Algoritmos Genéticos

Universidade Federal de Santa Catarina
Departamento de Automação e Sistemas - 2020/01
Prof. Dr. Eric Aislan Antonelo, Prof. Dr. Jomi Fred Hubner, Alexander Silva Barbosa

1 Introdução

Durante os anos 50 e 60, diversos cientistas se dedicaram a estudar, de forma independente, os sistemas evolutivos, tendo como premissa a ideia de que evolução poderia ser utilizada como uma ferramenta de otimização para problemas de engenharia. A ideia central para todos esses sistemas era evoluir uma população de soluções candidatas para um determinado problema, usando operadores inspirados na seleção natural (o mais adatapado sobrevive) e variação genética natural (combinação dos melhores genótipos e mutação).

Nos anos 60, Rechenberg introduziu **estratégias evolutivas**, um método para otimizar parâmetros de valor real para dispositivos como aerofólios. Fogel, Owens, and Walsh em 1966 desenvolveram **programação genética**, uma técnica onde as soluções candidatas são representadas como máquinas de estado finito, que são então evoluídas aleatoriamente efetuando mutações em seu diagrama de estados e transições, de forma a selecionar o melhor candidato. Juntos, os algoritmos de estratégias evolutivas, de programação genética e de algoritmos genéticos (*Genetic Algorithms* - GAs) formam a base da Computação Evolutiva (ou Evolucionária).

Os Algoritmos Genéticos (AGs) foram inventados por John Holland na década de 1960. Diferentemente de estratégias evolutivas e programação genética, o objetivo original de Holland não era projetar algoritmos para resolver problemas específicos, mas sim estudar formalmente o fenômeno da adaptação como ocorre na natureza e desenvolver maneiras pelas quais os mecanismos de adaptação natural pudessem ser importados para os sistemas computacionais. O livro *Adaptation in Natural and Artificial Systems* de Holland, de 1975, apresentou o algoritmo genético como uma abstração da evolução biológica e forneceu uma abordagem teórica para adaptação sob AGs.

O método original de algoritmos genéticos, apresentado por John Holland, consiste em iniciar com uma população de *cromossomos*, onde um indivíduo é representado por um cromossomo (por exemplo, sequências de uns e zeros, ou "bits"), e chegar a uma nova população usando uma espécie de **seleção natural** através de operadores genéticos de seleção, mutação, cruzamento e inversão.

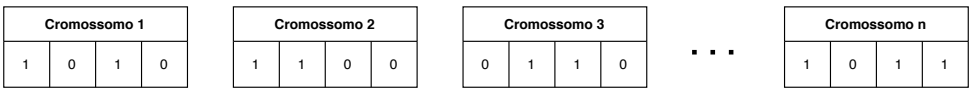
Cada cromossomo é formado por uma sequência de *genes* (bits por exemplo). O operador de **seleção** escolhe os indivíduos (cromossomos) na população que poderão se reproduzir, de forma que na média, os indivíduos mais aptos gerem mais descendentes do que os menos aptos. O operador de **cruzamento** (*crossover*) troca partes de um cromossomo com outro, imitando a recombinação de dois organismos de apenas um cromossomo. A mutação altera de forma aleatória alguns genes dos cromossomos. A inversão inverte a ordem de uma seção contígua do cromossomo, reorganizando assim a ordem em que os genes são agrupados.

As diferenças e limites entre estratégias evolutivas, programação genética e algoritmos genéticos vêm diminuindo ou nem sempre são claras. O termo algoritmo genético tem sido usado em diferentes contextos, desviando por vezes do conceito inicial apresentado por John Holland.

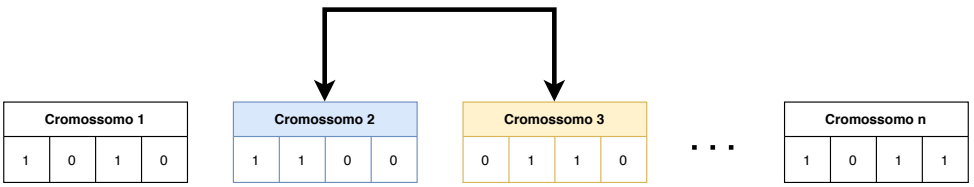
2 Elementos de Algoritmos Genéticos

A maioria dos métodos chamados "GAs" possuem pelo menos os seguintes elementos em comum: uma população de indíviduos (cromossomos), seleção conforme aptidão (*fitness function*), operadores genéticos de cruzamento para produzir novos descendentes e mutação de um novo descendente. Inversão é raramente usado.

