INSTITUTO FEDERAL
Catarinense
Campus Rio do Sul

# Algoritmo Genético DNA

Aluno: Adriano Zimmermann
Disciplina: Inteligência Artificial
Professor: Daniel Gomes Soares

***Abstract.*** *In this article we will see a brief explanation of genetic algorithms where an analysis will be made of a line with letters that correspond to a strand of DNA, a search will be made to verify if the algorithm manages to reach a satisfactory result. populations and generations, and at the end the generated results will be demonstrated.*
***Key-words:*** *Genetic algorithms; DNA; Tests.*

***Resumo.*** *Neste artigo veremos uma breve explicação de algoritmos genéticos onde será feito uma análise de uma linha com letras que correspondem a uma fita de DNA, será feita a busca para verificar se o algoritmo consegue chegar a um resultado satisfatório No total foram executados 5 testes com diferentes populações e gerações, e ao final será demonstrados os resultados gerados.*
***Palavras-chave:*** *Algoritmo genético; DNA; Testes.*

## 1. Introdução

Os algoritmos genéticos são muito utilizados para busca de estados ideais, sendo uma técnica de inteligência artificial ela foi baseada na sobrevivência de indivíduos mais fortes ou seja na evolução das espécies, sabe se que a natureza evolui com isso essa técnica seleciona cromossomos mais “fortes” de pais, gerando assim filhos melhores, se tenta chegar a um resultado próximo ou igual ao desejado.

## 2. DNA

O DNA (ácido desoxirribonucleico) é um tipo de ácido nucleico que possui destaque por armazenar a informação genética da grande maioria dos seres vivos. Essa molécula é formada por nucleotídeos e apresenta, geralmente, a forma de uma

dupla-hélice. Nos organismos eucariotos, o DNA é encontrado no núcleo da célula, nas mitocôndrias e nos cloroplastos. Nos procariontes, o DNA está localizado em uma região que não é delimitada por membrana, denominada de nucleóide.

### 2.1. Estrutura do DNA

O DNA é formado por duas cadeias de polinucleotídeos (fita), que são constituídas por vários nucleotídeos. Os nucleotídeos são unidos uns aos outros por ligações denominadas fosfodiéster (grupo fosfato ligando dois açúcares de dois nucleotídeos). Nessas ligações, um grupo fosfato conecta o carbono 3' de um açúcar ao carbono 5' do próximo açúcar.

Essa junção dos nucleotídeos forma um padrão típico de repetição de unidade de açúcar-fosfato, que forma a cadeia principal. A essa cadeia principal estão ligadas às bases nitrogenadas.

Essa estrutura nitrogenada é formada por, Adenina (A), Timina (T), Citosina (C) e Guanina (G).

## 3. Algoritmo Genético

Algoritmos genéticos foram inspirados na evolução das espécies, ou seja os indivíduos mais fortes sobrevivem na natureza, melhorando seus genes para que possam continuar existindo, conforme Pacheco cita, são algoritmos matemáticos inspirados nos mecanismos de evolução natural e recombinação genética. A técnica de Algoritmos Genéticos fornece um mecanismo de busca adaptativa que se baseia no princípio Darwiniano de reprodução e sobrevivência dos mais aptos.

## 4. Código

Neste artigo serão utilizados os algoritmos genéticos para fazer uma busca de uma sequência de DNA, essa sequência contará com um conjunto de 100 letras, correspondentes a estrutura de DNA formada pelas letras A, T, G e C. O modelo usado para a busca será retirado de um arquivo .txt onde contam com várias linhas de DNA.

O código foi desenvolvido na linguagem python e usado o editor de código Visual Studio Code, os testes foram executados em uma máquina com sistema operacional Windows 11 pro, com processador AMD Ryzen 5 2600 Six-Core 3.40 GHz e 16GB de memória RAM, não foi computado o tempo que o algoritmo levou para fazer as gerações.

Na figura 1 veremos as variáveis da população inicial, DNA, modelo, gerações e variável de tamanho do cromossomo.

```
arq = open("dna-0.txt")
linhas = arq.readlines()

dna = ['A','T','C','G']
model = linhas[0]
chromosome_size = len(model)
population_size = 200
generations = 5000
```

Figura 1 (Fonte autor).

