МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5**

**по дисциплине Методы поисковой оптимизации**

Работу выполнила\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ромашкина А.А.

Факультет Компьютерных технологий и прикладной математики

Направление подготовки 02.03.02 Фундаментальная информатика и

информационные технологии курс 4

Краснодар

2022

СОДЕРЖАНИЕ

[Задание 3](#_Toc122434046)

[Алгоритм 3](#_Toc122434047)

[Результат работы программы 3](#_Toc122434048)

[Листинг 5](#_Toc122434049)

Тема работы: Пчелиный алгоритм.

# Задание

Необходимо разработать программу оптимизации некоторой функции.

# Алгоритм

Программа содержит файлы:

1. pybee.py содержит базовый класс пчелы с вещественными координатами, класс улья, который управляет пчёлами в нём.
2. beeexamples.py содержит примеры пчёл для оптимизации различных функций.
3. lab5.py содержит функцию, осуществляющую запуск алгоритма для некоторой функции. Возвращает координаты лучшей найденной точки и значение функции в ней.

# Результат работы программы

Вся работа программы представлена на рисунках 1-2 ниже.

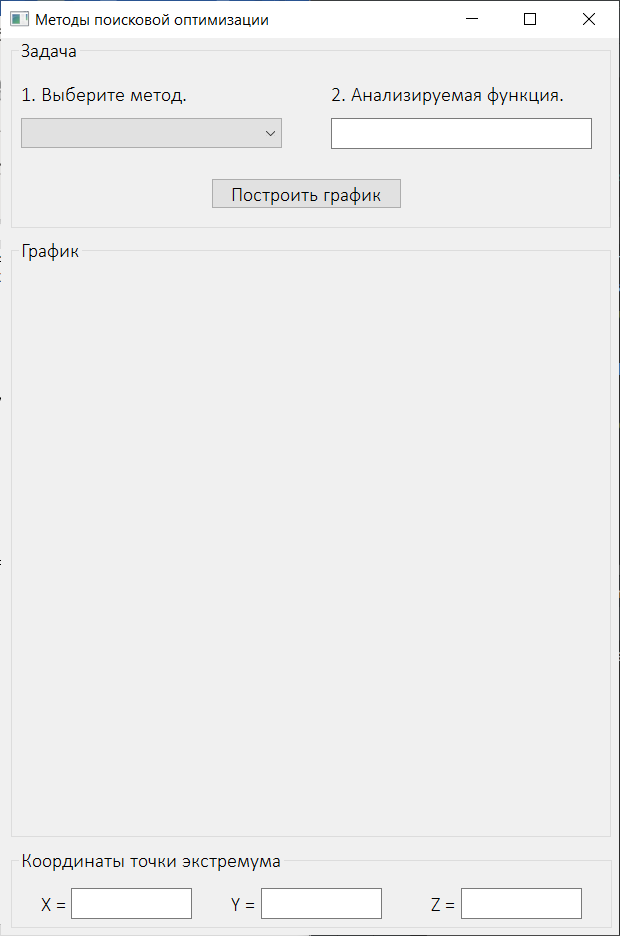


Рисунок 1 – Начальный вид программы при запуске

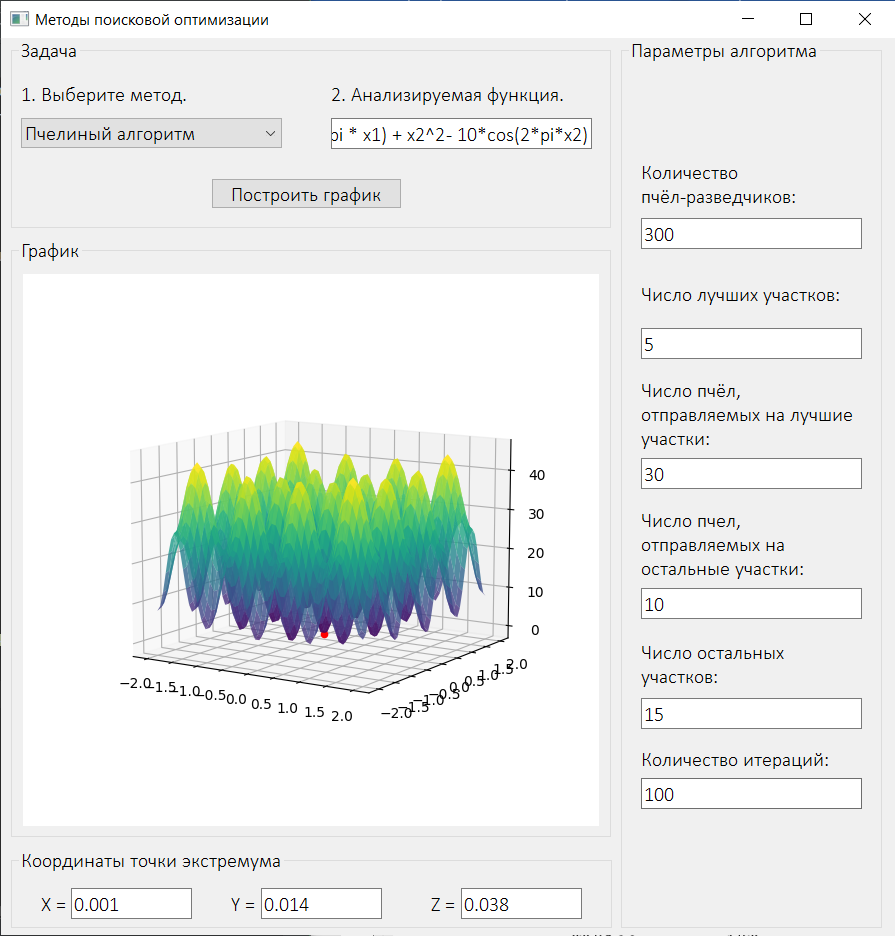


Рисунок 2 – Вид программы после выполнения выбранного алгоритма

# Листинг

**Файл pybee.py**

"""

Реализация алгоритма роя пчел

"""

import random

import math

class floatbee:

"""Класс пчел, где в качестве координат используется список дробных чисел"""

def \_\_init\_\_ (self):

# Положение пчелы (искомые величины)

self.position = None

# Интервалы изменений искомых величин (координат)

self.minval = None

self.maxval = None

# Значение целевой функции

self.fitness = 0.0

def calcfitness (self):

"""Расчет целевой функции. Этот метод необходимо перегрузить в производном классе.

Функция не возвращает значение целевой функции, а только устанавливает член self.fitness

Эту функцию необходимо вызывать после каждого изменения координат пчелы"""

pass

def sort (self, otherbee):

"""Функция для сортировки пчел по их целевой функции (здоровью) в порядке убывания."""

if self.fitness < otherbee.fitness:

return -1

elif self.fitness > otherbee.fitness:

return 1

else:

return 0

def otherpatch (self, bee\_list, range\_list):

"""Проверить находится ли пчела на том же участке, что и одна из пчел в bee\_list.

range\_list - интервал изменения каждой из координат"""

if len (bee\_list) == 0:

return True

