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 оптимизации пчелиным роем

Работу выполнили: _____ Парфинцов. Е. А.

_____ Татарян. Е. В.

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Математическое и программное обеспечение компьютерных технологий

Преподаватель: _____ Полупанова Е. Е.

Краснодар
2024

Цель работы: разработать алгоритм оптимизации пчелиным роем на функциях Химмельблау и Розенброка.

Ход работы

Основная идея алгоритма состоит в моделировании поведения пчел при поиске нектара.

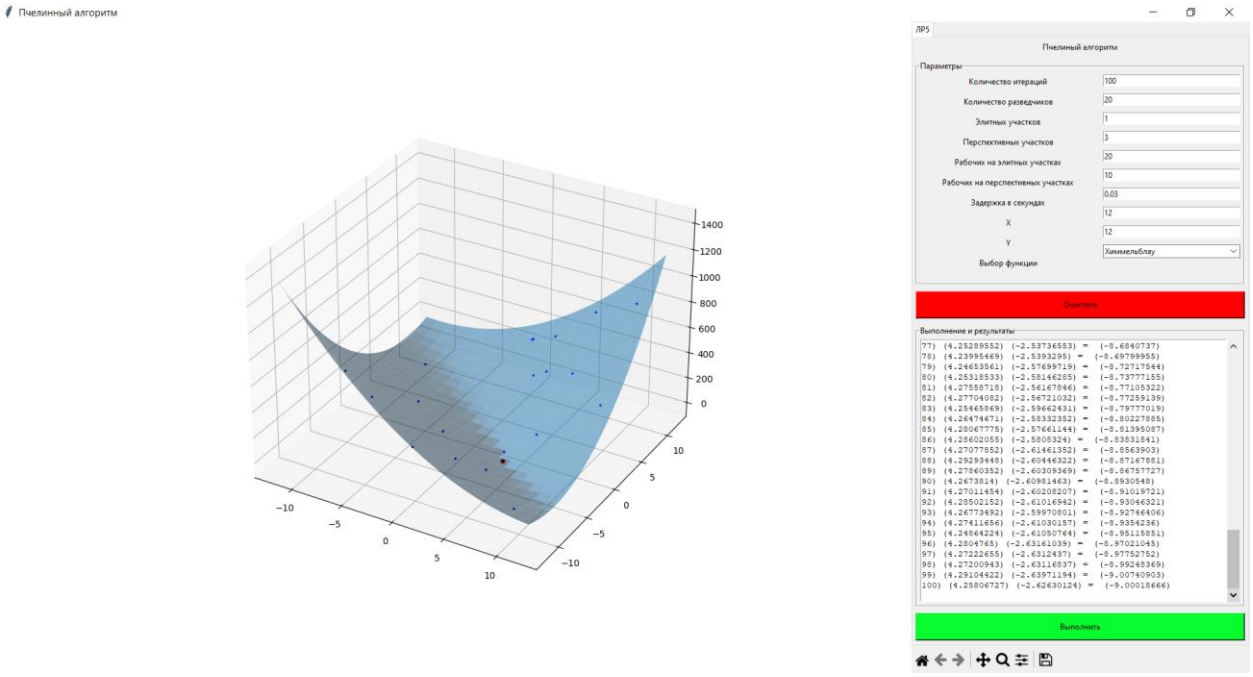
Схема алгоритма имеет следующий вид. Сначала из улья вылетает в случайном направлении некоторое число пчел-разведчиков, которые пытаются отыскать участки, где есть нектар. Через какое-то время пчелы возвращаются в улей и сообщают другим пчелам, где и сколько они нашли нектара. После этого на найденные участки отправляются рекрутированные (рабочие) пчелы, причем чем больше на данном участке предполагается найти нектара, тем больше пчел летит к этому участку. Пчелы-разведчики опять улетают искать другие участки, после чего процесс повторяется.

В случае оптимизации некоторой фитнес-функции кол-во нектара в некоторой точке пространства полагаем пропорциональным значению фитнес-функции в этой точке.

