ocorra, o algoritmo interpreta que a população está elitizada, ou seja, os indivíduos estão convergindo para um valor específico, mas diferente de uma raiz. Assim, 'elitismo' é incrementado.

Linhas 173 a 207 – IMPRIMINDO RELATÓRIOS DAS GERAÇÕES
Imprimem-se na tela as informações relativas a cada geração para visualizar a mudança dos indivíduos e a evolução do algoritmo. Caso *'elitismo'* seja diferente de 0, ele é mostrado ao usuário, bem como *'elitismo_max'*, de modo a indicar a ocorrência de populações elitizadas em comparação com o coeficiente de elitismo.

Linhas 208 a 228 – ENCONTRANDO A RAIZ
Verifica-se se *'melhor_y'* é 0. Caso isso ocorra, o melhor indivíduo é raiz da equação e, assim, o programa imprime a raiz encontrada na tela e realiza um [break] para encerrar a execução do laço principal.

Linhas 229 e 230 – INICIANDO UMA NOVA GERAÇÃO
Inicia-se uma nova população/geração. O contador *'gen'* é incrementado.

Linhas 232 à 243 – SELECIONANDO OS INDIVÍDUOS
Por meio de um [for], preenche-se a primeira metade do vetor '*genb*' com indivíduos de *'gena*'. Para cada indivíduo, avalia-se se um número aleatório gerado pela função [rand()] é menor ou igual sua probabilidade de seleção (registrada no vetor '*prob*'). Caso isso ocorra, o indivíduo é copiado em '*genb*', e sua probabilidade recebe -1, para que o mesmo indivíduo não possa ser selecionado mais de uma vez. Caso todo o vetor '*gena*' tenha sido percorrido e a metade de *'genb'* não tenha sido preenchida completamente, o contador *'i'* recebe 0, para que o vetor '*gena*' seja lido novamente.

Linhas 244 a 281 – REALIZANDO OS CRUZAMENTOS
Utiliza-se um [for] que é interrompido quando todos os filhos são gerados. A máscara é inicializada como 1, e, por meio de um bitshift, desloca-se sua representação binária ao fim de cada realização do laço. Inicializa-se o filho como 0 e, bit-a-bit, forma-se o valor do indivíduo. Para o cruzamento, trata-se, em primeiro lugar, os pais que possuem ao menos um bit em comum por meio de um [if]. Para esse caso, utiliza-se um [for] que é executado 9 vezes (8 bits + o sinal). Se os bits analisados dos pais forem iguais, adiciona-se (pai & máscara) ao filho. Caso os pais possuam o bit analisado igual a 0, o filho recebe 0. Caso contrário, o filho recebe 1. Para pais com bits diferentes, o filho possui 50% de chance de receber cada bit (0 ou 1) - chance gerada por meio de um [rand()]. Uma vez que o laço é realizado 8 vezes (j == 8), verifica-se o sinal dos pais. Para pais ambos negativos, o filho será negativo. Para pais positivos, o filho será positivo. Para pais com sinais opostos, o filho possui 50% de chance de ser negativo ou positivo - chance gerada por meio de um [rand()]. Em segundo caso, para pais totalmente diferentes, ou seja, com nenhum bit coincidente, o filho será a média aritimética entre os dois pais.

Linhas 282 a 31 – APLICANDO A MUTAÇÃO
Utiliza-se um [for] que percorre o vetor '*genb*' do primeiro ao último filho. Para cada filho, a máscara é inicializada como 1 e a probabilidade de mutação como 512. Por meio de um [if], avalia-se se um número aleatório gerado pela função [rand()] é menor ou igual a probabilidade de mutação do indivíduo analisado. Caso isso ocorra, o indivíduo sofrerá mutação. Para isso, verifica-se se o &-bitwise entre o módulo do indivíduo e a mascara é 0, ou seja, o indivíduo não possui aquele bit. Caso isso ocorra, o indivíduo receberá o bit. Caso contrário, o indivíduo possui aquele bit, e o bit em questão será subtraído dele. Para filhos negativos, utiliza-se o seu simétrico para facilitar as operações bitwise com indivíduos negativos. Ao final da mutação, dividi-se a probabilidade de mutação por 2 por meio de um bitshift, para que a probabilidade dos bits menos significativos seja maior que a dos bits mais significativos. Além disso, realiza-se um bitshift na máscara para analisar o próximo bit.

