

# Genetic Means

Felipe Colombelli

# Codificação dos cromossos

- Array de 7128 elementos booleanos
- True representa que o gene associado ao índice daquele elemento foi selecionado
- False, descartado

Exemplo de um indivíduo:

[True, False, False, True, ..... , False, False]

# Fitness function

- Construída a partir de uma análise incremental de necessidades
- Começamos com a relação óbvia

Seja     $acc$ : acurácia

$ngs$ : número de genes selecionados

$$\textbf{Fitness} = acc - ngs$$

# Fitness function

- Normaliza-se **ngs** para o mesmo intervalo de acc

$$\text{ngsNorm} = \frac{100 * (\text{ngs} - \text{min})}{\text{max} - \text{min}}$$

$$\begin{aligned}\text{min} &= 0 \\ \text{max} &= 7128\end{aligned}$$

$$\text{ngsNorm} = \frac{100 * \text{ngs}}{7128}$$

# Fitness function

- Assume-se acurácias abaixo de 90% como lixo

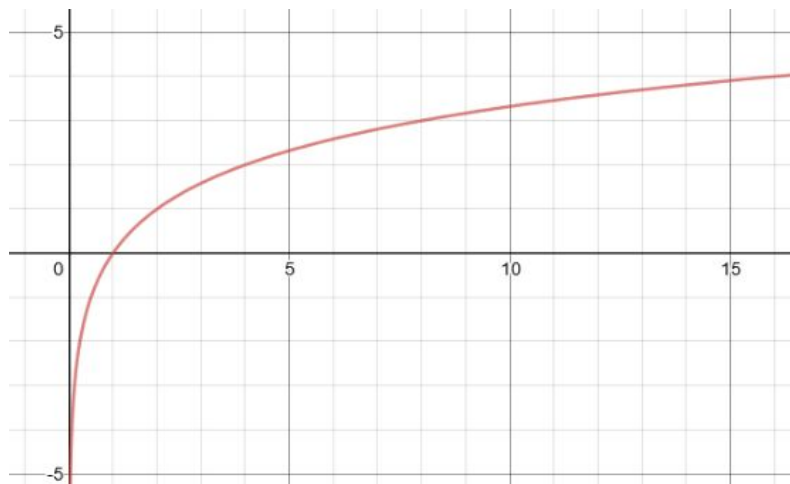
**if acc <= 100:**

**acc = acc - 100**

- Fazendo com que o novo intervalo seja [-100, -10]

# Fitness function

- Considerando os comportamentos assintóticos



$\log_2(x)$



$\log_2(-x)$

# Fitness function

- Adiciona-se tais comportamentos monotônicos como coeficientes multiplicativos da acurácia, destacando sua importância em relação ao número de genes selecionados

**if acc <= 90:**

**acc = acc - 100**

**accNew = acc \*  $-\log_2(-acc^*)$**

**else:**

**accNew = acc \*  $\log_2(acc)$**

\* -acc: [10, 100] => 80%, 0% de acurácia

# Fitness function

- Finalmente tem-se a função para cálculo de fitness

$$\text{Fitness} = \text{accNew} - \text{ngsNorm}$$



# Computação da função de fitness

- Gargalo da aplicação
- Independência de dados entre indivíduos
- Implementação explorando paralelismo

```
169     def __computeFitness(self):
170
171         self.fitness = [None] * len(self.population)
172         pool = mp.Pool(mp.cpu_count())
173
174         self.fitness = np.array([pool.apply(self.computeIndividualFitness, args=(individual, ))
175 |                               | for individual in self.population])
176
177         pool.close()
178         return
```

# Demonstração da utilização dos recursos

Activities Terminal nov 22 04:01 genetic\_means.py - genetic-means - Visual Studio Code

colombelli@colombelli: ~/Documents/genetic-means

```
1 [|||||] [86.8%] 5 [|||||] [92.2%]
2 [|||||] [87.4%] 6 [|||||] [92.1%]
3 [|||||] [94.1%] 7 [|||||] [92.8%]
4 [|||||] [86.8%] 8 [|||||] [88.1%]
Mem [|||||] [4.98G/7.63G] Tasks: 182, 901 thr; 8 running
Swp [|||||] [259M/2.00G] Load average: 2.56 1.59 1.29
Uptime: 14:32:47
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
10570	colombelli	20	0	1222M	134M	41676	S	5.9	1.7	0:04.11	python3 main.py
2623	colombelli	20	0	4552M	284M	114M	S	2.0	3.6	11:26.92	/snap/spotify/36/
1718	colombelli	20	0	3489M	320M	87008	S	1.3	4.1	34:26.32	/usr/bin/gnome-sh
1496	colombelli	9	-11	3241M	23840	18156	S	1.3	0.3	16:27.39	/usr/bin/pulseaud
2769	colombelli	20	0	2007M	307M	90960	S	2.0	3.9	4:45.63	/snap/spotify/36/
10398	colombelli	20	0	2335M	84080	65368	S	0.0	1.1	0:00.23	/usr/lib/firefox/
1506	colombelli	20	0	335M	93468	53952	S	0.7	1.2	29:28.72	/usr/lib/xorg/Xor
1507	colombelli	-6	0	3241M	23840	18156	S	1.3	0.3	8:34.03	/usr/bin/pulseaud
2763	colombelli	20	0	4552M	284M	114M	S	0.7	3.6	3:01.79	/snap/spotify/36/
2798	colombelli	20	0	4552M	284M	114M	S	0.7	3.6	4:21.03	/snap/spotify/36/
2751	colombelli	20	0	944M	99180	78892	S	0.7	1.2	2:12.76	/snap/spotify/36/
9834	colombelli	20	0	2698M	297M	110M	S	0.7	3.8	0:15.55	/usr/lib/firefox/
10578	colombelli	20	0	2335M	84080	65368	S	0.0	1.1	0:00.02	/usr/lib/firefox/

F1 help F2 Setup F3 Search F4 Filter F5 Free F6 Sort By F7 Filter F8 Kill F9 Kill F10 Quit

