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

Fitness = acc - ngs

Fitness function

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

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

min = 0

max = 7128

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

Fitness function

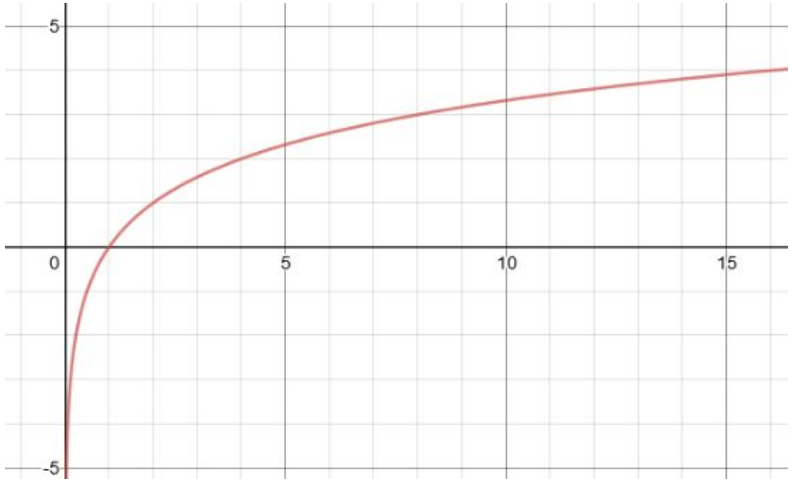
- Assume-se acurácias abaixo de 90% como lixo

