МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ

ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ

Кафедра математичних методів системного аналізу

ЗВІТ

про виконання лабораторної роботи № 3

з дисципліни «Еволюційні методи оптимізації»

з теми «Дослідження генетичного алгоритму

оптимізації багатоекстремальних функцій у дійсному просторі»

Виконала:

Студентка 3 курсу ІПСА

групи КА-24

Спекторовська Лада

Варіант №35

Київ 2025

Мета: Дослідити ефективність алгоритму адаптивного спірального пошуку для задачі оптимізації на заданих багатоекстремальних функціях.

Програма реалізована власноруч мовою програмування Python. При виконанні було використано бібліотеки: NumPy, Pandas, Matplotlib.

Використані параметри та елементи методу:

- Розмір популяції: 100 (для тестових запусків 200).
- Кількість поколінь: 50.
- Кут θ: π/6, π/3 (π/4 для тестових запусків).
- Діапазон (rl, ru): (0.8, 1.1), (0.8,1.3), (0.95,1.1), (0.95,1.3), (0.9, 1.05) – для тестових запусків.
- параметр чутливості c1: 0.1, 1, 10.

Тестовий запуск:

- Розмір популяції: 200
- Кількість поколінь: 50
- Кут θ: π/4
- Діапазон (rl, ru): (0.9, 1.05)

Тестовий запуск для функції Drop-Wave:

Оскільки в отриманій популяції 200 точок, виведемо перші 10:

X	Y	Z
-0,00587	-0,03909	-0,94442
-0,00587	-0,03909	-0,94442
-0,00587	-0,03909	-0,94442
-0,00587	-0,03909	-0,94442
-0,00587	-0,03909	-0,94442
-0,00587	-0,03909	-0,94442
-0,00587	-0,03909	-0,94442
-0,00587	-0,03909	-0,94442
-0,00587	-0,03909	-0,94442
-0,00587	-0,03909	-0,94442

Таблиця №1. Результати тестового запуску для Drop-Wave (перші 10 точок).

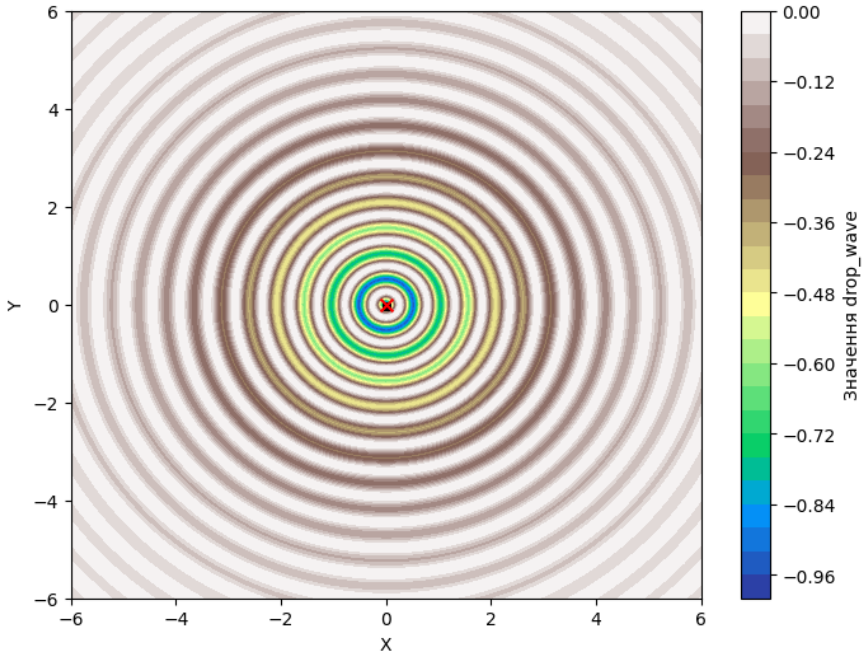

Рисунок №1. Результат тестового запуску для Drop-Wave.

