Ex2. Matveychuk, II-11

Maximization of Test Functions

code(Python):

```
import random
import math
def get_fitness(chromosome):
    strChrom = ""
    for ch in chromosome:
        strChrom += str(ch)
    strChrom = strChrom[::-1]
    x1 = 5 * (int(strChrom[1:11], 2) / 1023)
    x2 = 5 * (int(strChrom[12:22], 2) / 1023)
    if strChrom[0] == "1":
        x1 = -x1
    if strChrom[12] == "1":
        x2 = -x2
    return x1 * math.sin(abs(x1)) + x2 * math.sin(abs(x2))
def truncate_selection(chromosomes):
    return [chromosomes[random.randint(0, 9)],
            chromosomes[random.randint(0, 9)]]
def uniform_crossover(parents):
    child = []
    for x in range(22):
        point = random.random()
        if point > 0.5:
            child.append(parents[0][x])
        else:
            child.append(parents[1][x])
    # print(child)
    return child
def mutation(child):
    point = random.randint(0,21)
    child[point]=(child[point]+1)%2
    return child
def main():
    chromosomes = [[random.randint(0, 1) for _ in range(22)] for _ in range(20)]
    chromosomes.sort(key=get_fitness, reverse=True)
    new_chromosomes = []
    better_fitness = get_fitness(chromosomes[0])
    print(better_fitness)
    populations = 0
    while populations<20:
        for _ in range (20):
            parents=truncate_selection(chromosomes)
            child = uniform_crossover(parents)
            child = mutation(child)
            new_chromosomes.append(child)
        new_chromosomes.sort(key=get_fitness, reverse=True)
        #for chr in new_chromosomes:
        #    print(get_fitness(chr))
        #print("____________________")
        better_fitness=get_fitness(new_chromosomes[0])
        chromosomes = new_chromosomes
        new_chromosomes = []
        populations+=1
        print(better_fitness)
    # get_fitness(chromosomes[0])
        #print(new_chromosomes)
main()
```

Graph: