МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по лабораторной работе №6 по курсу**

**«МЕТОДЫ ПОИСКОВОЙ ОПТИМИЗАЦИИ»**

Работу выполнили

Студенты 36 группы

Агаджанян А.С.

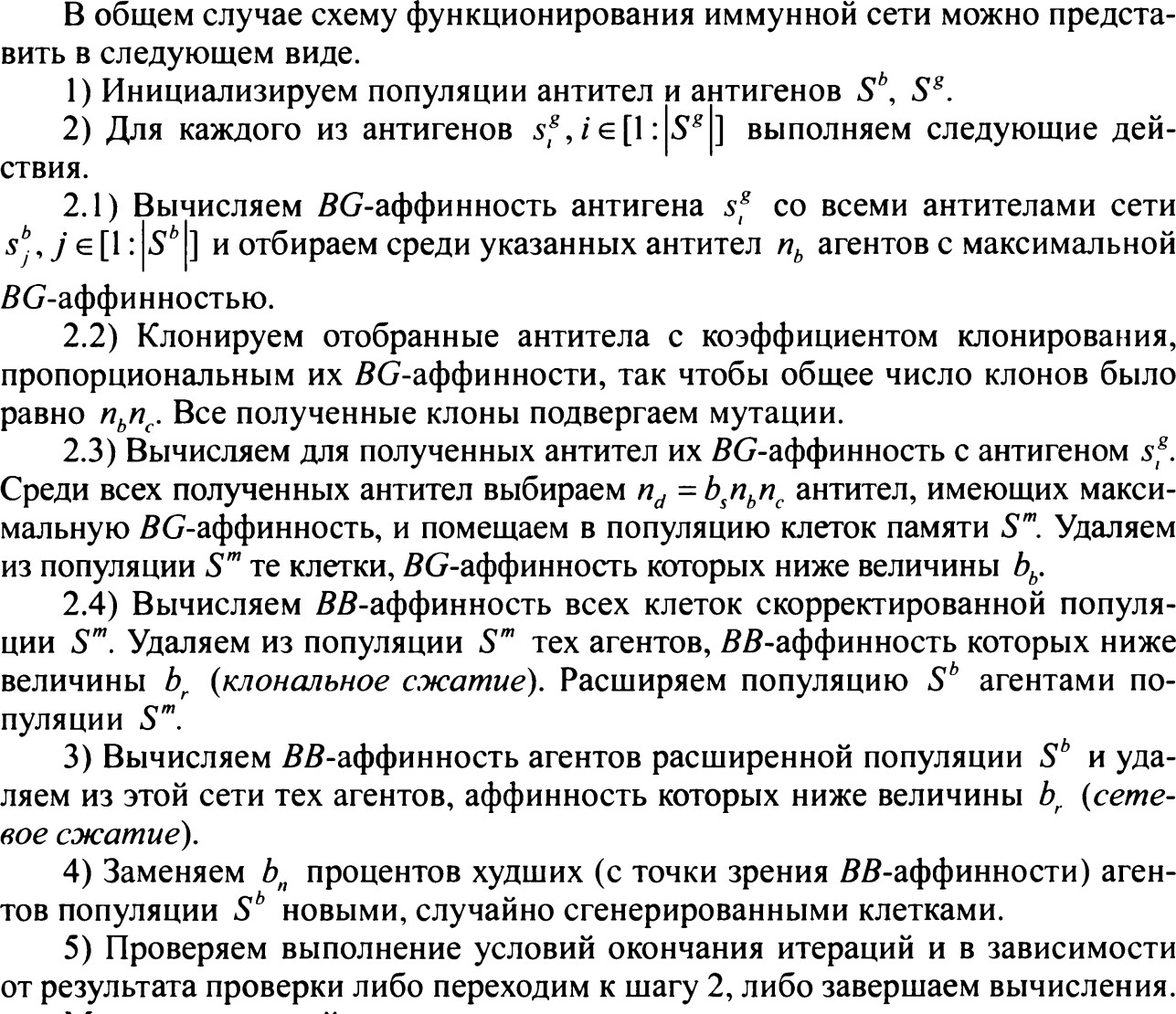
Гаранина Л.В.

Преподаватель:

Нигодин Е.А.

Краснодар 2025

**Цель работы:** разработать алгоритм оптимизации функции Розенброка искусственной иммунной сетью.

**Шаги алгоритма**

**Особенности реализации генетического алгоритма.**

Для создания программы использовался язык программирования Python. Для графической визуализации были подключены графические фреймворки Tkinter и Matplotlib.