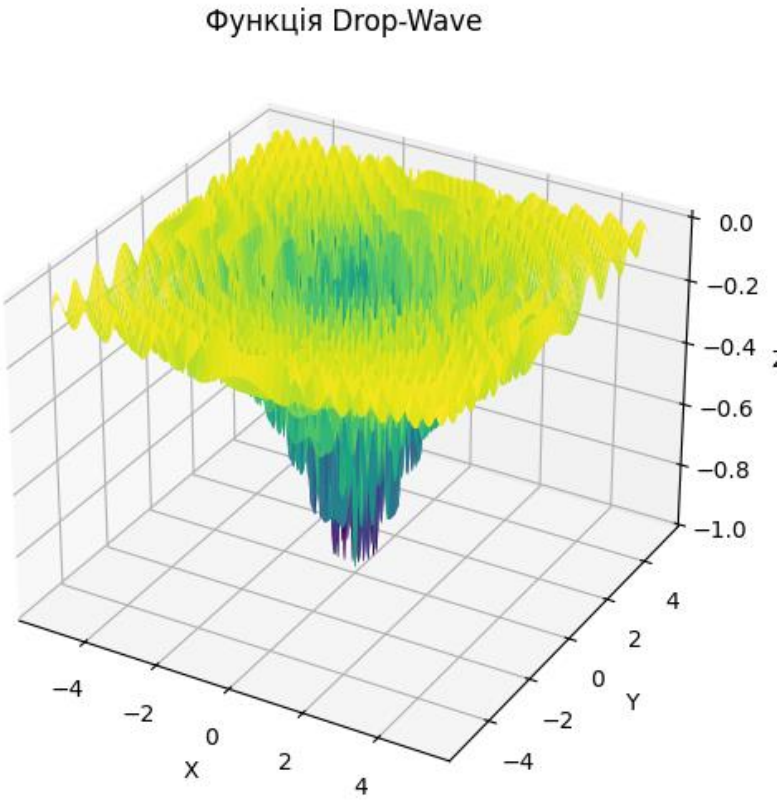

Рисунок №2. Функція Drop-Wave.

Тестовий запуск для функції Швефеля:

Оскільки в отриманій популяції 200 точок, виведемо перші 10:

X	Y	Z
429,1735	431,0843	21,38867
429,1735	431,0843	21,38867

429,1735	431,0843	21,38867
429,1735	431,0843	21,38867
429,1735	431,0843	21,38867
429,1735	431,0843	21,38867
429,1735	431,0843	21,38867
429,1735	431,0843	21,38867
429,1735	431,0843	21,38867
429,1735	431,0843	21,38867

Таблиця №2. Результати тестового запуску для Швефеля (перші 10 точок).

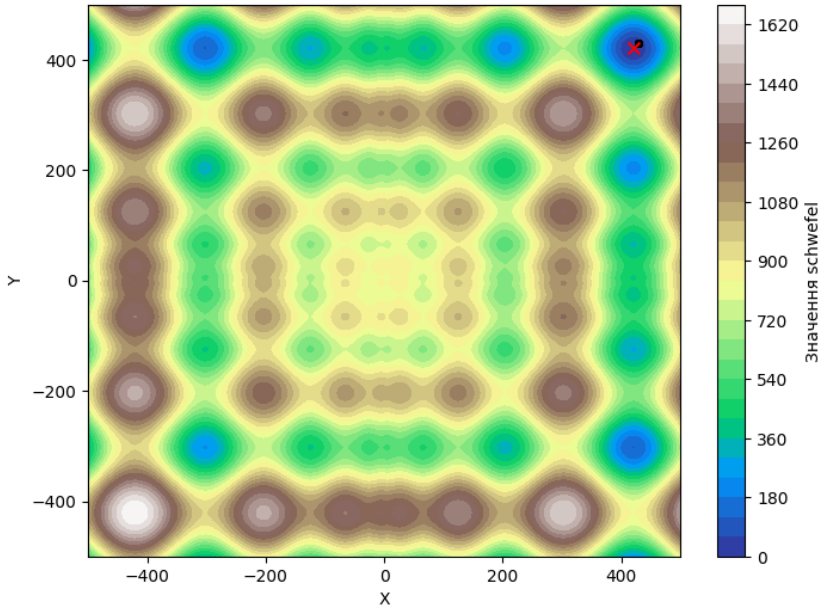

Рисунок №3. Результат тестового запуску для функції Швефеля.

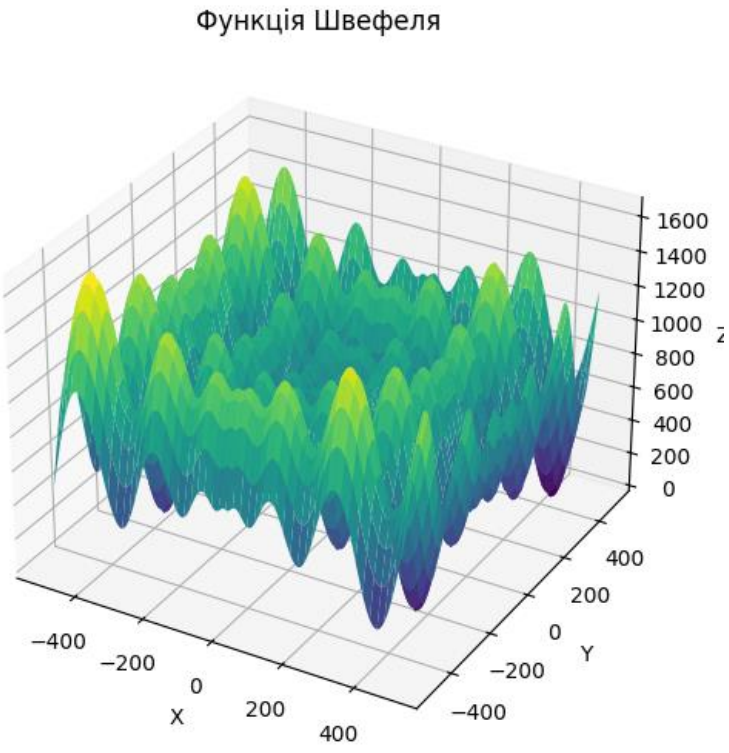

Рисунок №4. Функція Швефеля

Як видно з результатів, для обох функцій популяція збігається в одну точку. Це свідчить про ефективний пошук, але також може вказувати на ризик передчасної збіжності до локального мінімуму.

