Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**Дисциплина: Методы поисковой оптимизации**

**Тема: Алгоритм пчелиного роя**

Работу выполнили: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Д. Воробьев А.П. Андреев

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.Е.Полупанова

**СОДЕРЖАНИЕ**

[Описание задачи 3](#_Toc148042159)

[Алгоритм 5](#_Toc148042160)

[Листинг программы 6](#_Toc148042161)

[Результаты работы программы 8](#_Toc148042162)

[Вывод 9](#_Toc148042163)

# Описание задачи

# Метод оптимизации пчелиным роем, или алгоритм оптимизации пчелиным роем (Bee Algorithm), является метаэвристикой, вдохновленной поведением пчёл в природе. Этот метод применяется для решения задач оптимизации, включая поиск оптимальных значений функций.

# На первом шаге алгоритма в точки, описываемые случайными координатами, отправляется некоторое количество пчел-разведчиков (пусть будет *S* пчел, от слова scout). В зависимости от значения целевой функции, которое определяется координатами пчелы, выделяются два вида перспективных участков на поверхности функции, вблизи которых возможно располагается глобальный максимум. А именно:

* Выбирается n лучших участков, где значения целевой функции больше всех
* Выбирается m так называемых выбранных участков, где значения целевой функции поменьше, чем на лучших участках, но эти участки все-равно являются неплохими с точки зрения значения целевой функции.

Здесь нужно обратить внимание на одну особенность, которая может быть важна при реализации алгоритма. Несколько пчел могут попасть на один и тот же участок (близко друг к другу, размер участка, или возможная близость пчел, задается отдельным параметром). Поэтому можно выделить два варианта поведения:

* Считать, что эти две пчелы нашли два разных пересекающихся участка, и оба этих участка отметить как лучшие или выбранные.
* Считать, что это один участок, центр которого находится в точке, которая соответствует пчеле с большим значением целевой функции.

В реализации, которая будет описана ниже, используется второй вариант поведения, так как он показался менее подверженным застреванию в локальных экстремумах.

В окрестность n лучших участков посылается *N* пчел, а в окрестность *m* выбранных участков посылается *M* пчел, причем на каждый из лучших участков должно приходиться больше пчел, чем на каждый из выбранных участков. Можно сделать так, что чем больше значение целевой функции, тем большее количество пчел будет отправляться на соответствующий участок, а можно *N* и *M* сделать фиксированными величинами.

После того как пчелы были отправлены на лучшие и выбранные участки, можно отправить тех же пчел-разведчиков на другие случайные точки.

После всех этих операций снова находятся *n* лучших и *m* выбранных участков, на этот раз среди всех пчел из роя, а не только среди разведчиков, и запоминается самое лучшее место на функции, значение больше которого пока не было найдено, это и будет промежуточным решением.

Затем алгоритм повторяется до тех пор пока не сработает какой-нибудь из критериев останова. Критериев останова может быть несколько. Например, если мы знаем значение целевой функции в глобальном экстремуме, то можем повторять алгоритм до тех пор, пока функция не достигнет некоторого значения, близкого к желаемому. Если значение функции в экстремуме неизвестно, то можем повторять шаги алгоритма до тех пор, пока на протяжении какого-то достаточно большого количества итераций найденное решение не будет улучшаться.

**Результаты выполнения**

Изображение выглядит как снимок экрана, текст, диаграмма, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 1 – Интерфейс программы

# Листинг программы

import time

import tkinter as tk

from random import uniform

from tkinter import messagebox

from tkinter import ttk

import matplotlib.pyplot as plt

import numpy as np

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

# Функция Химмельблау

def himmelblau\_function(x, y):

return (x\*\*2 + y - 11)\*\*2 + (x + y\*\*2 - 7)\*\*2

# Градиент функции Химмельблау

def gradient\_himmelblau(x, y):

dx = 4 \* x \* (x\*\*2 + y - 11) + 2 \* (x + y\*\*2 - 7)

dy = 2 \* (x\*\*2 + y - 11) + 4 \* y \* (x + y\*\*2 - 7)

return dx, dy

# Функция сферы

def sphere\_function(x, y):