for curr\_bee in bee\_list:

position = curr\_bee.getposition()

for n in range ( len (self.position) ):

if abs (self.position[n] - position[n]) > range\_list[n]:

return True

return False

def getposition (self):

"""Вернуть копию (!) своих координат"""

return [val for val in self.position]

def goto (self, otherpos, range\_list):

"""Перелет в окрестность места, которое нашла другая пчела. Не в то же самое место! """

# К каждой из координат добавляем случайное значение

self.position = [otherpos[n] + random.uniform (-range\_list[n], range\_list[n]) \

for n in range (len (otherpos) ) ]

# Проверим, чтобы не выйти за заданные пределы

self.checkposition()

# Расчитаем и сохраним целевую функцию

self.calcfitness()

def gotorandom (self):

# Заполним координаты случайными значениями

self.position = [random.uniform (self.minval[n], self.maxval[n]) for n in range (len (self.position) ) ]

self.checkposition()

self.calcfitness()

def checkposition (self):

"""Скорректировать координаты пчелы, если они выходят за установленные пределы"""

for n in range ( len (self.position) ):

if self.position[n] < self.minval[n]:

self.position[n] = self.minval[n]

elif self.position[n] > self.maxval[n]:

self.position[n] = self.maxval[n]

class hive:

"""Улей. Управляет пчелами"""

def \_\_init\_\_ (self, scoutbeecount, selectedbeecount, bestbeecount, \

selsitescount, bestsitescount, \

range\_list, beetype):

"""scoutbeecount - Количество пчел-разведчиков

selectedbeecount - количество пчел, посылаемое на один из лучших участков

selectedbeecount - количество пчел, посылаемое на остальные выбранные участки

selsitescount - количество выбранных участков

bestsitescount - количество лучших участков среди выбранных

beetype - класс пчелы, производный от bee

range\_list - список диапазонов координат для одного участка"""

self.scoutbeecount = scoutbeecount

self.selectedbeecount = selectedbeecount

self.bestbeecount = bestbeecount

self.selsitescount = selsitescount

self.bestsitescount = bestsitescount

self.beetype = beetype

self.range = range\_list

# Лучшая на данный момент позиция

self.bestposition = None

# Лучшее на данный момент здоровье пчелы (чем больше, тем лучше)

self.bestfitness = -1.0e9

# Начальное заполнение роя пчелами со случайными координатами

beecount = scoutbeecount + selectedbeecount \* selsitescount + bestbeecount \* bestsitescount

self.swarm = [beetype() for n in range (beecount)]