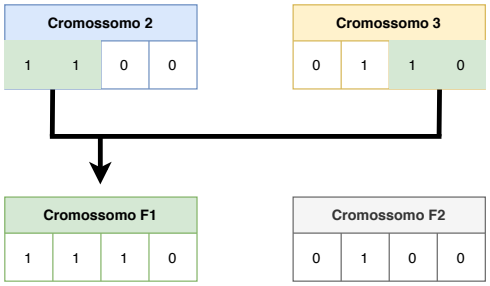
- **Uma população de cromossomos:** Os cromossomos em uma população de GA geralmente são formados por cadeias de bits. Cada locus no cromossomo tem dois alelos possíveis: 0 e 1. Cada cromossomo pode ser visto como um ponto no espaço de busca das soluções candidatas. Abaixo temos uma população de n indivíduos, onde cada um tem um cromossomo de quatro bits.



- **Seleção com base em uma função objetivo de aptidão (fitness):** Esta função é responsável por dar uma nota ou pontuação (fitness) para cada cromossomo na população, oferecendo assim uma forma do GA diferenciar um cromossomo mais apto de um menos apto. Esta pontuação de aptidão está diretamente relacionada ao quão bem aquele cromossomo resolve o problema ao qual se busca uma solução. O operador de seleção, responsável por selecionar os indivíduos da população que irão reproduzir, utiliza a pontuação de cada cromossomo como base. Quando mais apto um cromossomo está para a solução, maiores são suas chances de reproduzir, garantindo assim que toda a população evolua na direção de encontrar a melhor solução.

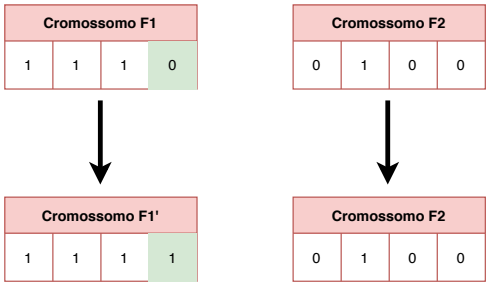


- **Cruzamento com um ponto de corte para produzir descendentes:** Este operador escolhe aleatoriamente um locus no cromossomo (chamado ponto de corte) e troca as subsequências antes e depois desse locus entre dois cromossomos para criar dois descendentes. Por exemplo, as cadeias 10000100 e 11111111 podem ser cruzadas após o terceiro locus para produzir os dois descendentes 10011111 e 11100100. O operador de cruzamento imita aproximadamente a recombinação biológica entre dois organismos de cromossomo único (haplóide).



- **Mutação aleatória de novos descendentes:**

  Este operador inverte um bit (gene) selecionado aleatoriamente de um cromossomo. Ou seja, o locus da mutação é sorteado de maneira uniforme, e o gene correspondente é alterado (0 vira 1, ou vice-versa). Por exemplo, a cadeia 1110 pode sofrer uma mutação no quarto e último locus (selecionado aleatoriamente), tornando-se 1111. Nem todos indíviduos sofrem mutação. A probabilidade de um indivíduo descendente, após o cruzamento, ser selecionado para mutação é dada pelo parâmetro p_m.



3 Exemplos de Funções de Aptidão

Uma aplicação comum de GAs é na otimização de funções, onde se deseja encontrar um conjunto de valores de parâmetros que maximize uma dada função complexa de múltiplos parâmetros. Por exemplo, suponha que desejamos maximizar a função real de uma dimensão:

$$f(y) = y + |sin(32y)|, \quad 0 \leq y < \pi \tag{1}$$

Para este problema, as soluções candidatas são valores de y, os quais podem ser codificados como cadeias de bits representando números reais. A função objetivo neste caso é uma função que decodifica uma dada cadeia de bits x em um número real y, e então avalia o valor da função em (1) neste número real. A aptidão de uma cadeia de bits é o valor da função naquele ponto. Ao aumentarmos a aptidão média dos indivíduos de uma população através de um AG, estamos também maximizando a função real de uma dimensão em (1).