return x\*\*2 + y\*\*2

# Градиент функции сферы

def gradient\_sphere(x, y):

dx = 2 \* x

dy = 2 \* y

return dx, dy

# Функция Бута

def booth\_function(x, y):

return (x + 2\*y - 7)\*\*2 + (2\*x + y - 5)\*\*2

# Градиент функции Бута

def booth\_gradient(x, y):

dx = 2 \* (x + 2\*y - 7) + 4 \* (2\*x + y - 5)

dy = 4 \* (x + 2\*y - 7) + 2 \* (2\*x + y - 5)

return dx, dy

# Функция Розенброка

def rosenbrock(x, y):

return (1 - x)\*\*2 + 100 \* (y - x\*\*2)\*\*2

# Задача для симплекс метода

def simplex\_task(x, y):

return 2\*x\*\*2 + 2\*x\*y + 2\*y\*\*2 - 4\*x - 6\*y

# Функция Растригина

def rastrigin(x, y):

return (x\*\*2 - 10 \* np.cos(2 \* np.pi \* x)) + (y\*\*2 - 10 \* np.cos(2 \* np.pi \* y)) + 20

def choose\_function(name\_function):

'''

Выбор функции по её названию

'''

if name\_function == "Химмельблау":

return himmelblau\_function

elif name\_function == "Сфера":

return sphere\_function

elif name\_function == "Функция Бута":

return booth\_function

elif name\_function == "Розенброк":

return rosenbrock

elif name\_function == "Растригина":

return rastrigin

else:

messagebox.showerror("Ошибка", "Выбранная функция пока не поддерживается")

return False

# Создание GUI окна

root = tk.Tk()

root.title("Визуализация функции с методом оптимизации")

points\_var = tk.StringVar(value=100)

points\_var\_3 = tk.StringVar(value=100)

individual\_var\_3 = tk.StringVar(value=20)

# Для симплекс метода

point\_x\_simplex = tk.StringVar(value=2)

point\_y\_simplex = tk.StringVar(value=2)

# Создание вкладок

tab\_control = ttk.Notebook(root)

tab5 = ttk.Frame(tab\_control)

tab\_control.add(tab5, text='Пчелиный алгоритм')

tab\_control.pack(expand=1, fill='both')

# 5 лаба, основной вызов функции

points\_var\_5 = tk.StringVar(value=200)

scout\_var\_5 = tk.StringVar(value=20)

perspective\_bee\_var\_5 = tk.StringVar(value=10)

best\_bee\_var\_5 = tk.StringVar(value=20)

perspective\_var\_5 = tk.StringVar(value=3)

best\_var\_5 = tk.StringVar(value=1)

size\_var\_5 = tk.StringVar(value=0.5)

def visualize\_5():

selected\_function = function\_var\_5.get()

function = choose\_function(selected\_function)

if (function == False):

return

points\_text\_5.delete("1.0", tk.END)

x = np.linspace(-5, 5, 500)

y = np.linspace(-5, 5, 500)

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

Z = function(X, Y)

ax5.cla()

ax5.plot\_surface(X, Y, Z, cmap='viridis', alpha=0.5)

ax5.set\_xlabel('X')

ax5.set\_ylabel('Y')

ax5.set\_zlabel('Z')

ax5.set\_title("Пчелинный алгоритм")

ax5.view\_init(elev=30, azim=45)

n\_point = int(points\_var\_5.get()) # количество итераций

n\_scout = int(scout\_var\_5.get()) # количество разведчиков

n\_perspective\_bee = int(perspective\_bee\_var\_5.get()) # количество пчёл на перспективном участке

n\_best\_bee = int(best\_bee\_var\_5.get()) # кол-во пчёл на элитных участках

n\_perspective = int(perspective\_var\_5.get()) # кол-во перспективных участков

n\_best = int(best\_var\_5.get()) # кол-во элитных участков

size = float(size\_var\_5.get()) # размер участка

def scout\_spawn(n\_scout):

'''

