МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №  43

ОТЧЁТ

ЗАЩИЩЁН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

### профессор                                   Скобцов Ю.А.

должность, уч. Степень, звание   подпись, дата           инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №7.

Оптимизация функций многих переменных с помощью роевых алгоритмов .

по курсу: Эволюционные методы проектирования программно-информационных систем

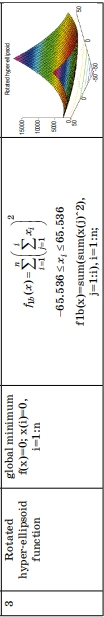
РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4136                                                                                Бобрович Н. С.

                                                                         подпись, дата                      инициалы, фамилия

Санкт-Петербург 2024

1. **Задание:**

Вариант 3:  
Скриншот 17-10-2024 230645

1. Создать программу, использующую РА для нахождения оптимума функции согласно таблице вариантов, приведенной в приложении А. Для всех Benchmark-ов оптимумом является минимум. Программу выполнить на встроенном языке пакета Matlab.
2. Для n=2 вывести на экран график данной функции с указанием найденного экстремума, точек популяции. Для вывода графиков использовать стандартные возможности пакета Matlab. Предусмотреть возможность пошагового просмотра процесса поиска решения.
3. Повторить нахождение решения с использованием стандартного Genetic Algorithm toolbox. Сравнить полученные результаты.
4. Исследовать зависимость времени поиска, числа поколений (генераций), точности нахождения решения от основных параметров генетического алгоритма: - число особей в популяции - вероятность кроссинговера, мутации. Критерий остановки вычислений – повторение лучшего результата заданное количество раз или достижение популяцией определенного возраста (например, 100 эпох).

5. Повторить процесс поиска решения для n=3, n =5, n = 10, сравнить результаты, скорость работы программы.

1. **Листинг:**

import numpy as np

import matplotlib.pyplot as plt

def function(\*args):

return (-20 \* np.exp(-0.2 \* np.sqrt(1/len(args) \* sum([arg\*\*2 for arg in args]))) -

np.exp(1/len(args) \* sum([np.cos(2 \* np.pi \* arg) for arg in args])) + 20 + np.e)

# def function(x, y):

# return (-20 \* np.exp(-0.2 \* np.sqrt(0.5 \* (x \*\* 2 + y \*\* 2))) -

# np.exp(0.5 \* (np.cos(2 \* np.pi \* x) + np.cos(2 \* np.pi \* y))) + 20 + np.e)

def correction\_speed(speed, position, personal\_best\_positions, global\_best\_position, c1=1.5, c2=1.5):

n = len(speed)

for i in range(n):

speed[i] += c1 \* np.random.rand() \* (personal\_best\_positions[i] - position[i]) + \

c2 \* np.random.rand() \* (global\_best\_position[i] - position[i])

if speed[i] < -3:

speed[i] = -3

if speed[i] > 3:

speed[i] = 3

return speed

def correction\_position(position, speed):

n = len(position)

for i in range(n):

position[i] += speed[i]

if position[i] < -20:

position[i] = -20

if position[i] > 20:

position[i] = 20

return position

def swarm\_alg(n):

num\_particles = 20 # Количество частиц

num\_iterations = 500 # Количество итераций

w = 0.5 # Вес инерции

c1 = 1 # Коэффициент личного опыта

c2 = 2 # Коэффициент опыта группы

# Инициализация частиц

#n = 2 # Количество переменных

positions = np.random.uniform(-10, 10, (num\_particles, n)) # Позиции частиц

velocities = np.random.uniform(-1, 1, (num\_particles, n)) # Начальные скорости частиц

personal\_best\_positions = positions.copy() # Личные лучшие позиции

personal\_best\_scores = np.array([function(\*p) for p in positions])

global\_best\_position = personal\_best\_positions[np.argmin(personal\_best\_scores)] # Глобальная лучшая позиция

print(global\_best\_position)

if n == 2:

plot\_with\_particles(positions)

for i in range(num\_iterations):

for j in range(num\_particles):

if function(\*positions[j]) < function(\*personal\_best\_positions[j]):

personal\_best\_positions[j] = positions[j].copy()

if function(\*personal\_best\_positions[j]) < function(\*global\_best\_position):

global\_best\_position = personal\_best\_positions[j].copy()

velocities[j] = correction\_speed(velocities[j], positions[j], personal\_best\_positions[j], global\_best\_position, c1, c2)

positions[j] = correction\_position(positions[j], velocities[j])

print(global\_best\_position)

#print(positions)

if n == 2:

plot\_with\_particles(positions)

return global\_best\_position

def plot\_with\_particles(positions, bounds=(-20, 20), resolution=100):

"""

:param positions: Позиции частиц, массив размера (num\_particles, 2).

:param bounds: Кортеж с границами области (-bounds, bounds).

:param resolution: Количество точек на ось для сетки.