```
delta compression using up to 8 threads
aemon.py", line 214, in start cliCompressing objects: 100% (6/6), done.
with self.startedWriting objects: 100% (6/6), 2.98 KiB | 1018.00 KiB/s, done.
File "/usr/lib/pythonTotal 6 (delta 3), reused 0 (delta 0)
return next(self.gremote: 100% (3/3), completed with 3 local objects.
File "/home/colombelliTo https://github.com/colombelli/genetic-means.git
32352ed..8d1d2a4 master -> master
aemon.py", line 110, in started
self.start()
File "/home/colombelliRuntimeError: already started
aemon.py", line 145, in start
raise RuntimeError('already started')
RuntimeError: already started
```

colombelli@colombelli: ~/Documents/genetic-means

```
python3 main.py
git add -A
git commit -m "fitness method i
ual accuracy in parallel)"
thod implemented (compute individual accuracy in para
lons(+), 6 deletions(-)
python3 main.py
git push
thub.com': ombelli
belli@github.com':
ne.
11), done.
delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 2.98 KiB | 1018.00 KiB/s, done.
Total 6 (delta 3), reused 0 (delta 0)
Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/colombelli/genetic-means.git
32352ed..8d1d2a4 master -> master
colombelli@colombelli:~/Documents/genetic-means$ python3 main.py
```

Ln 95, Col 1 Spaces: 4 UTF-8 LF Python

# Implementação 1 - Geração da população inicial

- Geração de 7128 booleanos aleatórios, 50 vezes (50 indivíduos com 7128 genes selecionados ou não)

```
160     def __generatePopulation(self):
161
162         self.population = np.array([
163             [bool(random.getrandbits(1)) for i in range(NUM_OF_GENES)]
164             for i in range(self.populationSize)])
165         return
```

# Seleção dos indivíduos da população

- Método do elitismo
- 30% de retenção