Como exemplo de um problema não numérico, considere que se deseja encontrar uma cadeia de 50 aminoácidos que se dobrará em uma estrutura proteica tridimensional desejada. Pode-se aplicar GA neste caso buscando em uma população de soluções candidatas,

onde cada uma é codificada como uma cadeia de 50 letras, por exemplo:

IHCCVASASDMIKPVFTVASYLKNWTKAKGPNFEICISGRTPYWDNFPGI

onde cada letra representa um dos possíveis 20 aminoácidos. Neste problema, uma forma de definir a função de aptidão de uma dada sequência é calcular o negativo da energia potencial da sequência com respeito à estrutura desejada. A energia potencial é uma medida de quanta resistência física a sequência apresentaria se forçada a ser dobrada na estrutura desejada - quanto menor a energia potencial, maior a aptidão.

Estes dois exemplos, apresentados no livro *An Introduction to Genetic Algorithms* de Mitchell Melanie, permitem visualizar em dois contextos diferentes, como soluções candidatas para os problemas podem ser representadas por cromossomos abstratos codificados como cadeias de símbolos, com uma função objetiva de aptidão definida neste espaço de cadeias resultante. Um algoritmo genético é um método para buscar em um espaço de soluções candidatas por cadeias altamente *aptas*.

4 Um Algoritmo Genético Simples

Dado um problema bem definido a ser solucionado e uma representação em cadeia de bits para as soluções candidatas, um AG simples funciona da seguinte maneira:

1. Comece com uma população gerada aleatoriamente de n cromossomos de l bits (soluções candidatas a um problema).
2. Calcule a aptidão $f(x)$ de cada cromossomo x na população.
3. Repita as etapas a seguir até que n filhos tenham sido criados:
   (a) **Selecione um par de cromossomos pais** (geradores) da população atual, com a probabilidade de seleção sendo uma função crescente da aptidão, por ex.: $P(x) = \frac{f(x)}{\sum_{j=1}^{n} f(x)}$.
   A seleção é feita "com substituição", o que significa que o mesmo cromossomo pode ser selecionado mais de uma vez para se tornar um pai.
   (b) Com a probabilidade p_c (a "**probabilidade de cruzamento**" ou "taxa de cruzamento"), cruze o par em um ponto escolhido aleatoriamente (escolhido com probabilidade uniforme nos *loci* dos cromossomos) para formar dois descendentes. Se não houver cruzamento, forme dois descendentes que são cópias exatas de seus respectivos pais. (Observe que aqui a taxa de cruzamento é definida como a probabilidade de que dois pais se cruzem em um único ponto. Existem também versões de *cruzamentos em multipontos* do GA em que a taxa de cruzamento para um par de pais é o número de pontos em que ocorre um cruzamento.)

(c) Realize a mutação dos dois descendentes gerados da etapa anterior. Com a **probabilidade p_m de mutação** (ou taxa de mutação), inverta o bit em cada locus do cromossomo . Insira os cromossomos resultantes na nova população.

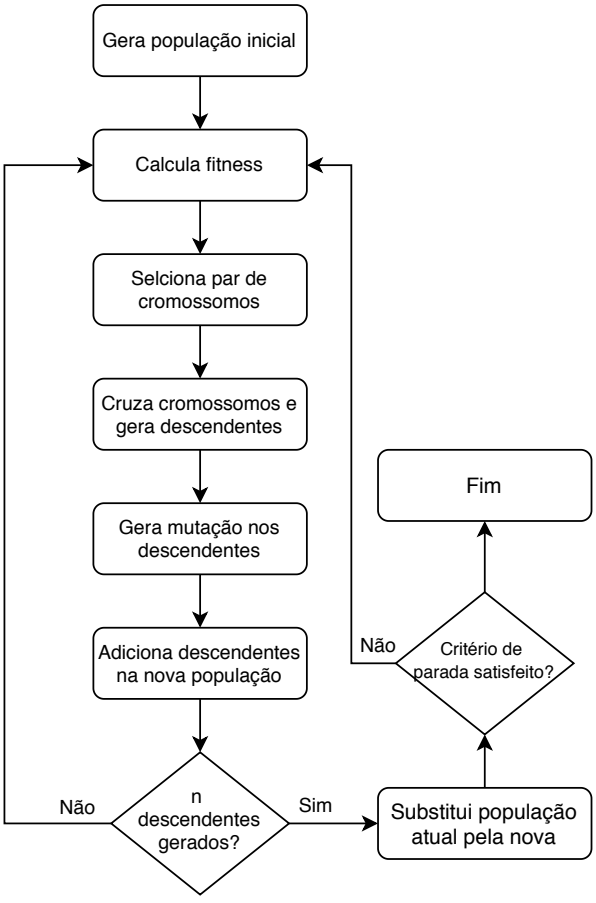
Se n for ímpar, um novo membro da população pode ser descartado aleatoriamente.

