UNIVERSIDADE ESTADUAL PAULISTA
Programa de Pós-Graduação em Ciência da Computação
Computação Inspirada pela Natureza - **Trabalho 1** - Professor Fabricio Breve
Giovanna Carreira Marinho

Introdução

A Computação Evolutiva é um campo de pesquisa que engloba todos os algoritmos inspirados na evolução (Algoritmos Evolutivos). Entre as principais categorias, destaca-se a Programação Genética, que constitui um tipo de algoritmo evolutivo desenvolvido como uma extensão dos Algoritmos Genéticos. Nele, as estruturas de dados que sofrem adaptação são representações de programas de computador, e assim, a avaliação de aptidão envolve a execução dos programas. Envolve uma busca baseada na evolução do espaço de possíveis programas de computador de forma que, quando executados, produzam uma saída adequada, que geralmente está relacionada à resolução de um determinado problema.

Entre os possíveis problemas que podem ser resolvidos por meio dos conceitos e teorias dos Algoritmos Genéticos, destacam-se a evolução de uma população para: reconhecer um número (representado visualmente por uma imagem e computacionalmente por uma *bitstring*), maximizar ou minimização uma função (de uma ou várias variáveis), entre outros.

No que se refere ao funcionamento, os algoritmos evolutivos padrão mantém uma população P de N indivíduos $P = \{x_1, x_2, \cdots, x_n\}$ em cada iteração t. Cada indivíduo é uma solução potencial para o problema a ser resolvido. Os indivíduos são avaliados em sua medida de adaptação ao ambiente. Dessa forma, uma nova população é gerada na iteração $t + 1$ selecionando alguns (geralmente os mais aptos) dos indivíduos da população e reproduzindo-os. Durante a reprodução, um operador de recombinação genética (*crossover*) pode ser usado, implicando em variações genéticas. Esse processo todo é iterado. O conjunto desses passos (reprodução → variação genética → seleção) constitui o que é chamado de uma geração.

As seções a seguir apresentam o uso dos Algoritmos Genéticos em alguns problemas citados anteriormente. Para o desenvolvimento dos algoritmos, foi utilizada a linguagem de programação *Python*, assim como alguns pacotes de bibliotecas disponíveis. O ambiente de desenvolvimento escolhido foi o *Google Colab*. As implementações podem ser acessadas pelo seguinte link: https://github.com/Giovannacm/nature-inspired-computing.

Reconhecimento de padrões

De modo a evoluir uma população para reconhecer o número 0, representado pela *bitstring* [1 1 1 1 0 1 1 0 1 1 1 1], as seguintes configurações do algoritmo foram utilizadas:

- População (inicializada aleatoriamente) de 8 indivíduos representados por um *vetor* de 12 *bits*;
- Aptidão calculada a partir da *Distância de Hamming* (que contabiliza a quantidade de bits diferentes de um indivíduo para o outro/ideal) e da aptidão ideal de valor 12;
- Seleção pelo método da roleta simples, calculada proporcionalmente a partir das aptidões de cada indivíduo. Esse tipo de seleção aumenta a chance de escolher os indivíduos mais aptos para a próxima geração;
- *Crossover* de 1 ponto (escolhido aleatoriamente) nos indivíduos selecionados;
- Mutação de 1 gene, quando aplicável.

A sequência de passos que constituem uma geração (reprodução → variação genética → seleção) foram executados até que algum indivíduo da população atingisse a aptidão ideal ou o número de iterações ultrapassasse um valor considerável de 100.000 iterações. Dessa forma, a cada execução do algoritmo a quantidade de iteração é variável, uma vez que o algoritmo se baseia em muitos valores aleatórios.

As Figuras 1, 2, 3, 4 e 5, apresentam o resultado de uma execução do algoritmo para diversas configurações de taxa de *crossover* e taxa de mutação. Para cada execução é exibida a aptidão média e melhor a cada iteração do algoritmo. Ainda, como esperado, é possível observar que, a cada iteração, a aptidão tende a aumentar e a *Distância de Hamming* diminui, uma vez que essa é utilizada do cálculo da aptidão a partir do valor ideal.

Como a intenção dessa implementação é observar o comportamento do algoritmo a cada iteração (principalmente no que se refere à aptidão e seleção) com base numa configuração inicial, ele foi executado apenas uma única vez. Dessa forma, como o algoritmo se baseia em muitos valores aleatórios, seu comportamento não se resume ao comportamento de uma única execução. Uma solução para essa análise é executá-lo várias vezes, tal abordagem é explorada na seção - **Minimização de uma função de duas variáveis**.

Figura 1 – Resultado de uma execução com taxa de *crossover* 0.6 e taxa de mutação 0.02.

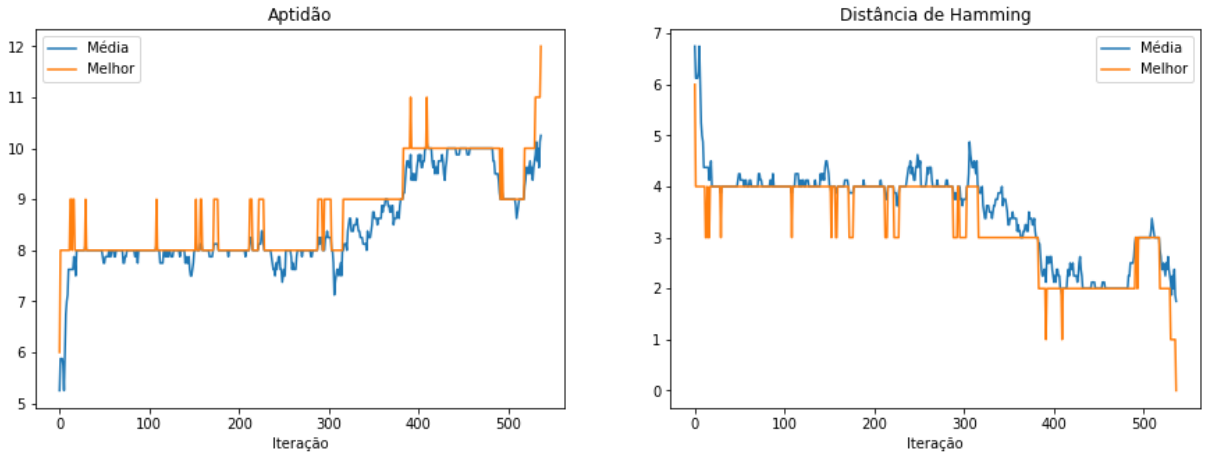


Figura 2 – Resultado de uma execução com taxa de *crossover* 0.6 e taxa de mutação 0.05.

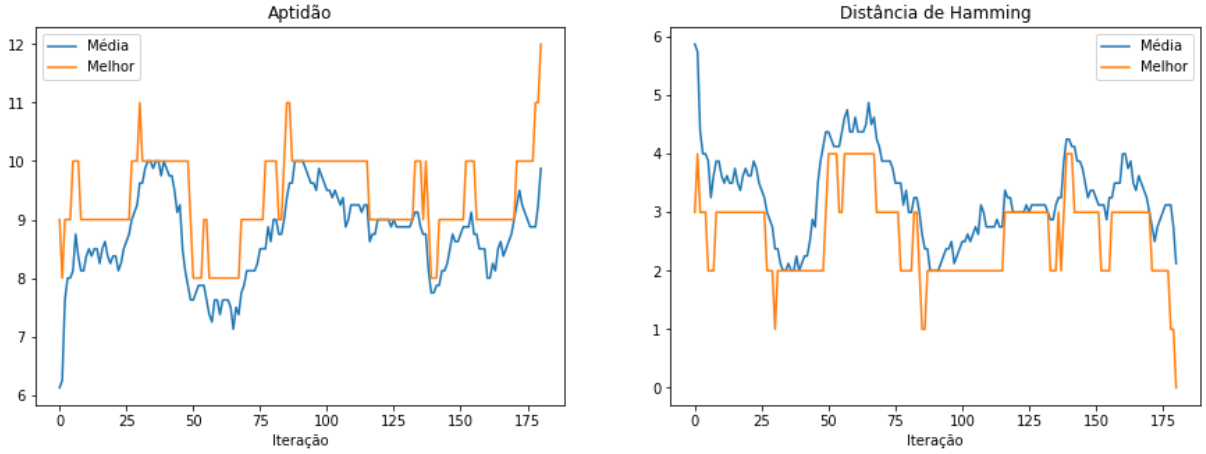


Figura 3 – Resultado de uma execução com taxa de *crossover* 0.6 e taxa de mutação 0.1.