```
235     def __selectPopulation(self):
236
237         numElite = round(self.elitism * self.populationSize)
238         # Get the index of the N greatest scores:
239         eliteIdx = np.argpartition(self.fitness, -numElite)[-numElite:]
240         elite = self.population[eliteIdx]
241
242         self.population = elite
243         return
```

# Cruzamento da população

- Geração de máscara aleatória
- Exemplo de máscara:  
[1, 0, 0, 1, ..... , 1, 1, 0]
- Genes 2, 3, 7128 herdados do pai 0
- Genes 1, 4, 7126, 7127 herdados do pai 1

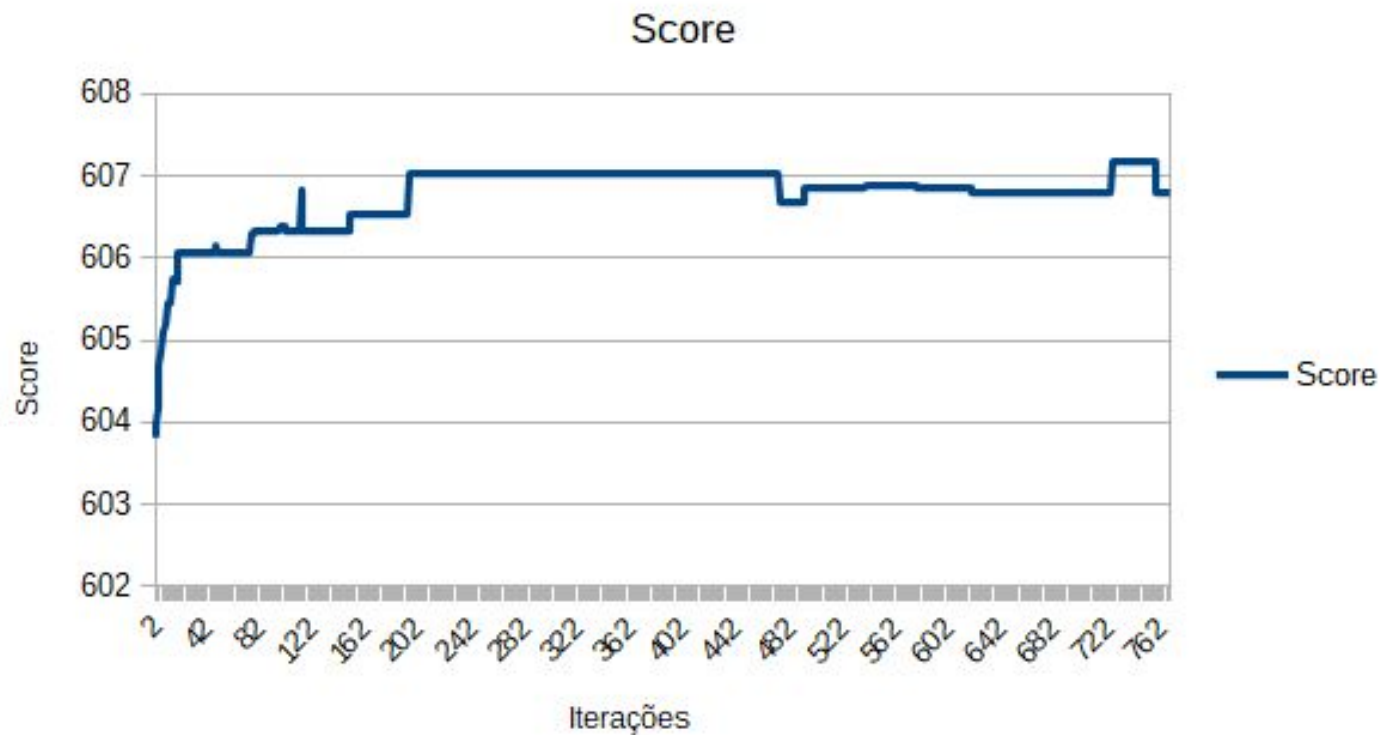
# Implementação 1 - Mutação

- Taxa: 0.2
- Escolha feita em relação a todos os genes de todos os indivíduos filhos gerados

# Resultados da Implementação 1

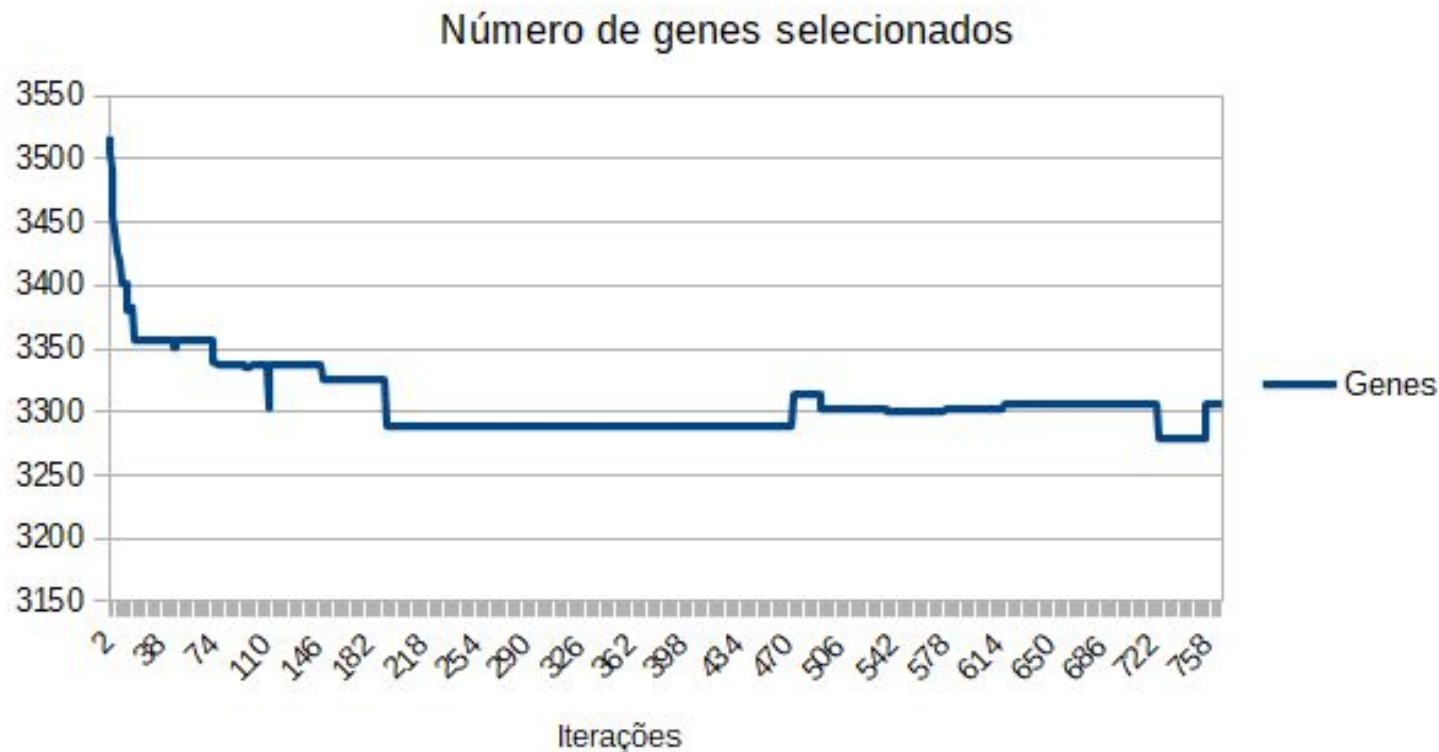
- 767 iterações feitas