Интерфейс программы имеет вид:

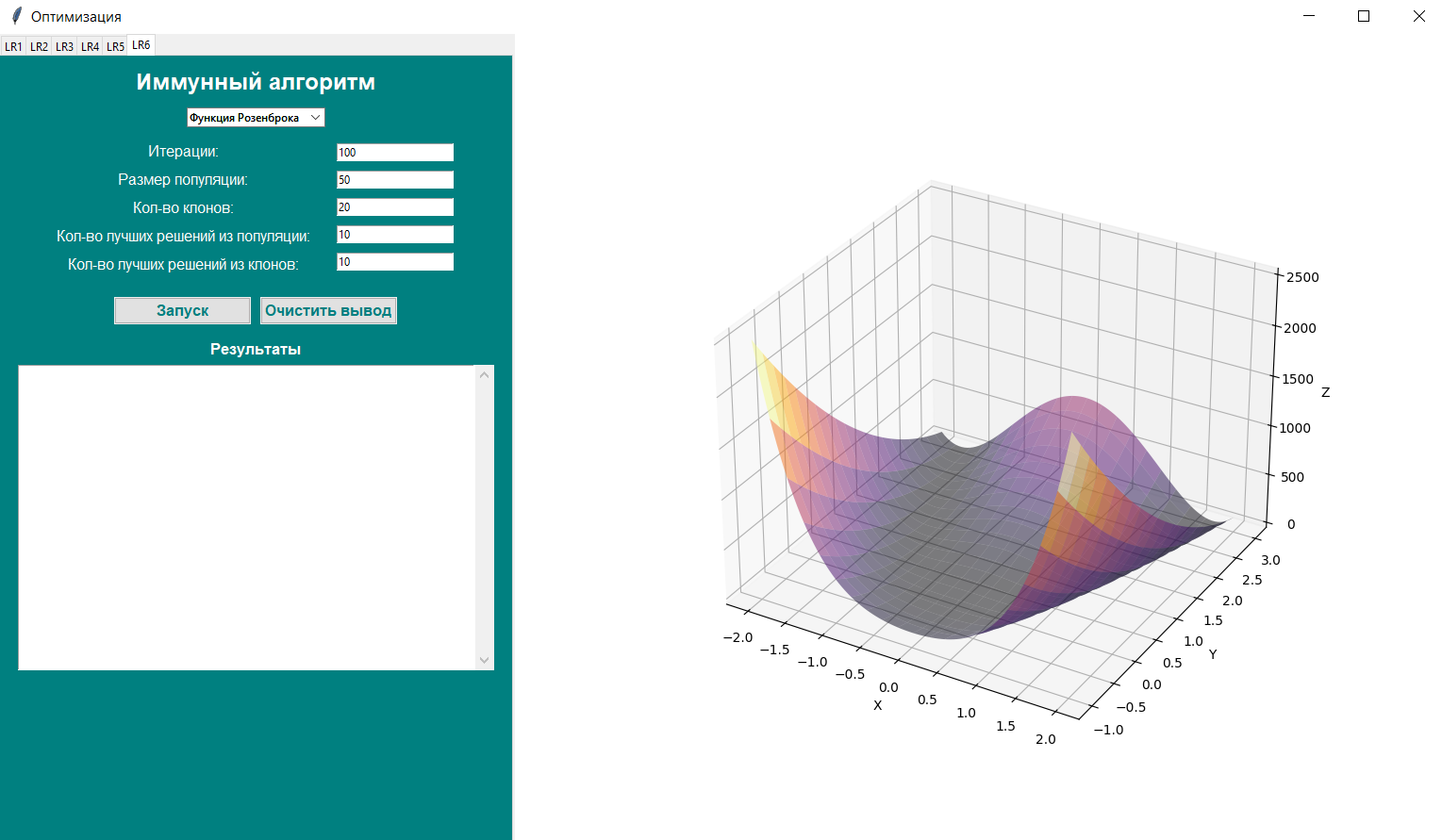


Рисунок 1 – Главное окно программы

В данном окне можно выбрать необходимую вкладку соответствующей лабораторной работы, в данном случае «6», ввести количество итераций, размер популяции, количество клонов, количество лучших решений из популяции и количество лучших решений из клонов. В поле «Результаты» динамически выводятся результаты работы алгоритма в виде шагов, представленных координатами и значениями оптимизируемой функции в этих координатах.

На рисунке 2 показана основная панель.