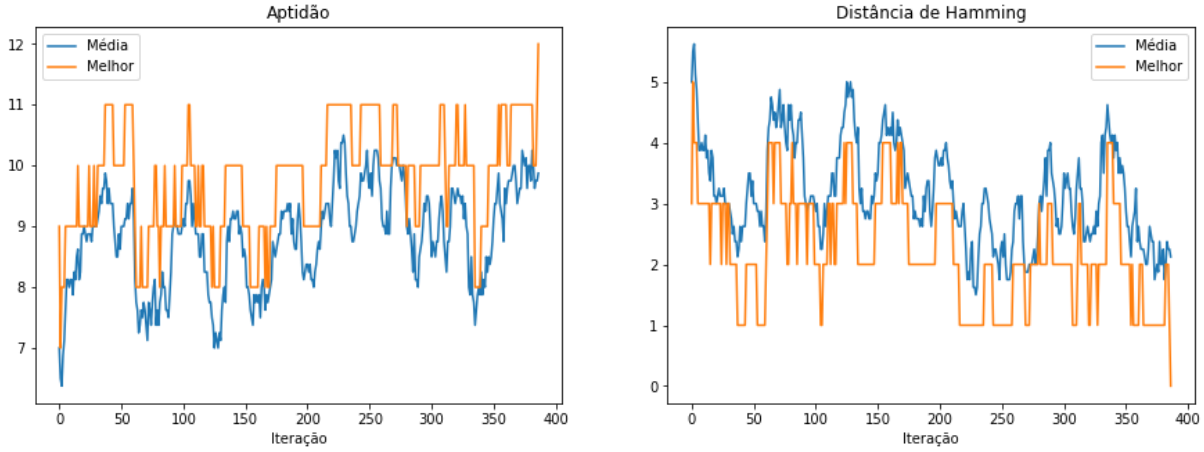


Figura 4 – Resultado de uma execução com taxa de *crossover* 0.8 e taxa de mutação 0.05.

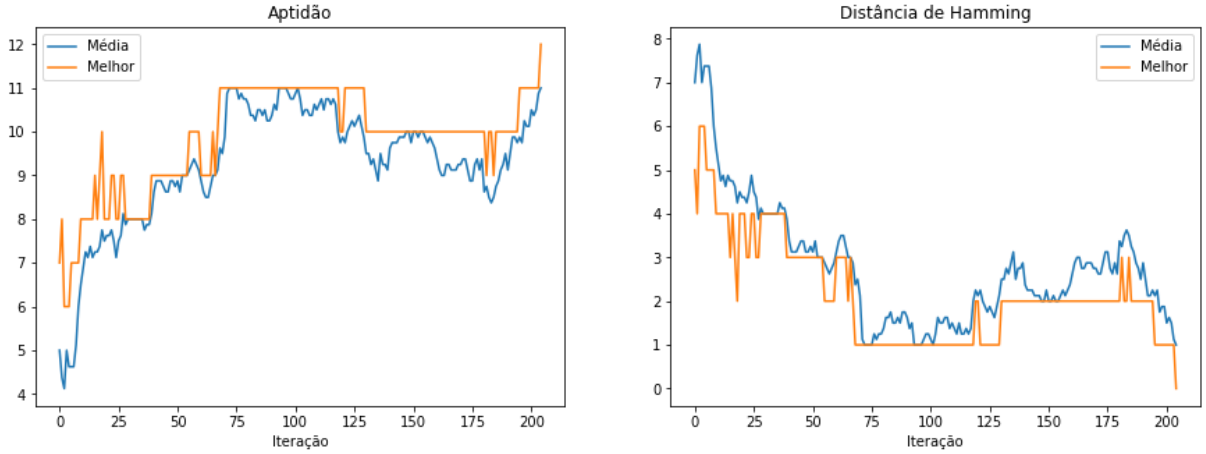
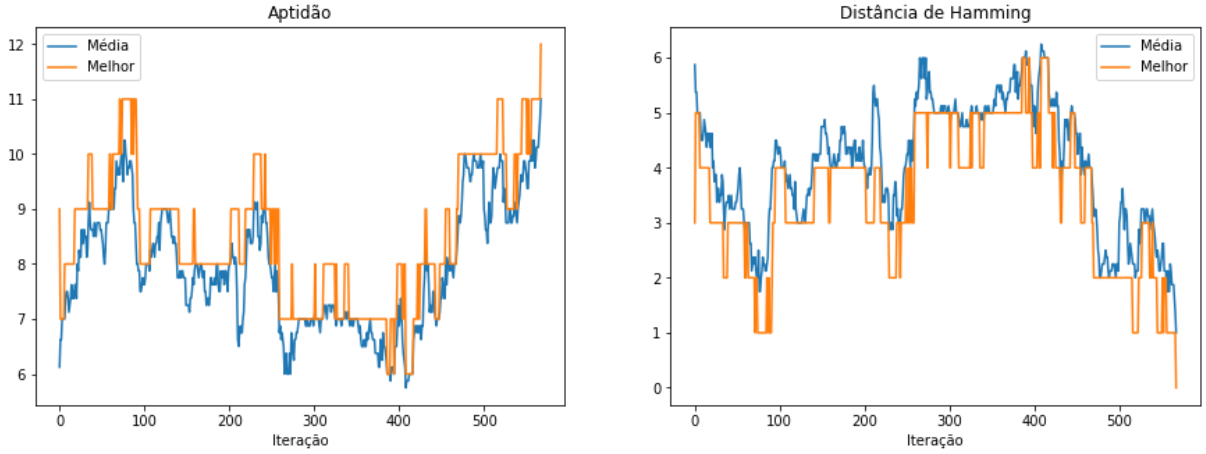


Figura 5 – Resultado de uma execução com taxa de *crossover* 0.4 e taxa de mutação 0.05.



Observa-se que a quantidade de iterações para atingir a aptidão ideal é variável e a população evolui a cada geração até chegar na aptidão desejada, mesmo esse processo não sendo linear em todas as configurações. Entretanto, um fato a ser analisado é em relação aos “picos” em cada uma das configurações apresentadas. Existe um comportamento diferente à cada valor das taxas de *crossover* e mutação. Mantendo a taxa de *crossover* e variando a taxa de mutação (Figura 1, 2 e 3) observa-se que a aptidão altera de forma mais brusca ao aumentar a taxa de mutação, uma vez que a chance de modificar um gene fica maior e essa mudança pode implicar num valor fora do comportamento de gerações anteriores, aumentando o espaço de busca, gerando mais perturbação. Ao manter a taxa de mutação fixa e variar a taxa de *crossover* (Figura 2, 4 e 5) observa-se que ao aumentar esse valor, como novos indivíduos serão gerados com mais chance a partir da troca genética de soluções anteriores (que podem estar convergindo para a melhor solução), a melhor solução pode ser enviada para as próximas gerações podendo implicar numa convergência mais rápida.

As Figuras 6, 7 e 8 apresentam o comportamento da execução durante as iterações ao manter apenas a taxa de *crossover* e mutação, respectivamente.

Figura 6 – Resultado de uma execução com taxa de *crossover* 0.6 e sem taxa de mutação.

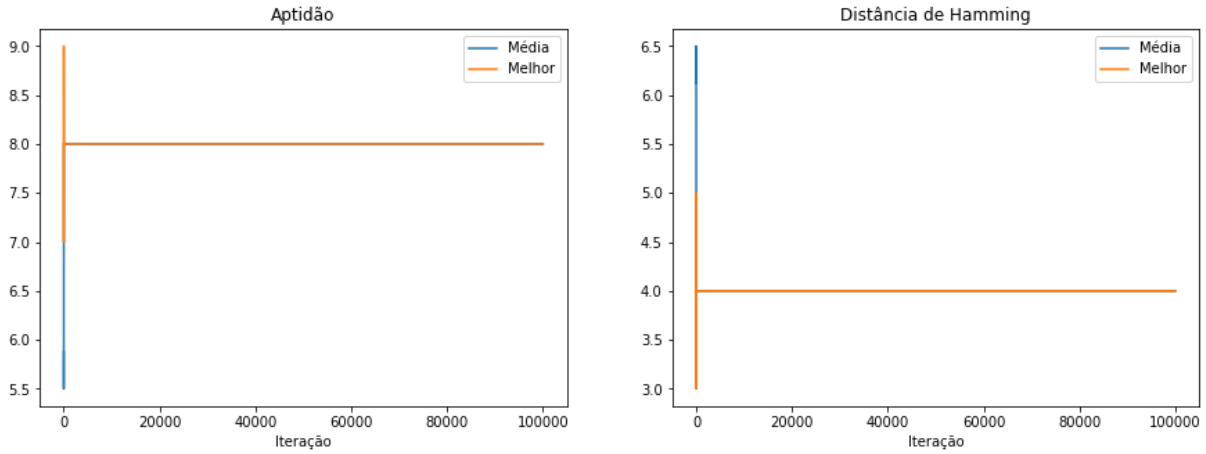


Figura 7 – Resultado de uma execução sem taxa de *crossover* e taxa de mutação 0.1.