  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

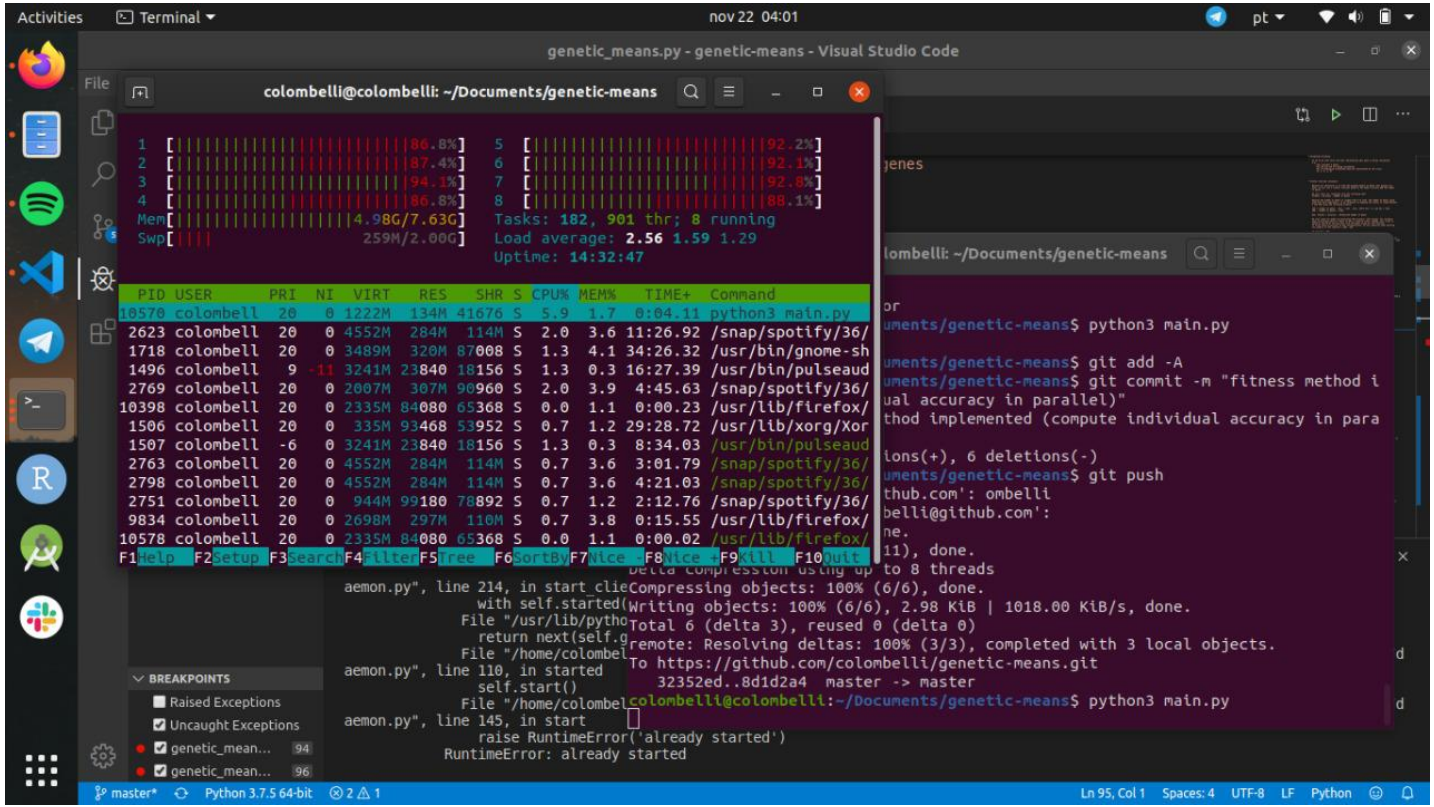
Fitness = accNew - ngsNorm

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



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