Експерименти:

Надалі проведемо 100 експериментів для кожної функції. Ефективність проведених досліджень буде вимірюватись у наступних критеріях:

- Здатність знайти глобальний мінімум (або стабільність). Обчислюємо скільки зі 100 випадків метод потрапляє в епсілон окіл потрібної точки. Для функції Швефеля епсілон взято 55 для функції Drop-Wave - 0.6.
- Точність визначення глобального мінімуму. Обчислюємо середню відстань від найкращої точки методу на 100 експериментах, тобто чим менше ця відстань тим краще вправляється метод.
- Кількість підрахунків функції.

Розглянемо результати:

Target function	Param_theta	Param_rl_ru	Param_c1	Stability	AvgDistance	Function counts
schwefel	0,523598776	[0.8, 1.1]	0,1	0,473920002	394,5882097	10100
schwefel	0,523598776	[0.8, 1.1]	1	0,484322002	383,0725233	10100
schwefel	0,523598776	[0.8, 1.1]	10	0,464916001	412,6660684	10100
schwefel	0,523598776	[0.8, 1.3]	0,1	0,474124002	394,1041307	10100
schwefel	0,523598776	[0.8, 1.3]	1	0,484521002	402,243142	10100
schwefel	0,523598776	[0.8, 1.3]	10	0,415107002	426,1796755	10100
schwefel	0,523598776	[0.95, 1.1]	0,1	0,606618001	296,5291008	10100
schwefel	0,523598776	[0.95, 1.1]	1	0,676321001	234,3065224	10100
schwefel	0,523598776	[0.95, 1.1]	10	0,586119002	305,4148362	10100
schwefel	0,523598776	[0.95, 1.3]	0,1	0,525823002	341,4847214	10100
schwefel	0,523598776	[0.95, 1.3]	1	0,646220001	269,8894526	10100
schwefel	0,523598776	[0.95, 1.3]	10	0,636210001	266,6325508	10100
schwefel	1,047197551	[0.8, 1.1]	0,1	0,415543005	429,1814548	10100
schwefel	1,047197551	[0.8, 1.1]	1	0,454139005	416,0095918	10100
schwefel	1,047197551	[0.8, 1.1]	10	0,495144004	372,9689966	10100
schwefel	1,047197551	[0.8, 1.3]	0,1	0,484044004	388,753813	10100
schwefel	1,047197551	[0.8, 1.3]	1	0,504935004	364,7593616	10100
schwefel	1,047197551	[0.8, 1.3]	10	0,434042003	432,0589677	10100
schwefel	1,047197551	[0.95, 1.1]	0,1	0,616330003	287,6860742	10100
schwefel	1,047197551	[0.95, 1.1]	1	0,524834003	328,1371839	10100
schwefel	1,047197551	[0.95, 1.1]	10	0,515724003	332,6910767	10100
schwefel	1,047197551	[0.95, 1.3]	0,1	0,586041004	295,58936	10100
schwefel	1,047197551	[0.95, 1.3]	1	0,565637004	298,9015683	10100
schwefel	1,047197551	[0.95, 1.3]	10	0,545125002	324,8163821	10100

Таблиця №3.1. Результати експериментів.

drop_wave	0,523598776	[0.8, 1.1]	0,1	0,96921412	0,521084055	10100
drop_wave	0,523598776	[0.8, 1.1]	1	0,64491515	0,703330996	10100
drop_wave	0,523598776	[0.8, 1.1]	10	0,7663152	0,66632837	10100
drop_wave	0,523598776	[0.8, 1.3]	0,1	0,89742122	0,554693244	10100
drop_wave	0,523598776	[0.8, 1.3]	1	0,61491722	0,73262805	10100
drop_wave	0,523598776	[0.8, 1.3]	10	0,62411815	0,74091522	10100
drop_wave	0,523598776	[0.95, 1.1]	0,1	0,67501717	0,675308261	10100
drop_wave	0,523598776	[0.95, 1.1]	1	0,71552015	0,682155682	10100
drop_wave	0,523598776	[0.95, 1.1]	10	0,76631825	0,651804933	10100
drop_wave	0,523598776	[0.95, 1.3]	0,1	0,62371724	0,749611047	10100
drop_wave	0,523598776	[0.95, 1.3]	1	0,63411418	0,711543965	10100
drop_wave	0,523598776	[0.95, 1.3]	10	0,69542411	0,707810333	10100
drop_wave	1,047197551	[0.8, 1.1]	0,1	0,98903132	0,401067701	10100
drop_wave	1,047197551	[0.8, 1.1]	1	0,73552116	0,660977668	10100
drop_wave	1,047197551	[0.8, 1.1]	10	0,78681321	0,642593973	10100
drop_wave	1,047197551	[0.8, 1.3]	0,1	0,96851118	0,518437146	10100
drop_wave	1,047197551	[0.8, 1.3]	1	0,52442626	0,772256844	10100
drop_wave	1,047197551	[0.8, 1.3]	10	0,70601428	0,677740364	10100
drop_wave	1,047197551	[0.95, 1.1]	0,1	0,74562428	0,653114458	10100
drop_wave	1,047197551	[0.95, 1.1]	1	0,78571925	0,628428613	10100
drop_wave	1,047197551	[0.95, 1.1]	10	0,79601522	0,598118561	10100
drop_wave	1,047197551	[0.95, 1.3]	0,1	0,5453202	0,773688576	10100
drop_wave	1,047197551	[0.95, 1.3]	1	0,73562119	0,655917622	10100
drop_wave	1,047197551	[0.95, 1.3]	10	0,705442	0,674923626	10100

