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 №4**

**Дисциплина: Методы поисковой оптимизации**

**Тема: Алгоритм роя частиц**

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

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

# Алгоритм оптимизации роя частиц (Particle Swarm Optimization, PSO) представляет собой мощный метод, вдохновленный природными явлениями, такими как движение стаи птиц или роя частиц. Он используется для решения задач оптимизации в разнообразных областях, включая машинное обучение, инженерию и финансы. Основной целью PSO является поиск оптимального решения в многомерном пространстве параметров, минимизируя целевую функцию.

# Алгоритм PSO базируется на следующих ключевых принципах:

# Частицы: Каждая частица представляет собой потенциальное решение задачи оптимизации и характеризуется своим текущим положением в пространстве параметров и скоростью движения.

# Лучшее личное решение: Каждая частица хранит информацию о лучшем решении, которое она смогла найти в течение своего пути.

# Лучшее глобальное решение: Все частицы также имеют информацию о лучшем решении, найденном где-либо в рое, и это решение является глобальным оптимумом.

# Процесс оптимизации с использованием PSO можно описать следующим образом:

# Инициализация: Создается начальное множество частиц с случайными положениями и скоростями. Каждая частица также инициализируется своим лучшим локальным решением и лучшим глобальным решением.

# Обновление положения и скорости: Каждая частица обновляет свои положение и скорость на основе её текущего положения, скорости, лучшего локального решения и лучшего глобального решения. Это обновление происходит с использованием определенных формул, зависящих от параметров алгоритма.

# Оценка целевой функции: Для каждой частицы вычисляется значение целевой функции, которую необходимо минимизировать.

# Обновление лучших решений: Если текущее решение частицы лучше её лучшего локального решения, то оно становится новым лучшим локальным решением. Если текущее локальное решение лучше лучшего глобального решения в рое, то оно становится новым лучшим глобальным решением.

# Повторение: Шаги 2-4 выполняются в течение нескольких итераций или до достижения критерия остановки, такого как заданное число итераций или достижение определенной точности.

# Завершение: По завершении алгоритма, лучшее глобальное решение является приближенным оптимумом задачи.

# Алгоритм

Алгоритм роя частиц (PSO) — это метод оптимизации, в котором множество частиц ищет оптимальное решение в многомерном пространстве параметров. Основные формулы для алгоритма роя частиц в его каноническом виде включают:

1. Формула коррекции скорости

Здесь vi, t – i-я компонента скорости при t-ой итерации алгоритма xi, t – i-я координата частицы при t- ой итерации алгоритма pi – i-я координата лучшего решения, найденного частицей gi – i-я координата лучшего решения, найденного всеми частицами rp, rg – случайные числа в интервале (0, 1) φp, φg – весовые коэффициенты, которые надо подбирать под конкретную задачу

1. Обновление положения частицы

**Результаты выполнения**

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 1 – Первоначальная итерация

Изображение выглядит как текст, снимок экрана, компьютер, дизайн

Автоматически созданное описание

Рисунок 2 – 4 итерация

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 3 – Последняя итерация

# Вывод

Преимущества PSO включают его простоту реализации, способность к работе в многомерных пространствах параметров и способность находить глобальные оптимумы в некоторых случаях. Тем не менее, эффективность алгоритма может зависеть от выбора параметров и конкретной задачи оптимизации.

В заключение, алгоритм роя частиц - это мощный метод оптимизации, который находит применение в различных областях, где требуется поиск минимума целевой функции в многомерных пространствах параметров.

# Листинг программы

import math

import random

import numpy as np

def my\_function(x, y, limits\_x, limits\_y):

if x < limits\_x[0] or x > limits\_x[1]:

return 10000 + math\_function(x, y)

if y < limits\_y[0] or y > limits\_y[1]:

return 10000 + math\_function(x, y)

return math\_function(x, y)

def math\_function(x, y):

return (-x \* np.sin(np.sqrt(abs(x)))) + (-y \* np.sin(np.sqrt(abs(y))))

def trigger():

particles\_count = 300 # Количество частиц

iterations\_count = 50 # Количество итераций

limits\_x = (-10, 10) # Ограничения X

limits\_y = (-10, 10) # Ограничения Y

coeff\_personal = 2 # Фи пе (коэффициент при своем лучшем значении)

coeff\_global = 9 # Фи ге (коэффициент при глобальном лучшем значении)

coeff\_speed = 0.4 # Общий коэффициент скорости (от 0 до 1)