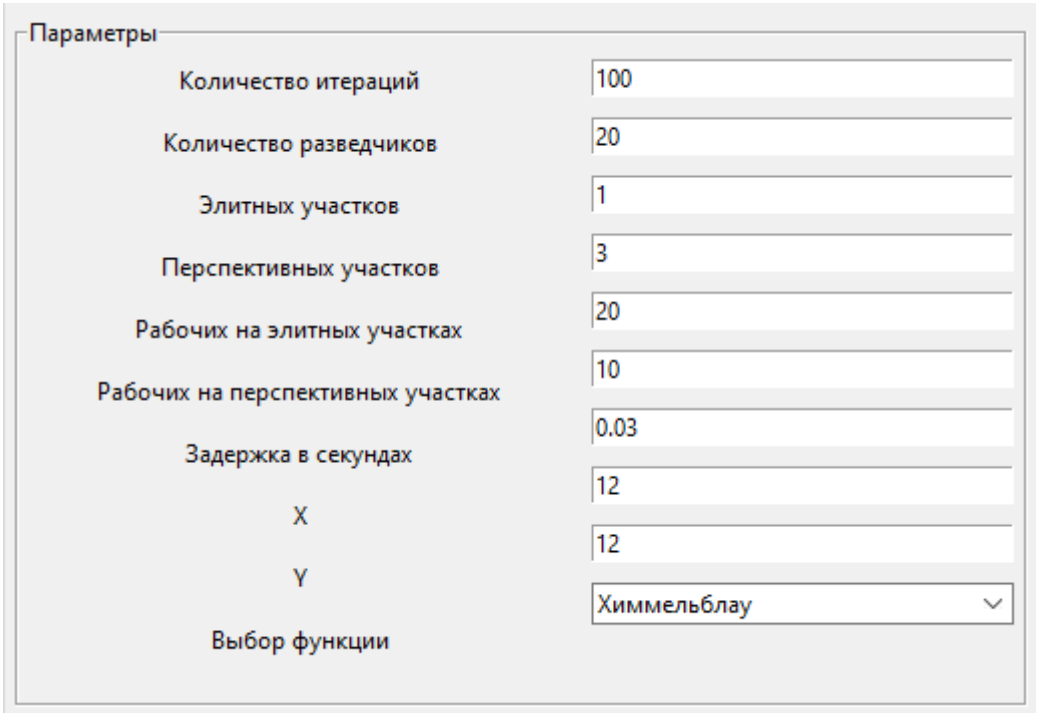
Одна итерация алгоритма включает в себя следующие основные шаги:

1. В случайные точки пространства поиска отправляем пчел-разведчиков.
2. На основании значений фитнес-функции, вычисленных в указанных точках, выделяем некоторое число элитных участков – подобластей пространства поиска, соответствующих максимальным значениям этой функции. Аналогично определяем некоторое число перспективных участков, которые соответствуют значениям фитнес-функции, близким к максимальным.
3. На каждый из элитных и перспективных участков посылаем определенное число рабочих пчел. Используя соответствующие значения фитнес-функции, находим новые элитные и перспективные участки. При этом выборе учитываем результаты, полученные как пчелами-разведчиками, так и рабочими пчелами.

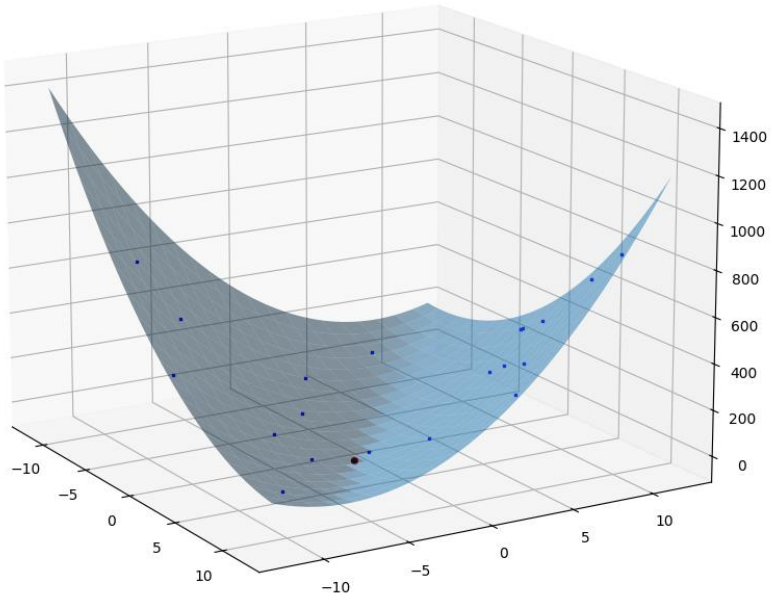
Реализация данного алгоритма на Python:



Скриншот 1. Внешний вид приложения.