"""

# Создаем сетку координат для графика

x = np.linspace(bounds[0], bounds[1], resolution)

y = np.linspace(bounds[0], bounds[1], resolution)

X, Y = np.meshgrid(x, y)

# Вычисляем значения функции для сетки

Z = function(X, Y)

# Создаем объект графика

fig = plt.figure(figsize=(10, 8))

ax = fig.add\_subplot(111, projection='3d')

# Строим поверхность

ax.plot\_surface(X, Y, Z, cmap='viridis', alpha=0.6)

# Извлекаем координаты частиц

particle\_x = positions[:, 0]

particle\_y = positions[:, 1]

particle\_z = np.array([function(px, py) for px, py in positions])

# Отображаем только точки частиц

ax.scatter(particle\_x, particle\_y, particle\_z, color='red', s=50, label='Particles')

# Настройка меток осей

ax.set\_xlabel('X axis')

ax.set\_ylabel('Y axis')

ax.set\_zlabel('Z axis')

# Показать график

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

n = 2

swarm\_alg(n)

1. **Результаты выполнения индивидуального задания:**

[-1.94043617 -0.47175245]

[-0.01045974 -0.0165579 ]

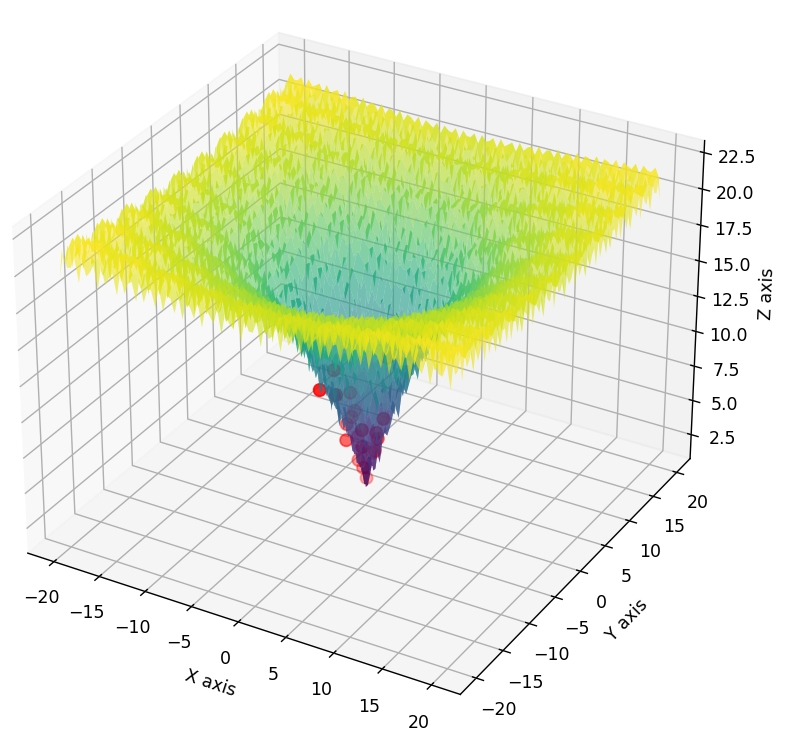


Рис. 1 – Начальная генерация частиц (для функции 2-ух переменных)

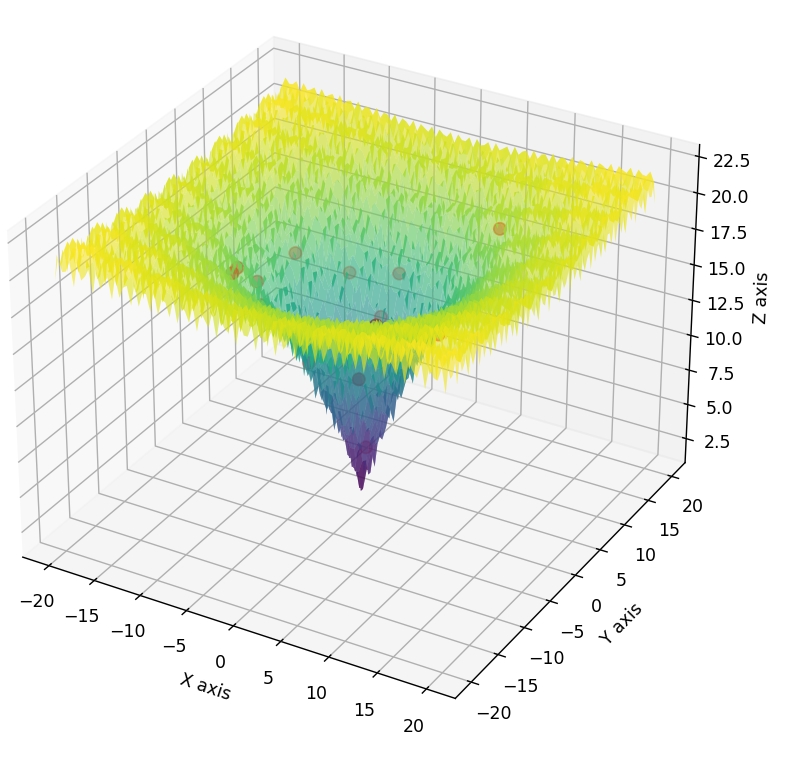


Рис. 2 – Частицы для функции 2-ух переменных на момент конца работы программы