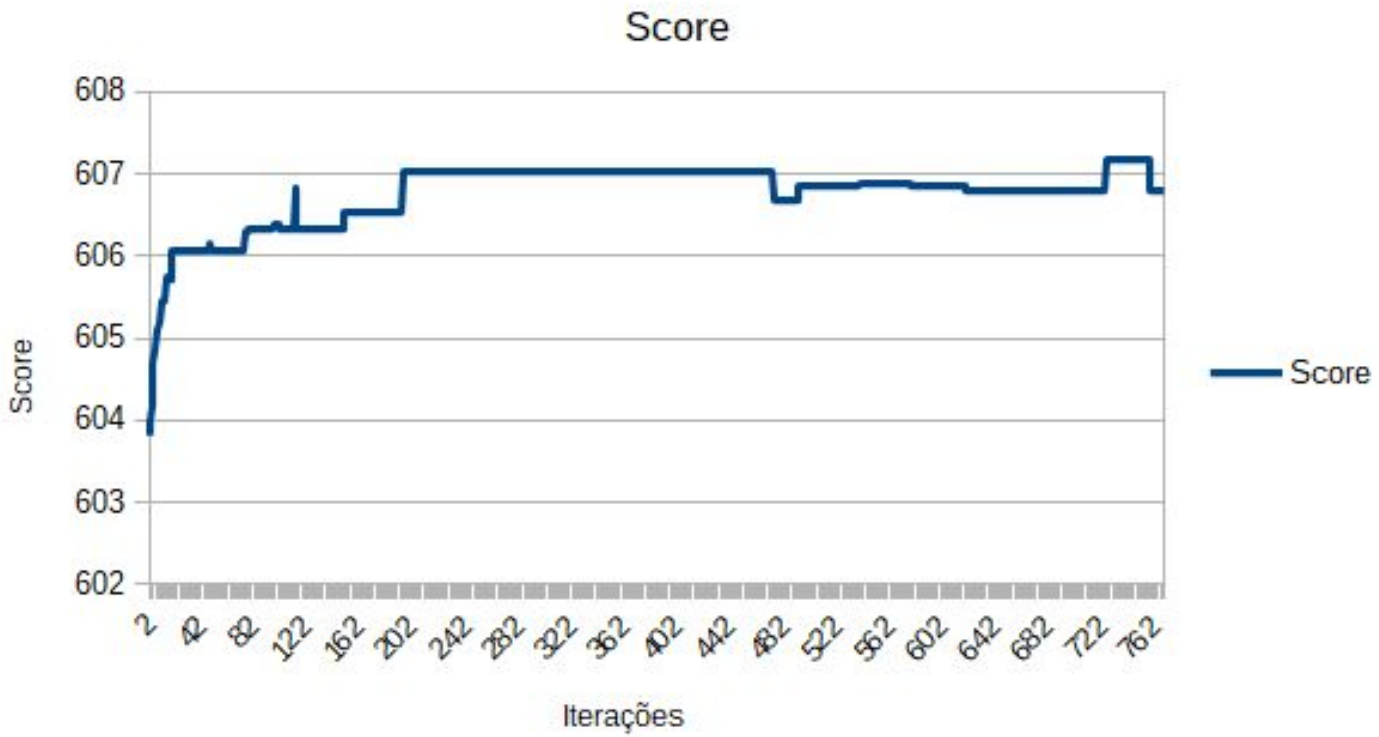
Implementação 1 - Mutação

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

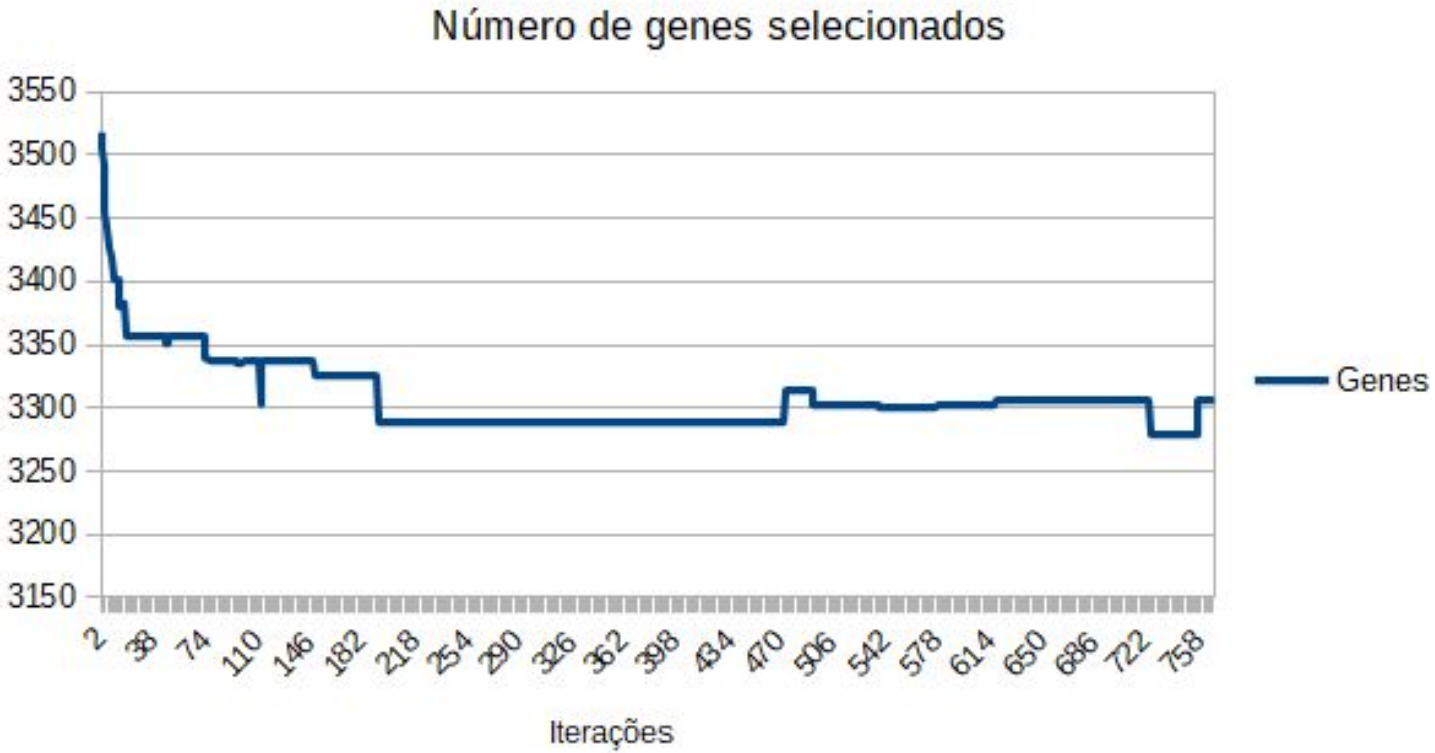
Resultados da Implementação 1

- 767 iterações feitas
- 3306 genes selecionados
- 98.6%