Таблиця №3.2. Результати експериментів. Частина 2.

Візуалізуємо зміни стабільності та точності для обох функцій:

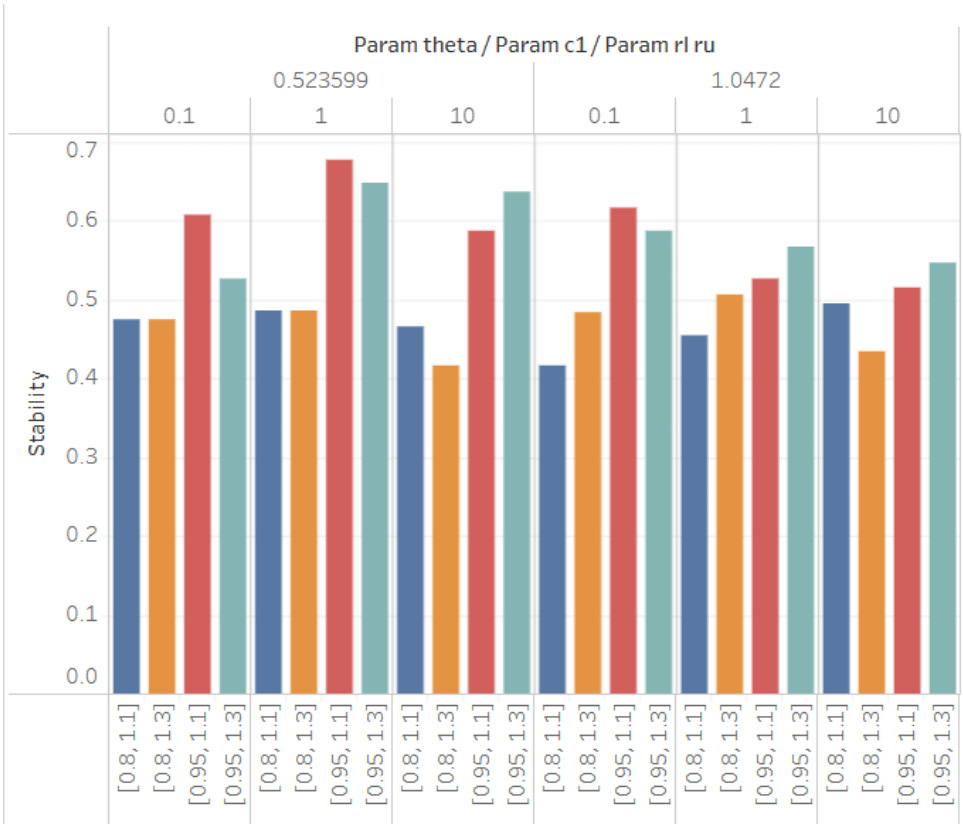

Рисунок №5. Функція Швефеля. Візуалізація змін стабільності.