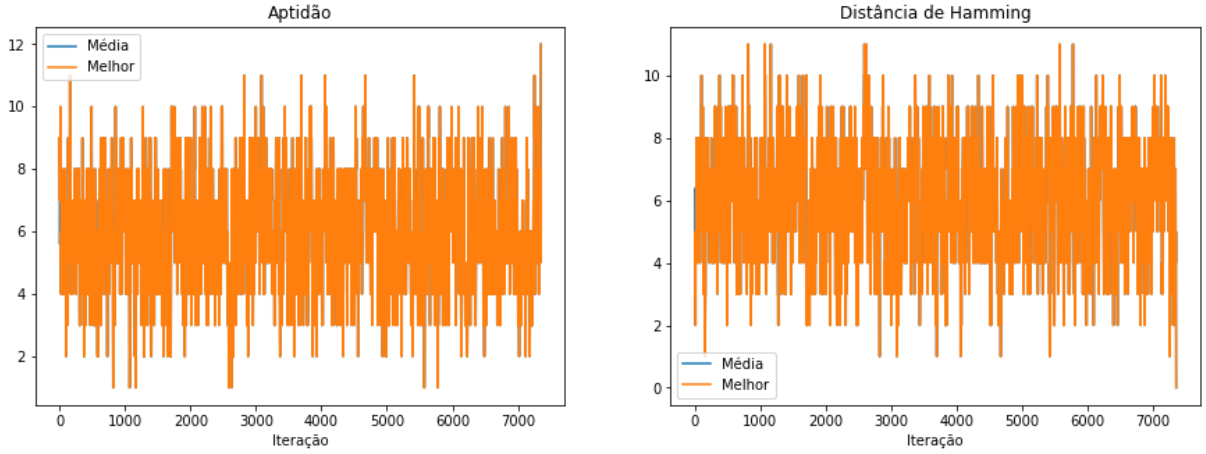
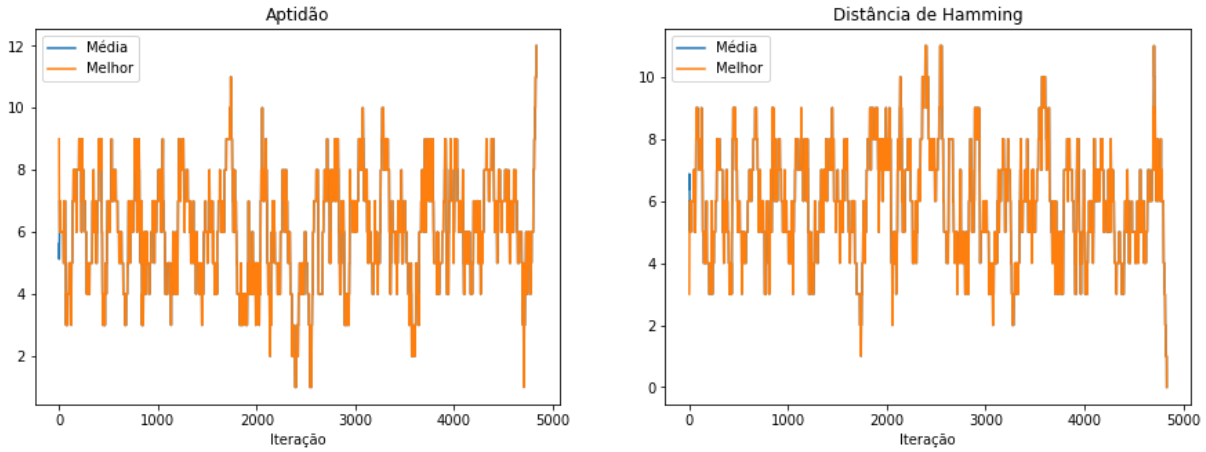


Figura 8 – Resultado de uma execução sem taxa de *crossover* e taxa de mutação 0.02.



A Figura 6 mostra que apenas o *crossover* não é suficiente para chegar na solução ideal, uma vez que o algoritmo não convergiu, apresentou uma aptidão constante (que era diferente da ideal) e chegou no limite máximo de iterações. Já as outras duas Figuras 7 e 8 mostram que a solução foi encontrada e a diminuição desse valor implica numa menor variabilidade de indivíduos. Dessa forma, conclui-se que só a mutação, por meio da busca de indivíduos fora do espaço, pode ser capaz de encontrar a solução ideal, já só o *crossover* não é suficiente, uma vez que os indivíduos possuem pouca variância, diminuindo o espaço de busca e convergindo num valor que pode não ser o ótimo local, pois outras soluções não são exploradas.

Outro ponto a ser observado comparando as Figuras 1 a 8 é em relação à quantidade de iterações até atingir o resultado final. As diferentes dosagens das taxas de mutação e *crossover* influenciam significativamente nesse valor.

Maximização de uma função de uma variável

Além da aplicação anterior, destaca-se o uso dos Algoritmos Genéticos para maximização de uma função. A implementação apresentada nessa seção maximiza a função $g(x) =$

$2^{-2((x-0.1)/0.9)^2}(\sin(5\pi x))^6$, no intervalo $[0, 1]$, cujo valor máximo é 0.1 (a partir da análise do gráfico da função). Tal valor é desconhecido pelo algoritmo. Para esse problema, as configurações a seguir foram utilizadas:

- População de tamanho variável onde cada indivíduo é representado por meio de *bitstring* de 16 bits. Cada *bitstring* corresponde ao valor real da variável, tal conversão é feita pelo código de Gray. Dessa forma, o *bitstring* é convertido para real antes a avaliação da aptidão, e os respectivos valores binários são utilizados para as outras operações;
- Aptidão baseada em ranking (com base na avaliação da função no ponto);
- Seleções diversas: método de roleta simples, torneio e amostragem estocástica universal - SUS;
- *Crossover* de 2 pontos (escolhidos aleatoriamente) nos indivíduos selecionados;
- Mutação de 1 gene, quando aplicável.

A condição de parada nessa implementação é determinada por meio de uma quantidade máxima de iterações apenas. Assim como dito no exemplo anterior, como o algoritmo se baseia em vários valores aleatórios, a quantidade de iteração é variável a cada execução.

Também, foi feita apenas uma única execução do algoritmo em determinadas configurações para analisar o impacto de cada método de seleção e comparar o comportamento dessa execução com a dos algoritmos Subida da Colina e Recozimento Simulado.

As Figuras 9 a 17 apresentam diferentes configurações para analisar o impacto do tamanho da população e o comportamento dos métodos de seleção. Todas elas utilizam 500 iterações máximas, uma taxa de *crossover* de 0.6 e mutação de 0.02.

Figura 9 – Resultado de uma execução com população de 10 indivíduos e seleção por roleta simples.

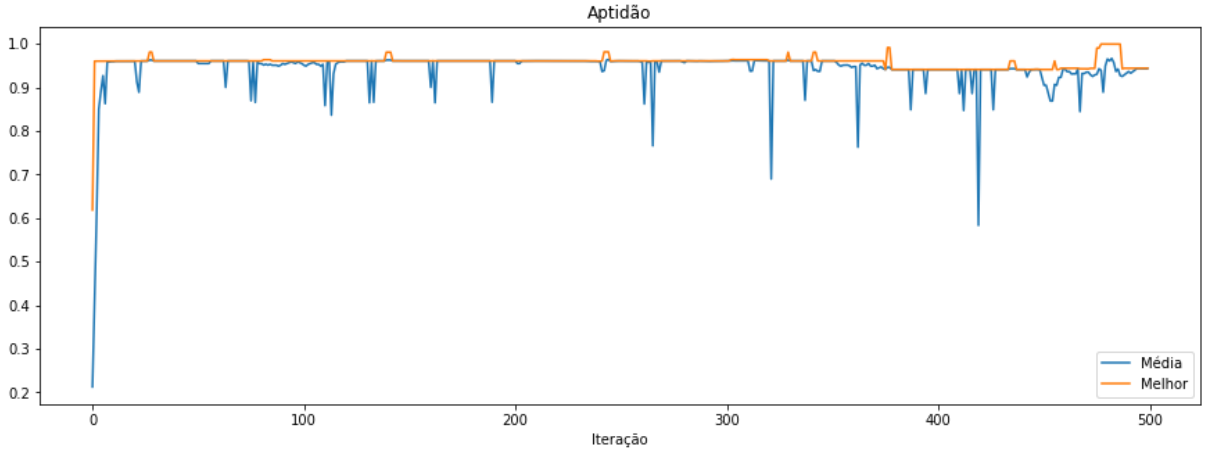


Figura 10 – Resultado de uma execução com população de 10 indivíduos e seleção por torneio.