Функция получения участков с помощью пчёл разведчиков

'''

scouts = [[0] \* 3 for i in range(n\_scout)]

for i in range(n\_scout):

scouts[i][1] = uniform(-5,5)

scouts[i][2] = uniform(-5,5)

scouts[i][0] = function(scouts[i][1], scouts[i][2])

return scouts

def best\_bee\_spawn(bp, n\_best\_bee, size):

'''

Функция выборки точек на лучших участках

'''

bb = [[0] \* 3 for i in range(n\_best\_bee)]

bb[0][1] = bp[1]

bb[0][2] = bp[2]

bb[0][0] = function(bb[0][1], bb[0][2])

for i in range(1, n\_best\_bee):

bb[i][1] = bp[1] + uniform(-size, size)

bb[i][2] = bp[2] + uniform(-size, size)

bb[i][0] = function(bb[i][1], bb[i][2])

return bb

def perspective\_bee\_spawn(pp, n\_perspective\_bee, size):

'''

Функция выборки точек на перспективных учатсках

'''

pb = [[0] \* 3 for i in range(n\_perspective\_bee)]

pb[0][1] = pp[1]

pb[0][2] = pp[2]

pb[0][0] = function(pb[0][1], pb[0][2])

for i in range(1, n\_perspective\_bee):

pb[i][1] = pp[1] + uniform(-size, size)

pb[i][2] = pp[2] + uniform(-size, size)

pb[i][0] = function(pb[i][1], pb[i][2])

return pb

scouts = scout\_spawn(n\_scout) # сначала отправляем разведчиков

points=[]

for i in range(n\_scout):

point = ax5.scatter(scouts[i][1], scouts[i][2], function(scouts[i][1], scouts[i][2]), c="b", alpha=0.3)

points.append(point)

scouts.sort() # сортируем полученные ими участки по значению фитнес-функции

points\_text\_5.insert(tk.END, f"0. x= {scouts[0][1]:.4f}, y= {scouts[0][2]:.4f} f= {scouts[0][0]:.4f}\n")

canvas5.draw()

root.update()

time.sleep(0.01)

best\_point = []

perspective\_point = []

for i in range(n\_best): # первые n участков считаем элитными, добавляем их центральные точки в список

best\_point.append(scouts[i])

for i in range(n\_best,n\_best+n\_perspective): # остальные считаем перспективными

perspective\_point.append(scouts[i])

for point in points:

point.remove()

for i in range(n\_point):

points = []

best\_bee = [[0] for i in range(n\_best)]

perspective\_bee= [[0] for i in range(n\_perspective)]

for j in range(n\_best):

best\_bee[j] = best\_bee\_spawn(best\_point[j], n\_best\_bee, size) # выбираем точки среди элитных участков

for j in range(n\_perspective):

perspective\_bee[j] = perspective\_bee\_spawn(perspective\_point[j], n\_perspective\_bee, size) # затем выбираем точки среди перспективных участков

scouts = scout\_spawn(n\_scout) # заново отправляем разведчиков

for j in range(n\_scout): # отрисовка разведчиков

point = ax5.scatter(scouts[j][1], scouts[j][2], function(scouts[j][1], scouts[j][2]), c="b", alpha=0.3)

points.append(point)

for j in range(n\_best): # отрисовка элитных

for q in range(n\_best\_bee):

point = ax5.scatter(best\_bee[j][q][1], best\_bee[j][q][2], function(best\_bee[j][q][1], best\_bee[j][q][2]), c="b", alpha=0.3)

points.append(point)

for j in range(n\_perspective): # отрисовка перспективных

for q in range(n\_perspective\_bee):

point = ax5.scatter(perspective\_bee[j][q][1], perspective\_bee[j][q][2], function(perspective\_bee[j][q][1], perspective\_bee[j][q][2]), c="b", alpha=0.3)

points.append(point)

b = [] # собираем все точки в 1 список и сортируем

for j in range(n\_perspective):

b.extend(perspective\_bee[j])

for j in range(n\_best):

b.extend(best\_bee[j])

for j in range(n\_scout):

b.extend(scouts)

b.sort()