4. Substitua a população atual pela nova população.

5. Vá para a etapa 2.

Cada iteração desse processo é chamada de **geração**. Um AG é geralmente iterado em torno de 50 a 500 ou mais gerações. Uma execução (*run*) do AG consiste na produção de todas as gerações de descendentes. No final de uma execução, geralmente há um ou mais cromossomos altamente adaptados na população. Uma vez que a aleatoriedade desempenha um papel importante em cada execução, duas execuções com sementes de números aleatórios diferentes geralmente produzirão execuções distintas. Devido à esta natureza aleatória, ao se reportar a execução de um algoritmo genético, deve-se apresentar a média estatística de várias execuções diferentes sobre o mesmo problema, levando em conta por exemplo a melhor aptidão para aquela execução e a geração em que este melhor indivíduo foi encontrado.

O procedimento simples que acabamos de descrever é a base para a maioria das aplicações de AGs. Existem vários **hiperparâmetros** para serem sintonizados, como o *tamanho da população* e as *probabilidades de cruzamento e mutação*, e o sucesso do algoritmo muitas



vezes depende muito desses detalhes. Existem também versões mais complicadas de AGs (por exemplo, AGs que funcionam com representações diferentes de cadeias de símbolos ou AGs que têm diferentes tipos de operadores de cruzamento e mutação).

Para um exemplo mais detalhado de um GA simples, suponha que para determinada aplicação o tamanho da cadeia de bits seja $l = 8$, e que a função objetivo $f(x)$ seja igual ao número de $1s$ na cadeia (uma função de aptidão extremamente simples, usada aqui apenas para fins ilustrativos). Considere também que o tamanho da população seja $n = 4$, com uma probabilidade de cruzamento $p_c = 0.7$ e uma probabilidade de mutação $p_m = 0.001$. Note que estes valores de l e n foram escolhidos por simplicidade, pois em uma aplicação típica os valores l e n estão na faixa de $50 - 1000$. Os valores de p_c e p_m são mais condizentes com a realidade.

Uma exemplo de população inicial gerada aleatoriamente para este problema pode ser:

Legenda do Cromossomo	Cadeia do cromossomo	Aptidão
A	00000110	2
B	11101110	6
C	00100000	1
D	00110100	3

Um método comum de seleção em AGs é a **seleção proporcional à aptidão**, conhecida na biologia como seleção por viabilidade, onde o número de vezes esperado que um indivíduo na população se reproduza é igual a sua aptidão dividida pela aptidão média de toda a população.

Uma forma simples de implementar este método é a **amostragem por roleta** "*roulette-wheel sampling*", que equivale a dar a cada indivíduo da população uma fatia de uma roleta com área equivalente a sua aptidão. Quando a roleta é girada, um ponteiro ou uma bola irá parar em um ponto de uma das fatias, e um indivíduo correspondente é selecionado. Vale notar que indivíduos mais aptos terão mais chance de serem selecionados, uma vez que ocupam uma fatia maior na roleta.

