Министерство науки и высшего образования Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Институт прикладной математики и компьютерных наук

Кафедра теоретических основ информатики

**Применение генетического алгоритма для решения задач оптимизации**

Интеллектуальные системы

Автор работы

студент группы 931702

А.В. Волков

Томск 2020

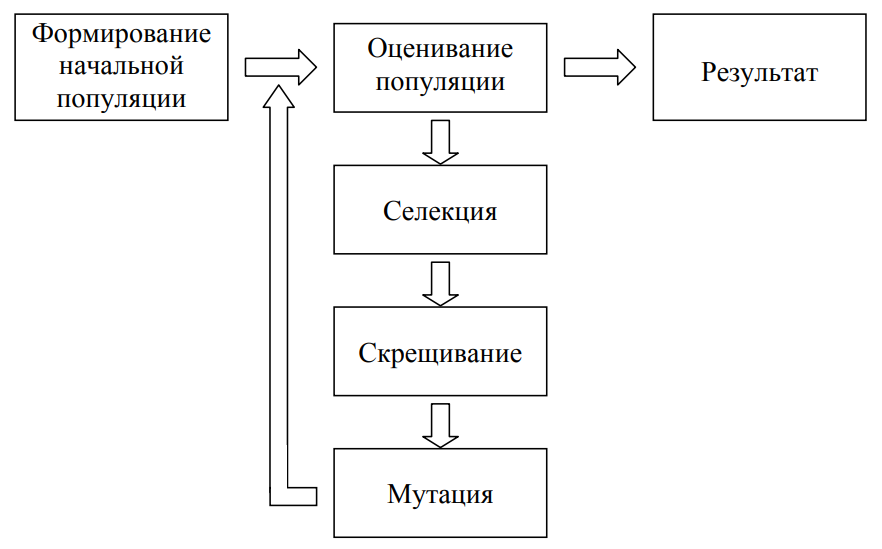
# Цели и задачи

**Цель работы** – создание программы, применяющей генетический алгоритм для нахождения максимума функции y(x) = 1/x. x [-4, 0).

**Задачи:**

1. Изучить принципы работы генетического алгоритма.
2. Изучить структурную схему алгоритма.
3. Написать программу, применяющую генетический алгоритм.
4. Протестировать программу, настроить параметры.
5. Предоставить результаты работы программы и сделать выводы.

# Метод решения задачи и структурная схема алгоритма



Для решения задачи нахождения максимума функции будет применятся генетический алгоритм. Генетический алгоритм работает с популяцией особей, в хромосоме (генотип) каждой из которых закодировано возможное решение задачи (фенотип). В начале работы алгоритма популяция формируется случайным образом (блок «Формирование начальной популяции»). Для того чтобы оценить качество закодированных решений используют функцию приспособленности, которая необходима для вычисления приспособленности каждой особи (блок «Оценивание популяции»). По результатам оценивания особей наиболее приспособленные из них выбираются (блок «Селекция») для скрещивания. В результате скрещивания выбранных особей посредством применения генетического оператора кроссинговера создается потомство, генетическая информация которого формируется в результате обмена хромосомной информацией между родительскими особями (блок «Скрещивание»). Созданные потомки формируют новую популяцию, причем часть потомков мутирует (используется генетический оператор мутации), что выражается в случайном изменении их генотипов (блок «Мутация»). Этап, включающий последовательность «Оценивание популяции» – «Селекция» – «Скрещивание» – «Мутация», называется поколением. Эволюция популяции состоит из последовательности таких поколений.

# Листинг программы

В ходе работы было получено две реализации с целочисленным и вещественным кодированием.

