Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра комп’ютерного моделювання процесів і систем

ЗВІТ

з лабораторної роботи №1

“ЗАСТОСУВАННЯ ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ ОПТИМІЗАЦІЇ ФУНКЦІЇ”

з курсу

«Нейронні мережі та методи проектування»

Виконав: студент групи ІКМ-М222к Черкас Ю.В.

Перевірив: професор, д.т.н. Успенський В.Б.

Харків 2023р

Постановка задачі

Розробити програму, яка за допомогою генетичного алгоритму знаходить максимум функції у заданому діапазоні. Розмір популяції встановити таким, як буде зручно, але не менше, ніж 50 особин. Точність 0.1, кількість кроків (ітерацій) не менше 500. Результати надати у вигляді тексту програми та графіку найкращих значень функції на кожній ітерації. В кінці - координати найкращої точки.

Таблиця 1. Вихідні дані

Номер варіанту	Функція $f(x,y)$, яку максимізуємо	Діапазон [a,b] для змінної x	Діапазон [c,d] для змінної y
15	$3\|x\| - y^2$	[0;6]	[-4;0]

Виконання

Фітнес-функцію F виберемо рівною цільовій:

$$F(ind) = f(x,y) = 3|x| - y^2$$

```
import numpy as np
def fitness_function(ind):
    return 3 * np.abs(ind.x) - ind.y**2
```

тобто F буде збільшуватися зі збільшенням f. Виберемо кількість особин N=50. Кожна особина матиме по 2 хромосоми (x* і y*) , що відповідають аргументам цільової функції x та y.

Для зручності створимо допоміжний клас Individual.

```
class Individual:
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

Наведемо допоміжні методи, котрі використовуються в подальших розрахунках.

```
import numpy as np
from numpy.random import randint
from numpy.random import rand


def decode(min, max, bitstring):
    n_bits = len(bitstring)
    chars = ''.join([str(s) for s in bitstring])
    quanted_value = int(chars, 2)
    value = min + quanted_value * ((max - min)/(2**n_bits - 1))
    return value


def selectOne(individuals, scores):
    scores_for_selection = scores.copy()
    min_score = np.min(scores_for_selection)
    if min_score < 0:
        scores_for_selection = scores_for_selection + 2 * np.abs(min_score)
    max = sum(scores_for_selection)
    selection_probability = [c/max for c in scores_for_selection]
    return individuals[np.random.choice(len(individuals),
p=selection_probability)]


def crossover(p1, p2, crossover_probability):
    if rand() < crossover_probability:
        point_x = randint(1, len(p1.x)-2)
        point_y = randint(1, len(p1.x)-2)
        c1 = Individual(p1.x[:point_x] + p2.x[point_x:], p1.y[:point_y] +
p2.y[point_y:])
        c2 = Individual(p2.x[:point_x] + p1.x[point_x:], p2.y[:point_y] +
p1.y[point_y:])
        return [c1, c2]
    c1, c2 = Individual(p1.x, p1.y), Individual(p2.x, p2.y)
    return [c1, c2]


def mutation(ind, mutation_probability):
    for i in range(len(ind.x)):
        if rand() < mutation_probability:
            ind.x[i] = 1 - ind.x[i]
    for i in range(len(ind.y)):
        if rand() < mutation_probability:
            ind.y[i] = 1 - ind.y[i]
```

Кінцева програма знаходження максимуму функції за допомогою генетичного алгоритму на мові Python буде мати вигляд:

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import randint
from helpers import Individual, fitness_function, decode, selectOne,
crossover, mutation


q = 1 # precision (number of digits after the dot)
N_epochs = 500
N_individuals = 50
a = 0
b = 6
c = -4
d = 0
crossover_probability = 0.9
mutation_probability = 0.01
Lx = np.int(np.ceil(np.log2((b - a) * 10**q + 1)))
Ly = np.int(np.ceil(np.log2((d - c) * 10**q + 1)))

# initial population
individuals = [Individual(randint(0, 2, Lx).tolist(), randint(0, 2,
Ly).tolist()) for _ in range(N_individuals)]

best_ind, best_score = 0, fitness_function(Individual(decode(a, b,
individuals[0].x), decode(c, d, individuals[0].y)))
F_max_per_epoch, F_avg_per_epoch = [], []

for epoch in range(N_epochs):
    decoded = [Individual(decode(a, b, ind.x), decode(c, d, ind.y)) for ind in
individuals]
    scores = [fitness_function(d) for d in decoded]

    F_max = 0
    F_sum = 0

    for i in range(N_individuals):
        F_sum += scores[i]
        if scores[i] > best_score:
            best_ind, best_score = individuals[i], scores[i]
            print(f'{epoch} epoch: new best f({decoded[i].x}, {decoded[i].y})
= {scores[i]}')
        if scores[i] > F_max:
            F_max = scores[i]

    F_avg = F_sum / N_individuals
```

```
    F_avg_per_epoch.append(F_avg)
    F_max_per_epoch.append(F_max)

    # select parents
    selected = [selectOne(individuals, scores) for _ in range(N_individuals)]

    # create the next generation
    children = list()
    for i in range(0, N_individuals, 2):
        # get selected parents in pairs
        p1, p2 = selected[i], selected[i+1]

        # crossover and mutation
        for child in crossover(p1, p2, crossover_probability):
            # mutation
            mutation(child, mutation_probability)
            # store for next generation
            children.append(child)
    # replace population
    individuals = children

best_decoded = Individual(decode(a, b, best_ind.x), decode(c, d, best_ind.y))
print(f'Best case: f({best_decoded.x}, {best_decoded.y})) = {best_score}')

x = np.arange(0, 500, 1)
plt.plot(x, F_max_per_epoch, 'r')
plt.plot(x, F_avg_per_epoch, 'b')
plt.legend([r'Fmax - максимальна пристосованість у популяції', 'Favg - середня
пристосованість по популяції'])
plt.grid(True)
plt.ylabel(r'F')
plt.xlabel('Ітерація')
plt.show()
```