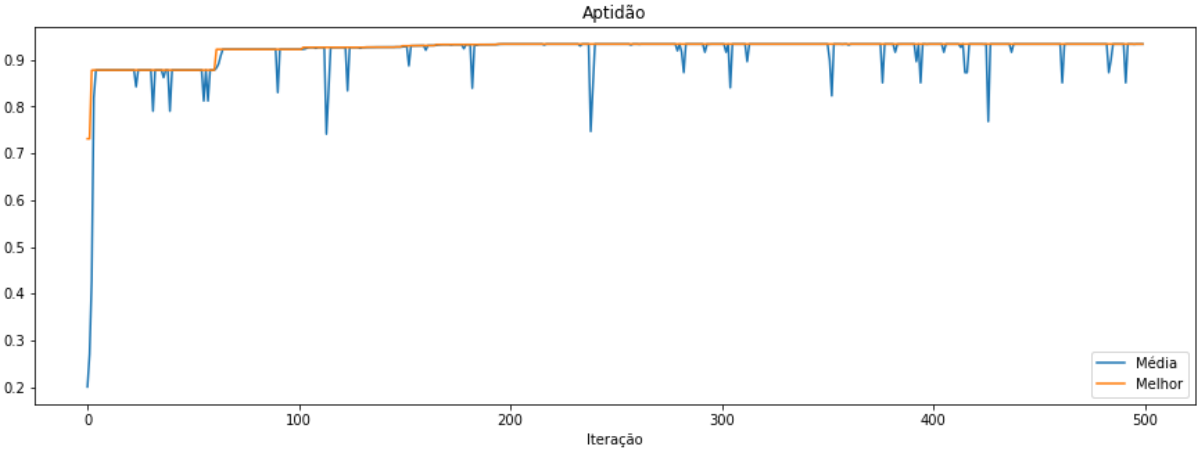


Figura 11 – Resultado de uma execução com população de 10 indivíduos e seleção por SUS.

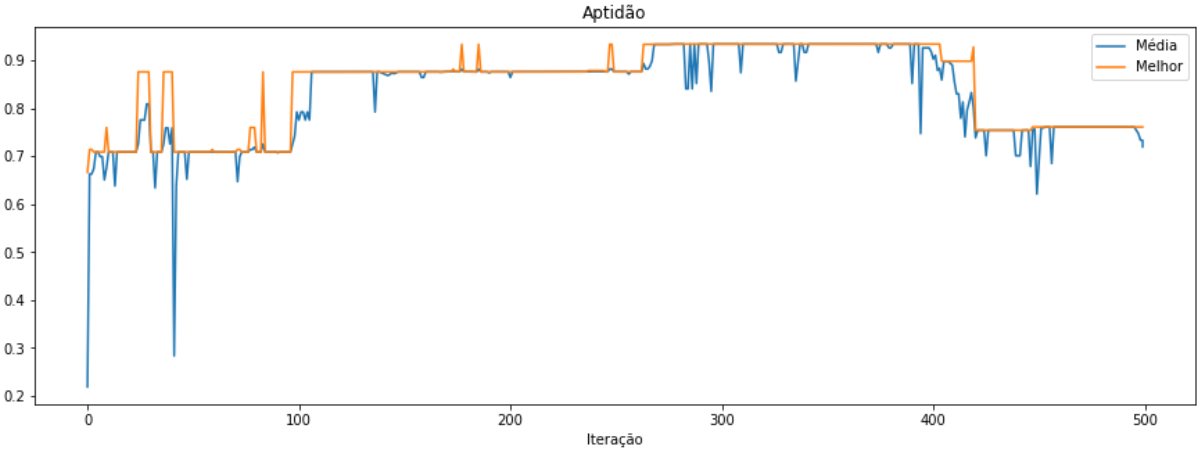


Figura 12 – Resultado de uma execução com população de 20 indivíduos e seleção por roleta simples.

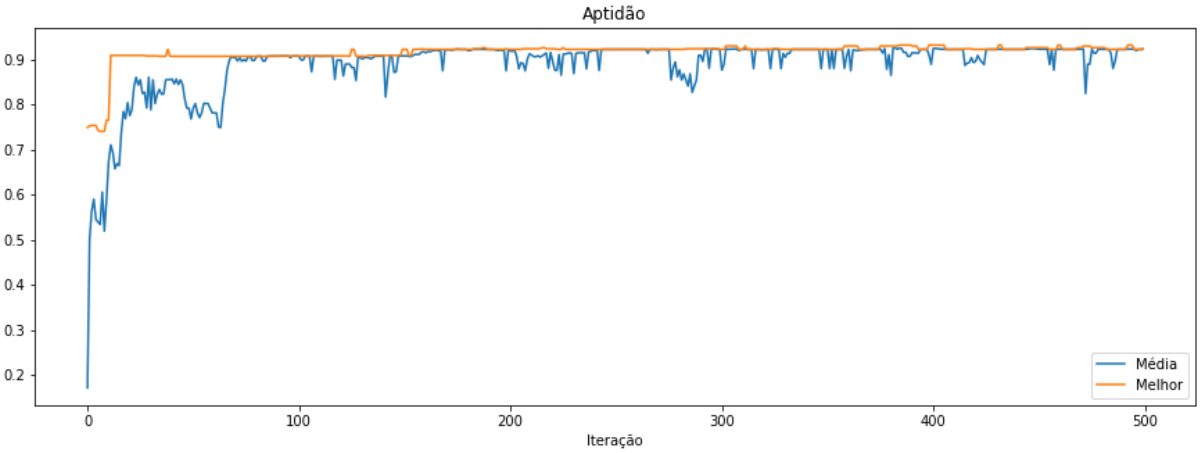


Figura 13 – Resultado de uma execução com população de 20 indivíduos e seleção por SUS.

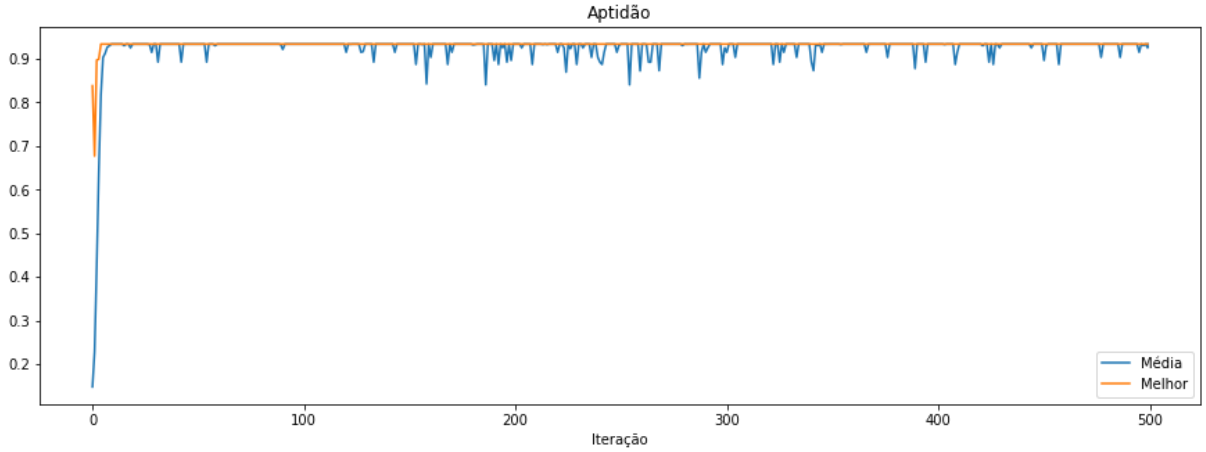


Figura 14 – Resultado de uma execução com população de 20 indivíduos e seleção por SUS.