**Целочисленное кодирование:**

#Импортим генетический алгоритм

from pyeasyga.pyeasyga import GeneticAlgorithm

import random

import numpy as np

#Представление данных

#Данные представляют из себя множество нулей с одной единицей на какой-либо позиции

#Это позиция отражает положение x на оси абсцисс от -4 до 0

seed\_data = [0] \* 399

seed\_data.append(1)

#Иницилизируем генетический алгоритм с заданными данными, а также размером

#популяции, количеством поколений, шансом на кроссинговер и мутацию

ga = GeneticAlgorithm(seed\_data, 400, 200, 0.7, 0.05, True, True)

#Представляем отдельную особь как случайное положение х

def create\_individual(data):

individual = data[:]

random.shuffle(individual)

return individual

ga.create\_individual = create\_individual

#Одноточечный кроссинговер

def crossover(parent\_1, parent\_2):

crossover\_index = random.randrange(1, len(parent\_1))

child\_1a = parent\_1[:crossover\_index]

child\_1b = [i for i in parent\_2 if i not in child\_1a]

child\_1 = child\_1a + child\_1b

child\_2a = parent\_2[crossover\_index:]

child\_2b = [i for i in parent\_1 if i not in child\_2a]

child\_2 = child\_2a + child\_2b

return child\_1, child\_2

ga.crossover\_function = crossover

#Целочисленная мутация

def mutate(individual):

mutate\_index1 = random.randrange(len(individual))

mutate\_index2 = random.randrange(len(individual))

individual[mutate\_index1], individual[mutate\_index2] = individual[mutate\_index2], individual[mutate\_index1]

ga.mutate\_function = mutate

#Рулеточная селекция

def selection(population):

return random.choice(population)

ga.selection\_function = selection

#Реализация функции приспособленности

#Ищет такой х, при котором значение функции максимально

def fitness (individual, data):

fitness = -float("inf")

x = -4

for item in individual:

if item != 1:

x += 0.01

else:

fitness = 1 / x

break

return fitness

ga.fitness\_function = fitness

ga.run()

print(ga.best\_individual()) #Выводит самую приспособленную особь

**Вещественное кодирование:**

from pyeasyga.pyeasyga import GeneticAlgorithm

import random

import numpy as np

data = (-0.5, [-2.0]) #Формат задания значений (y(x), [x])

#Задаём параметры работы генетического алгоритма

ga = GeneticAlgorithm(data, 20, 30, 0.7, 0.01, True, True)

#(начальные данные, количество особей в популяции, количество генераций, вероятность применения оператора скрещивания, вероятность применения мутации к гену, вкл. выбор лучшей особи, максимизация целевой функции)

def create\_individual(data): #Функция создания начальной популяции

ind = [random.uniform(-4.0, -1e-10)] #Вещественное кодирование

print(1 / ind[0], ind) #Вывод начальной популяции на экран

return ind #[random.uniform(-4.0, -0.1) for \_ in range(len(data))]

ga.create\_individual = create\_individual

def crossover(parent\_1, parent\_2): #Функция кроссинговера ГА (арифметический кроссинговер, lambda = 0.2)

child\_1 = [parent\_1[0] \* 0.2 + parent\_2[0] \* 0.8]

child\_2 = [parent\_2[0] \* 0.2 + parent\_1[0]\* 0.8]

return child\_1, child\_2

ga.crossover\_function = crossover

def mutate(individual): #Функция мутации для вещественного кодирования

rnd = np.random.normal(0, 0.5)

if rnd < -0.5:

rnd = -0.5

elif rnd > 0.5:

rnd = 0.5

individual[0] = individual[0] + rnd

ga.mutate\_function = mutate

def selection(population): #Функция селекции (турнирный отбор)

ind1 = random.choice(population)

ind2 = random.choice(population)

if ind1.fitness > ind2.fitness:

return ind1

else:

return ind2

ga.selection\_function = selection

def fitness (individual, data): #Целевая функция

if individual[0] < -4.0: #Ограничения переменных ЦФ

individual[0] = -4.0

if individual[0] >= 0:

individual[0] = 1e-10

return 1 / individual[0] #Значение целевой функции

ga.fitness\_function = fitness

ga.run() #Запуск ГА

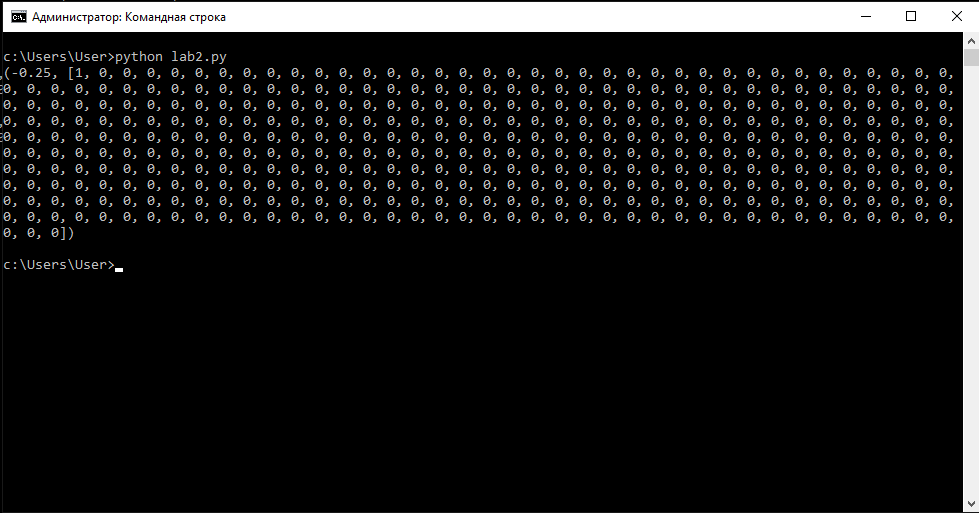
print("\nBest individual: ", ga.best\_individual(), "\n") #Вывод лучшей особи популяции (Решение)

for individual in ga.last\_generation(): #Вывод всех особей популяции в последнем поколении

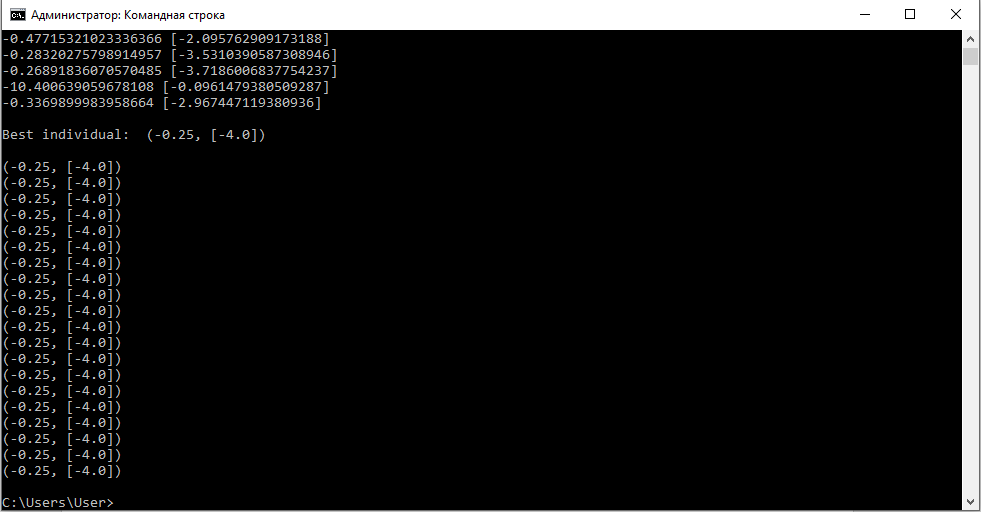
print(individual)

# Результаты работы генетического алгоритма

Целочисленное кодирование:



Вещественное кодирование:



В результате работы алгоритм, нашел максимум функции -0.25 при х равном -4

# Выводы

В результате проделанной работы была достигнута цель создания программы, применяющей генетический алгоритм для нахождения максимума функции y(x) = 1/x. x [-4, 0).

В ходе работы были выполнены все задачи, изучены принципы и структура работы генетического алгоритма, написана, настроена и протестирована программа, реализующая генетический алгоритм.