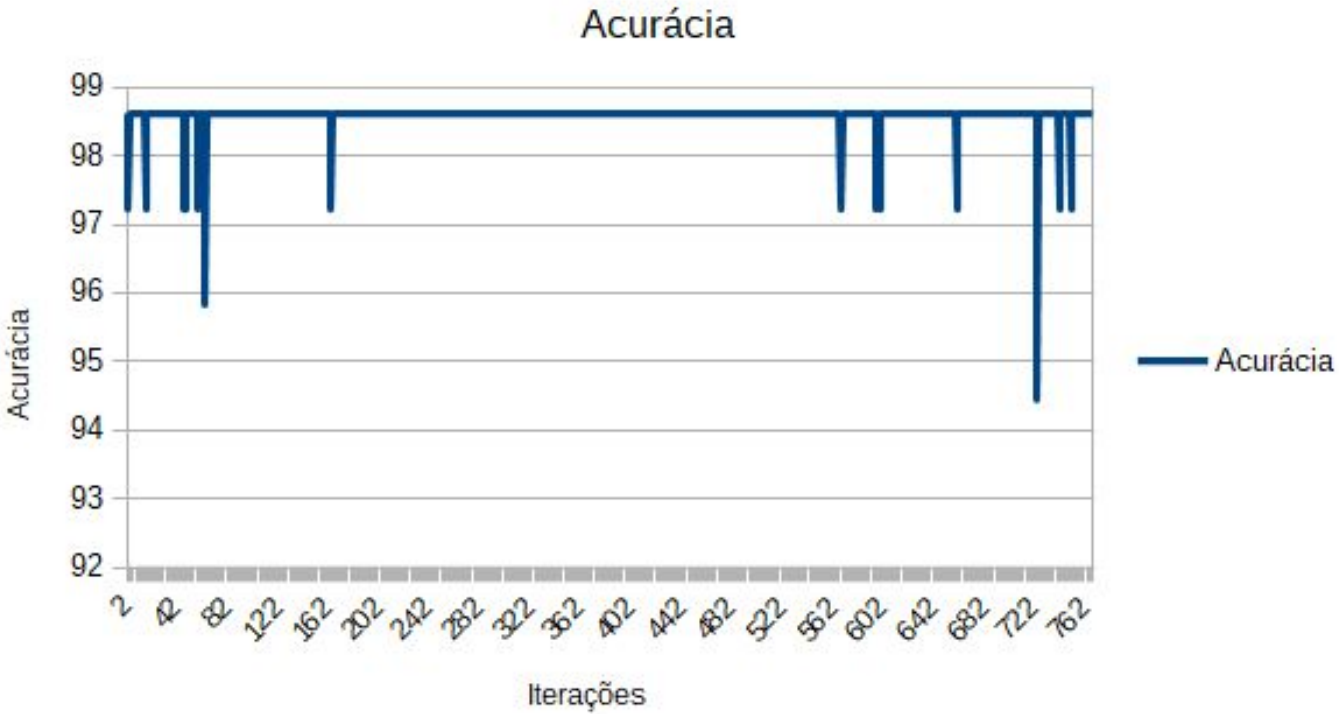
Resultados da Implementação 1



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

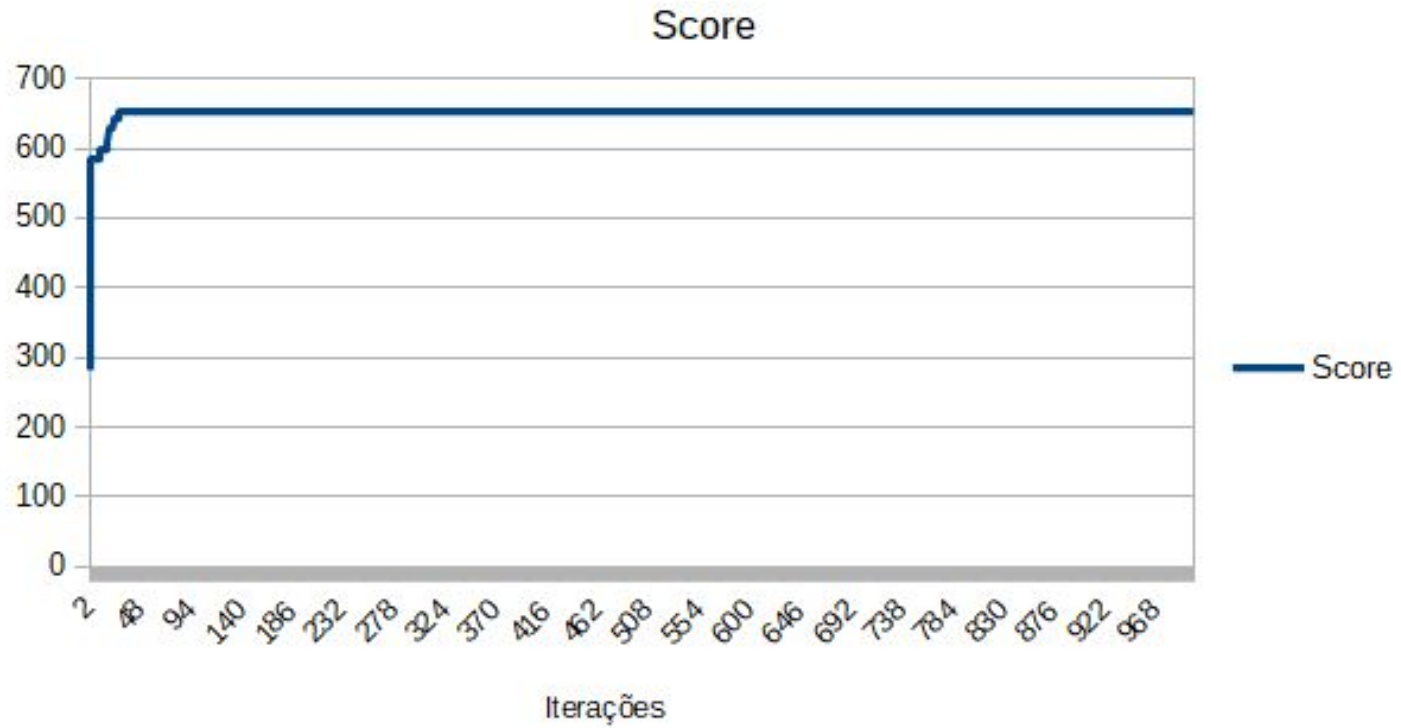
Implementação 2 - Modificação 3

- **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”