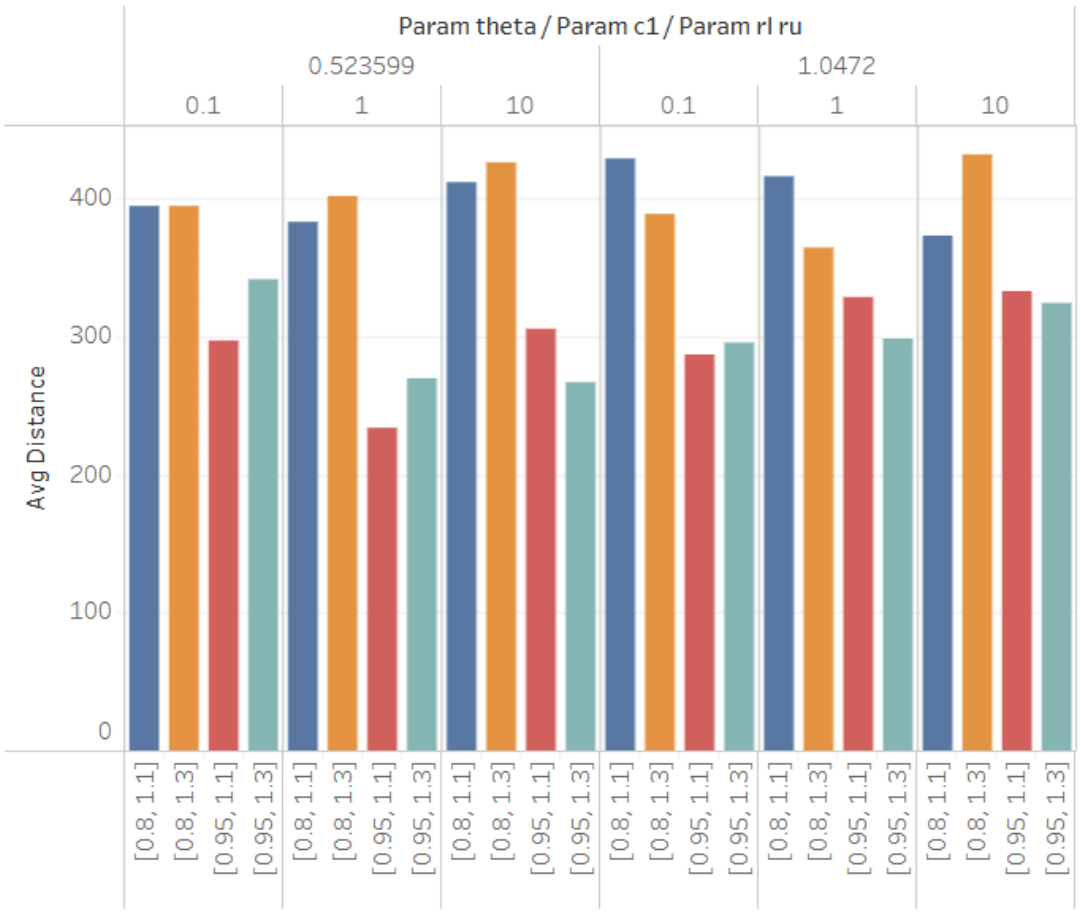

Рисунок №6. Функція Швефеля. Візуалізація змін точності.

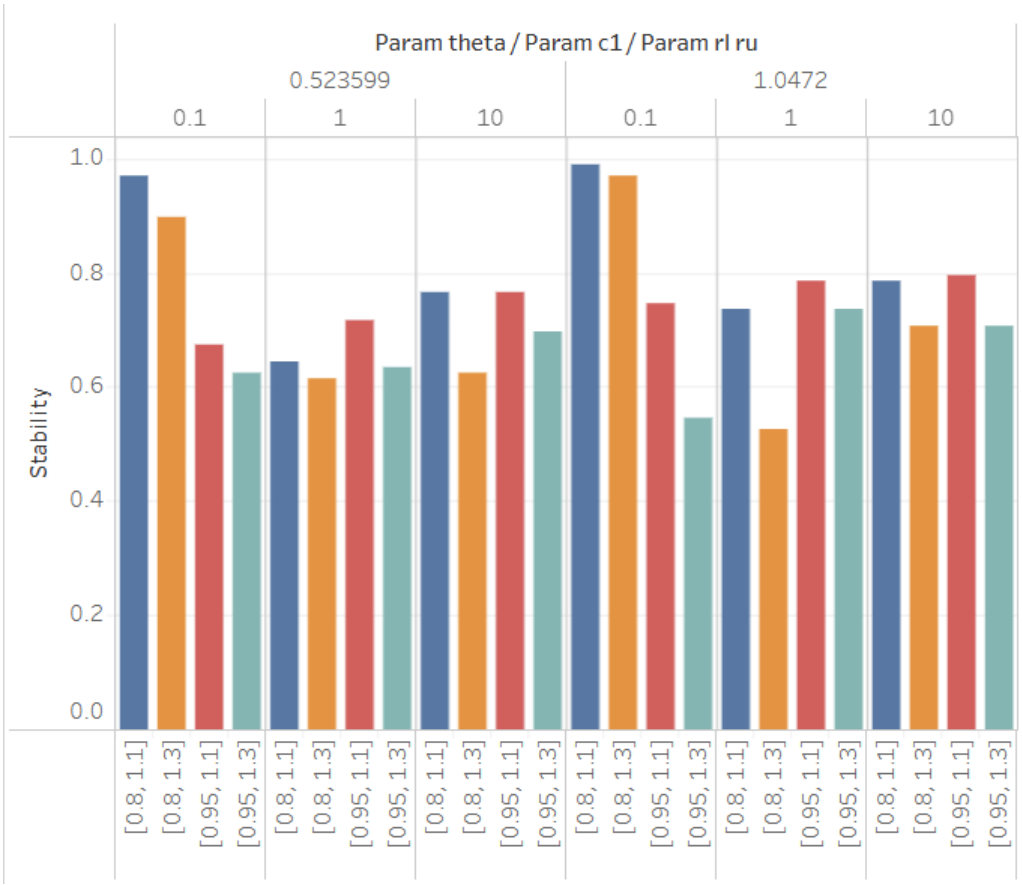

Рисунок №7. Функція Drop-Wave. Візуалізація змін стабільності.