Na figura 2 veremos as funções de criação da população, onde a função criará a população aleatoriamente conforme estipulado pela lista dna, usando apenas as letras correspondentes a fita de DNA.

```
def random_character():
  return random.choice(dna)

def random_population():
  population = []
  for i in range(population_size):
    chromosome = ""
    for j in range(chromosome_size):
      chromosome += random_character()
    population.append(chromosome)
  return population
```

Figura 2 (Fonte autor).

Na figura 3 podemos ver a função de mutação, essa função é especial porque é utilizada para garantir uma maior varredura do espaço de estados e evitar que o algoritmo genético convirja muito cedo para mínimos locais. A mutação é efetuada alterando-se o valor de um gene de um indivíduo sorteado aleatoriamente com uma determinada probabilidade, denominada probabilidade de mutação, ou seja, vários indivíduos da nova população podem ter um de seus genes alterado aleatoriamente.

```
def mutation(chromosome):
  chromosome_outside = ""
  mutation_chance = 100
  for i in range(chromosome_size):
    if int(random.random() * mutation_chance) == 1:
      chromosome_outside += random_character()
    else:
      chromosome_outside += chromosome[i]
  return chromosome_outside
```

Figura 3 (Fonte autor).

## 5. Testes

O primeiro teste foi utilizado uma população inicial de 200 e o algoritmo poderá gerar até 5000 descendentes ou se caso chegar ao modelo ideal terminará antes, o segundo teste foi utilizado a mesma quantidade de população e um limite de 10000 descendentes, o terceiro teste foi aumentado para até 30000 descendentes com a mesma quantidade de população inicial, o quarto teste foi utilizado um limite de 10000 descendentes com uma população inicial de 500, e o último teste foi aumentado para uma população inicial de 1000 com a mesma quantidade de descendentes do teste 4. Ao final será apresentado a última geração e o modelo ideal de cada teste.

```
Geração 4998 | População: 'CTTCGGAACGGGTCGTCAGCACCGGCCTCGCATTAATGCTGTACTCCCGTGAAAGAAGGAACGATATCTATTTTCCATCTCCGATCACAAAGCCGTGCGAA'
Geração 4999 | População: 'GTTCGAACAGGATCGTCAGCACCGGCGTCGCATTACTGGTATACTTGGGTGACCGGGGTAACGATAAATGATTTCCGTCCCCCTAGACATAGCCGTAAGCA'
População Final: GTTCGAAACGGATCGTGAGCACCAGCGTCACCTTCTTGCTGGAATTCAGTCACAGAAGTCCCGATAGATGTTTTCCATCTCCTAACCCCAAGGCGTCCGCA
Modelo: CTTCGCAACGGATCGTGAGCAACAGCGTCGCCTTCTTGGTGGAATTCGGTGACCGAAGTAACGATAGATGTTTTCCATCTACTAACCCAAAGCCGTCCGC
```

Figura 4 - Primeiro teste população inicial 200, gerações 5000 (fonte autor).

```
Geração 9998 | População: 'CCTCGCAAAGGATCGTGCGCAGAAGCGTCCCCTTCTTGGTAGACTTCGGTGACCCACGTAACCACCGATGTTTTCAATCTACTACCACAAAGCAATACGTA'
Geração 9999 | População: 'CTTCGCCCGGGATAGTCCACAACCGCATGGCCTTGAGGGTGGAATTAGTTGACCAAGGGACCGATTGATGTTTTCAATCTACTAAACCACCGCCGTCAGAA'
População Final: CTTCGCCACGGATAGTGCGCACCCGCGTAGCGTTCTTGGTGGAATTCGGTGACCAACGTAAGGATAGATGTTTTCCATCTAATAACCCAAACCCGTCCGCC
Modelo: CTTCGCAACGGATCGTGAGCAACAGCGTCGCCTTCTTGGTGGAATTCGGTGACCGAAGTAACGATAGATGTTTTCCATCTACTAACCCAAAGCCGTCCGC
```

Figura 5 - Segundo teste população inicial 200, gerações 10000 (fonte autor).

```
Geração 29998 | População: 'CTTCGAAAGGGATCGTAAGCAGCGGCGTCGCGTTATTGCTGGGCTTCCGTGACCGACGTGAAAATGCATCTTTTACATAGACTAAGCCAAAACGCTGCGCC'
Geração 29999 | População: 'CTTCGCAAGGAATCGCGAGTCACAGCTTCGAGTTGTTGGTGGACTTTGGTGATTGAAGTAACGTTAGATATTTTCCATATACTAACCCAACCCCGTCCGCC'
População Final: ATTCGAAAAGGATCGTGAGCAACAGCGTCGCGTTCTTGGTGCACTTCGGTCACCGAAGTAAGGATAGATGTTTTCCATCTCCTAAACCAAAACACTCCCGC
Modelo: CTTCGCAACGGATCGTGAGCAACAGCGTCGCCTTCTTGGTGGAATTCGGTGACCGAAGTAACGATAGATGTTTTCCATCTACTAACCCAAAGCCGTCCGC
```

