МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное

учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Инженерная школа информационных технологий и робототехники

Отчет по лабораторной работе № 3 по дисциплине

«Интеллектуальные системы»

**«**Генетические алгоритмы**»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент группы 8ВМ21 |  | Лобанов Д. А. |
| Проверил: |  | Брагин А. Д. |

Томск 2022

**Цель работы**: целью лабораторной работы является создание студентом экспертной системы (ЭС).

**Задание:** cоздание студентом программы, реализующей генетический алгоритм (ГА) для решения задачи оптимизации.

Исходя из личных предпочтений для выполнения лабораторной работы был выбран 8 вариант с вещественным кодированием. Необходимо найти точку пересечения функции с осью Ox.

**Ход работы:**

Генетический алгоритм будет реализован на языке программирования Python.

Введем глобальные константы:

Вероятность мутации ;

Вероятность скрещивания ;

1. Формируем начальную популяцию.

Начальная популяция состоит из 10 особей. Гены данной популяции инициализируются случайным образом. Формирование популяции представлено на рисунке 1. Исходя из области определения функции, значение гена не может быть меньше 1.

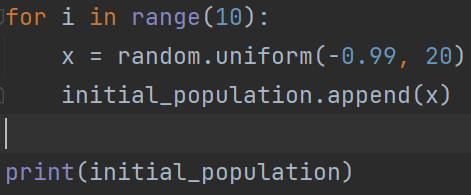
****

Рисунок 1 – Формирование начальной популяции.

Далее определим функцию приспособленности. Для нахождения пересечения функции с осью Ox необходимо прировнять ее к нулю. Для удобства разделим ее еще на 1. Т.е. чем больше значение функции принадлежности, тем больше приспособленность особи. Произведем расчёт функции принадлежности для всех особей из популяции. Также рассчитаем вероятность i-ой особи принять участие в скрещивании. Расчет будет производится при помощи рулеточной селекции. Данные функции представлены на рисунке 2.

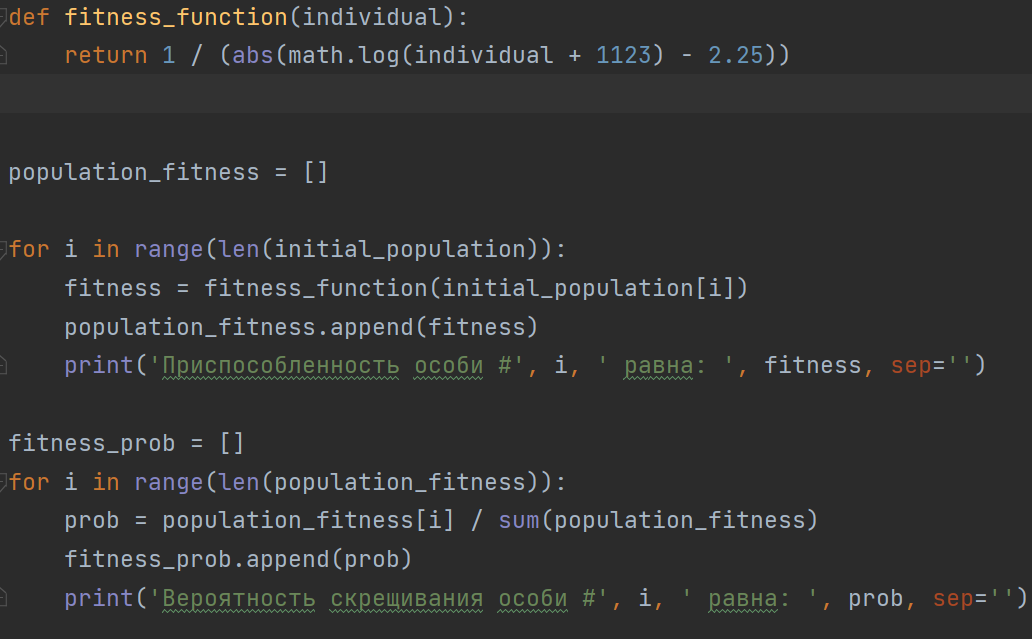


Рисунок 2 – Функция принадлежности и определение вероятности i-ой особи принять участие в скрещивании.

Далее определим функцию, которая будет реализовывать рулеточную селекцию (Рисунок 3).

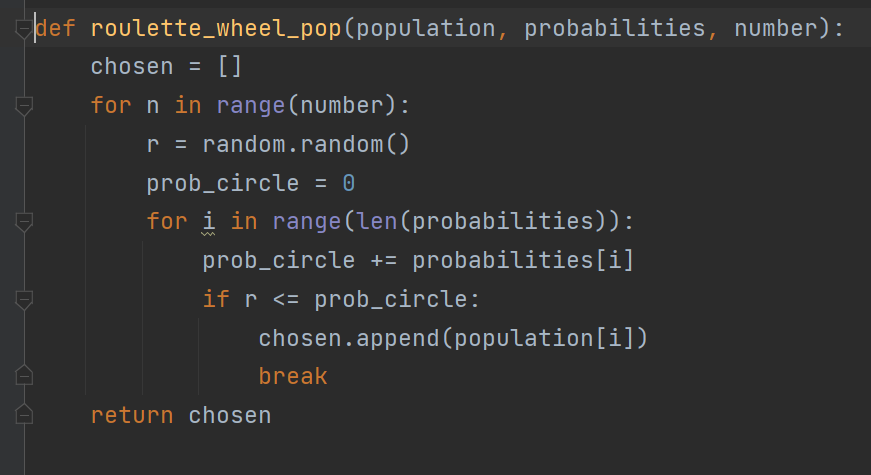


Рисунок 3 – Реализация алгоритма рулеточной селекции.