# Создание первоначального роя частиц

particles = []

global\_best = (None, None, None)

if coeff\_personal + coeff\_global < 4:

s = coeff\_personal + coeff\_global

coeff\_personal = coeff\_personal \* (4 / s)

coeff\_global = coeff\_global \* (4 / s)

for i in range(particles\_count):

particle\_x = random.uniform(limits\_x[0], limits\_x[1])

particle\_y = random.uniform(limits\_y[0], limits\_y[1])

particle\_value = my\_function(particle\_x, particle\_y, limits\_x, limits\_y)

# Задаём конфигурацию каждой частице

particles.append({

'x': particle\_x,

'y': particle\_y,

'value': particle\_value,

# Скорость в декартовом пространстве (вектор)

'speed': (random.uniform(limits\_x[0] - particle\_x, limits\_x[1] - particle\_x),

random.uniform(limits\_y[0] - particle\_y, limits\_y[1] - particle\_y)),

'best': (particle\_x, particle\_y, particle\_value)

})

# Устанавливаем наилучший глобальный минимум

if (global\_best[2] is None) or (particle\_value < global\_best[2]):

global\_best = (particle\_x, particle\_y, particle\_value)

x, y = [], []

result = []

# Итерации

for j in range(iterations\_count):

x.clear()

y.clear()

# Проход по каждой частице

for i in range(particles\_count):

particle = particles[i]

# Считаем лучшее собственное значение

particle\_value = my\_function(particle['x'], particle['y'], limits\_x, limits\_y)

if particle\_value < particle['best'][2]:

particle['best'] = (particle['x'], particle['y'], particle\_value)

# Считаем лучшее глобальное значение

if particle\_value < global\_best[2]:

global\_best = (particle['x'], particle['y'], particle\_value)

# Считаем новый вектор скорости

particle\_coords = (particle['x'], particle['y'])

rand\_personal = random.uniform(0, 1)

rand\_global = random.uniform(0, 1)

coeff\_sum = coeff\_personal + coeff\_global

tmp = (2 \* coeff\_speed) / abs(2 - coeff\_sum - math.sqrt(coeff\_sum \* coeff\_sum - 4 \* coeff\_sum))

new\_speed = []

for coord\_i in range(len(particle['speed'])):

v = particle['speed'][coord\_i]

new\_speed.append(tmp \* (v + coeff\_personal \* rand\_personal \* (

particle['best'][coord\_i] - particle\_coords[coord\_i]) + coeff\_global \* rand\_global \* (

global\_best[coord\_i] - particle\_coords[coord\_i])))

new\_speed = tuple(new\_speed)

particle['speed'] = new\_speed

# Двигаем частицу

particle['x'] += particle['speed'][0]

particle['y'] += particle['speed'][1]

x.append(particle['x'])

y.append(particle['y'])

result.append({

"iteration": j,

"x": global\_best[0],

"y": global\_best[1],

"z": global\_best[2]

})

print(f'Итерация {j} => Best: {global\_best}')

print(f'Итоговый минимум: {global\_best}')

return result

if \_\_name\_\_ == '\_\_main\_\_':

trigger()import tkinter

from tkinter import \*

from tkinter import scrolledtext, messagebox

from tkinter.ttk import Combobox, Notebook, Style

import numpy as np

from matplotlib import pyplot as plt

from matplotlib.backends.backend\_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)

from lab4.core import trigger

def func\_to\_display(data):

x = data[1:]

y = data[:-1]

return np.sum((-x \* np.sin(np.sqrt(abs(x)))) + (-y \* np.sin(np.sqrt(abs(y)))),axis=0)

def make\_data\_lab\_4():

x = np.linspace(-10, 10, 100)

y = np.linspace(-10, 10, 100)

x\_grid, y\_grid = np.meshgrid(x, y)

z = func\_to\_display(np.array([x\_grid, y\_grid]))

return x\_grid, y\_grid, z

def main():

window = Tk()

width = window.winfo\_screenwidth()

height = window.winfo\_screenheight()

window.geometry("%dx%d" % (width, height))

window.title("SEARCH OPTIMISATION METHODS")

fig = plt.figure(figsize=(11, 11))