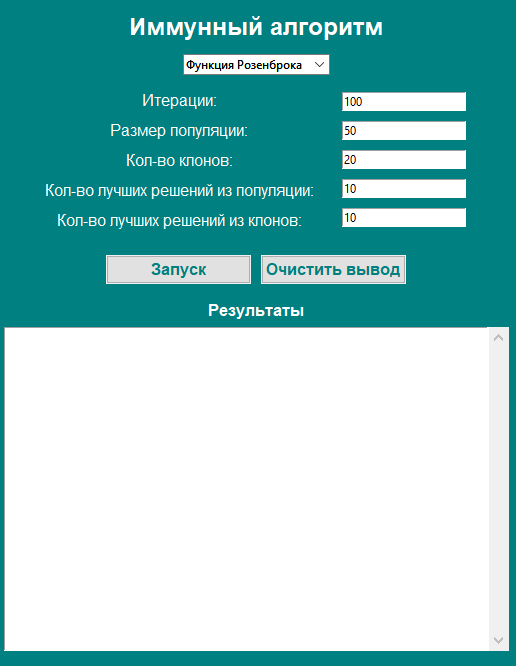


Рисунок 2 – Основная панель

Для запуска алгоритма необходимо нажать кнопку «Выполнить», при этом поле «Результаты» начнёт динамически заполняться точками, а на отображаемой функции можно наглядно увидеть функционирование алгоритма - искомые точки, которые также отображаются динамически как показано на рисунке 3. Результирующая точка выделена красным цветом.