- 3306 genes selecionados
- 98.6%

# Resultados da Implementação 1

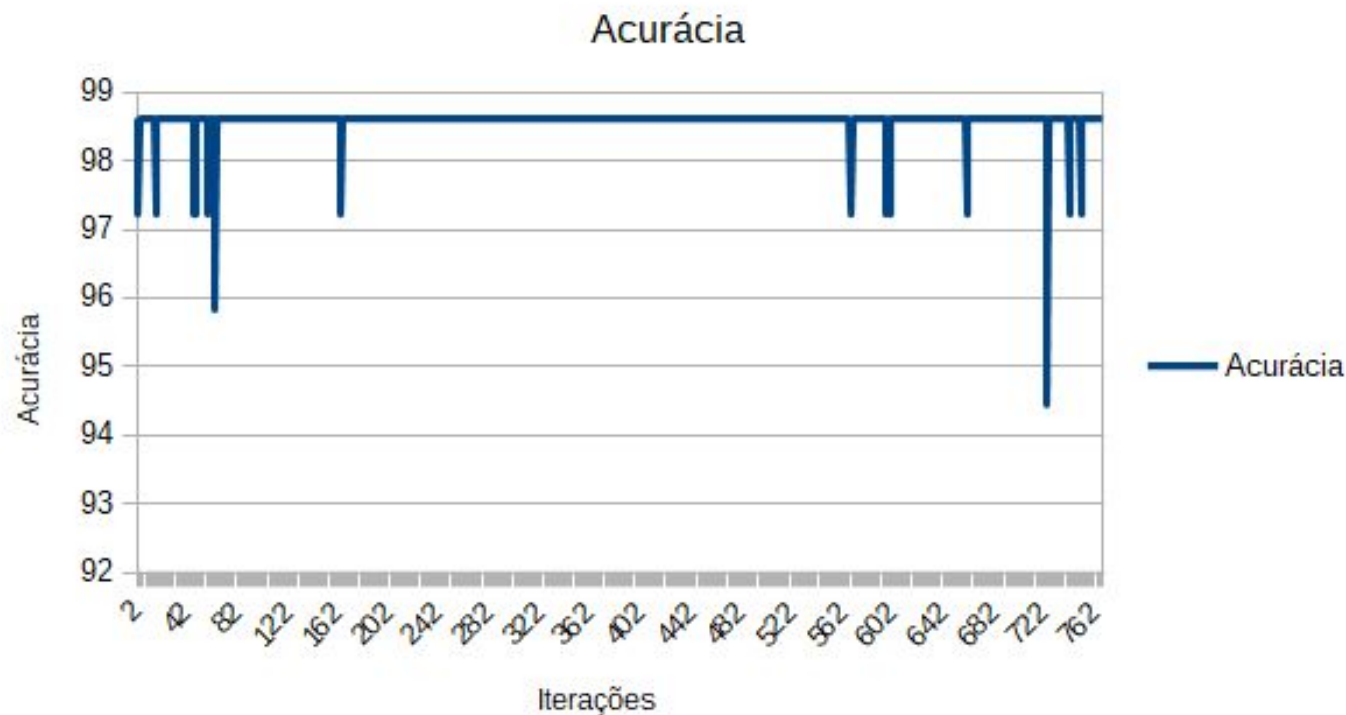




# Resultados da Implementação 1



# Resultados da Implementação 1



# Implementação 2 - Modificação 1

- Taxa de mutação: 0.001
- Escolha feita em relação a todos os genes de todos os indivíduos filhos gerados
- Rationale:
  - Com 0.2, em média 49896 são mutados
  - Com 0.001, apenas 249