Нижче наведений результат роботи програми, котрий був виведений в термінал:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

0 epoch: new best f(4.857142857142857, -1.6507936507936511) = 11.846308893927938
0 epoch: new best f(4.857142857142857, -0.8253968253968256) = 13.890148652053412
0 epoch: new best f(5.333333333333333, -1.2698412698412698) = 14.387503149407912
0 epoch: new best f(5.7142857142857135, -0.5079365079365079) = 16.884857646762406
5 epoch: new best f(5.7142857142857135, 0.0) = 17.14285714285714
7 epoch: new best f(5.904761904761904, -0.6984126984126986) = 17.226505416981606
10 epoch: new best f(6.0, -0.1904761904761907) = 17.96371882086168
16 epoch: new best f(6.0, 0.0) = 18.0
Best case: f(6.0, 0.0)) = 18.0
```

На рисунку 1 зображені графіки максимальної пристосованості у популяції та середньої пристосованості по популяції для кожної ітерації роботи програми.

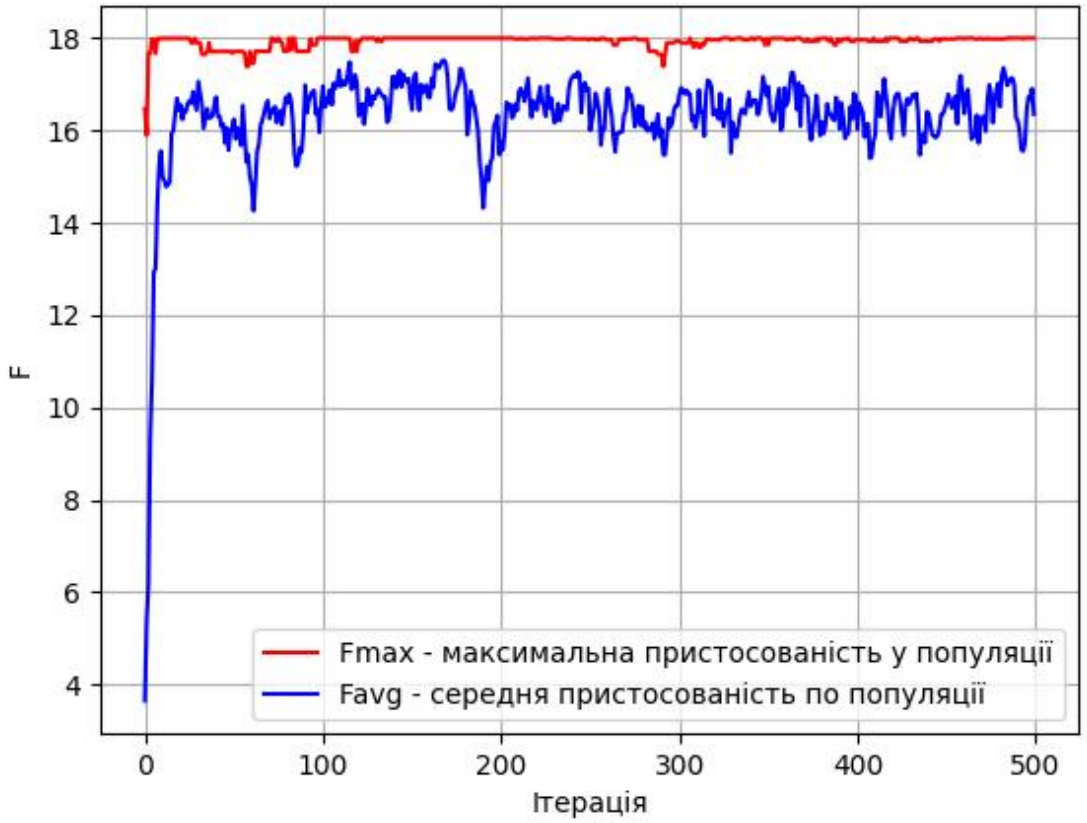

Рисунок 1 – Графік значень середньої та максимальної пристосованості в залежності від ітерації алгоритму

З рисунку 1 та з даних виведених в термінал бачимо, що максимум фітнес-функції набуває на 16 ітерації в точці $x = 6$, $y = 0$, в котрій він має величину $F = 18$.

Висновки

На даній лабораторній роботі ми дослідили роботу генетичного алгоритму на прикладі знаходження максимуму функції в заданих діапазонах змін аргументів. Реалізували даних алгоритм на мові Python та проаналізували його роботу при різних початкових умовах та налаштуваннях.

Враховуючи простоту функції, генетичний алгоритм вже на 16-й ітерації знайшов максимум функції: $x = 6, y = 0, f(x, y) = 18$. По формулі досліджуваної функції ($f(x, y) = 3|x| - y^2$) та по аналізованих діапазонах ($x \in [0; 6]$, $y \in [-4; 0]$), ми бачимо, що отримані результати відповідають дійсності, а отже, генетичний алгоритм доказав коректність своєї роботи.