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

canvas = FigureCanvasTkAgg(fig, master=window)

canvas.draw()

canvas.get\_tk\_widget().pack(side=tkinter.RIGHT, fill=tkinter.BOTH)

toolbar = NavigationToolbar2Tk(canvas, window)

toolbar.update()

canvas.get\_tk\_widget().pack(side=tkinter.RIGHT, fill=tkinter.BOTH)

style = Style()

style.theme\_create("dummy", parent="alt", settings={

"TNotebook": {"configure": {"tabmargins": [2, 5, 2, 0]}},

"TNotebook.Tab": {

"configure": {"padding": [5, 1], "background": "#808080"},

"map": {"background": [("selected", "FFFF00")],

"expand": [("selected", [1, 1, 1, 0])]}}})

style.theme\_use("dummy")

tab\_control = Notebook(window)

def draw\_lab\_3():

fig.clf()

x, y, z = make\_data\_lab\_4()

if combo\_tab\_3.get() == "Min":

min\_max = True

else:

min\_max = False

ax = fig.add\_subplot(projection='3d')

ax.plot\_surface(x, y, z, rstride=5, cstride=5, alpha=0.3, cmap="inferno")

canvas.draw()

result = trigger()

canvas.draw()

window.update()

# Эти 4 строки ниже это считай удалить точку/точки

fig.clf()

ax = fig.add\_subplot(projection='3d')

ax.plot\_surface(x, y, z, rstride=5, cstride=5, alpha=0.5, cmap="inferno")

canvas.draw()

for item in result:

x = item.get("x")

y = item.get("y")

z = item.get("z")

ax.scatter(x, y, z, c="red")

txt\_tab\_1.insert(INSERT,

f"{item.get('iteration')}) ({round(x, 3)})({round(y, 3)}) = {z}\n")

canvas.draw()

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

window.update()

messagebox.showinfo('Уведомление', 'Готово')

def delete\_lab\_3():

txt\_tab\_1.delete(1.0, END)

tab\_1 = Frame(tab\_control)

tab\_control.add(tab\_1, text="LAB\_1")

main\_f\_tab\_1 = LabelFrame(tab\_1, text="Parameters")

left\_f\_tab\_1 = Frame(main\_f\_tab\_1)

right\_f\_tab\_1 = Frame(main\_f\_tab\_1)

txt\_f\_tab\_1 = LabelFrame(tab\_1, text="Execution and results")

lbl\_5\_tab\_1 = Label(tab\_1, text="Лаба 4")

txt\_tab\_1 = scrolledtext.ScrolledText(txt\_f\_tab\_1)

btn\_del\_tab\_1 = Button(tab\_1, text="Clear", command=delete\_lab\_3, foreground="black", background="grey")

btn\_tab\_1 = Button(tab\_1, text="Execute", foreground="black", background="yellow", command=draw\_lab\_3)

lbl\_5\_tab\_1.pack(side=TOP, padx=5, pady=5)

main\_f\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH, expand=True)

left\_f\_tab\_1.pack(side=LEFT, fill=BOTH, expand=True)

right\_f\_tab\_1.pack(side=RIGHT, fill=BOTH, expand=True)

txt\_tab\_1.pack(padx=5, pady=5, fill=BOTH, expand=True)

combo\_tab\_3 = Combobox(right\_f\_tab\_1)

combo\_tab\_3['values'] = ("Min", "Max")

combo\_tab\_3.set("Min")

btn\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

btn\_del\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

txt\_f\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

tab\_control.pack(side=RIGHT, fill=BOTH, expand=True)

window.mainloop()

window.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

main()import tkinter

from tkinter import \*

from tkinter import scrolledtext, messagebox

from tkinter.ttk import Combobox, Notebook, Style

import numpy as np

from matplotlib import pyplot as plt

from matplotlib.backends.backend\_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)

from lab4.core import trigger

def func\_to\_display(data):

x = data[1:]

y = data[:-1]

return np.sum((-x \* np.sin(np.sqrt(abs(x)))) + (-y \* np.sin(np.sqrt(abs(y)))),axis=0)

def make\_data\_lab\_4():

x = np.linspace(-10, 10, 100)

y = np.linspace(-10, 10, 100)

x\_grid, y\_grid = np.meshgrid(x, y)

z = func\_to\_display(np.array([x\_grid, y\_grid]))

return x\_grid, y\_grid, z

def main():

window = Tk()

width = window.winfo\_screenwidth()

height = window.winfo\_screenheight()

window.geometry("%dx%d" % (width, height))

window.title("SEARCH OPTIMISATION METHODS")

fig = plt.figure(figsize=(11, 11))