## Implementação 2 - Modificação 2

- Número de iterações do K-Means
- Antes: 25
- Agora: 50
- Rationale: acurácia oscilante para os mesmos indivíduos (mantidos através do elitismo) devido à escolha estocástica inicial de clusters

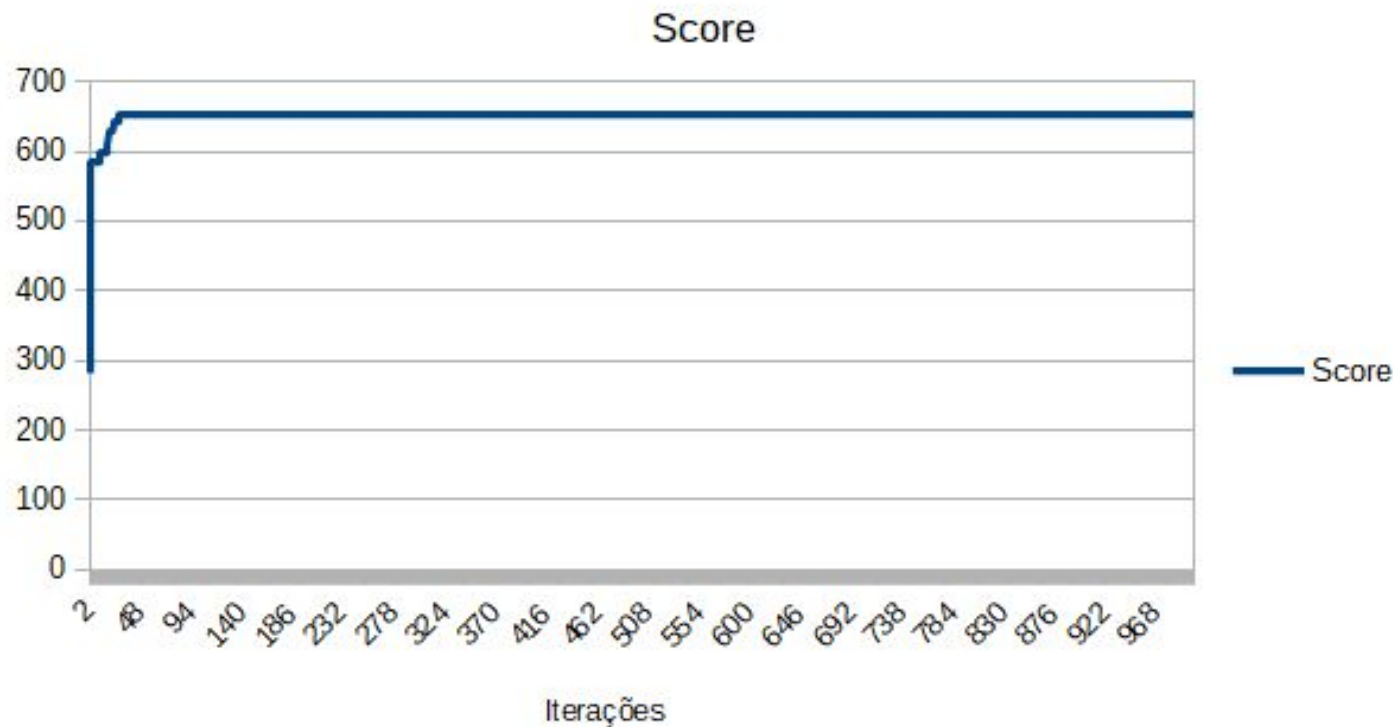
## Implementação 2 - Modificação 3

- **Geração da população inicial**
- 50 indivíduos com todos os genes em False