Для операций скрещивания и мутации будем использовать вещественное кодирование и арифметический оператор кроссинговера. Оператор кроссинговера принимает на вход две родительские особи и возвращающую результирующую особь. Функция мутации вносит в хромосому случайную мутацию. Также для скрещивания всей популяции реализуем соответствующую функцию. Данные функции представлены на рисунке 4.



Рисунок 4 – Функции реализующее скрещивание.

Объединив вышеописанные функции реализуем итоговую версию генетического алгоритма. Программный код итоговой версии представлен на листинге 1.

import random  
import math  
  
  
Pc = 0.6  
Pm = 0.1  
Lambda = random.uniform(0, 1)  
print(Lambda)  
# Найти точку пересечения функции с осью Ох.  
# f(х) = ln (x+1) – 2,25, x > –1.  
# Вариант А) Использовать целочисленное кодирование.  
# Вариант Б) Использовать вещественное кодирование.  
  
# начальная популяция из 10 особей  
initial\_population = []  
for i in range(10):  
 x = random.uniform(-0.99, 20)  
 initial\_population.append(x)  
  
print(initial\_population)  
  
  
def fitness\_function(individual):  
 return 1 / (abs(math.log(individual + 1) - 2.25))  
  
  
def roulette\_wheel\_pop(population, probabilities, number):  
 chosen = []  
 for n in range(number):  
 r = random.random()  
 prob\_circle = 0  
 for i in range(len(probabilities)):  
 prob\_circle += probabilities[i]  
 if r <= prob\_circle:  
 chosen.append(population[i])  
 break  
 return chosen  
  
  
# roulette\_wheel\_pop(initial\_population, fitness\_prob, 10)  
  
  
def crossover(parent\_1, parent\_2):  
 return Lambda \* parent\_1 + (1 - Lambda) \* parent\_2  
  
  
def mutation(individual):  
 Pm = 0.1  
 r = random.uniform(-0.01, 0.01)  
 if Pm > r:  
 individual = individual + r  
 return individual  
  
  
def population\_crossover(population):  
 new\_pop = []  
 for i in range(0, len(population), 2):  
 if Pc > random.random():  
 new\_pop.append(mutation(crossover(population[i], population[i + 1])))  
 new\_pop.append(mutation(crossover(population[i + 1], population[i])))  
 else:  
 new\_pop.append(population[i])  
 new\_pop.append(population[i + 1])  
  
 return new\_pop  
  
  
result\_population = initial\_population  
  
for j in range(100000):  
  
 population\_fitness = []  
 for i in range(len(result\_population)):  
 fitness = fitness\_function(result\_population[i])  
 population\_fitness.append(fitness)  
  
 fitness\_prob = []  
 for i in range(len(population\_fitness)):  
 prob = population\_fitness[i] / sum(population\_fitness)  
 fitness\_prob.append(prob)  
  
 population\_for\_crossover = roulette\_wheel\_pop(result\_population, fitness\_prob, 10)  
 new\_population = population\_crossover(population\_for\_crossover)  
 population\_for\_crossover.sort(key=fitness\_function, reverse=True)  
 new\_population.sort(key=fitness\_function, reverse=True)  
 result\_population = population\_for\_crossover + new\_population  
  
 if abs(math.log(new\_population[0] + 1) - 2.25) < 1E-5:  
 print("Итоговая Популяция" + str(new\_population))  
 for i in range(len(new\_population)):  
 print('Целевая функция "ln(x + 1) - 2.25" особи #', i, ' равна: ',  
 abs(math.log(new\_population[i] + 1) - 2.25), sep='')  
 print("Прошло итераций: " + str(j))  
 break

Листинг 1 – Программный код генетического алгоритма.

В случае, когда точность найденного решения становится приемлемой, то выполнение алгоритма прекращается. Результат работы генетического алгоритма представлен на рисунке 5.

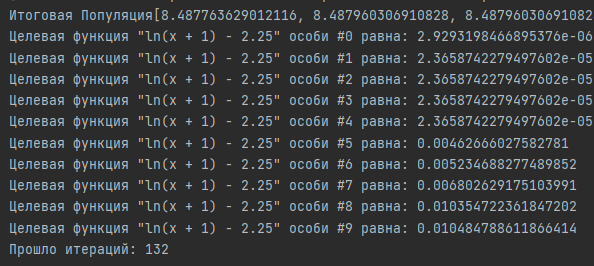


Рисунок 5 – Результат работы генетического алгоритма.

**Вывод**: в ходе работы был реализован генетический алгоритм для поиска пересечения функции с осью Ox. Реализованный генетический алгоритм имеет следующие характеристики: вещественное кодирование, рулеточная селекция, арифметический оператор кроссинговера.