Linhas 312 a 322 – AVALIANDO O ELITISMO
Avalia-se o contador '*elitismo'*. Caso 'elitismo' seja igual ao coeficiente de elitismo digitado pelo usuário, a geração é inteiramente substituída por números aleatórios e 'elitismo' é reinicializado como 0.

Linhas 323 a 354 – ENCERRANDO A ITERAÇÃO NO LAÇO PRINCIPAL
O vetor '*gena'* recebe os elementos do vetor '*genb'* para dar início a uma nova iteração do laço principal. Caso o contador '*gen*' seja maior ou igual a *'gen_max'*, o algoritmo encerra o laço principal.

Linhas 355 a 375 – IMPRIMINDO O RELATÓRIO FINAL
A última etapa do programa é imprimir todos os melhores indivíduos de cada geração e suas equações, para acompanhamento da evolução do algoritmo. Após isso, o programa se encerra.

2.3 Recursos extra

Para melhorar a leitura dos dados informados pelo usuário, o programa identifica valores inválidos para os parâmetros de tamanho da população, número máximo de gerações e coeficiente de elitismo conforme a Figura 9.

```
>> Informe o tamanho da população (nº par): 1
Atenção! Informe um valor válido: 0
Atenção! Informe um valor válido: -1
Atenção! Informe um valor válido: 4
>> Informe o número máximo de gerações: 0
Atenção! Informe um valor válido: -1
Atenção! Informe um valor válido: 8
>> Informe o coeficiente de elitismo: -2
Atenção! Informe um valor válido: 0
Atenção! Informe um valor válido: 4
>> Informe o coeficiente a:
```

Figura 9: Retorno do programa após a leitura de valores inválidos em terminal do Linux, Ubuntu 20.0.04. [Fonte: Autores]

A fim de analisar o comportamento das populações com o passar das gerações, o programa imprime, em forma de tabela, os indivíduos, seus scores e suas equações, para cada geração conforme a Figura 10.

```
Informações sobre a geração 0
----------------------------------

 Indivíduo | Score |   Valor na equação
-----------+-------+--------------------
 [00]  103 | 33.1% |         10797160495
 [01] -105 | 31.2% |        -13605643073
 [02]  -81 | 43.0% |         -3784436729
 [03]   74 | 50.4% |          2006225166
 [04]  -41 | 99.0% |          -135165569
 [05] -186 | 15.3% |       -230953657574
 [06] -242 | 11.1% |       -853906982750
 [07]  227 | 12.2% |        584070083139
 [08]  124 | 26.1% |         27647846116
 [09]   84 | 42.9% |          3829410556

-> O melhor indivíduo é x = -41 com y = -135165569.
```

Figura 10: Retorno do programa sobre a geração 0 do caso 3 em terminal do Linux, Ubuntu 20.0.04. [Fonte: Autores]

Além disso, há casos em que o elitismo é mostrado ao usuário, de modo a acompanhar as populações elitizadas, inclusive alertando quando o algoritmo gera uma nova população conforme a Figura 11.

```
Informações sobre a geração 10
----------------------------------

 Indivíduo | Score |   Valor na equação
-----------+-------+--------------------
 [00]    0 | 99.0% |                 -8
 [01]    1 | 68.4% |                -35
 [02]   -1 | 96.1% |                 -9
 [03]    0 | 99.0% |                 -8
 [04]    0 | 99.0% |                 -8
 [05]    0 | 99.0% |                 -8
 [06]   -9 |  9.3% |            -100385
 [07]   -6 | 14.9% |             -15554
 [08]    0 | 99.0% |                 -8
 [09]    0 | 99.0% |                 -8

-> O melhor indivíduo é x = 0 com y = -8.
-> A população está elitizada em 6/6
ELITISMO DETECTADO! Uma nova população será gerada!
```

