ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| профессор |  |  |  | Ю.А. Скобцов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1  Простой генетический алгоритм |
| по дисциплине:  Эволюционные методы проектирования программно-информационных систем |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2024

# Цель Работы

1. Разработать простой генетический алгоритм для нахождения оптимума заданной по варианту функции одной переменной (таб. 1.1). Вид экстремума: Максимум

2. Исследовать зависимость времени поиска, числа поколений (генераций), точности нахождения решения от основных параметров генетического алгоритма: - число особей в популяции - вероятность кроссинговера, мутации.

3. Вывести на экран график данной функции с указанием найденного экстремума для каждого поколения.

4. Сравнить найденное решение с действительным.

Вариант 4

|  |  |  |
| --- | --- | --- |
| 4 | Sin(2x)/x2 | x ∈ [-20,-3.1] |

# Краткие теоретические сведения

Генети́ческий алгори́тм (англ. genetic algorithm) — эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений, с помощью которых решаются оптимизационные задачи с использованием методов естественной эволюции, таких как наследование, мутации, отбор и кроссинговер. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

# Программа и результаты выполнения

Листинг программы

import numpy as np

import matplotlib.pyplot as plt

def f(x):

    return np.sin(2 \* x) / (x \*\* 2)

population\_size = 100

generations = 50

mutation\_rate = 0.2

mutation\_chance = 0.9

x\_bounds = [-20, -3.1]

population = np.random.uniform(x\_bounds[0], x\_bounds[1], population\_size)

fitness\_history = []

population\_history=[]

for generation in range(generations):

    fitness = f(population)

    fitness\_history.append(fitness)

    population\_history.append(population)

    selected\_indices = np.argsort(fitness)[-population\_size // 2:]

    selected\_population = population[selected\_indices]

    offspring = []

    for i in range(len(selected\_population) // 2):

        parent1 = selected\_population[2 \* i]

        parent2 = selected\_population[2 \* i + 1]

        crossover\_point = np.random.rand()

        child = crossover\_point \* parent1 + (1 - crossover\_point) \* parent2

        offspring.append(child)

    offspring = np.array(offspring)

    if np.random.uniform(0,1)<=mutation\_chance:

        mutation = np.random.uniform(-1, 1, offspring.shape) \* mutation\_rate

        offspring += mutation

    population = np.concatenate((selected\_population, offspring))

def plot\_generation(generation):

    print(fitness\_history[generation])

    print(population\_history[generation])

    plt.figure(figsize=(10, 6))

    plt.plot( population\_history[generation], fitness\_history[generation],'o', label=f'Поколение {generation + 1}', alpha=0.7)

    # Построение графика исходной функции

    x\_values = np.linspace(x\_bounds[0], x\_bounds[1], 400)

    plt.plot(x\_values, f(x\_values), label='Исходная функция', color='red', linewidth=2)

    plt.title(f'Приспособленность индивидов в поколении {generation + 1}')

    plt.xlabel('Индивид')

    plt.ylabel('Приспособленность')

    plt.ylim(-0.5, 1)  # Установите пределы по оси Y для лучшей визуализации

    plt.legend(loc='upper right', fontsize='small')

    plt.grid()

    plt.show()

while 1:

    # Запрос ввода от пользователя

    user\_input = input("Введите номер поколения (1-100) или 'all' для отображения всех поколений: ")

    if user\_input.lower() == 'q':

        break

    if user\_input.lower() == 'all':

        # Визуализация всех поколений на одном графике

        plt.figure(figsize=(12, 8))

        for i in range(generations):

            plt.plot(population\_history[i],fitness\_history[i], 'o', label=f'Поколение {i + 1}' if i < generations else "", alpha=0.5)

        # Построение графика исходной функции

        x\_values = np.linspace(x\_bounds[0], x\_bounds[1], 400)

        plt.plot(x\_values, f(x\_values), label='Исходная функция', color='red', linewidth=2)

        plt.title(f'Приспособленность индивидов на протяжении {generations} поколений')

        plt.xlabel('Индивид')

        plt.ylabel('Приспособленность')

        plt.ylim(-0.5, 1)  # Установите пределы по оси Y для лучшей визуализации

        plt.legend(loc='upper right', fontsize='small')

        plt.grid()

        plt.show()

    else:

        try:

            generation\_number = int(user\_input) - 1  # Преобразуем в индекс (0-99)

            if 0 <= generation\_number < generations:

                plot\_generation(generation\_number)