- Operação de turning x5
- Ao final das 5 operações, cada um dos 50 indivíduos terá, provavelmente, 5 genes selecionados
- Rationale: espaço de busca começa “de baixo”

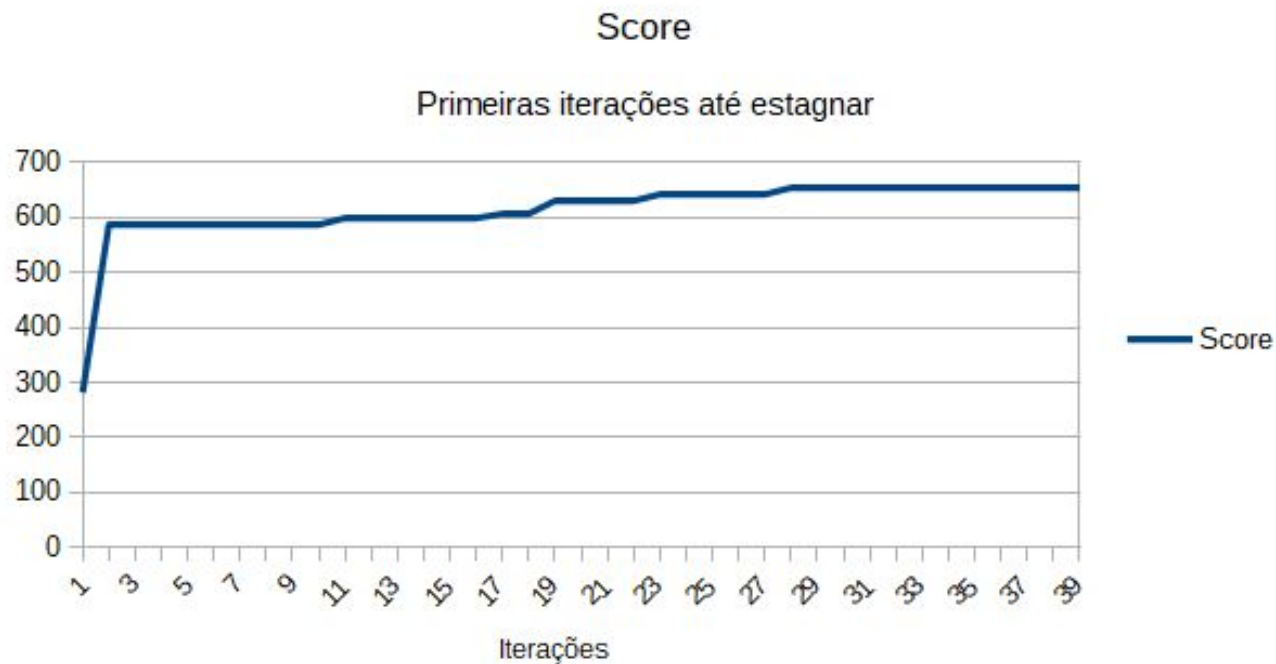
# Resultados da Implementação 1

- 1000 iterações feitas
- 42 genes selecionados
- 98.6%

# Resultados da Implementação 1

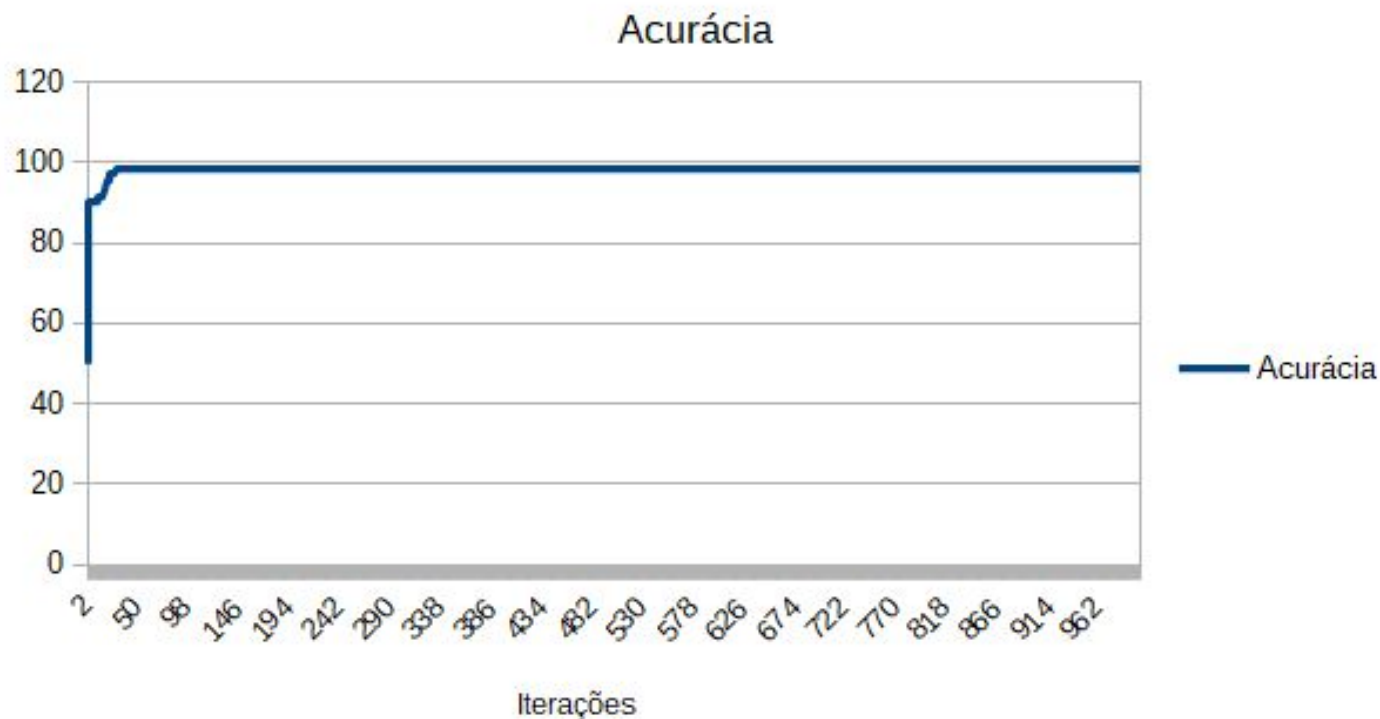