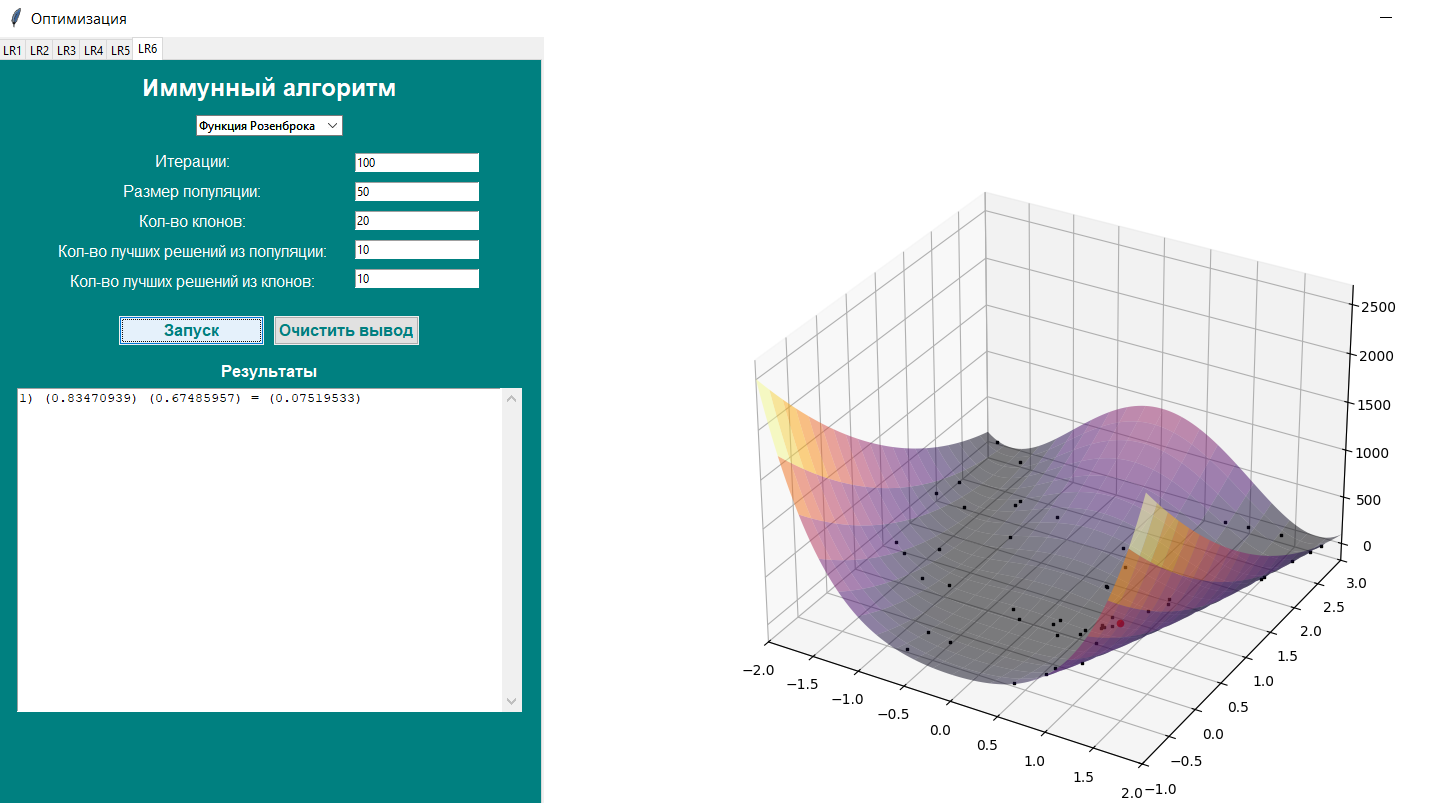


Рисунок 3 – Результат работы программы.

**Вывод:** в ходе работы изучили иреализовали алгоритм искусственной иммунной сети для нахождения глобального минимума функции Розенброка.

**Листинг программы**

Файл immune.py

import random

from operator import itemgetter

def init\_population(func, agents\_numb, bounds):

population = [[random.uniform(bounds[0][0], bounds[0][1]),

random.uniform(bounds[1][0], bounds[1][1])] for \_ in range(agents\_numb)]

evaluated = [[x, y, func([x, y])] for x, y in population]

return evaluated

def immune\_step(agents, func, best\_numb, clon\_numb, best\_clon\_numb, coef, bounds):

best = sorted(agents, key=itemgetter(2))[:best\_numb]

clones = []

for agent in best:

for \_ in range(clon\_numb):

x = agent[0] + coef \* random.uniform(-0.5, 0.5)

y = agent[1] + coef \* random.uniform(-0.5, 0.5)

# Ограничиваем значения в пределах границ

x = max(bounds[0][0], min(bounds[0][1], x))

y = max(bounds[1][0], min(bounds[1][1], y))

z = func([x, y])

clones.append([x, y, z])

top\_clones = sorted(clones, key=itemgetter(2))[:best\_clon\_numb]

new\_population = agents + top\_clones

new\_population = sorted(new\_population, key=itemgetter(2))[:len(agents)]

return new\_population

def get\_best(agents):

return min(agents, key=itemgetter(2))

Файл main.py

def draw\_lab\_6():

elev = None

azim = None

fig.clf()

txt\_f\_tab\_6.delete("1.0", tk.END)

pop\_number = int(size6.get())

iter\_number = int(it.get())

clon = int(clones.get())

best\_clon = int(resClones6.get())

best\_pop = int(res6.get())

selected\_function = function\_im.get()

# Определяем функцию и границы для каждой функции

if selected\_function == "Функция Химмельблау":

func = himmelblau

x, y, z = make\_data\_himmelblau()

bounds = [(-5, 5), (-5, 5)] # Границы для Химмельблау

elif selected\_function == "Функция Растригина":

func = rastrigin

x, y, z = make\_data\_rastrigin()

bounds = [(-5.12, 5.12), (-5.12, 5.12)] # Границы для Растригина

else: # По умолчанию Розенброка

func = rosenbrock

x, y, z = make\_data\_rosenbrock()

bounds = [(-2, 2), (-1, 3)] # Границы для Розенброка

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

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

ax.set\_xlim(bounds[0][0], bounds[0][1])

ax.set\_ylim(bounds[1][0], bounds[1][1])

canvas.draw()

# Инициализация популяции с учетом границ

agents = init\_population(func, pop\_number, bounds)

for ag in agents:

ax.scatter(ag[0], ag[1], ag[2], c="black", s=1, marker="s")

b = get\_best(agents)

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, cmap="inferno")

ax.set\_xlim(bounds[0][0], bounds[0][1])

ax.set\_ylim(bounds[1][0], bounds[1][1])

canvas.draw()

for i in range(iter\_number):

agents = immune\_step(agents, func, best\_pop, clon, best\_clon, 1 / (i + 1), bounds)

for ag in agents:

ax.scatter(ag[0], ag[1], ag[2], c="black", s=1, marker="s")

b = get\_best(agents)

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

txt\_f\_tab\_6.insert(tk.END,

f"{i + 1}) ({round(b[0], 8)}) "

f"({round(b[1], 8)}) = "

f"({round(b[2], 8)})\n")

txt\_f\_tab\_6.yview(1)

canvas.draw()

window.update()

time.sleep(0.0001) # Фиксированная задержка

elev = ax.elev

azim = ax.azim

fig.clf()