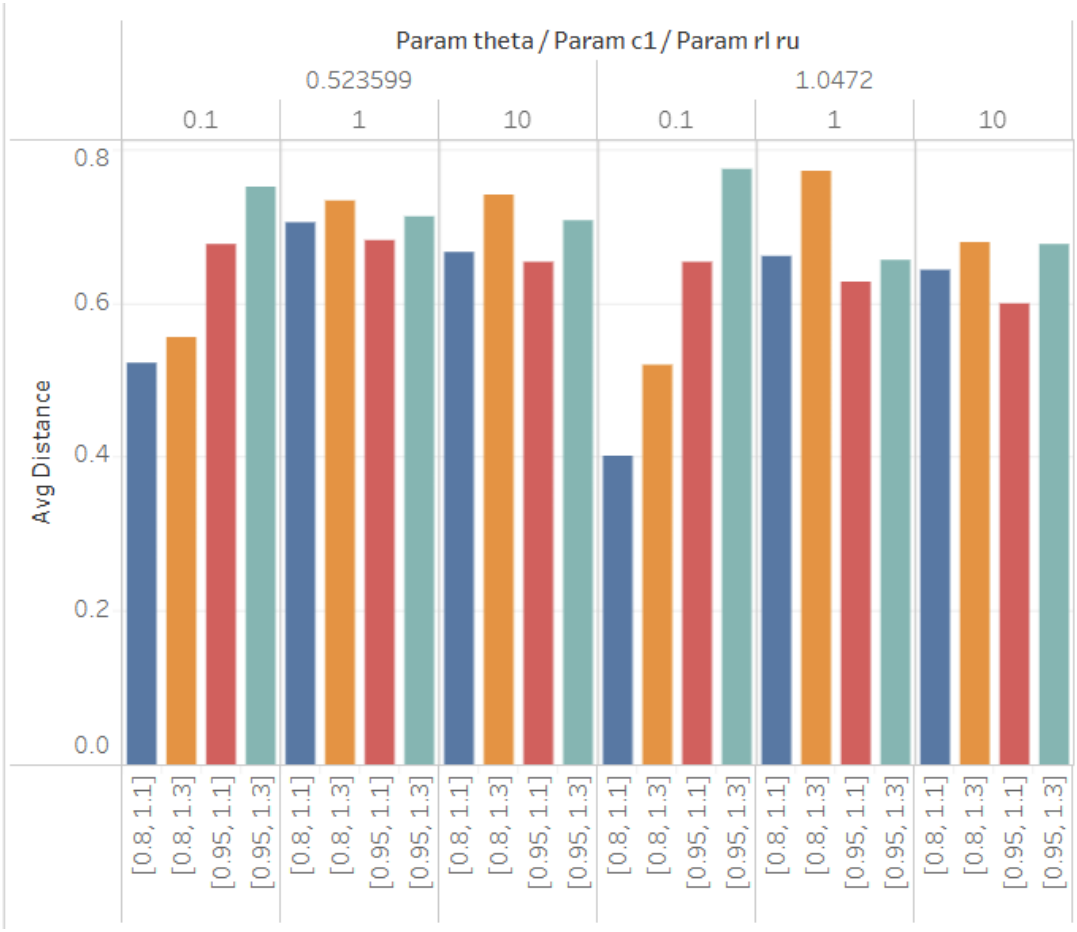

Рисунок №8. Функція Drop-Wave. Візуалізація змін точності.

Висновок:

У ході лабораторної роботи досліджено ефективність адаптивного спірального пошуку для функцій Schwefel і Drop-Wave. Для оцінки якості використовувались два критерії — точність (середня відстань до глобального мінімуму, чим менше — тим краще) та стабільність (частка потраплянь в ε-окол глобального мінімуму).

Варто зауважити, що на тестових запусках для Drop-Wave та Швефеля, при значеннях епсілон 0.55 та 50 відповідно, середня відстань до мінімума та стабільність мали такі значення: 0.501155150139744 і 0.825; 139.3065396785108 і 0.805 відповідно. І також було помітно збіжність в одну точку. Як можна бачити, стабільність доволі висока, хоча тут може бути присутній і фактор випадковості.

У результаті серії експериментів змінювалися параметри методу:

- кут обертання θ ($\pi/6 \approx 0.5236$ та $\pi/3 \approx 1.0472$),
- межі адаптації,
- параметр чутливості c1.

У ході експериментів досліджено вплив різних параметрів алгоритму:

- Для функції Schwefel найкраща точність та стабільність досягалися при (0.95,1.1) та (0.95, 1.3).
- Для функції Drop-Wave найкраща точність та стабільність досягалися при (0.8, 1.3) та (0.8, 1.1)
- Менший кут $\theta = \pi/6$ незначно покращував стабільність для обох функцій, тоді як більший $\theta = \pi/3$ позитивно впливав на точність.
- Значення c1 = 1 дало найкращий баланс між точністю і стабільністю. Значення c1 = 0.1 теж позитивно впливало на результати.

Під час виконання лабораторної роботи було видно, що алгоритм адаптивного спірального пошуку працює набагато менш стабільно для функції Швефеля, що обумовлено тим, що це доволі складна функція з безліччю локальними мінімумами, а алгоритм якраз має доволі високу схильність передчасної збіжності. На мою думку, алгоритм стохастичного спірального пошуку може покращити результати та виправити цю проблему.