Figura 11: Retorno do programa sobre a geração 10 do caso 3 em terminal do Linux, Ubuntu 20.0.04. [Fonte: Autores]

3 Casos de teste

3.1 Caso 1

Para o caso 1, deve-se informar as entradas: **20 80 3 1 1 1 1 1 1**. A função informada será dada pelo gráfico mostrado na Figura 1 (em destaque um ponto cuja ordenada é uma raiz da equação). Nesse caso, o algoritmo é capaz de detectar a raiz, já que é inteira e está no intervalo [-255, 255]. A saída esperada para esse caso é exibida na Figura 2.

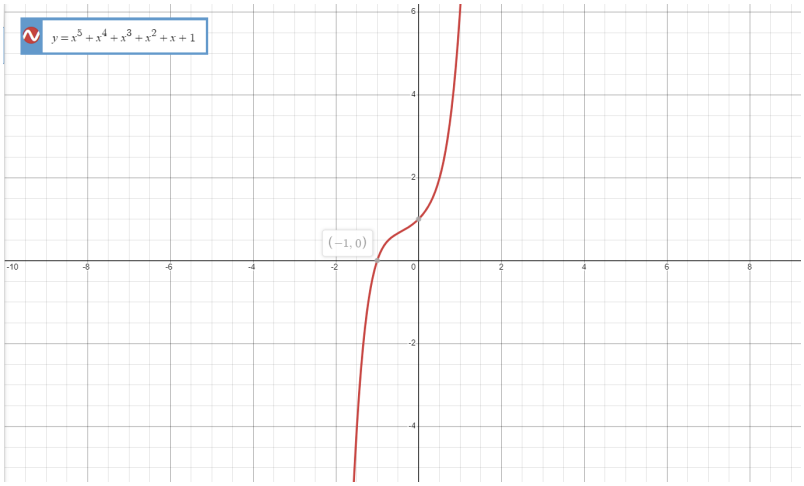

Figura 1: Função informada no caso 1, com destaque para o ponto (-1,0). [Fonte: desmos.com]

```
============================================================
Em  3 gerações, o algoritmo encontrou que x = -1 é raiz da
equação informada: 1x⁵ + 1x⁴ + 1x³ + 1x² + 1x + 1
============================================================

Evolução das gerações
---------------------

 Gen | Melhor x | Valor na equação
-----+----------+------------------
   0 |      -13 |          -344772
   1 |       -2 |               -21
   2 |       -1 |                 0
```

Figura 2: Saída para o caso 1 em terminal do Linux, Ubuntu 20.0.04. [Fonte: Autores]

3.2 Caso 2

Para o caso 2, deve-se informar as entradas: **18 80 4 1 -110 4825 -105500 1150000 -5000000**. A função informada será dada pelo gráfico mostrado na Figura 3 (em destaque pontos cujas ordenadas são raizes da equação). Nesse caso, o algoritmo é capaz de detectar alguma dessas raizes, já que são inteiras e estão no intervalo [-255, 255]. A saída esperada para esse caso é exibida na Figura 4.