# заново формируем список с элитными и с перспективными точками

best\_point = []

perspective\_point = []

for j in range(n\_best):

best\_point.append(b[j])

for j in range(n\_best, n\_best + n\_perspective):

perspective\_point.append(b[j])

point = ax5.scatter(best\_point[0][1], best\_point[0][2], function(best\_point[0][1], best\_point[0][2]), c="r", alpha=1)

points.append(point)

points\_text\_5.insert(tk.END, f"{i}. x= {best\_point[0][1]:.4f}, y= {best\_point[0][2]:.4f} f= {best\_point[0][0]:.4f}\n")

canvas5.draw()

root.update()

time.sleep(0.001)

if i != n\_point - 1:

for point in points:

point.remove()

# 6 лаба - основной вызов

points\_var\_6 = tk.StringVar(value=200)

anti\_var\_6 = tk.StringVar(value=50)

best\_var\_6 = tk.StringVar(value=10)

clone\_var\_6 = tk.StringVar(value=20)

cmut\_var\_6 = tk.StringVar(value=0.2)

rand\_var\_6 = tk.StringVar(value=10)

# Создание 5 вкладки

fig = plt.figure(figsize=(8, 6), dpi=100)

ax5 = fig.add\_subplot(111, projection='3d')

canvas5 = FigureCanvasTkAgg(fig, master=tab5)

canvas5.draw()

canvas5.get\_tk\_widget().pack()

function\_label\_5 = tk.Label(tab5, text="Функция:")

function\_label\_5.place(x=10, y=10)

function\_var\_5 = tk.StringVar(value="Химмельблау")

function\_dropdown\_5 = tk.OptionMenu(tab5, function\_var\_5, "Химмельблау", "Сфера", "Функция Бута", "Розенброк", "Растригина")

function\_dropdown\_5.place(x=200, y=10)

points\_label\_5 = tk.Label(tab5, text="Количество итераций:")

points\_label\_5.place(x=10, y=50)

points\_entry\_5 = tk.Entry(tab5, textvariable=points\_var\_5)

points\_entry\_5.place(x=200, y=50)

scout\_label\_5 = tk.Label(tab5, text="Количество разведчиков:")

scout\_label\_5.place(x=10, y=90)

scout\_entry\_5 = tk.Entry(tab5, textvariable=scout\_var\_5)

scout\_entry\_5.place(x=200, y=90)

perspective\_bee\_label\_5 = tk.Label(tab5, text="Пчёл на перспективном участке:")

perspective\_bee\_label\_5.place(x=10, y=130)

perspective\_bee\_entry\_5 = tk.Entry(tab5, textvariable=perspective\_bee\_var\_5)

perspective\_bee\_entry\_5.place(x=200, y=130)

best\_bee\_label\_5 = tk.Label(tab5, text="Пчел на лучшем участке:")

best\_bee\_label\_5.place(x=10, y=170)

best\_bee\_entry\_5 = tk.Entry(tab5, textvariable=best\_bee\_var\_5)

best\_bee\_entry\_5.place(x=200, y=170)

perspective\_label\_5 = tk.Label(tab5, text="Перспективных участков:")

perspective\_label\_5.place(x=10, y=210)

perspective\_entry\_5 = tk.Entry(tab5, textvariable=perspective\_var\_5)

perspective\_entry\_5.place(x=200, y=210)

best\_label\_5 = tk.Label(tab5,text="Лучших участков:")

best\_label\_5.place(x=10, y=250)

best\_entry\_5 = tk.Entry(tab5, textvariable=best\_var\_5)

best\_entry\_5.place(x=200, y=250)

size\_label\_5 = tk.Label(tab5,text="Размер участков:")

size\_label\_5.place(x=10, y=290)

size\_entry\_5 = tk.Entry(tab5, textvariable=size\_var\_5)

size\_entry\_5.place(x=200, y=290)

visualize\_button\_5 = tk.Button(tab5, text="Визуализировать", command=visualize\_5)

visualize\_button\_5.place(x=100, y=350)

points\_text\_5 = tk.Text(tab5, height=10, width=40)

points\_text\_5.place(x=10, y=380)

# Запуск GUI

root.mainloop()