Отже, алгоритм добре працює, але є ризик передчасного зближення, особливо для функцій з багатьма локальними мінімумами, таких як функція Швефеля.

Додаток А

```
import numpy as np

import random

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

import random

import copy

import pandas as pd


# --- Функції ---

def schwefel(x, y):

    return 837.9658 - (x * np.sin(np.sqrt(abs(x))) + y * np.sin(np.sqrt(abs(y))))


def drop_wave(x, y):

    numerator = 1 + np.cos(12 * np.sqrt(x**2 + y**2))
```

```
    denominator = 0.5 * (x**2 + y**2) + 2
    return -numerator / denominator
# --- Глобальні змінні ---
min_schwefel = (420.9687, 420.9687, 0)
min_drop_wave = (0, 0, -1)
bounds_schwefel = [-500, 500]
bounds_drop_wave = [-5.12, 5.12]
temp = [0,0,0]
def euclidean_distance(p1, p2):
    return np.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)

def initialize_population(pop_size, bounds):
    population=[]
    for i in range(pop_size):
            temp[0] = random.uniform(bounds[0], bounds[1])
            temp[1] = random.uniform(bounds[0], bounds[1])
            population.append(copy.deepcopy(temp))
    return population

n_f = 0
def fitness(individual, func_name):
    global n_f
    n_f += 1
    if func_name == 'schwefel':
        return schwefel(individual[0], individual[1])
    elif func_name == 'drop_wave':
        return drop_wave(individual[0], individual[1])
```

```
def adaptive_spiral_search(func_name, bounds, pop_size=30, max_gen=50, r_l=0.9, r_u=1.05, theta=np.pi/4, c1=1.0):
    global n_f
    n_f = 0

    population = initialize_population(pop_size, bounds)
    for ind in population:
        ind[2] = fitness(ind, func_name)

    population.sort(key=lambda ind: ind[2])
    best = population[0][:2]

    for gen in range(max_gen):
        new_population = []

        f_min = min([ind[2] for ind in population])

        for ind in population:
            dx = ind[0] - best[0]
            dy = ind[1] - best[1]

            x_rot = dx * np.cos(theta) - dy * np.sin(theta)
            y_rot = dx * np.sin(theta) + dy * np.cos(theta)

            x_new = best[0] + x_rot
            y_new = best[1] + y_rot

            x_new = np.clip(x_new, bounds[0], bounds[1])
            y_new = np.clip(y_new, bounds[0], bounds[1])
```

```
            f_new = fitness([x_new, y_new], func_name)

            diff = f_new - f_min + 1e-8
            r = r_u + (r_l - r_u) / (1 + c1 / diff)

            dx = x_new - best[0]
            dy = y_new - best[1]
            x_final = best[0] + r * dx
            y_final = best[1] + r * dy

            x_final = np.clip(x_final, bounds[0], bounds[1])
            y_final = np.clip(y_final, bounds[0], bounds[1])
            f_final = fitness([x_final, y_final], func_name)

            new_population.append([x_final, y_final, f_final])

        combined = population + new_population
        combined.sort(key=lambda ind: ind[2])
        population = combined[:pop_size]
        best = population[0][:2]

    return population, n_f

def plot_function(func, bounds, title):
    x = np.linspace(bounds[0], bounds[1], 400)
    y = np.linspace(bounds[0], bounds[1], 400)
    X, Y = np.meshgrid(x, y)
```

```
    Z = func(X, Y)


    fig = plt.figure(figsize=(8, 6))
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(X, Y, Z, cmap='viridis')
    ax.set_title(title)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    plt.show()
accuracy=0
stability=0
for i in range(0,200):
    print(i)
    test,serv=adaptive_spiral_search("drop_wave", bounds_drop_wave, pop_size=200)
    dist=euclidean_distance(test[0],min_drop_wave)
    accuracy+=dist
    if dist<=0.55:
        stability+=1
accuracy=accuracy/200
stability=stability/200
print(accuracy)
print(stability)
test,serv=adaptive_spiral_search("drop_wave", bounds_drop_wave, pop_size=200)
print(test)
print(serv)
drop_wave_test_x=[]
drop_wave_test_y=[]
```

```
drop_wave_test_z=[]
for i in test:
    drop_wave_test_x.append(i[0])
    drop_wave_test_y.append(i[1])
    drop_wave_test_z.append(i[2])
temp_exp={
    "X": drop_wave_test_x,
    "Y": drop_wave_test_y,
    "Z": drop_wave_test_z
}
export_test=pd.DataFrame(temp_exp)
export_test.to_excel("EMO3_output1.xlsx", index=False)
x = np.linspace(-6, 6, 500)  # Діапазон X
y = np.linspace(-6, 6, 500)  # Діапазон Y
X, Y = np.meshgrid(x, y)     # Створюємо координатну сітку
Z = drop_wave(X, Y)              # Обчислюємо значення функції