Figura 6 - Terceiro teste população inicial 200, gerações 30000 (fonte autor).

```
Geração 9998 | População: 'CTTCGCAACGGCTGCTGAGCAACCGTGTCGCCTTCTTCGTGGGCTTAGGTCACCGAAGTACAGATCGATGTTTTCGATCTACTAACCCAAAGCGGTCCGCA'
Geração 9999 | População: 'CGTTGGAACGGATCGGGGGCAACAGCGTCGCCTTCTTGGTAGAATGCGGTGACCAAGGTAACAATAGATGTTTTCCCGCTACTCAACCCAATCAGTGCGCA'
População Final: CTTCGCAACGCATCGTGAGCAACAGCGTCGCCTTCTTGGTCCAATTCGGTGACCGAAGTACAGATAGATGTTTTCAATCTACTAACCCAAAGCAGTCCGCA
Modelo: CTTCGCAACGGATCGTGAGCAACAGCGTCGCCTTCTTGGTGGAATTCGGTGACCGAAGTAACGATAGATGTTTTCCATCTACTAACCCAAAGCCGTCCGC
```

Figura 7 - Quarto teste população inicial 500, gerações 10000 (fonte autor).

```
Geração 9998 | População: 'CTTCGCAACGGATCGTGAGCACCAGCGTCGCATCCTTGGTGGAATTCGGTGACCGGAGTCACGATAGCTGTTTTACCTCTAAGAAACGAAAGCCGTCCGCA'
Geração 9999 | População: 'CTTCGCAGCAGGGCGTGAGCCACAGCCTAGCCTTCTTGGTCGAATTTGGTGACCAATATACCAATAGATGTTTTCCGTCTACTAACCCAAAGGCGTCCGCC'
População Final: CTTCGCAACGGATCGTGAGCAACAGCGTCGCCTTCTTGGTGGAATTAGCTCACCGAAGTAACGATAGATGTTTTCCATGTACTAACCCAGCGCCGTCCGCA
Modelo: CTTCGCAACGGATCGTGAGCAACAGCGTCGCCTTCTTGGTGGAATTCGGTGACCGAAGTAACGATAGATGTTTTCCATCTACTAACCCAAAGCCGTCCGC
```

Figura 8 - Quinto teste população inicial 1000, gerações 10000 (fonte autor).

## 5. Conclusão

Como podemos verificar nos testes apresentados nenhum chegou ao modelo ideal, no primeiro teste com população inicial de 200 e 5000 gerações pode verificar

que a última geração apresentou 22 diferenças de um total de 100, o segundo teste com 10000 gerações apresentou uma diferença de 12, já o terceiro teste com 30000 gerações apresentou uma diferença 15, o quarto teste com uma população inicial de 500 e 10000 gerações apresentou uma diferença de 7, o último teste com uma população de 1000 e 10000 gerações apresentou uma diferença de 6 apresentando o melhor resultado, porém neste teste por existir uma população de 1000 indivíduos o algoritmo obteve o maior tempo para gerar as 10000 gerações. O código completo pode ser acessado no link, https://github.com/adrianorsl/AlgoritmoGenetico.git.

## 4. Referências


MAGALHÃES, Lana (ed.). **DNA**. Disponível em: https://www.todamateria.com.br/dna/. Acesso em: 10 nov. 2022.

SANTOS, Vanessa Sardinha dos. **"DNA"**; *Brasil Escola*. Disponível em: https://brasilescola.uol.com.br/biologia/dna.htm. Acesso em 10 de novembro de 2022.

PACHECO, Marco Aurélio Cavalcanti. **ALGORITMOS GENÉTICOS: PRINCÍPIOS E APLICAÇÕES**. Disponível em: http://www.inf.ufsc.br/~mauro.roisenberg/ine5377/Cursos-ICA/CE-intro_apost.pdf. Acesso em: 10 nov. 2022.