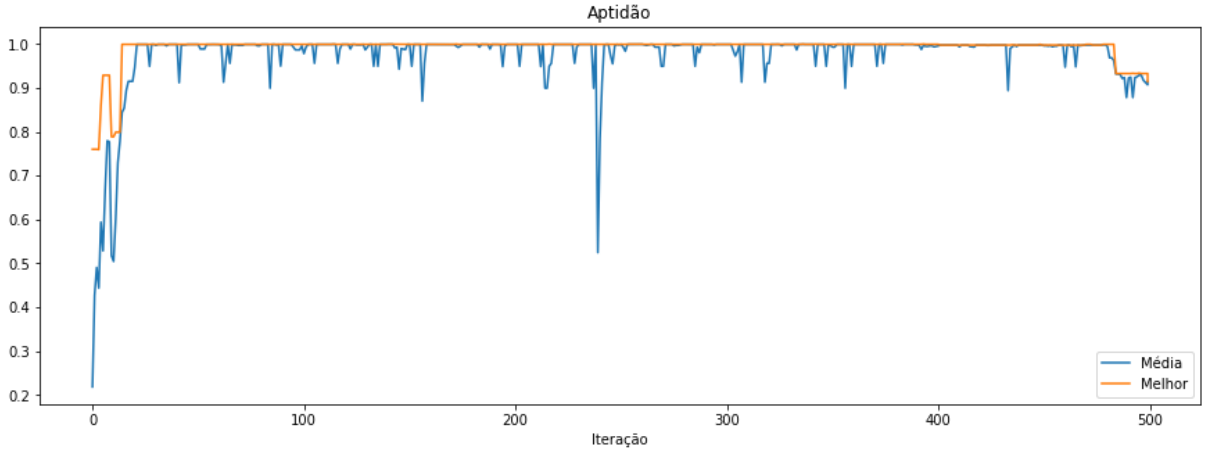


Figura 15 – Resultado de uma execução com população de 50 indivíduos e seleção por roleta simples.

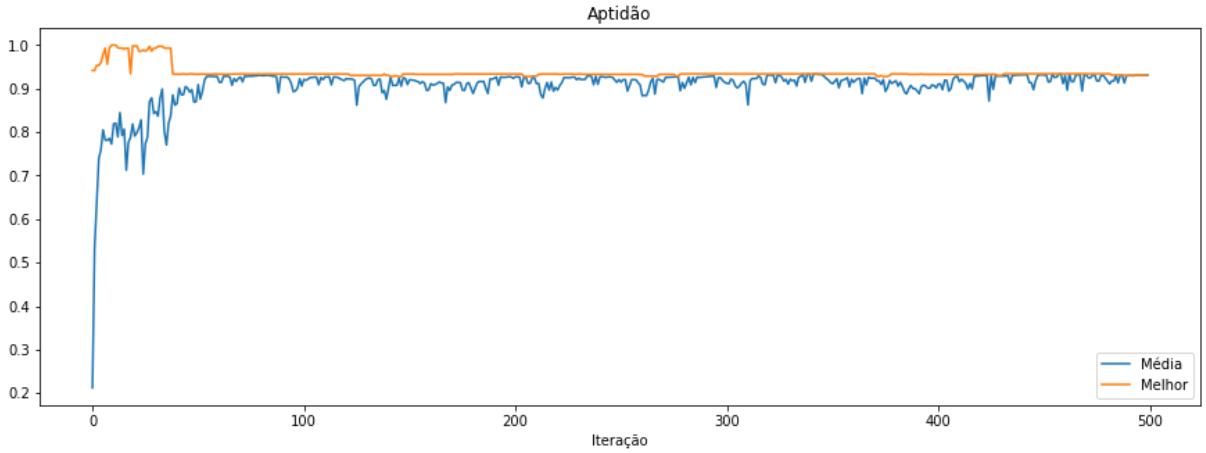


Figura 16 – Resultado de uma execução com população de 50 indivíduos e seleção por torneio.

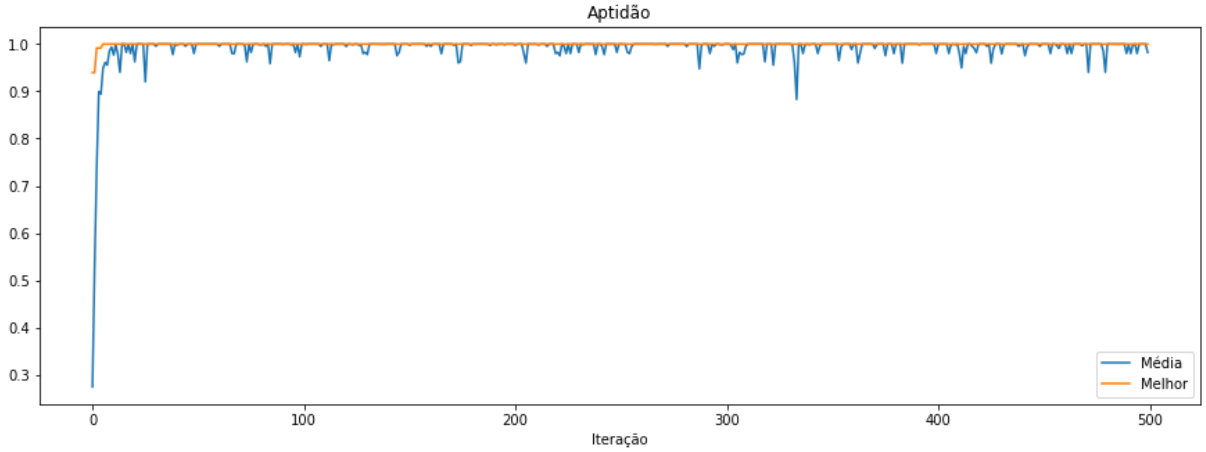
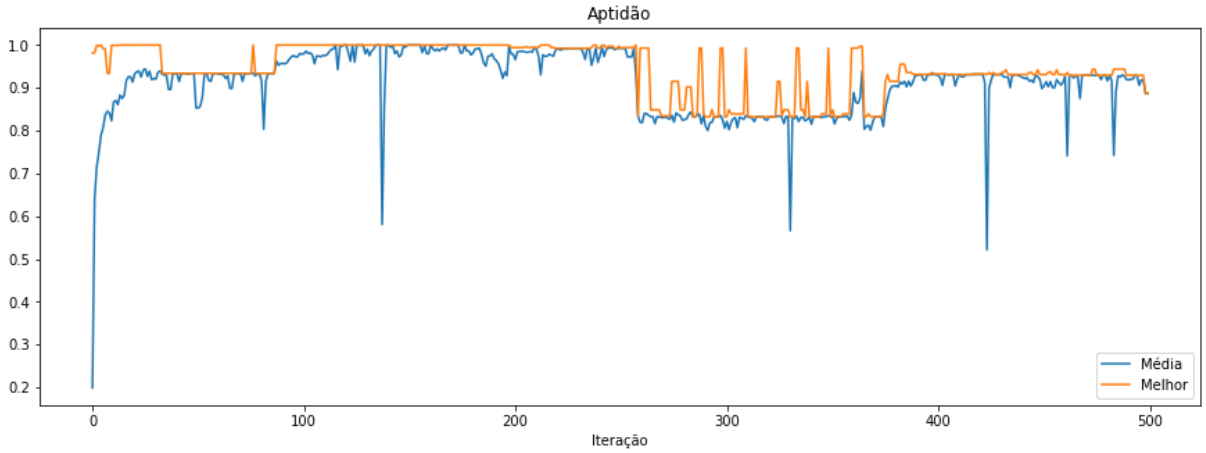


Figura 17 – Resultado de uma execução com população de 50 indivíduos e seleção por SUS.



Analisando os métodos de seleção em cada quantidade de indivíduos, observa-se que as diferentes combinações desses parâmetros implicam em uma convergência mais rápida para atingir o resultado final, além de um comportamento diferente nos valores da aptidão média e melhor. Outro fenômeno interessante a ser observado é o equilíbrio entre a pressão seletiva e diversidade. Quando essa pressão é baixa, ou seja, os melhores indivíduos não são favorecidos, a convergência é demorada e a solução ótima demora para ser encontrada. Quando esse favorecimento é elevado, a convergência é muito prematura e é necessário controlar a taxa de *crossover* de cada indivíduo. A diversidade, pode ser promovida por meio do tamanho da população ou da taxa de mutação. As Figuras 9 a 17 mostram a necessidade desse equilíbrio de acordo com a variação do tamanho da população e do método de seleção. O aumento da população implica, também, em um aumento do tempo de execução, uma vez que a quantidade de operações a serem feitas a cada geração é proporcional a esse valor.