# Лучшие и выбранные места

self.bestsites = []

self.selsites = []

self.swarm.sort (key=lambda bee: bee.fitness, reverse = True)

self.bestposition = self.swarm[0].getposition()

self.bestfitness = self.swarm[0].fitness

def sendbees (self, position, index, count):

""" Послать пчел на позицию.

Возвращает номер следующей пчелы для вылета """

for n in range (count):

# Чтобы не выйти за пределы улея

if index == len (self.swarm):

break

curr\_bee = self.swarm[index]

if curr\_bee not in self.bestsites and curr\_bee not in self.selsites:

# Пчела не на лучших или выбранных позициях

curr\_bee.goto (position, self.range)

index += 1

return index

def nextstep (self):

"""Новая итерация"""

# Выбираем самые лучшие места и сохраняем ссылки на тех, кто их нашел

self.bestsites = [ self.swarm[0] ]

curr\_index = 1

for currbee in self.swarm [curr\_index: -1]:

# Если пчела находится в пределах уже отмеченного лучшего участка, то ее положение не считаем

if currbee.otherpatch (self.bestsites, self.range):

self.bestsites.append (currbee)

if len (self.bestsites) == self.bestsitescount:

break

curr\_index += 1

self.selsites = []

for currbee in self.swarm [curr\_index: -1]:

if currbee.otherpatch (self.bestsites, self.range) and currbee.otherpatch (self.selsites, self.range):

self.selsites.append (currbee)

if len (self.selsites) == self.selsitescount:

break

# Отправляем пчел на задание.

# Отправляем сначала на лучшие места.

# Номер очередной отправляемой пчелы. 0-ую пчелу никуда не отправляем

bee\_index = 1

for best\_bee in self.bestsites:

bee\_index = self.sendbees (best\_bee.getposition(), bee\_index, self.bestbeecount)

for sel\_bee in self.selsites:

bee\_index = self.sendbees (sel\_bee.getposition(), bee\_index, self.selectedbeecount)

# Оставшихся пчел пошлем куда попадет

for curr\_bee in self.swarm[bee\_index: -1]:

curr\_bee.gotorandom()

self.swarm.sort(key = lambda bee: bee.fitness, reverse = True)

self.bestposition = self.swarm[0].getposition()

self.bestfitness = self.swarm[0].fitness

**Файл beeexamples.py**

import random

import math

import pybee

class spherebee (pybee.floatbee):

"""Функция - сумма квадратов по каждой координате"""

# Количество координат

count = 2

@staticmethod

def getstartrange ():

return [150.0] \* spherebee.count

@staticmethod

def getrangekoeff ():

return [0.98] \* spherebee.count

def \_\_init\_\_ (self):

pybee.floatbee.\_\_init\_\_ (self)

self.minval = [-150.0] \* spherebee.count

self.maxval = [150.0] \* spherebee.count

self.position = [random.uniform (self.minval[n], self.maxval[n]) for n in range (spherebee.count) ]

self.calcfitness()

def calcfitness (self):

"""Расчет целевой функции. Этот метод необходимо перегрузить в производном классе.

Функция не возвращает значение целевой функции, а только устанавливает член self.fitness

Эту функцию необходимо вызывать после каждого изменения координат пчелы"""

self.fitness = 0.0

for val in self.position:

self.fitness -= val \* val

class Himmelblaybee(pybee.floatbee):

count = 2

@staticmethod

def getstartrange():

return [6.0] \* Himmelblaybee.count

@staticmethod

def getrangekoeff():

return [0.9] \* Himmelblaybee.count

def \_\_init\_\_(self):

pybee.floatbee.\_\_init\_\_(self)

self.minval = [-6.0] \* Himmelblaybee.count

self.maxval = [6.0] \* Himmelblaybee.count

self.position = [random.uniform(self.minval[n], self.maxval[n]) for n in range(Himmelblaybee.count)]

self.calcfitness()

def calcfitness(self):

"""Расчет целевой функции. Этот метод необходимо перегрузить в производном классе.

