Dupla: Abdul Kevin Alexis e Victor Lara

Algoritmos Genéticos

1. Implemente um algoritmo genético para resolver um problema de maximização de função Alpine 2:

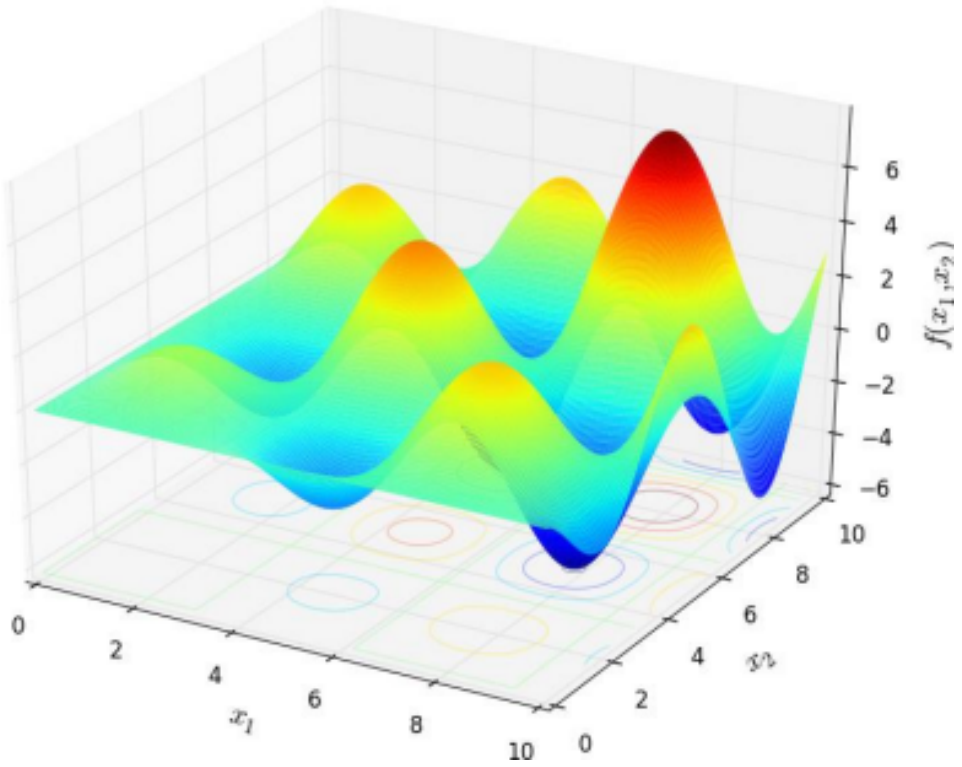
$$f_{\text{Alpine02}}(\mathbf{x}) = \prod_{i=1}^{n} \sqrt{x_i} \sin(x_i)$$

Nesta equação, n representa o número de dimensões da função e $x_i \in [0, 10$ para $i = 1, \ldots, n$. Utilizar, neste trabalho, n=2.

Máximo Global da função é igual a $f(x^*)=2.808^n$, em

$x^*=(7.917,\ldots,7.917)$. Para n=2, o máximo é de 7.88.

Dupla: Abdul Kevin Alexis e Victor Lara



Nesta prática, foi implementado um algoritmo genético em python, com o objetivo descobrir um valor máximo aproximado na função Alpine

Para o algoritmo, foram utilizadas duas condições de parada, a primeira é um valor máximo de iterações, o outro é calcular a diferença entre o indivíduo de melhor fitness da população atual e da anterior, caso este valor seja menor que 0.01 3 vezes consecutivas, o algoritmo para.

A função de fitness utilizada é o ranking linear, escolhido em virtude da capacidade de redução da pressão de seleção em cima do melhor indivíduo. Já a seleção dos indivíduos que

Dupla: Abdul Kevin Alexis e Victor Lara

farão o cruzamento, esta é definida pelo critério da roleta, que seleciona um valor aleatório entre o somatório dos fitness acumulados dos indivíduos. Também foi utilizado o elitismo, porém este necessitou de um estudo mais aprofundado, pois dependendo do valor escolhido, o resultado poderia divergir do resultado desejado. Isso ocorre pois, como a Alpine Function possui vários máximos locais, estes mínimos podem ter sido encontrados e priorizados pelo algoritmo ao longo das gerações, não permitindo uma convergência para os valores desejados.

Com estas decisões tomadas, foram realizados alguns testes, para identificar qual quantidade de população e iterações se tem um resultado expressivo. Para isso, foram testadas várias amostras, desde população = 50, iterações = 10 e elitismo = 3, até população = 200, iterações = 5000 e elitismo = 100. Com todos estes testes, percebeu-se que o melhor resultado foi para população = 500, iterações = 2000 e elitismo = 10. Com estes parâmetros, foi possível obter um resultado muito próximo do mínimo da função, além de ter um tempo de execução ideal, em torno de 1.62 segundos e um número de 6 iterações para encontrar o resultado atual. Assim, abaixo é possível ver o resultado obtido pelo algoritmo com estes parâmetros, além da quantidade de iterações gastas para chegar até a solução:

```
[Running] python -u "c:\Users\Abdul Kevin Alexis\Downloads\drive-download-20220602T231927Z-001 (1)\gen.py"
iteracoes: 6
x = -7.957908469804648, y = -8.011695906432749, f(x,y) = -7.84310964357608
```

Abaixo tem-se uma tabela resumindo os parâmetros utilizados e o gráfico gerado pelo programa:

Dupla: Abdul Kevin Alexis e Victor Lara

Genético	
Tamanho da população	*500*
Forma de seleção	*roleta*
Tipo de crossover	*uniforme*
Função objetivo	*Alpine 02*
Função de Fitness	*linear ranking*
Número de Gerações	*2000*
Taxa de Crossover	*70%*
Taxa de Mutação	*0.5%*

Dupla: Abdul Kevin Alexis e Victor Lara