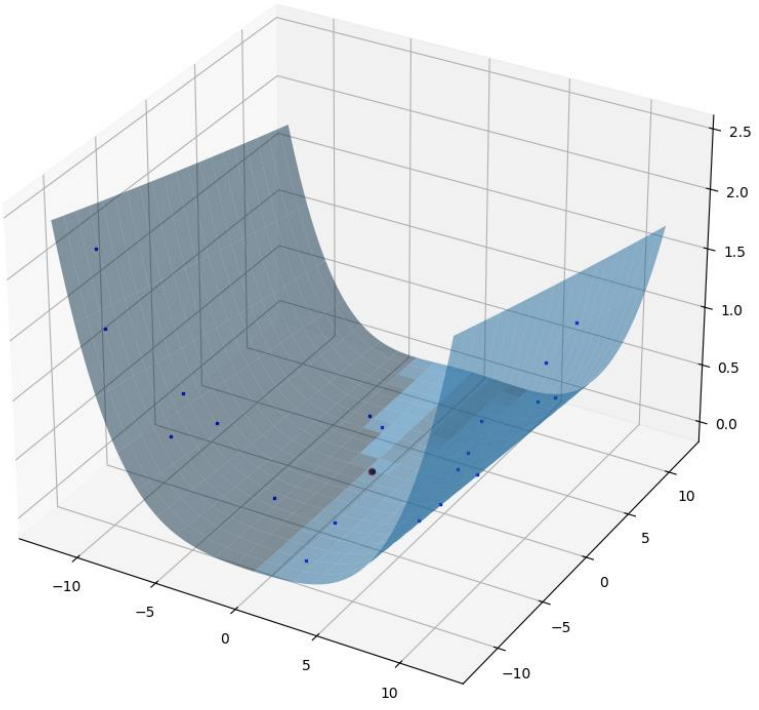


Скриншот 2. Окно задания параметров.




```
100)  (4.25806727)  (-2.62630124)  =   (-9.00018666)
```

Скриншот 3. Результаты работы для функции Химмельблау.




```
100)  (0.4320253)  (-0.00674188)  =   (-0.35131261)
```

Скриншот 4. Результаты работы для функции Розенброка.

Листинг 1. (Реализация алгоритма)

```
import random
from operator import itemgetter

class Bees:
    # Инициализируем пчелиную семью с заданными параметрами: целевая функция,
количество разведчиков, элитные пчелы,
    # перспективные пчелы, радиус, координаты.
    def __init__(self, func, scouts, elite, perspect, bees_to_leet,
bees_to_persp, radius, position_x, position_y):
        self.func = func

        self.pos_x = float(position_x)
        self.pos_y = float(position_y)

        self.scouts = [[random.uniform(-self.pos_x, self.pos_x), random.uniform(-
self.pos_y, self.pos_y), float(0.0)] for _ in
                       range(scouts)]

        for i in self.scouts:
            i[2] = self.func(i[0], i[1])

        self.n_workers = elite * bees_to_leet + perspect * bees_to_persp
        self.e = elite
        self.p = perspect
        self.b_leet = bees_to_leet
        self.b_persp = bees_to_persp

        max_b = max(self.scouts, key=itemgetter(2))
        self.workers = [[self.pos_x, self.pos_y, max_b[2]] for _ in
range(self.n_workers)]

        self.bees = list()

        self.selected = list()

        self.rad = radius

    def send_scouts(self):
        #Случайным образом генерируются новые позиции для скаутов и оценивается
их пригодность с помощью целевой функции.
        for unit in self.scouts:
            unit[0] = random.uniform(-self.pos_x, self.pos_x)
            unit[1] = random.uniform(-self.pos_y, self.pos_y)
            unit[2] = self.func(unit[0], unit[1])

    def research_reports(self):
```

```
        # Объединяет разведчиков и рабочих пчел, сортирует их по пригодности и
выбирает элитных и перспективных пчел.
        self.bees = self.scouts + self.workers
        self.bees = sorted(self.bees, key=itemgetter(2), reverse=False)

        self.selected = self.bees[:self.e + self.p]

    def get_best(self):
        # Возвращает наилучшее решение, найденное пчелиной семьей.
        return self.bees[0]

    def send_workers(self, bee_part, sector, radius):
        # Отправляет рабочих пчёл для использования или изучения определенных
регионов в пространстве поиска
        # на основе выбранных элитных и перспективных пчел.
        for bee in bee_part:
            bee[0] = random.uniform(sector[0] - radius, sector[0] + radius)
            bee[1] = random.uniform(sector[1] - radius, sector[1] + radius)
            bee[2] = random.uniform(sector[2] - radius, sector[2] + radius)

    def selected_search(self, param):
        # Выполняет процесс поиска, отправляя рабочих пчёл в разные регионы на
основе выбранных элитных и перспективных пчел.
        for i in range(self.e):
            Bees.send_workers(self.func,
                              self.workers[i * self.b_leet:i * self.b_leet +
self.b_leet],
                              self.selected[i],
                              self.rad * param)

        for i in range(self.p):
            Bees.send_workers(self.func,
                              self.workers[self.e * self.b_leet + i *
self.b_persp:self.e * self.b_leet +

i * self.b_persp + self.b_persp],
                              self.selected[self.e + i],
                              self.rad * param)
```