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

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

ax.set\_xlim(bounds[0][0], bounds[0][1])

ax.set\_ylim(bounds[1][0], bounds[1][1])

canvas.draw()

if elev is not None and azim is not None:

ax.view\_init(elev=elev, azim=azim)

for ag in agents:

ax.scatter(ag[0], ag[1], ag[2], c="black", s=1, marker="s")

b = get\_best(agents)

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

canvas.draw()

window.update()

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

tab\_control = Notebook(menu\_frame)

window = tk.Tk()

width = window.winfo\_screenwidth()

height = window.winfo\_screenheight()

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

window.title("Оптимизация")

menu\_frame = tk.Frame(window, width=300, bg="#008080")

menu\_frame.pack(side=tk.LEFT, fill=tk.Y)

# Для графика

plot\_frame = tk.Frame(window, bg="#008080")

plot\_frame.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

# График

fig = plt.figure(figsize=(10, 10))

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

canvas = FigureCanvasTkAgg(fig, master=plot\_frame)

canvas.draw()

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Панель инструментов для графика

toolbar = NavigationToolbar2Tk(canvas, plot\_frame)

toolbar.update()

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

tab6 = tk.Frame(tab\_control, bg="#008080")

tab\_control.add(tab6, text="LR6")

label\_title5 = tk.Label(tab6, text="Иммунный алгоритм", font=("Arial", 18, "bold"), fg="#FFFFFF", bg="#008080")

label\_title5.pack(pady=10)

function\_im = ttk.Combobox(tab6, values=["Функция Розенброка", "Функция Химмельблау", "Функция Растригина"])

function\_im.set("Функция Розенброка")

function\_im.pack()

function\_im.bind("<<ComboboxSelected>>", update\_plot)

input\_frame6 = tk.Frame(tab6, bg="#008080")

input\_frame6.pack(pady=10)

# Фрейм слева для лейблов

labels\_frame6 = tk.Frame(input\_frame6, bg="#008080")

labels\_frame6.pack(side=tk.LEFT, padx=10)

# Фрейм справа для полей ввода

entries\_frame6 = tk.Frame(input\_frame6, bg="#008080")

entries\_frame6.pack(side=tk.LEFT, padx=10)

label\_iterations6 = tk.Label(labels\_frame6, text="Итерации:", bg="#008080", font=("Arial", 12), fg="#FFFFFF")

label\_iterations6.pack(pady=2.5)

it = tk.Entry(entries\_frame6)

it.insert(0, "100")

it.pack(padx=5, pady=5)

label\_62 = tk.Label(labels\_frame6, text="Размер популяции:", bg="#008080", font=("Arial", 12), fg="#FFFFFF")

label\_62.pack(pady=2.5)

size6 = tk.Entry(entries\_frame6)

size6.insert(0, "50")

size6.pack(padx=5, pady=5)

label\_63 = tk.Label(labels\_frame6, text="Кол-во клонов:", bg="#008080", font=("Arial", 12), fg="#FFFFFF")

label\_63.pack(pady=2.5)

clones = tk.Entry(entries\_frame6)

clones.insert(0, "20")

clones.pack(padx=5, pady=5)

label\_64 = tk.Label(labels\_frame6, text="Кол-во лучших решений из популяции:", bg="#008080", font=("Arial", 12), fg="#FFFFFF")

label\_64.pack(pady=2.5)

res6 = tk.Entry(entries\_frame6)

res6.insert(0, "10")

res6.pack(padx=5, pady=5)

label\_65 = tk.Label(labels\_frame6, text="Кол-во лучших решений из клонов:", bg="#008080", font=("Arial", 12), fg="#FFFFFF")

label\_65.pack(pady=2.5)

resClones6 = tk.Entry(entries\_frame6)

resClones6.insert(0, "10")

resClones6.pack(padx=5, pady=5)

button\_im = tk.Frame(tab6, bg="#008080")

button\_im.pack(pady=10)

ttk.Button(button\_im, text="Запуск", style="Custom.TButton", command=draw\_lab\_6).pack(side="left", padx=5)

ttk.Button(button\_im, text="Очистить вывод", style="Custom.TButton", command=lambda: txt\_f\_tab\_6.delete(1.0, tk.END)).pack(side="left", padx=5)

ttk.Label(tab6, text="Результаты", font=("Helvetica", 12, "bold"), background="#008080", foreground="#FFFFFF").pack(pady=5)

txt\_f\_tab\_6 = scrolledtext.ScrolledText(tab6, width=60, height=20)

txt\_f\_tab\_6.pack()

plot\_surface(ax, "Функция Химмельблау")

canvas.draw()

window.mainloop()