Функция не возвращает значение целевой функции, а только устанавливает член self.fitness

Эту функцию необходимо вызывать после каждого изменения координат пчелы"""

self.fitness = -Himmelblaybee.fun(self.position[0], self.position[1])

@staticmethod

def fun(x1, x2):

return (x1 \* x1 + x2 - 11) \*\* 2 + (x1 + x2 \* x2 - 7) \*\* 2

class Rastriginbee(pybee.floatbee):

count = 2

@staticmethod

def getstartrange():

return [5.0] \* Rastriginbee.count

@staticmethod

def getrangekoeff():

return [0.9] \* Rastriginbee.count

def \_\_init\_\_(self):

pybee.floatbee.\_\_init\_\_(self)

self.minval = [-5.0] \* Rastriginbee.count

self.maxval = [5.0] \* Rastriginbee.count

self.position = [random.uniform(self.minval[n], self.maxval[n]) for n in range(Rastriginbee.count)]

self.calcfitness()

def calcfitness(self):

"""Расчет целевой функции. Этот метод необходимо перегрузить в производном классе.

Функция не возвращает значение целевой функции, а только устанавливает член self.fitness

Эту функцию необходимо вызывать после каждого изменения координат пчелы"""

self.fitness = -Rastriginbee.fun(self.position[0], self.position[1])

@staticmethod

def fun(x1, x2):

return 20 + x1 \*\* 2 - 10 \* math.cos(2 \* math.pi \* x1) + x2 \*\* 2 - 10 \* math.cos(2 \* math.pi \* x2)

class rosenbrockbee (pybee.floatbee):

"""Функция Rosenbrock"""

# Количество координат

count = 2

@staticmethod

def getstartrange ():

return [10.0] \* rosenbrockbee.count

@staticmethod

def getrangekoeff ():

return [0.98] \* rosenbrockbee.count

def \_\_init\_\_ (self):

pybee.floatbee.\_\_init\_\_ (self)

self.minval = [-10.0] \* rosenbrockbee.count

self.maxval = [10.0] \* rosenbrockbee.count

self.position = [random.uniform (self.minval[n], self.maxval[n]) for n in range (rosenbrockbee.count) ]

self.calcfitness()

def calcfitness (self):

"""Расчет целевой функции. Этот метод необходимо перегрузить в производном классе.

Функция не возвращает значение целевой функции, а только устанавливает член self.fitness

Эту функцию необходимо вызывать после каждого изменения координат пчелы"""

self.fitness = 0.0

for n in range (1):

xi = self.position[n]

xi1 = self.position[n + 1]

self.fitness -= 100.0 \* ( ( (xi \* xi - xi1) \*\* 2) + ( (1 - xi) \*\* 2 ))

**Файл lab5.py**

import pybee

import beeexamples

def bee\_algorithm(scoutbeecount, bestsitescount, bestbeecount, selectedbeecount, selsitescount, maxiteration):

# Класс пчел, который будет использоваться в алгоритме

# beetype = beeexamples.spherebee

# beetype = beeexamples.rosenbrockbee

beetype = beeexamples.Rastriginbee

# beetype = beeexamples.Himmelblaybee

# Количество пчел-разведчиков

\_scoutbeecount = scoutbeecount

# Количество пчел, отправляемых на выбранные, но не лучшие участки

\_selectedbeecount = selectedbeecount

# Количество пчел, отправляемые на лучшие участки

\_bestbeecount = bestbeecount

# Количество выбранных, но не лучших, участков

\_selsitescount = selsitescount

# Количество лучших участков

\_bestsitescount = bestsitescount

# Максимальное количество итераций

\_maxiteration = maxiteration

# Через такое количество итераций без нахождения лучшего решения уменьшим область поиска

max\_func\_counter = 10

# Во столько раз будем уменьшать область поиска

koeff = beetype.getrangekoeff()

currhive = pybee.hive(\_scoutbeecount, \_selectedbeecount, \_bestbeecount, \_selsitescount, \_bestsitescount,

beetype.getstartrange(), beetype)

# Начальное значение целевой функции

best\_func = -1.0e9

# Количество итераций без улучшения целевой функции

func\_counter = 0

for n in range(\_maxiteration):

currhive.nextstep()

if currhive.bestfitness != best\_func:

# Найдено место, где целевая функция лучше

best\_func = currhive.bestfitness

func\_counter = 0

else:

func\_counter += 1

if func\_counter == max\_func\_counter:

# Уменьшим размеры участков

currhive.range = [currhive.range[m] \* koeff[m] for m in range(len(currhive.range))]

func\_counter = 0

return currhive.bestposition

# bee\_algorithm(300, 5, 30, 10, 15, 2000)