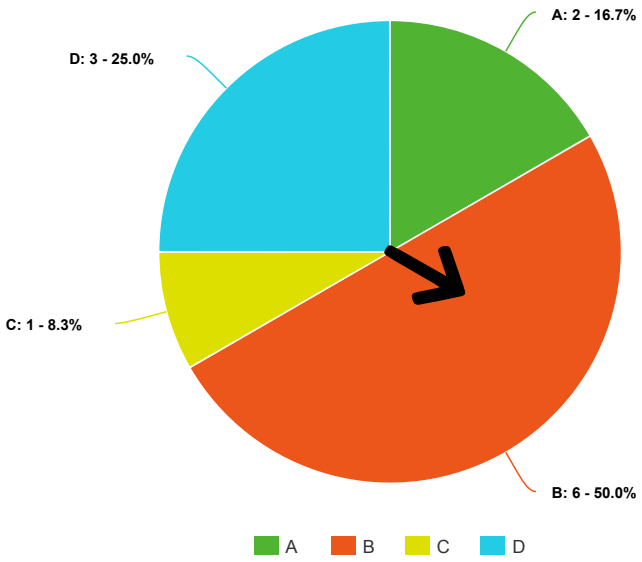


Figure 1: Representação da amostragem por roleta

No exemplo citado acima em que $n = 4$, a roleta iria girar 4 vezes; os dois primeiros giros poderiam resultar nos cromossomos B e D escolhidos, e os dois últimos giros poderiam escolher os cromossomos B e C (na média, essas escolhas feitas repetidamente aproximam a probabilidade dada pela roleta). Uma vez que um par de cromossomos pais é escolhido, estes se cruzam com probabilidade p_c para gerar descendentes. Se não cruzarem, os descendentes gerados serão as cópias exatas dos cromossomos pais.

Suponha, no exemplo acima, que os cromossomos pais B e D cruzam após a posição do primeiro bit para formar a prole $E = 10110100$ e $F = 01101110$, e os pais B e C não se cruzam, em vez disso, formando descendentes que são cópias exatas de B e C. Até o momento, antes da mutação, a nova população é dada por:

Cromossomo	Cadeia de bits	Fitness
E	10110100	4
F	01101110	5
C	00100000	1
B	11101110	6

Em seguida, cada descendente está sujeito a mutação em cada locus do cromossomo com probabilidade p_m. Por exemplo, suponha que o descendente E sofra mutação no sexto locus para formar $E' = 10110000$, o descendente F e C não sofram mutação, e o descendente B sofra mutação no primeiro locus para formar $B' = 01101110$. Desta forma, a nova população será a seguinte:

Cromossomo	Cadeia de bits	Fitness
E'	10110000	3
F	01101110	5
C	00100000	1
B'	01101110	5

Observe que, na nova população, embora o melhor indivíduo (aquele com aptidão 6) tenha sido perdido, a aptidão média da população aumentou de 4/12 para 14/4. Ao repetir este processo por várias gerações, eventualmente em alguma população será gerado um cromossomo com todos os genes iguais à 1.

5 Programação genética

Na programação genética (PG), os indivíduos de uma população não são mais cadeias de genes, mas sim programas de computador completos, geralmente representados como árvores ou máquinas de estado finito.

Os programas manipulados por PG são normalmente representados por árvores correspondendo à árvore de análise (*parse tree*) do programa. Cada chamada de função é representada por um nó na árvore, e os argumentos para a função são fornecidos por seu nós descendentes.

Por exemplo, considerando-se a função $\cos x - \sqrt{x+y}$, sua representação em árvore seria como a apresentada na Figura 2. Para se utilizar PG neste contexto, inicialmente devem ser definidas as funções primitivas a serem consideradas ($\sin$, $\cos$, $\sqrt{}$, $+$, $-$, $*$, $\wedge$, $\cdots$), além de também definir os terminais (x, y, π, constantes, $\cdots$). Então, o algoritmo use uma busca evolucionária para explorar o vasto espaço de programas que podem ser representados com estas primitivas.