Листинг 2. (Приложение и запуск алгоритма)

```
import tkinter
import time
from tkinter import *
from tkinter import scrolledtext, messagebox
from tkinter.ttk import Combobox, Notebook
from matplotlib import pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,
NavigationToolbar2Tk
from functions import *
from bees import Bees

def main():
    window = Tk()

    window.geometry("1000x800")

    window.title("Пчелинный алгоритм")

    fig = plt.figure(figsize=(14, 14))
    fig.add_subplot(projection='3d')

    canvas = FigureCanvasTkAgg(fig, master=window)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tkinter.LEFT, fill=tkinter.BOTH)

    toolbar = NavigationToolbar2Tk(canvas, window)
    toolbar.update()
    canvas.get_tk_widget().pack(side=tkinter.LEFT, fill=tkinter.BOTH)

    tab_control = Notebook(window)

    def draw_lab_5():
        fig.clf()

        iter_number = int(txt_1_tab_5.get())
        scouts_number = int(txt_2_tab_5.get())
        elite = int(txt_3_tab_5.get())
        perspective = int(txt_4_tab_5.get())
        b_to_leet = int(txt_5_tab_5.get())
        b_to_persp = int(txt_6_tab_5.get())
        pos_x = int(txt_8_tab_5.get())
        pos_y = int(txt_9_tab_5.get())
        delay = txt_7_tab_5.get()

        if combo_tab_5.get() == "Химмельблау":
            func = himmelblau_2
            x, y, z = make_data_himmelblau(pos_x, pos_y)
        else:
            func = rosenbrock_2
```

```
            x, y, z = make_data_rosenbrock(pos_x, pos_y)

        ax = fig.add_subplot(projection='3d')
        ax.plot_surface(x, y, z, rstride=5, cstride=5, alpha=0.5)
        canvas.draw()

        bees_swarm = Bees(func, scouts_number, elite, perspective, b_to_leet,
b_to_persp, 1, pos_x, pos_y)

        for scout in bees_swarm.scouts:
            ax.scatter(scout[0], scout[1], scout[2], c="blue", s=1, marker="s")

        bees_swarm.research_reports()
        bees_swarm.selected_search(1)

        for worker in bees_swarm.workers:
            ax.scatter(worker[0], worker[1], worker[2], c="black", s=1,
marker="s")

        b = bees_swarm.get_best()
        ax.scatter(b[0], b[1], b[2], c="red")

        canvas.draw()
        window.update()

        fig.clf()
        ax = fig.add_subplot(projection='3d')
        ax.plot_surface(x, y, z, rstride=5, cstride=5, alpha=0.5)
        canvas.draw()

        for i in range(iter_number):

            bees_swarm.send_scouts()
            for scout in bees_swarm.scouts:
                ax.scatter(scout[0], scout[1], scout[2], c="blue", s=1,
marker="s")

            bees_swarm.research_reports()
            bees_swarm.selected_search(1 / (i + 1))

            for sec in bees_swarm.selected:
                rx, ry, rz = make_square(sec[0], sec[1], 1 / (i + 1), func)
                ax.plot(rx, ry, rz, label='parametric curve')
            canvas.draw()
            window.update()

            for worker in bees_swarm.workers:
                ax.scatter(worker[0], worker[1], worker[2], c="black", s=1,
marker="s")

            b = bees_swarm.get_best()
```

```
            ax.scatter(b[0], b[1], b[2], c="red")

            txt_tab_5.insert(INSERT,
                             f"{i + 1}) ({round(b[0], 8)})"
                             f" ({round(b[1], 8)}) = "
                             f" ({round(b[2], 8)})\n")

            canvas.draw()
            window.update()
            time.sleep(float(delay))

            fig.clf()
            ax = fig.add_subplot(projection='3d')
            ax.plot_surface(x, y, z, rstride=5, cstride=5, alpha=0.5)
            canvas.draw()

        for scout in bees_swarm.scouts:
            ax.scatter(scout[0], scout[1], scout[2], c="blue", s=1, marker="s")

        for worker in bees_swarm.workers:
            ax.scatter(worker[0], worker[1], worker[2], c="black", s=1,
marker="s")

        b = bees_swarm.get_best()
        ax.scatter(b[0], b[1], b[2], c="red")

        canvas.draw()
        window.update()

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

    def make_square(x, y, rad, func):
        r_1 = [x - rad, x - rad, x + rad, x + rad]  # x
        r_2 = [y - rad, y + rad, y + rad, y - rad]  # y