Ao comparar o desempenho do Algoritmo Genético com os algoritmos de Subida da Colina e Recozimento Simulado, é possível observar o comportamento de cada um dos métodos. Como todos esses algoritmos baseiam em valores aleatórios, o resultado apresentado nas

Figuras 18 a 21 é aquele obtido por uma única execução, que pode ou não ser a melhor configuração. Todos eles foram executados com 500 iterações e as configurações utilizadas estão apresentadas na legenda de cada Figura.

Figura 18 – Resultado de uma execução do Algoritmo Genético com população de 50, taxa de *crossover* de 0.6, taxa de mutação de 0.02 e seleção por roleta simples. Valor da melhor aptidão da última geração: 0.9939434248782348.

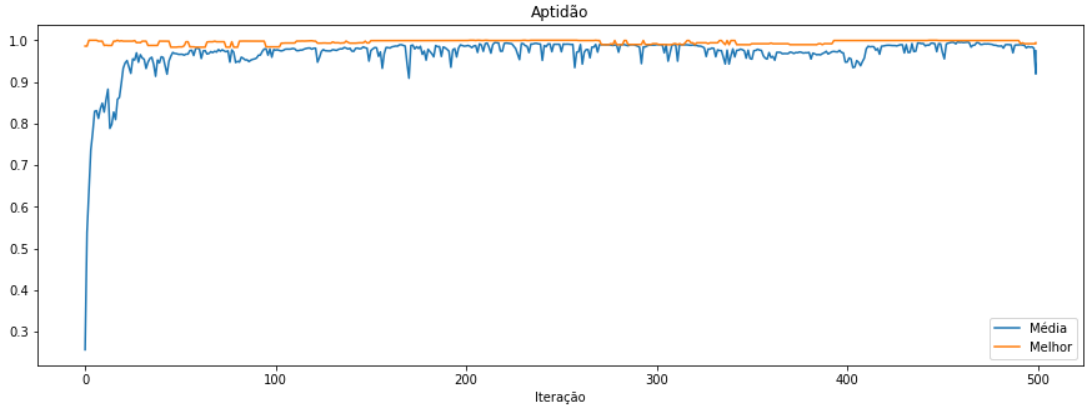


Figura 19 – Resultado de uma execução da Subida da Colina Simples. Valor da aptidão da última geração: 0.7609365884240712.

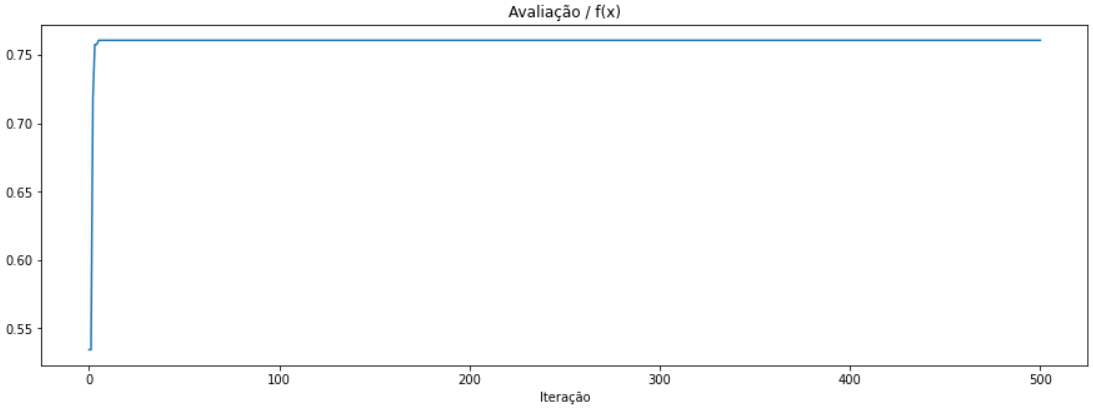


Figura 20 – Resultado de uma execução da Subida da Colina Iterativo com 10 iterações. Valor da aptidão da última geração: 0.9999999018807753.

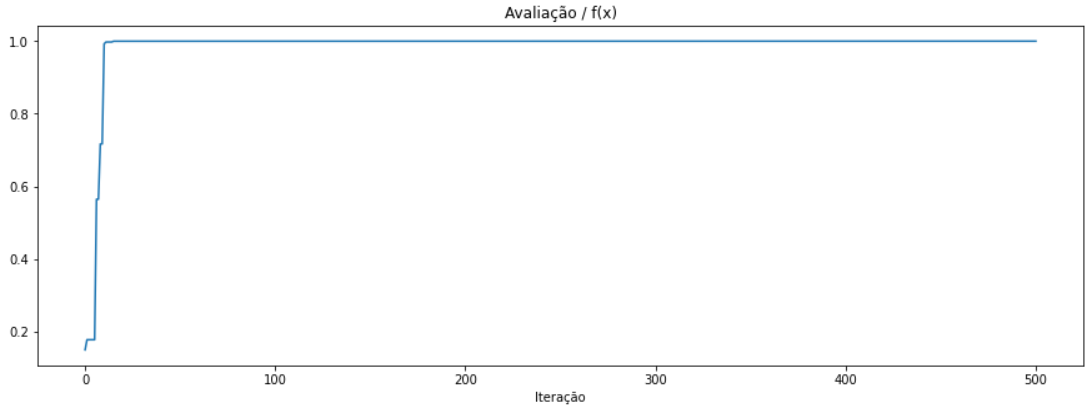
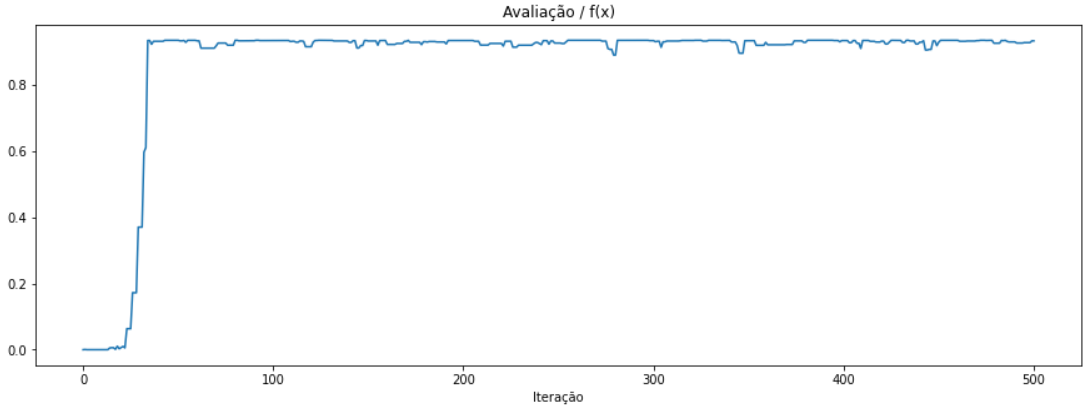


Figura 21 – Resultado de uma execução do Recozimento Simulado com $\beta = 0.01$ e temperatura 10. Valor da aptidão da última geração: 0.9321938992821791.



No que se refere ao funcionamento de cada algoritmo, analisando as Figuras 18 a 21, observa-se que para o Subida da Colina (Simples e Iterativo), ele converge para uma solução e a leva de forma constante para as próximas iterações. Já o Algoritmo Genético e Recozimento simulado apresentam perturbações durante as iterações, o primeiro, por conta das seleções, *crossover* e mutação, e, o segundo, pela sua perturbação no ponto mesmo quando a solução não seja melhor que a anterior. A Tabela 1 apresenta apenas a aptidão obtida em outros dois testes (além do apresentado pelas Figuras 18 a 21 – Teste 1). A segunda coluna mostra a melhor aptidão na última iteração e a terceira coluna, a melhor aptidão obtida durante todo o processo da execução.

Tabela 1 – Resultado numérico dos testes.