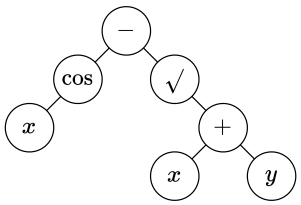


Figure 2: Representação em árvore da função $\cos x - \sqrt{x+y}$

Assim como nos algoritmos genéticos, existe uma população que neste caso é um conjunto de programas representados por árvores. Em cada iteração, o algoritmo de PG produz uma nova geração de indivíduos, utilizando os operadores de seleção, cruzamento e mutação. A **aptidão** de um determinado programa na população é tipicamente determinada pela execução do programa em um conjunto de dados de treinamento. Em PG, a **operação de cruzamento** consiste em substituir uma sub-árvore escolhida aleatoriamente de uma das

árvores-pai por uma sub-árvore da outra árvore-pai. Da mesma forma, a mutação pode ser feita substituindo aleatoriamente uma primitiva de um nó da árvore. As operações de cruzamento e mutação podem ser vistas nas Figuras 3 e 4.

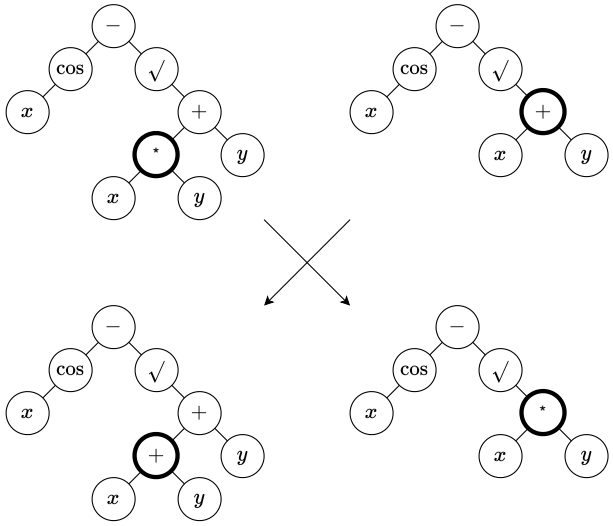


Figure 3: Operação de cruzamento entre dois programas

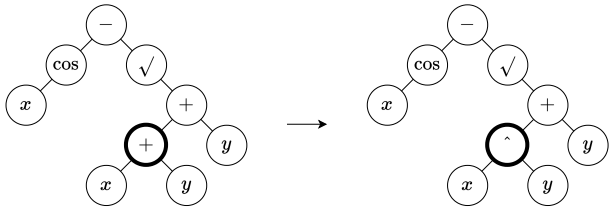


Figure 4: Operação de mutação em um programa

6 Bibliografia complementar

Mitchell, Melanie. An introduction to genetic algorithms. MIT press, 1998.

Mitchell, Tom M. Machine Learning. WCB McGraw-Hill, 1997 (capítulo 9).

7 Exercício

(a) Implemente um algoritmo genético simples (como definido na seção 4) em Python ou em sua linguagem preferida para maximizar a função dada em (1), ou seja:

$$f(y) = y + |sin(32y)|, \quad 0 \leq y < \pi$$

Use as funções *get_bits* e *get_float* em Python apresentadas abaixo para converter números reais em cadeias de bits e vice-versa (ou faça uma implementação própria sem usar strings).

```python
L = 4 * 8 # size of chromossome in bits

import struct

def floatToBits(f):
    s = struct.pack('>f', f)
    return struct.unpack('>L', s)[0]

def bitsToFloat(b):
    s = struct.pack('>L', b)
    return struct.unpack('>f', s)[0]

# Exemplo:  1.23 -> '00010111100'
def get_bits(x):
    x = floatToBits(x)
    N = 4 * 8
    bits = ''
    for bit in range(N):
        b = x & (2**bit)
        bits += '1' if b > 0 else '0'
    return bits

# Exemplo:  '00010111100' ->  1.23
def get_float(bits):
    x = 0
    assert(len(bits) == L)
    for i, bit in enumerate(bits):
        bit = int(bit)  # 0 or 1
        x += bit * (2**i)
    return bitsToFloat(x)
```

(b) Varie o tamanho da população, a taxa de mutação e a taxa de cruzamento e observe os resultados.

(c) Plote a aptidão média da população ao longo da evolução. Veja o plot para uma execução na figura 5.