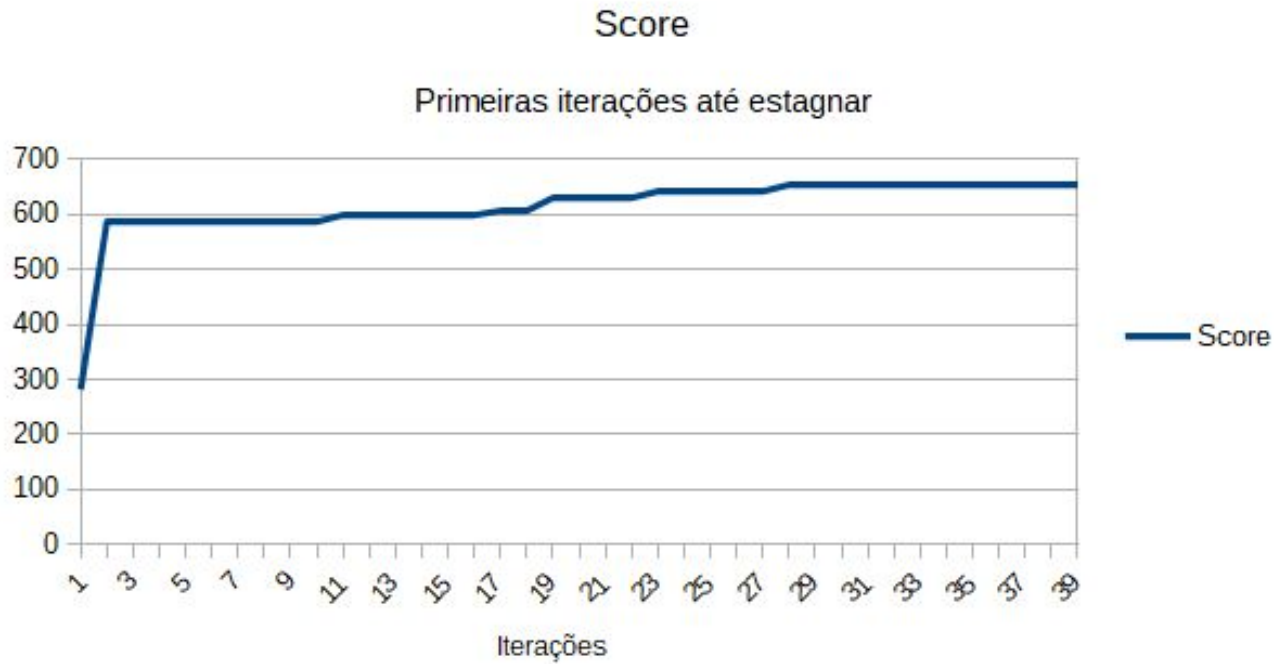
Resultados da Implementação 1

- 1000 iterações feitas
- 42 genes selecionados
- 98.6%

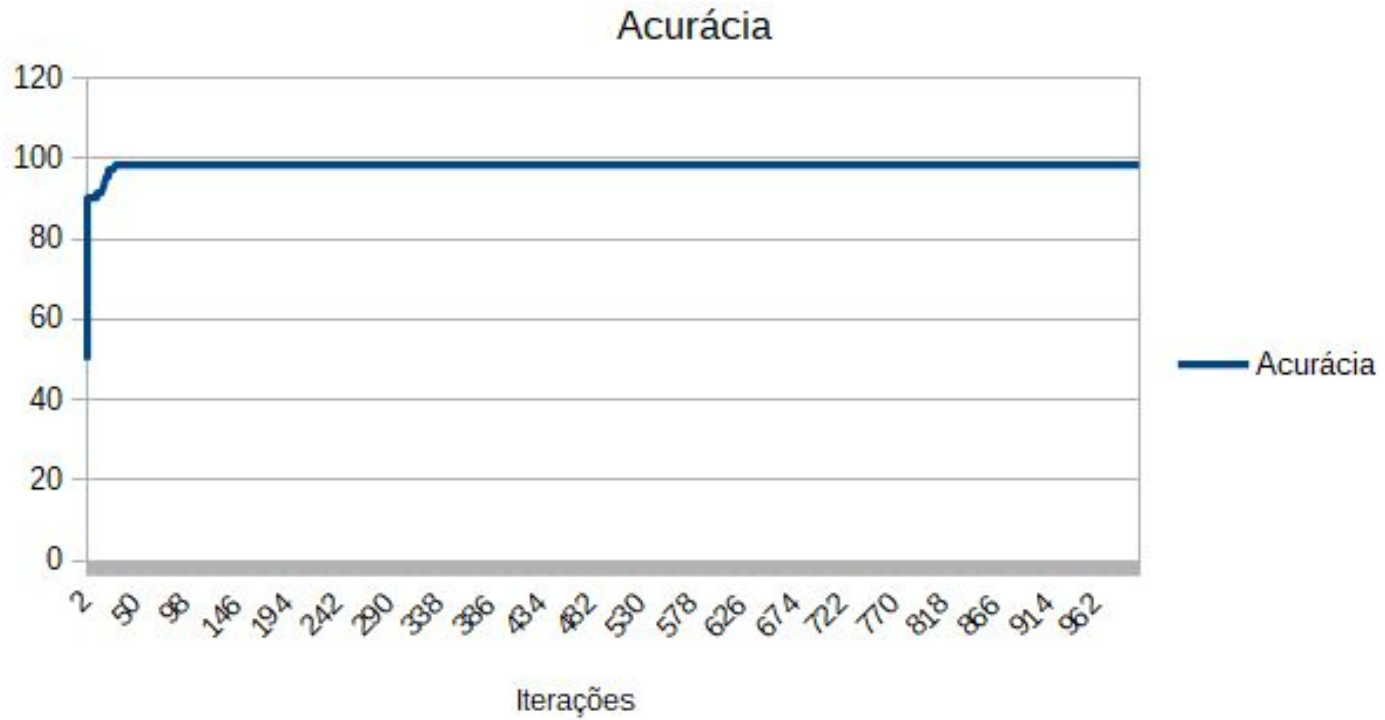