        r_3 = [func(r_1[0], r_2[0]), func(r_1[1], r_2[1]), func(r_1[2], r_2[2]),
func(r_1[3], r_2[3])]  # z

        r_1.append(r_1[0])
        r_2.append(r_2[0])
        r_3.append(r_3[0])

        return r_1, r_2, r_3

    def delete_lab_5():
        txt_tab_5.delete(1.0, END)

    tab_5 = Frame(tab_control)
    tab_control.add(tab_5, text="ЛР5")

    main_f_tab_5 = LabelFrame(tab_5, text="Параметры")
    left_f_tab_5 = Frame(main_f_tab_5)
```

```
    right_f_tab_5 = Frame(main_f_tab_5)
    txt_f_tab_5 = LabelFrame(tab_5, text="Выполнение и результаты")

    lbl_5_tab_5 = Label(tab_5, text="Пчелиный алгоритм")
    lbl_1_tab_5 = Label(left_f_tab_5, text="Количество итераций")
    lbl_2_tab_5 = Label(left_f_tab_5, text="Количество разведчиков")
    lbl_3_tab_5 = Label(left_f_tab_5, text="Элитных участков")
    lbl_4_tab_5 = Label(left_f_tab_5, text="Задержка в секундах")
    lbl_6_tab_5 = Label(left_f_tab_5, text="Перспективных участков")
    lbl_7_tab_5 = Label(left_f_tab_5, text="Выбор функции")
    lbl_8_tab_5 = Label(left_f_tab_5, text="Рабочих на элитных участках")
    lbl_9_tab_5 = Label(left_f_tab_5, text="Рабочих на перспективных участках")

    lbl_10_tab_5 = Label(left_f_tab_5, text="X")
    lbl_11_tab_5 = Label(left_f_tab_5, text="Y")

    txt_1_tab_5 = Entry(right_f_tab_5)
    txt_1_tab_5.insert(0, "100")
    txt_2_tab_5 = Entry(right_f_tab_5)
    txt_2_tab_5.insert(0,"20")
    txt_3_tab_5 = Entry(right_f_tab_5)
    txt_3_tab_5.insert(0,"1")
    txt_4_tab_5 = Entry(right_f_tab_5)
    txt_4_tab_5.insert(0,"3")
    txt_5_tab_5 = Entry(right_f_tab_5)
    txt_5_tab_5.insert(0,"20")
    txt_6_tab_5 = Entry(right_f_tab_5)
    txt_6_tab_5.insert(0,"10")
    txt_7_tab_5 = Entry(right_f_tab_5)
    txt_7_tab_5.insert(0,"0.03")

    txt_8_tab_5 = Entry(right_f_tab_5)
    txt_8_tab_5.insert(0, "12")
    txt_9_tab_5 = Entry(right_f_tab_5)
    txt_9_tab_5.insert(0, "12")

    combo_tab_5 = Combobox(right_f_tab_5)
    combo_tab_5['values'] = ("Химмельблау", "Розенброка")
    combo_tab_5.set("Химмельблау")

    txt_tab_5 = scrolledtext.ScrolledText(txt_f_tab_5)
    btn_del_tab_5 = Button(tab_5, text="Очистить",
background="red",command=delete_lab_5)
    btn_tab_5 = Button(tab_5, text="Выполнить", foreground="black",
background="#08fc30", command=draw_lab_5)

    lbl_5_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    main_f_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH, expand=True)
    left_f_tab_5.pack(side=LEFT, fill=BOTH, expand=True)
    right_f_tab_5.pack(side=RIGHT, fill=BOTH, expand=True)
```

```
    lbl_1_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    lbl_2_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    lbl_3_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    lbl_6_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    lbl_8_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    lbl_9_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    lbl_4_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

    lbl_10_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    lbl_11_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

    lbl_7_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

    txt_1_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    txt_2_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    txt_3_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    txt_4_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    txt_5_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    txt_6_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    txt_7_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

    txt_8_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)
    txt_9_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

    combo_tab_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

    txt_tab_5.pack(padx=5, pady=5, fill=BOTH, expand=True)

    btn_tab_5.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)
    txt_f_tab_5.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)
    btn_del_tab_5.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

    tab_control.pack(side=RIGHT, fill=BOTH, expand=True)
    window.mainloop()

if __name__ == '__main__':
    main()
```