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

canvas = FigureCanvasTkAgg(fig, master=window)

canvas.draw()

canvas.get\_tk\_widget().pack(side=tkinter.RIGHT, fill=tkinter.BOTH)

toolbar = NavigationToolbar2Tk(canvas, window)

toolbar.update()

canvas.get\_tk\_widget().pack(side=tkinter.RIGHT, fill=tkinter.BOTH)

style = Style()

style.theme\_create("dummy", parent="alt", settings={

"TNotebook": {"configure": {"tabmargins": [2, 5, 2, 0]}},

"TNotebook.Tab": {

"configure": {"padding": [5, 1], "background": "#808080"},

"map": {"background": [("selected", "FFFF00")],

"expand": [("selected", [1, 1, 1, 0])]}}})

style.theme\_use("dummy")

tab\_control = Notebook(window)

def draw\_lab\_3():

fig.clf()

x, y, z = make\_data\_lab\_4()

if combo\_tab\_3.get() == "Min":

min\_max = True

else:

min\_max = False

ax = fig.add\_subplot(projection='3d')

ax.plot\_surface(x, y, z, rstride=5, cstride=5, alpha=0.3, cmap="inferno")

canvas.draw()

result = trigger()[1]

canvas.draw()

window.update()

# Эти 4 строки ниже это считай удалить точку/точки

fig.clf()

ax = fig.add\_subplot(projection='3d')

ax.plot\_surface(x, y, z, rstride=5, cstride=5, alpha=0.5, cmap="inferno")

canvas.draw()

iter = 0

print(result[iter].get("data"))

for item in result[iter].get("data"):

data = item

x = data.get("x")

y = data.get("y")

z = data.get("value")

ax.scatter(x, y, z, c="red")

txt\_tab\_1.insert(INSERT,

f"{result[iter].get('iteration')}) ({round(x, 3)})({round(y, 3)}) = {z}\n")

canvas.draw()

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

window.update()

messagebox.showinfo('Уведомление', 'Готово')

def delete\_lab\_3():

txt\_tab\_1.delete(1.0, END)

tab\_1 = Frame(tab\_control)

tab\_control.add(tab\_1, text="LAB\_1")

main\_f\_tab\_1 = LabelFrame(tab\_1, text="Parameters")

left\_f\_tab\_1 = Frame(main\_f\_tab\_1)

right\_f\_tab\_1 = Frame(main\_f\_tab\_1)

txt\_f\_tab\_1 = LabelFrame(tab\_1, text="Execution and results")

lbl\_5\_tab\_1 = Label(tab\_1, text="Лаба 4")

txt\_tab\_1 = scrolledtext.ScrolledText(txt\_f\_tab\_1)

btn\_del\_tab\_1 = Button(tab\_1, text="Clear", command=delete\_lab\_3, foreground="black", background="grey")

btn\_tab\_1 = Button(tab\_1, text="Execute", foreground="black", background="yellow", command=draw\_lab\_3)

lbl\_5\_tab\_1.pack(side=TOP, padx=5, pady=5)

main\_f\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH, expand=True)

left\_f\_tab\_1.pack(side=LEFT, fill=BOTH, expand=True)

right\_f\_tab\_1.pack(side=RIGHT, fill=BOTH, expand=True)

txt\_tab\_1.pack(padx=5, pady=5, fill=BOTH, expand=True)

combo\_tab\_3 = Combobox(right\_f\_tab\_1)

combo\_tab\_3['values'] = ("Min", "Max")

combo\_tab\_3.set("Min")

btn\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

btn\_del\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

txt\_f\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

tab\_control.pack(side=RIGHT, fill=BOTH, expand=True)

window.mainloop()

window.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

main()