

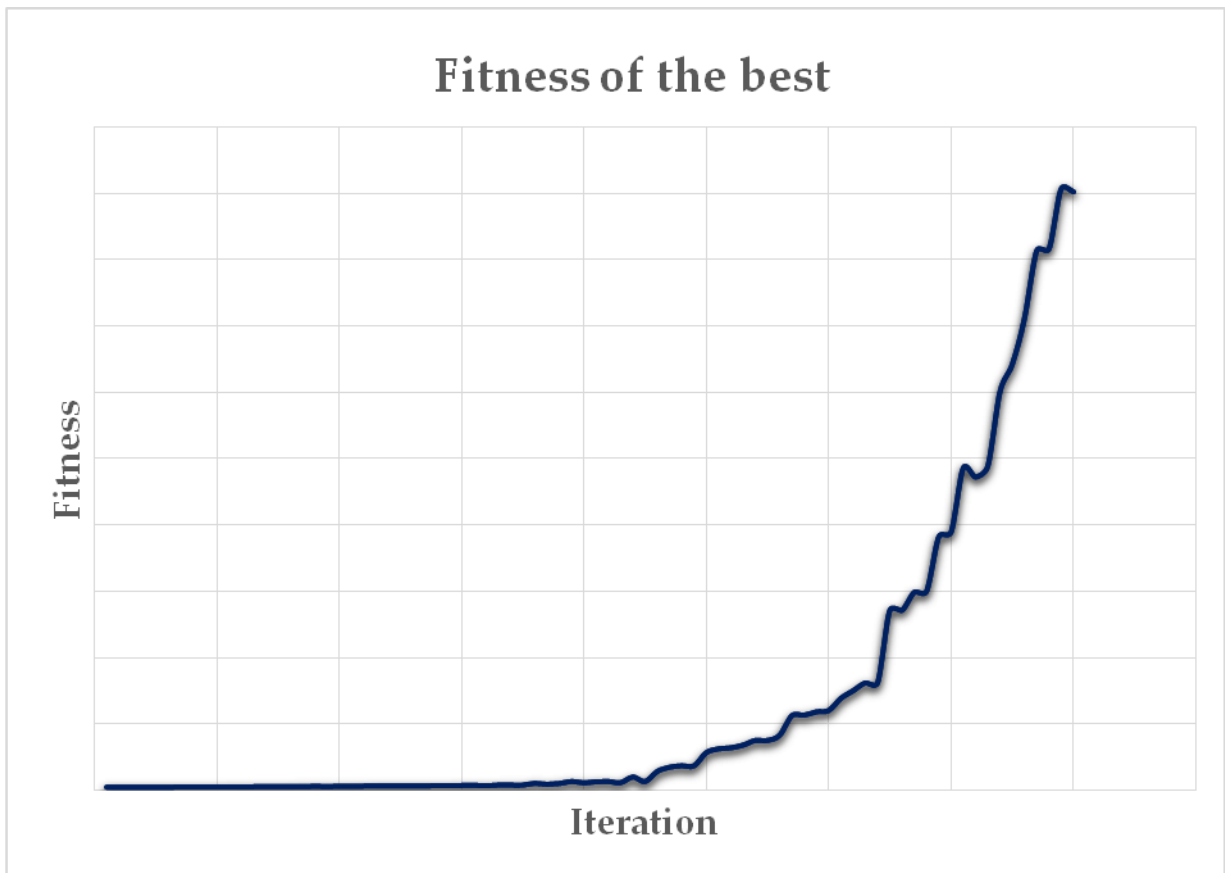
Task 3
Student – Oxana Zanko
ALGORITHM

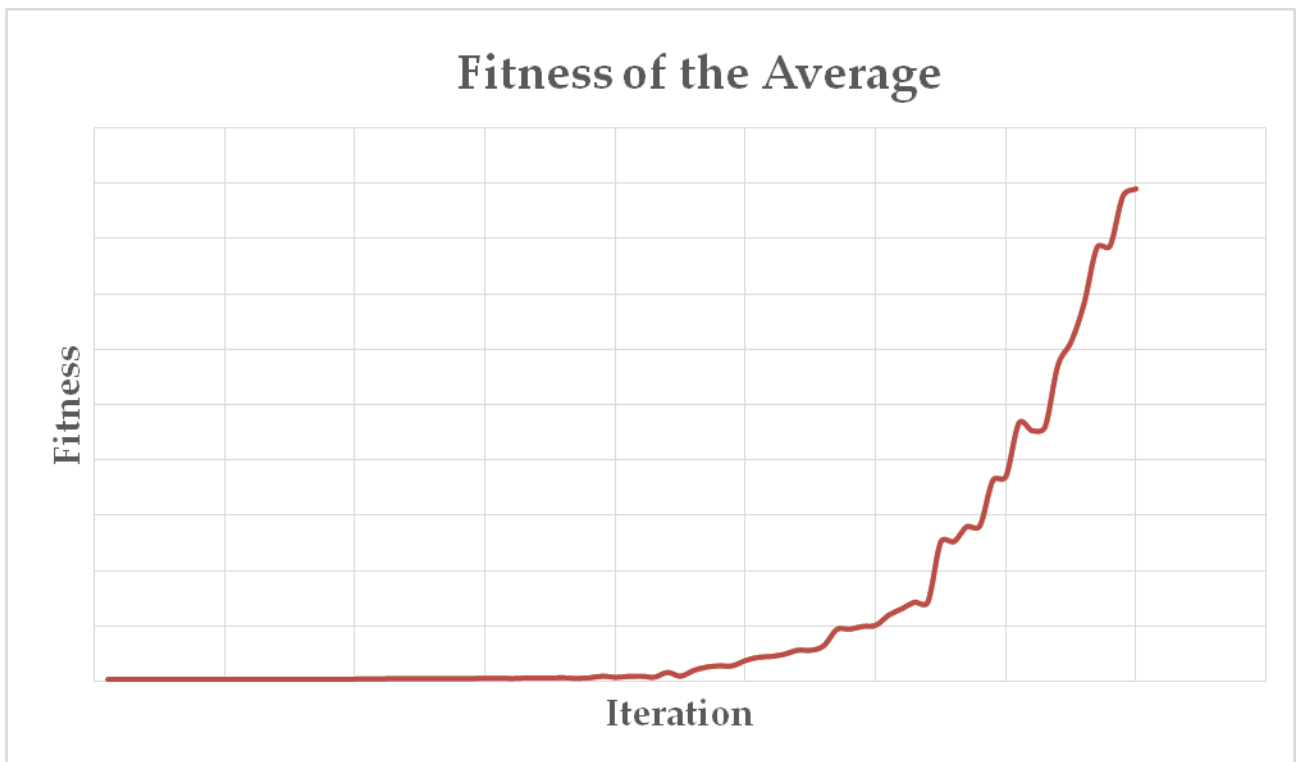
1. Maximize the following y in 3-D cases.

$$y = x_1 \cdot \sin(|x_1|) + x_2 \cdot \sin(|x_2|)$$

2. Create a population of 20 chromosomes at random, with fitness being z.
3. Evolve this population till fitness doesn't change

Results:





Source code:

```
import random
from math import sin
def get_average_value(generation):
    av_list = []
    for i in generation:
        sum = fitness_function(i)
    av_list.append(sum)
    average = reduce(lambda x, y: x + y, av_list) / len(av_list)
    min_value = min(av_list)
    return (average, min_value)
def fitness_function(i):
    sum = 0
    for j in i:
        sum += j * sin(abs(j))
    return sum
def main():
    generation = []
    for i in range(0, 20):
        hromosome = []
        for j in range(0, 20):
            hromosome.append(random.uniform(-5, 5))
        generation.append(hromosome)
        min_value = get_average_value(generation)[1]
    print(min_value)
    count = 0
```

```

while count < 10:
    generation.sort(key=lambda x: fitness_function(x), reverse=False)
    fathers = generation[0:10]
    new_generation = []
    for i in range(0, 10):
        mother = random.randrange(0, 10)
        father = random.randrange(0, 10)
        rand_index = random.randrange(0, 20)
        first = fathers[mother][0:rand_index]
        first.extend(fathers[father][rand_index:20])
        second = fathers[father][0:rand_index]
        second.extend(fathers[mother][rand_index:20])
        new_generation.append(first)
        new_generation.append(second)
        new_generation.extend(fathers)
    new_max = get_average_value(new_generation)
    print(new_max)
    if new_max[1] < min_value:
        min_value = new_max[1]
        count = 0
    else:
        count += 1
generation = new_generation
if __name__ == '__main__':
    main()

```