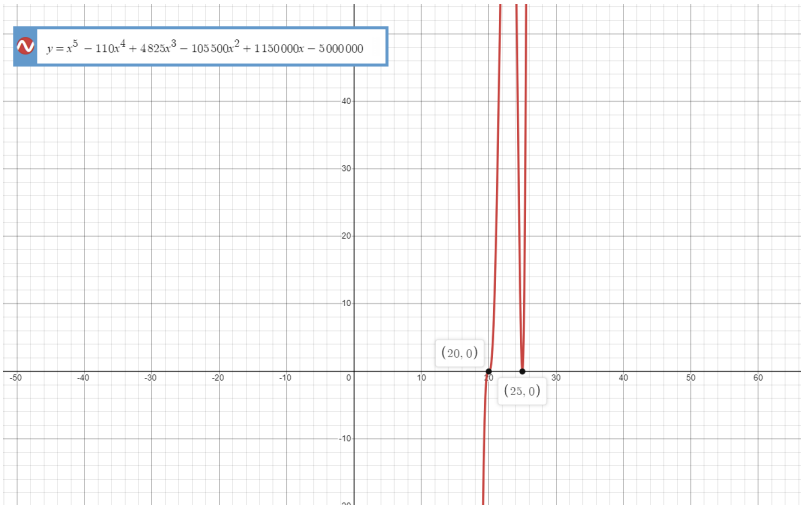

Figura 3: Função informada no caso 1, com destaque para os pontos (20,0) e (25,0). [Fonte: desmos.com]

```
============================================================
Em 12 gerações, o algoritmo encontrou que x = 20 é raiz da
equação informada: 1x⁵ + -110x⁴ + 4825x³ + -105500x² + 1150000x + -5000000
============================================================

Evolução das gerações
------------------------

 Gen | Melhor x | Valor na equação
-----+----------+------------------
   0 |       27 |             1372
   1 |       18 |             -392
   2 |       18 |             -392
   3 |       18 |             -392
   4 |       18 |             -392
   5 |       26 |              216
   6 |       19 |              -36
   7 |       19 |              -36
   8 |       19 |              -36
   9 |       21 |               16
  10 |       21 |               16
  11 |       20 |                0
```

Figura 4: Saída para o caso 2 em terminal do Linux, Ubuntu 20.0.04. [Fonte: Autores]

3.3 Caso 3

Para o caso 3, deve-se informar as entradas: **10 80 6 1 -7 -7 -7 -7 -8**. A função informada será dada pelo gráfico mostrado na Figura 5 (em destaque ponto (8,0) cujas ordenada é raiz da equação). Nesse caso, o algoritmo é capaz de detectar essa raiz, já que são inteiras e estão no intervalo [-255, 255]. A saída esperada para esse caso é exibida na Figura 6.