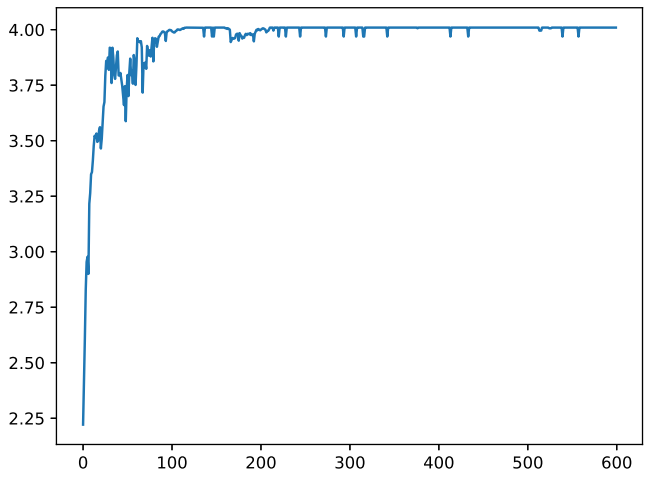
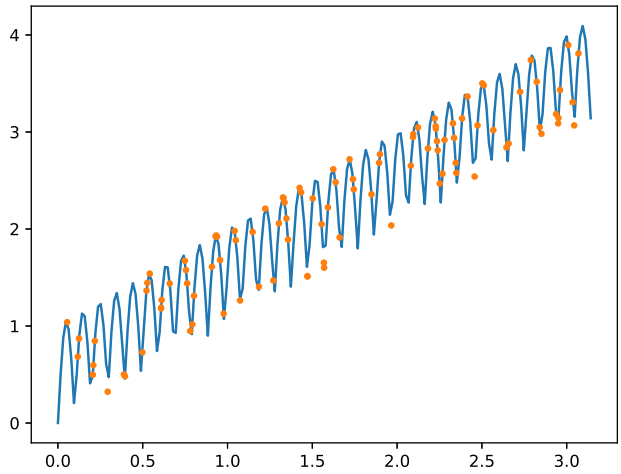
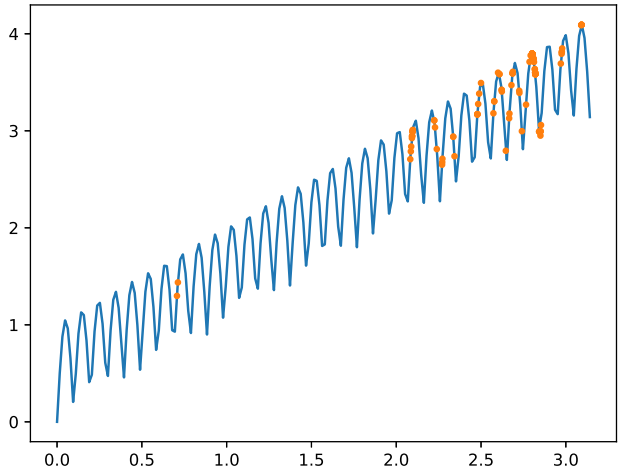


Figure 5: Aptidão média ao longo de 600 gerações.

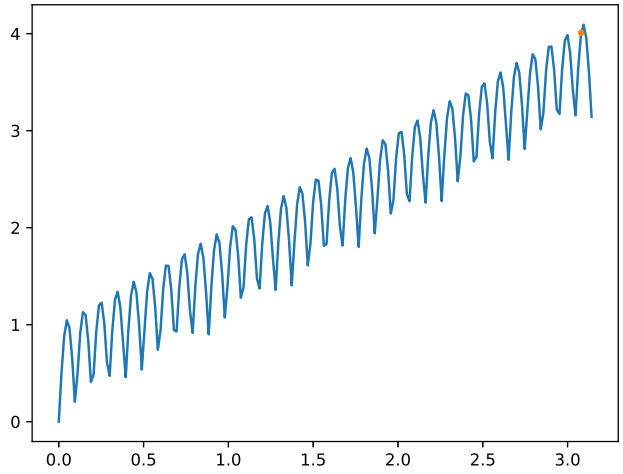
(d) Plote a aptidão em função do cromossomo para as diversas gerações numa execução. Veja exemplos na figura 6.



(a)



(b)



(c)

Figure 6: Função a ser maximizada em azul e cromossomos (pontos em laranja) da população inicial (a), da décima geração (b) e da última geração (600) (c), respectivamente. Eixo vertical dá a aptidão, que é a função aplicada no próprio cromossomo como número real de 0 a π.