Resultados da Implementação 1



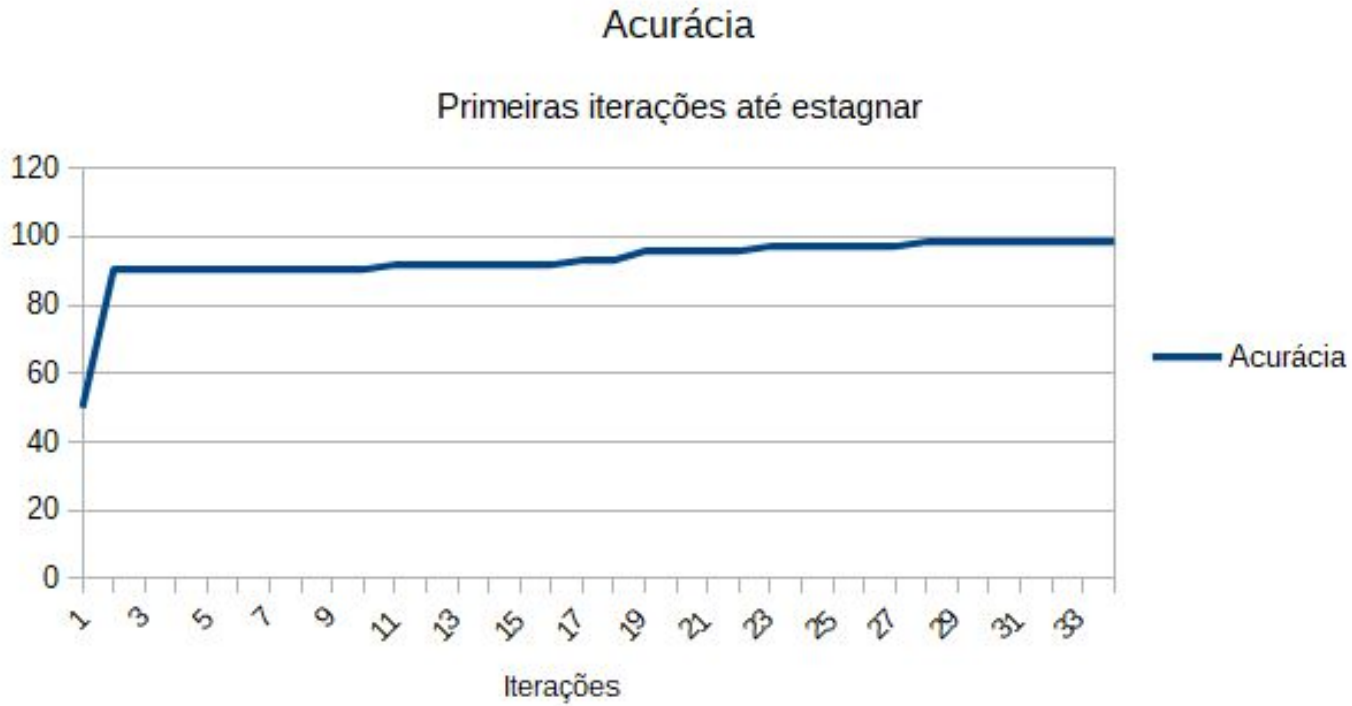
Resultados da Implementação 1



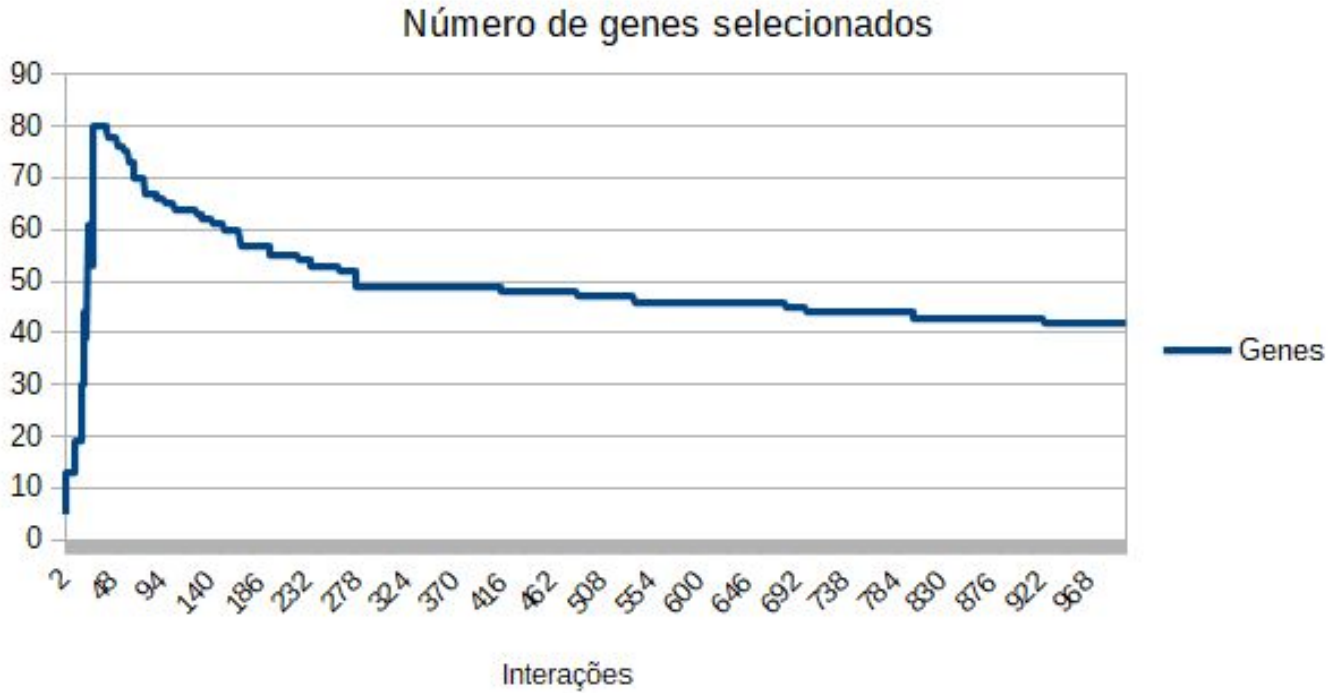