Algoritmo	**Melhor aptidão na última iteração**	**Melhor aptidão obtida durante as iterações**
Teste 1: 500 iterações		
Algoritmo Genético	0.9939434248782348	0.9999996075181437
Subida da Colina Simples	0.7609365884240712	0.7609365884240712
Subida da Colina Iterativo	0.9999999018807753	0.9999999018807753
Recozimento Simulado	0.9321938992821791	0.933979189228703
Teste 2: 200 iterações		
Algoritmo Genético	0.9996729328157673	0.9999999525163652
Subida da Colina Simples	0.9339792119135572	0.9339792119135572
Subida da Colina Iterativo	0.9999997906396924	0.9999997906396924
Recozimento Simulado	0.3089845782334643	0.3352679471935237
Teste 3: 1000 iterações		
Algoritmo Genético	0.9999927283498976	0.9999996075181437
Subida da Colina Simples	0.9999971957512581	0.9999971957512581
Subida da Colina Iterativo	0.9339776581357375	0.9339776581357375
Recozimento Simulado	0.9939991956877354	0.9999999372964001

Observa-se que as soluções são precisas. Devido ao comportamento citado anteriormente, a melhor aptidão da última geração do Algoritmo Genético e Recozimento Simulado, pode não ser a melhor obtida durante o processo da execução. Como os algoritmos dependem de valores aleatórios, a mesma configuração pode implicar em resultados diferentes. A seção a

seguir - **Minimização de uma função de duas variáveis**, analisa o Algoritmo Genético de outra maneira, uma vez que o mesmo é executado várias vezes para estudar seu comportamento e tempo de execução.

Minimização de uma função de duas variáveis

Por fim, a última aplicação dos Algoritmos Genéticos explorada nessa implementação é a minimização de uma função de duas variáveis no intervalo $[-10, 10]$: $f(x, y) = (1-x)^2 + 100(y-x^2)^2$. Analisando o gráfico da função, o valor mínimo é 0, entretanto, esse valor é desconhecido no algoritmo. Para isso, as seguintes configurações foram utilizadas:

- População de tamanho variável onde cada indivíduo é representado por meio de *bitstring* de 16 bits. Cada *bitstring* corresponde ao valor real da variável, tal conversão é feita pelo código de Gray. O primeiro bit corresponde ao sinal (1 para valores positivos, 0 para valores negativos). Dessa forma, o *bitstring* é convertido para real antes a avaliação da aptidão, e os respectivos valores binários são utilizados para as outras operações;
- Uma vez que a função é de duas variáveis, é utilizada uma população para x e outra para y, as operações são aplicadas nas suas populações;
- Aptidão baseada em ranking (com base na avaliação da função no ponto);
- Seleção por torneio;
- Crossover de 2 pontos;
- Mutação de 1 gene.

A condição de parada, nessa implementação específica, foi iterar até que um número máximo de iterações fosse atingido ou iterar mais de $1/3$ do número máximo de iterações sem encontrar uma solução melhor que a obtida até o momento. Essa última condição implica numa melhor performance do algoritmo. Diferentemente das duas implementações anteriores, essa implementação ainda executa o algoritmo várias vezes para estudar seu comportamento geral e informa o tempo final de execução.

As Figuras 22, 23 e 24 apresentam o resultado de uma execução do algoritmo considerando taxa de *crossocover* 0.6, taxa de mutação 0.02 e 1000 iterações para diferentes tamanhos da população. A ideia desse teste é mostrar o impacto desse valor no tempo de execução do algoritmo.

Figura 22 – Resultado de uma execução do algoritmo com uma população de 30 indivíduos (0.6003279685974121 segundos com 354 iterações). O melhor resultado obtido é 4.655350225216008 na iteração 19.

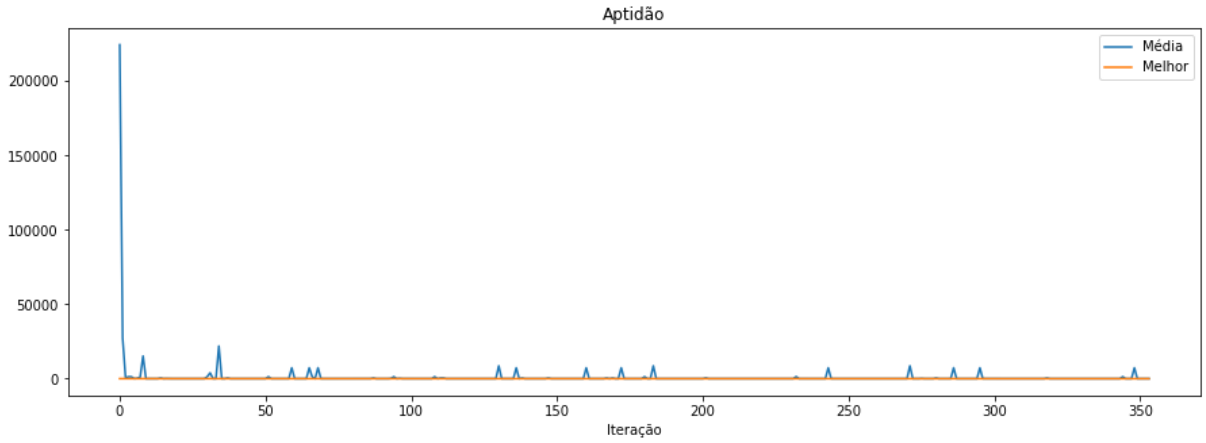


Figura 23 – Resultado de uma execução do algoritmo com uma população de 50 indivíduos (0.9468283653259277 segundos com 410 iterações). O melhor resultado obtido é 2.1312620263426125 na iteração 75.

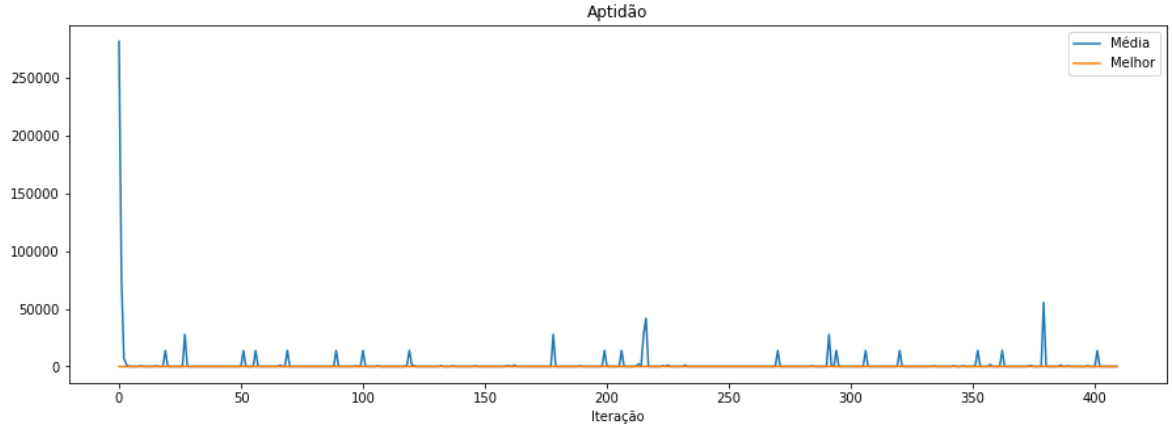
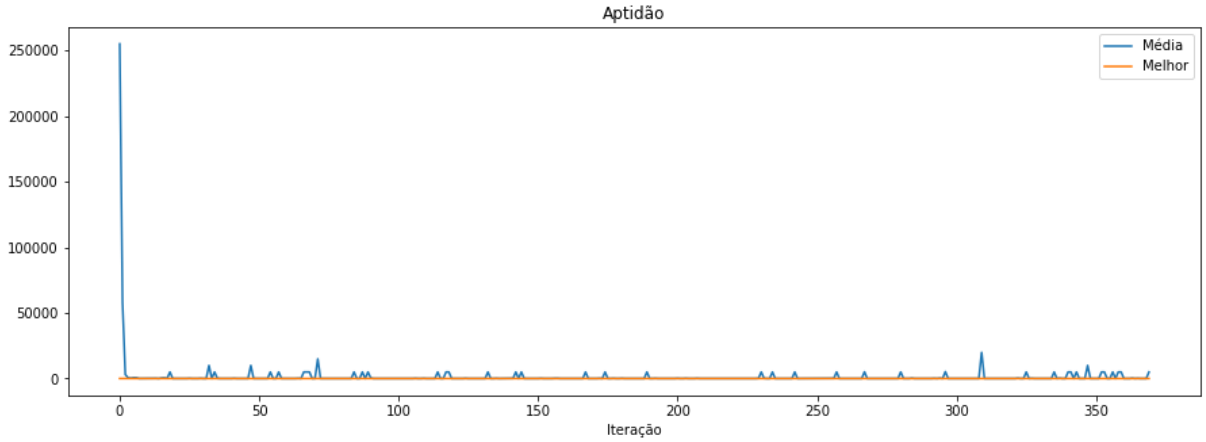


Figura 24 – Resultado de uma execução do algoritmo com uma população de 100 indivíduos (1.719858169555664 segundos com 370 iterações). O melhor resultado obtido é 0.008577173396131099 na iteração 35.