# Resultados da Implementação 1

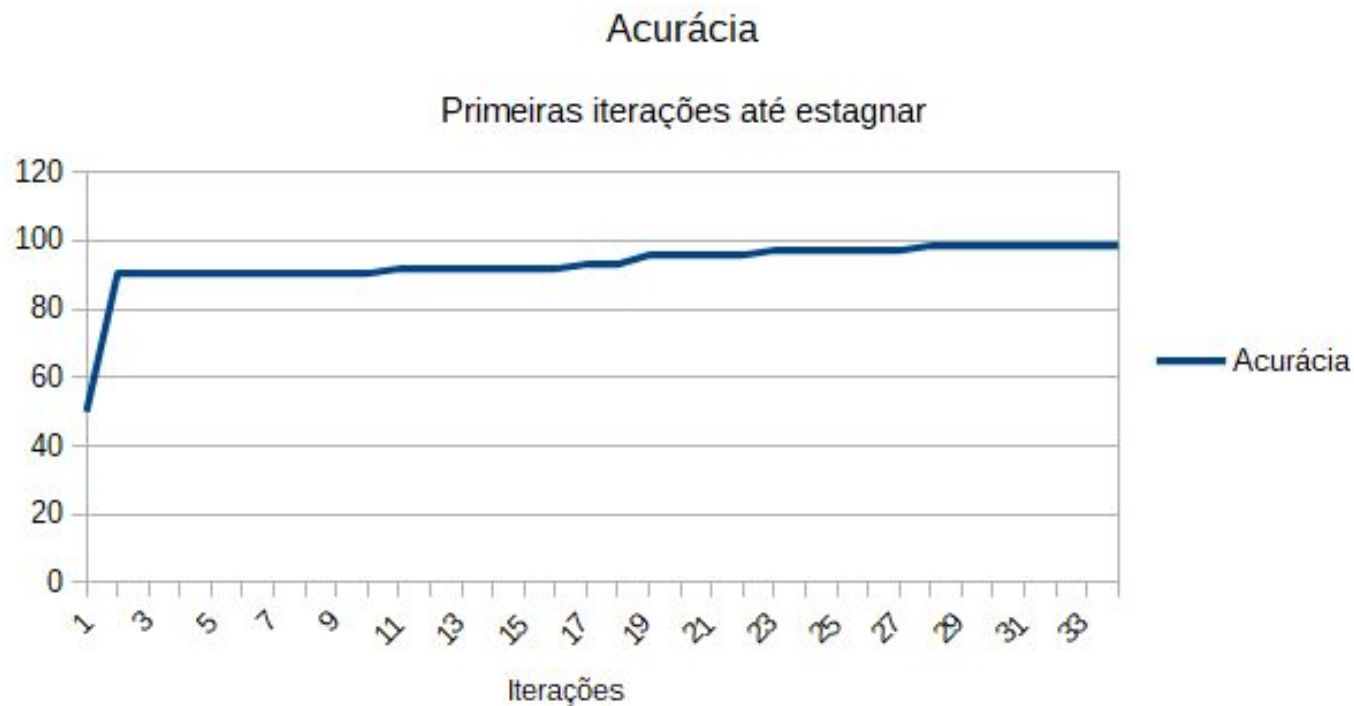




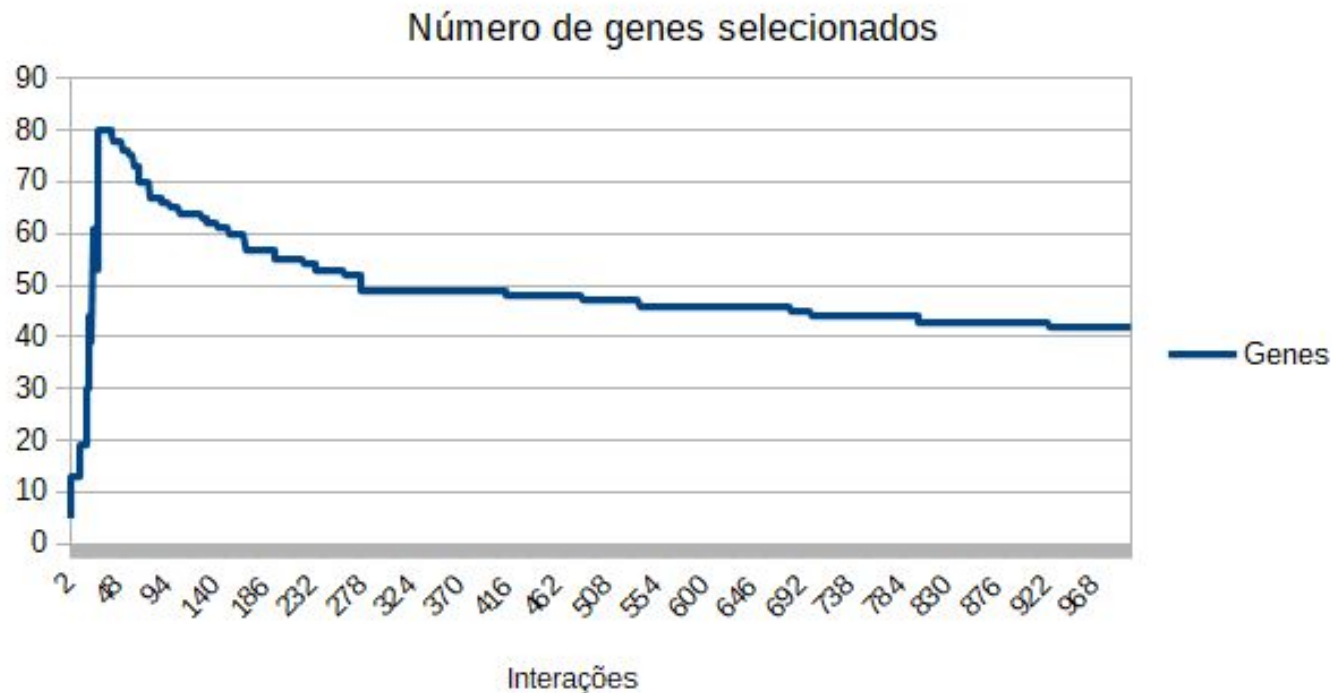
# Resultados da Implementação 1



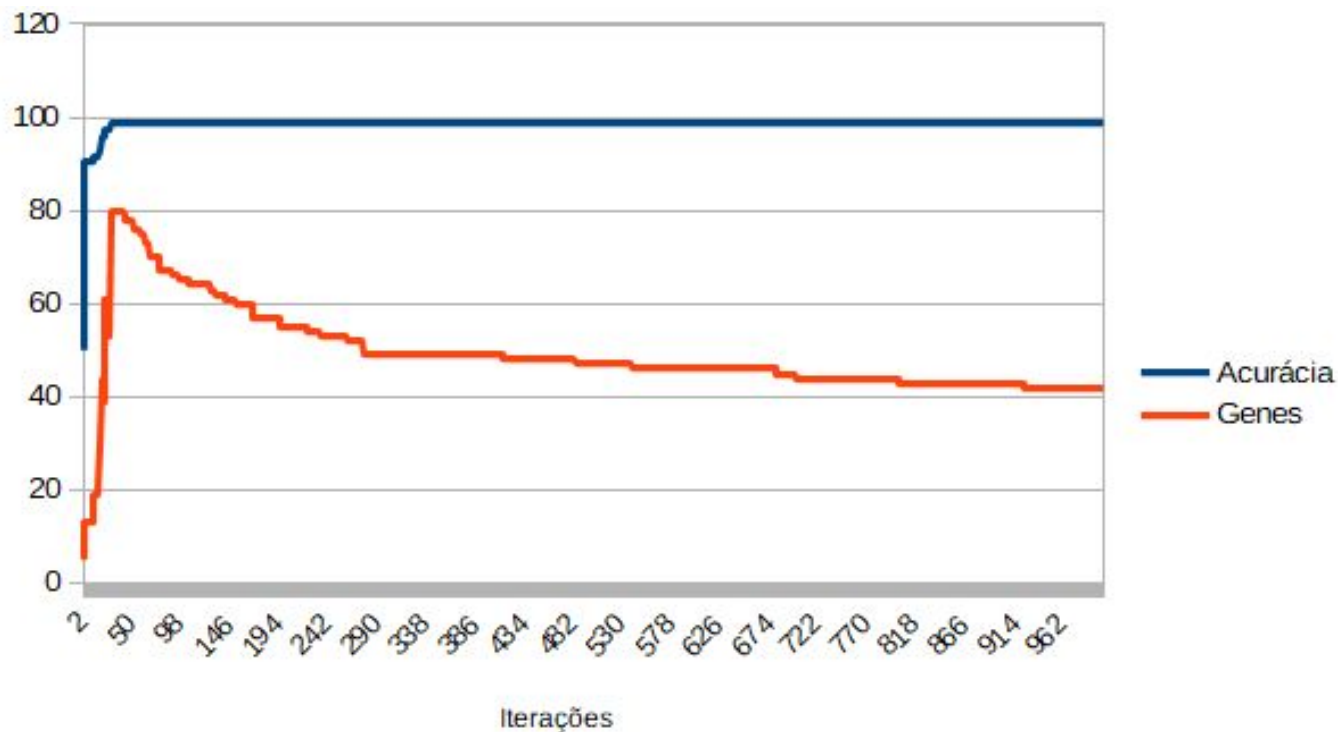
# Resultados da Implementação 1



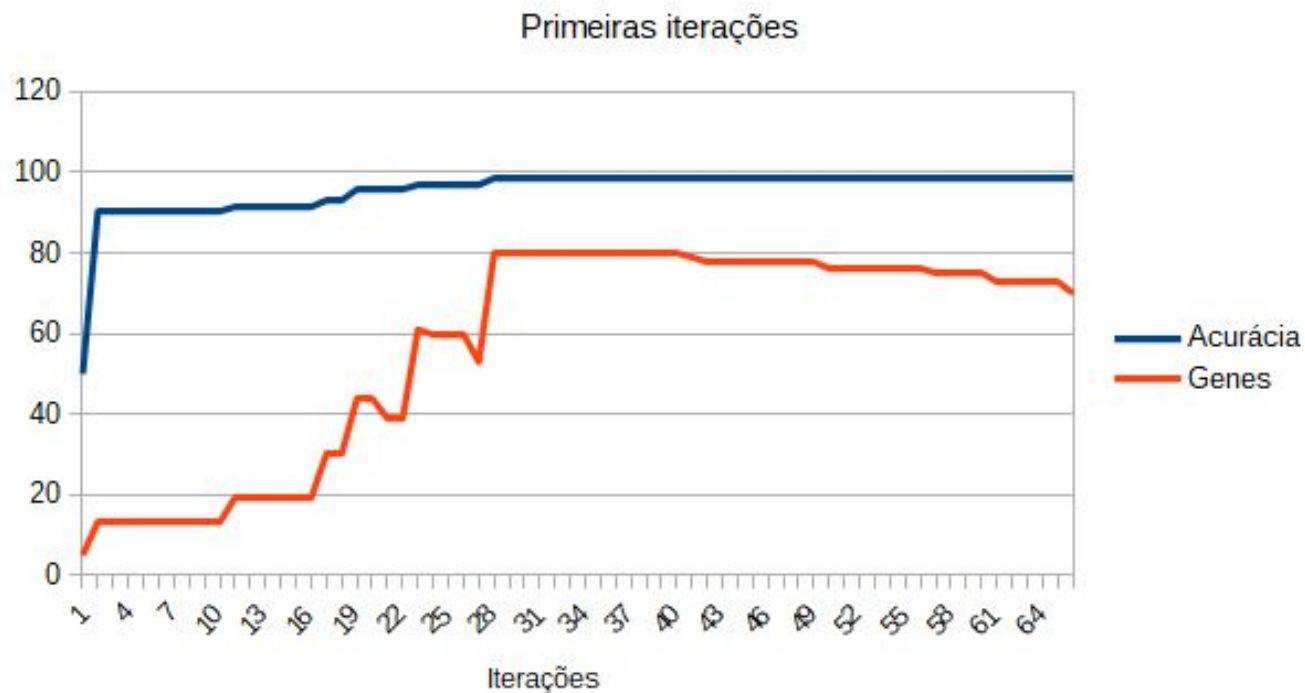
# Resultados da Implementação 1



# Resultados da Implementação 1



# Resultados da Implementação 1



Repo:

<https://github.com/colombelli/genetic-means>