Resultados da Implementação 1



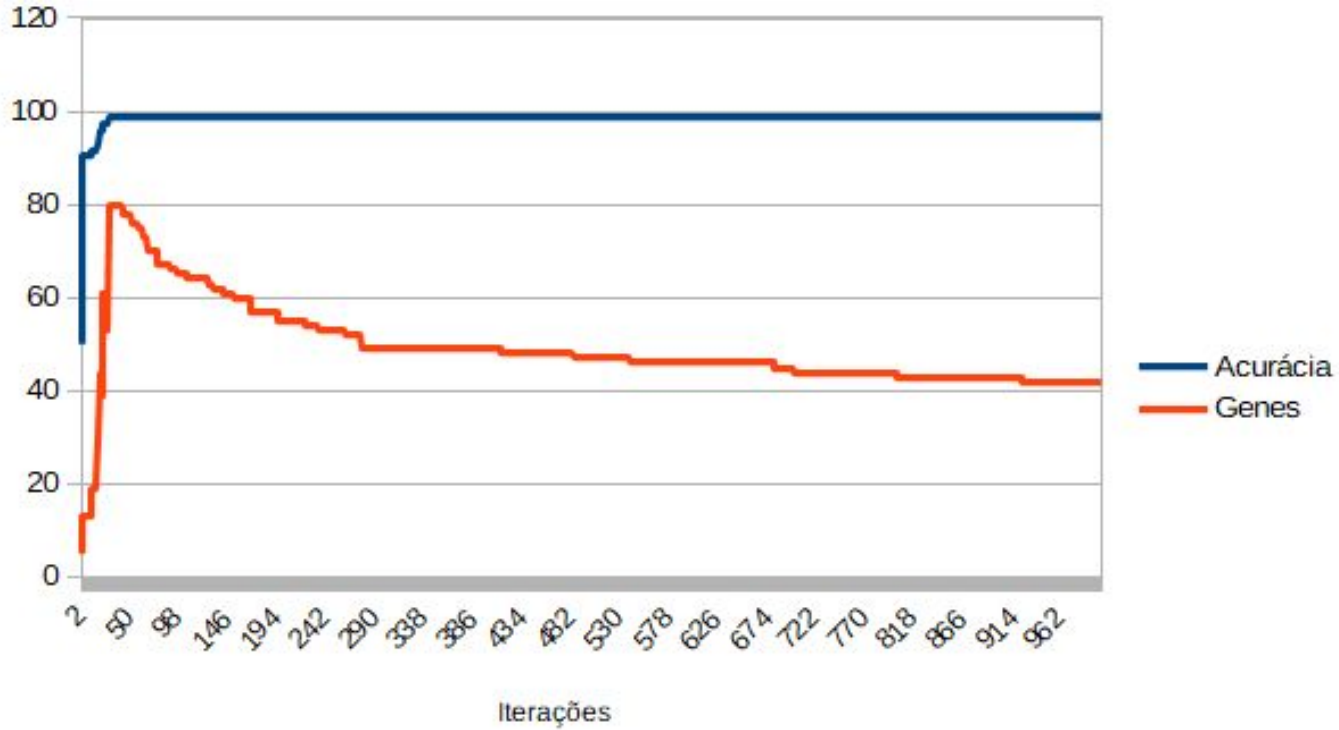
Resultados da Implementação 1



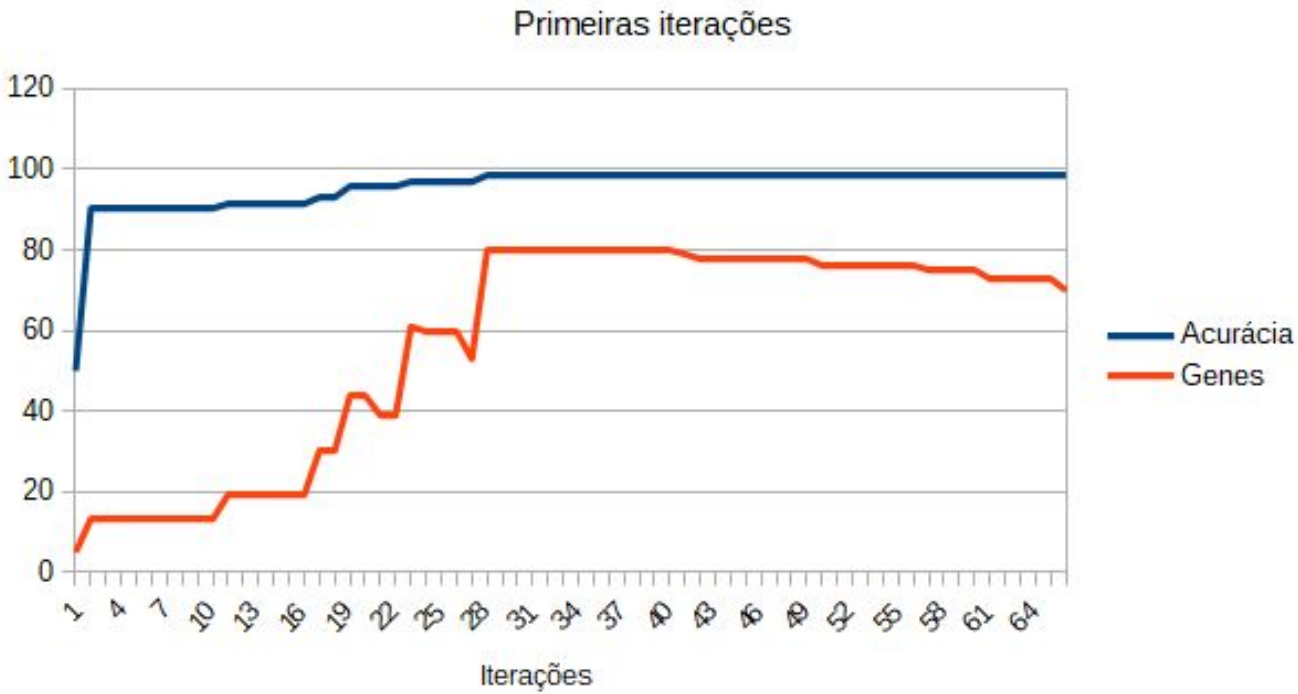
Resultados da Implementação 1



Resultados da Implementação 1



Resultados da Implementação 1



Repo:
https://github.com/colombelli/genetic-means