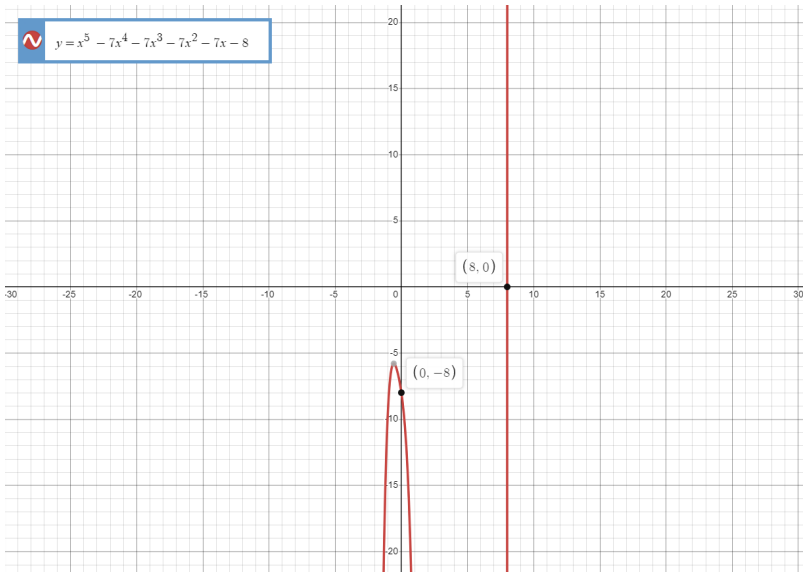

Figura 5: Função informada no caso 3, com destaque para os pontos (0,-8) e (8,0). [Fonte: desmos.com]

```
============================================================
Em 17 gerações, o algoritmo encontrou que x = 8 é raiz da
equação informada: 1x⁵ + −7x⁴ + −7x³ + −7x² + −7x + −8
============================================================

Evolução das gerações
-------------------------

 Gen | Melhor x | Valor na equação
-----+----------+-----------------
   0 |      -41 |       -135165569
   1 |        5 |            -2343
   2 |       -3 |             -671
   3 |       -2 |             -110
   4 |       -2 |             -110
   5 |        0 |               -8
   6 |        0 |               -8
   7 |        0 |               -8
   8 |        0 |               -8
   9 |        0 |               -8
  10 |        0 |               -8
  11 |       11 |            48315
  12 |        5 |            -2343
  13 |        4 |            -1364
  14 |       -1 |               -9
  15 |        0 |               -8
  16 |        8 |                0
```

Figura 6: Saída para o caso 3 em terminal do Linux, Ubuntu 20.0.04. [Fonte: Autores]

Nesse caso, o algoritmo converge para $x = 0$, mas o mecanismo do elitismo consegue gerar novas populações que convergem para $x = 8$, que é uma raiz da equação.

3.4 Caso 4

Para o caso 4, deve-se informar as entradas: **16 30 5 1 0 0 0 0 -24**. A função informada será dada pelo gráfico mostrado na Figura 7 (em destaque um ponto cuja ordenada é raiz da equação). Nesse caso, o algoritmo não é capaz de detectar essas raiz, já que não é inteira, apesar de estar no intervalo [-255, 255]. A saída esperada para esse caso é exibida na Figura 8.

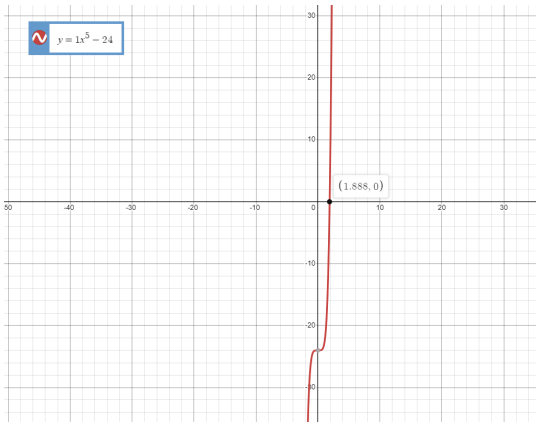

Figura 7: Função informada no caso 4, com destaque para o ponto (1.888, 0). [Fonte: desmos.com]

```
==========================================================
Em 30 gerações, o algoritmo não encontrou uma raiz inteira
entre -255 e 255 para a equação informada. Contudo, o valor
mais próximo de uma raiz dessa equação foi encontrado, pri-
meiramente na geração 3, sendo x = 2 e y = 8.
==========================================================

Evolução das gerações
----------------------

 Gen | Melhor x | Valor na equação
-----+----------+-----------------
   0 |       27 |         14348883
   1 |        3 |              219
   2 |        2 |                8
   3 |        2 |                8
   4 |        2 |                8
   5 |        2 |                8
   6 |        2 |                8
   7 |        2 |                8
   8 |        2 |                8
   9 |        2 |                8
  10 |        2 |                8
  11 |        2 |                8
  12 |        2 |                8
  13 |       19 |          2476075
  14 |        0 |              -24
  15 |        1 |              -23
  16 |        1 |              -23
  17 |        1 |              -23
  18 |        1 |              -23
  19 |        1 |              -23
  20 |        1 |              -23
  21 |        2 |                8
  22 |        1 |              -23
  23 |       -9 |           -59073
  24 |       -2 |              -56
  25 |       -2 |              -56
  26 |       -1 |              -25
  27 |        1 |              -23
  28 |        1 |              -23
  29 |        1 |              -23
```

Figura 8: Saída para o caso 4 em terminal do Linux, Ubuntu 20.0.04. [Fonte: Autores]