Observa-se que o tempo de execução aumenta significativamente ao incluir mais indivíduos na população, isso pode implicar ou não num melhor resultado. O número de iterações necessárias também é variável de acordo com as condições iniciais. Os três exemplos não chegaram ao limite máximo de iterações (1000) já que não foi encontrado outro resultado ótimo em $^1/_3$ do limite.

Executando o algoritmo diversas vezes e exibindo a média de aptidões médias e melhores em cada execução, é possível analisar alguns comportamentos. As Figuras 25, 26 e 27 apresentam ainda, no gráfico direito, em qual iteração da execução a melhor solução foi encontrada.

Figura 25 – Resultado de 100 execuções do algoritmo com uma população de 30 indivíduos (46.86056303977966 segundos). Valor da média das melhores aptidões: 7.962680668830639. Menor valor entre as melhores aptidões: 0.000014.

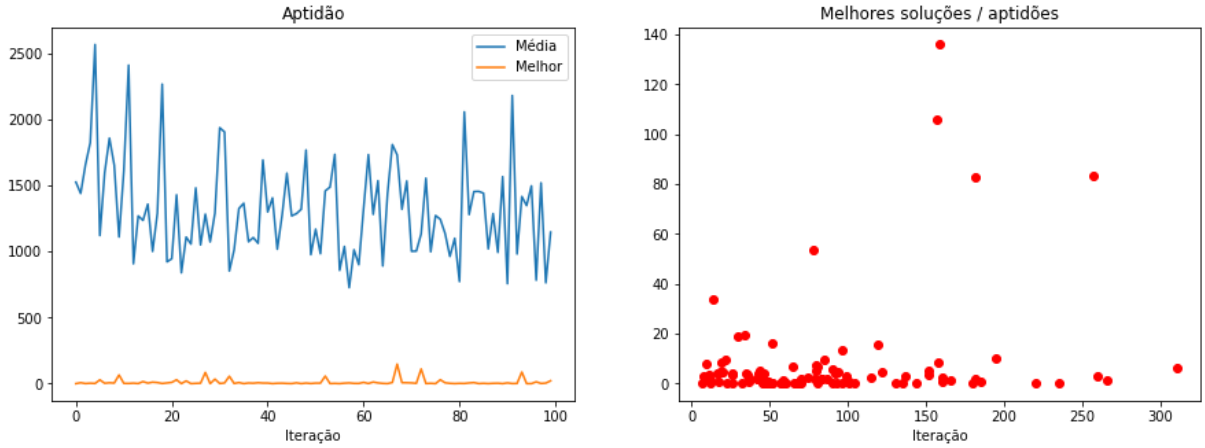


Figura 26 – Resultado de 100 execuções do algoritmo com uma população de 50 indivíduos (73.6172571182251 segundos). Valor da média das melhores aptidões: 3.1107474068019774. Menor valor entre as melhores aptidões: 0.00021236769758872665.

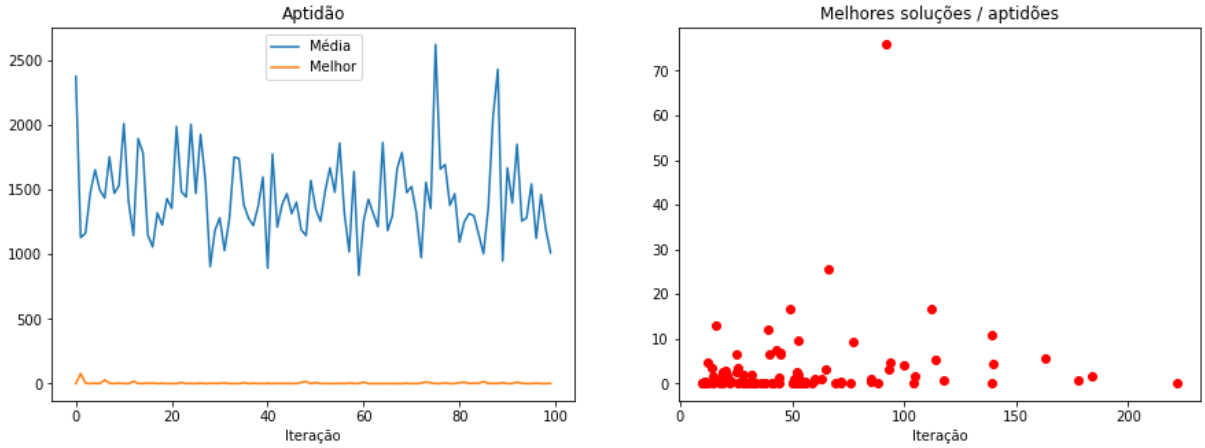
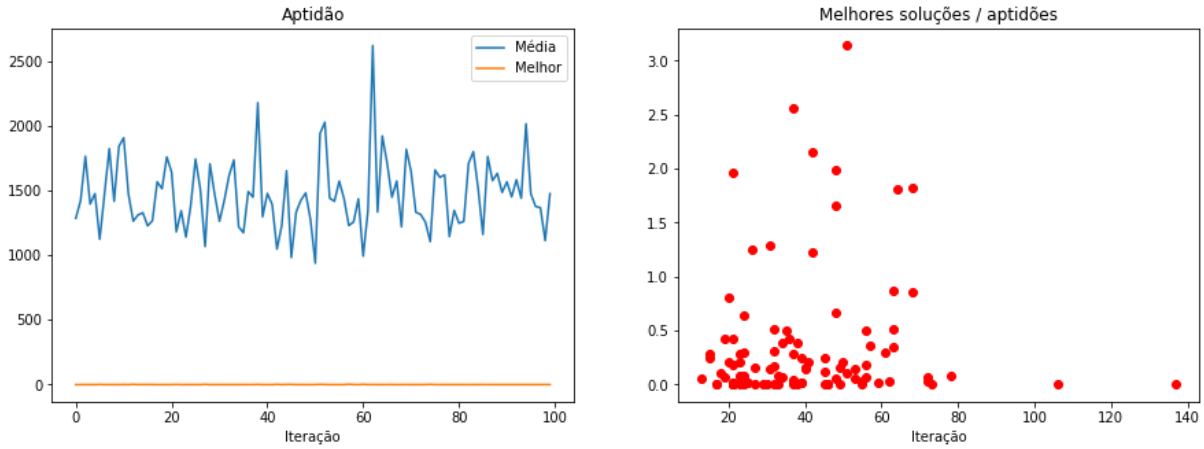


Figura 27 – Resultado de 100 execuções do algoritmo com uma população de 100 indivíduos (145.0887999534607 segundos). Valor da média das melhores aptidões: 0.444563843403753. Menor valor entre as melhores aptidões: 0.000011.



A relação entre o tamanho da população e tempo de execução também pode ser analisado nesse exemplo. Analisando a linha de melhor aptidão, no gráfico da esquerda das Figuras 25, 26 e 27, com 30 indivíduos houveram mais picos em relação a população com 50 e 100 indivíduos, onde os valores ficaram mais próximos do valor mínimo, mantendo-se mais estáveis (fato comprovado também pela média do melhor valor). No que se refere à aptidão média em cada execução de cada configuração dos exemplos, houve uma oscilação entre os valores, devido ao processo de seleção, *crossover* e mutação se basearem em valores aleatórios.

O gráfico da direita mostra um comportamento interessante: os melhores valores quase sempre foram encontrados nas iterações mais iniciais de cada execução. Isso pode indicar a necessidade de colocar um limitante de iterações para que o algoritmo fique mais eficiente (como foi colocado o intervalo de $^1/_3$ como limite para encontrar uma solução melhor, se não for encontrado, o algoritmo para). Com o aumento do tamanho da população, observa-se que essa melhor aptidão começa a se dispersar, já que o espaço de busca aumenta.

Outro indicador proporcional ao tamanho da população, é que a média das melhores aptidões se aproxima cada vez mais de um valor menor à medida que a população aumenta. O menor valor entre as melhores aptidões de cada situação anterior indica que os três exemplos conseguiram uma solução precisa. Assim, os Algoritmos Genéticos, quando implementados corretamente, são capazes de evoluir uma população de modo que as aptidões do melhor indivíduo aumentem em direção à um ótimo global.