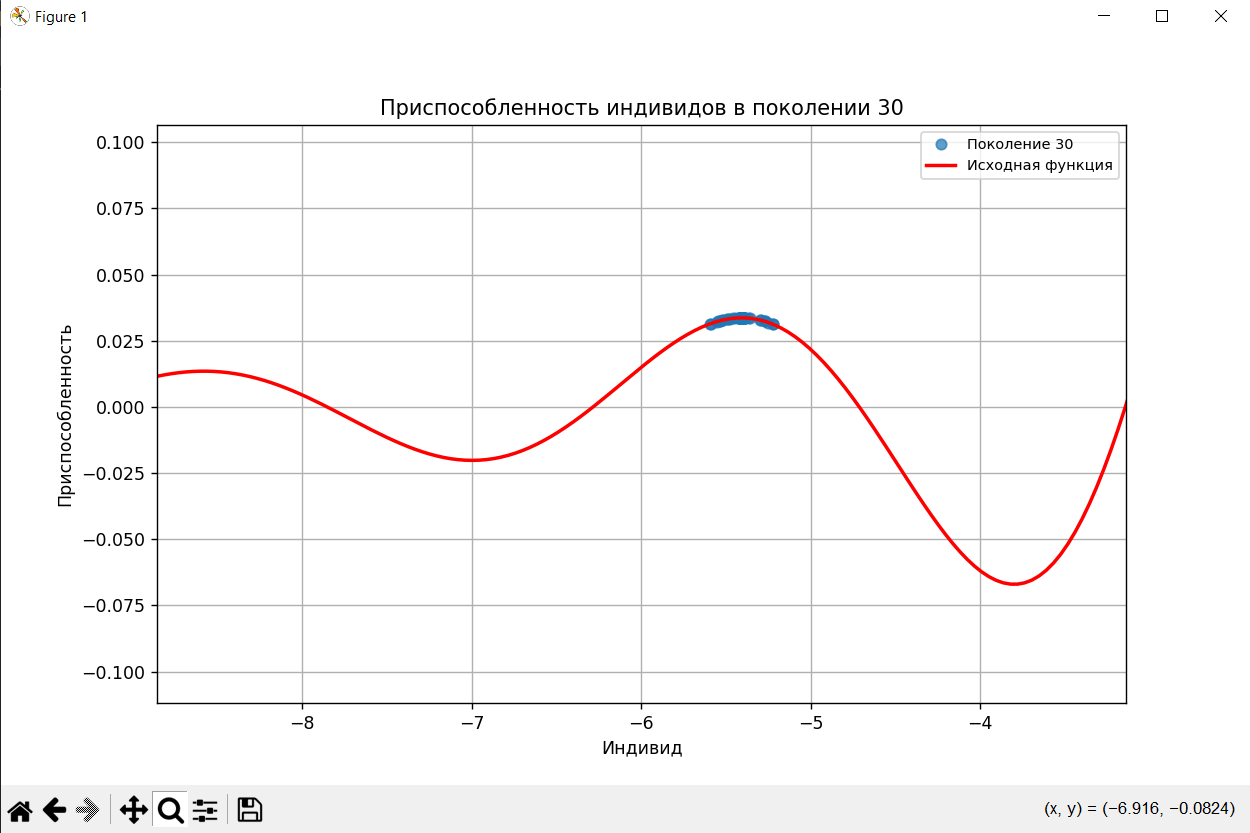
            else:

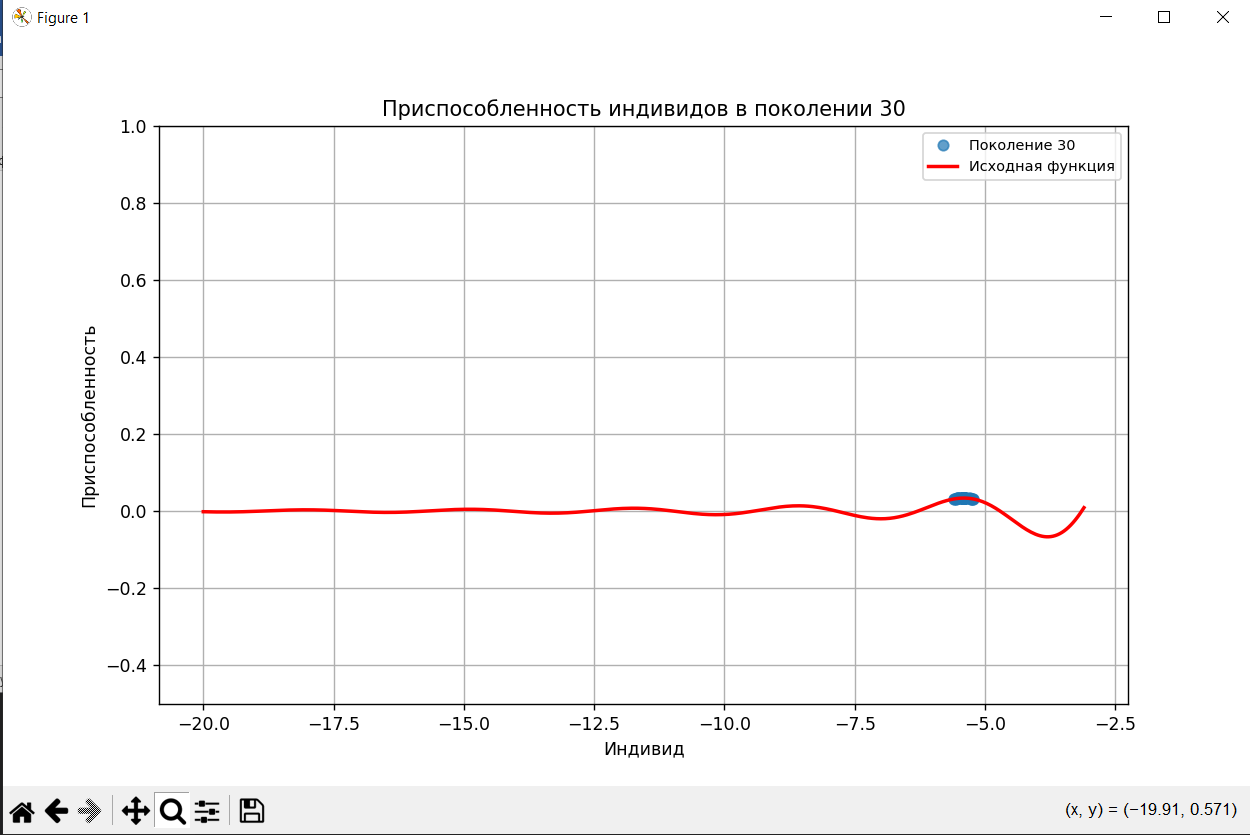
                print(f"Пожалуйста, введите номер поколения от 1 до {generations}.")

        except ValueError:

            print("Некорректный ввод. Пожалуйста, введите номер поколения или 'all'.")

Скриншоты графика





# Письменный ответ на теоретический вопрос

Опишите реализацию ОР в виде колеса рулетки и приведите пример его работы.

Реализация оператора репродукции в виде колеса рулетки

1. Оценка особей: Каждой особи в популяции присваивается значение приспособленности (fitness), которое отражает, насколько хорошо она решает задачу.

2. Нормализация приспособленности: Присвоенные значения приспособленности нормализуются, чтобы получить вероятности выбора каждой особи. Это делается путем деления значения приспособленности каждой особи на сумму всех значений приспособленности.

3. Создание колеса рулетки: На основе нормализованных значений создается "колесо рулетки", где каждая особь занимает сегмент, пропорциональный своей вероятности выбора.

4. Выбор особей: Для выбора особей для репродукции генерируется случайное число в диапазоне от 0 до 1. Это число используется для определения, в каком сегменте колеса оно попадает, что соответствует выбору конкретной особи.

5. Создание потомства: Выбранные особи могут быть скрещены (например, с помощью одноточечного или двухточечного кроссовера) для создания новых особей.

Пример работы оператора репродукции

Предположим, у нас есть популяция из 4 особей с следующими значениями приспособленности:

- Особь A: 10

- Особь B: 20

- Особь C: 30

- Особь D: 40

1. Сумма приспособленности: 10 + 20 + 30 + 40 = 100.

2. Нормализованные значения:

- A: 10/100 = 0.1

- B: 20/100 = 0.2

- C: 30/100 = 0.3

- D: 40/100 = 0.4

3. Создание колеса рулетки:

- A занимает 10% колеса,

- B — 20%,

- C — 30%,

- D — 40%.

4. Выбор особей: Генерируем случайное число, например, 0.35. Это число попадает в диапазон C (0.1 + 0.2 + 0.3 = 0.6), значит, выбираем особь C.

5. Повторный выбор: Генерируем еще одно случайное число, например, 0.05. Это число попадает в диапазон A, значит, выбираем особь A.

6. Создание потомства: Теперь особи C и A могут быть скрещены для создания новой особи.

Таким образом, оператор репродукции в виде колеса рулетки позволяет эффективно выбирать особей для создания нового поколения, основываясь на их приспособленности.