# Будуємо графік
plt.figure(figsize=(8, 6))
contour = plt.contourf(X, Y, Z, levels=30, cmap='terrain')  # 'terrain' - кольори, схожі на фізичну карту
plt.colorbar(contour, label="Значення drop_wave")  # Додаємо кольорову шкалу
plt.xlabel("X")
plt.ylabel("Y")
for i in test:
    plt.scatter(i[0],i[1], color="black", marker=".", s=50)
plt.scatter(0, 0, color="Red", marker="x", s=50)
# Відображення функцій
```

```
plot_function(drop_wave, bounds_drop_wave, "Функція Drop-Wave")
accuracy=0
stability=0
for i in range(0,200):
    print(i)
    test,serv=adaptive_spiral_search("schwefel", bounds_schwefel, pop_size=200)
    dist=euclidean_distance(test[0],min_schwefel)
    accuracy+=dist
    if dist<=50:
        stability+=1
accuracy=accuracy/200
stability=stability/200
print(accuracy)
print(stability)

test,serv=adaptive_spiral_search("schwefel", bounds_schwefel, pop_size=200)
print(test)
print(serv)
schwefel_test_x=[]
schwefel_test_y=[]
schwefel_test_z=[]
for i in test:
    schwefel_test_x.append(i[0])
    schwefel_test_y.append(i[1])
    schwefel_test_z.append(i[2])
temp_exp={
    "X": schwefel_test_x,
    "Y": schwefel_test_y,
```

```
    "Z": schwefel_test_z
}
export_test=pd.DataFrame(temp_exp)
export_test.to_excel("EMO3_output2.xlsx", index=False)
x = np.linspace(-500, 500, 1000)  # Діапазон X
y = np.linspace(-500, 500, 1000)  # Діапазон Y
X, Y = np.meshgrid(x, y)     # Створюємо координатну сітку
Z = schwefel(X, Y)              # Обчислюємо значення функції

# Будуємо графік
plt.figure(figsize=(8, 6))
contour = plt.contourf(X, Y, Z, levels=30, cmap='terrain')  # 'terrain' - кольори, схожі на фізичну карту
plt.colorbar(contour, label="Значення schwefel")  # Додаємо кольорову шкалу
plt.xlabel("X")
plt.ylabel("Y")
for i in test:
    plt.scatter(i[0],i[1], color="black", marker=".", s=50)
plt.scatter(420.9687, 420.9687, color="Red", marker="x", s=50)
plot_function(schwefel, bounds_schwefel, "Функція Швефеля")
functions=["schwefel", "drop_wave"]
theta_s=[np.pi/6, np.pi/3]
rlu_s=[[0.8, 1.1],[0.8,1.3],[0.95,1.1],[0.95,1.3]]
c1_s=[0.1, 1.0, 10.0]

def grid_test():
    temp_counter=[]
    temp_pop=[]
    temp_funcs=[]
```

```
    for func in functions:
        for theta_t in theta_s:
            for rlu in rlu_s:
                for c1_t in c1_s:
                    if func=="schwefel":
                        bounds=bounds_schwefel
                    else:
                        bounds=bounds_drop_wave
                    x, y=adaptive_spiral_search(func, bounds, pop_size=100, r_l=rlu[0], r_u=rlu[1], theta=theta_t, c1=c1_t)
                    temp_pop.append(x)
                    temp_counter.append(y)
                    temp_funcs.append(func)
    return temp_pop, temp_counter, temp_funcs
adapt_stability=[]
adapt_accuracy=[]
adapt_functions=[]
adapt_counter=[]
adapt_rlu=[]
adapt_theta=[]
adapt_c1=[]

for func in functions:
        for theta_t in theta_s:
            for rlu in rlu_s:
                for c1_t in c1_s:
                    adapt_functions.append(func)
                    adapt_theta.append(theta_t)
                    adapt_rlu.append(rlu)
```

```
                adapt_c1.append(c1_t)
                adapt_stability.append(np.float64(0))
                adapt_accuracy.append(np.float64(0))
                adapt_counter.append(np.float64(0))
dict={
    "Target function": adapt_functions,
    "Param_theta": adapt_theta,
    "Param_rl_ru": adapt_rlu,
    "Param_c1":adapt_c1,
    "Stability": adapt_stability,
    "AvgDistance":adapt_accuracy,
    "Function counts": adapt_counter
    }
results_df=pd.DataFrame(dict)
print(results_df)
dist=0
for i in range(100):
    pop,count,funcs =grid_test()
    print(i)
    for j in range(len(pop)):
        if funcs[j]=="schwefel":
            dist=euclidean_distance(pop[j][0],min_schwefel)
            #print(dist)
            if dist<=55:
                results_df.at[j,'Stability']+=1
                #print("a")
            results_df.at[j,'AvgDistance']=results_df.at[j,'AvgDistance']+dist
            results_df.at[j,'Function counts']=count[j]
```

```
            #print("b")
        if funcs[j]=="drop_wave":
            dist=euclidean_distance(pop[j][0],min_drop_wave)
            if dist<=0.6:
                results_df.at[j,'Stability']+=1
                #print("c")
            results_df.at[j,'AvgDistance']=results_df.at[j,'AvgDistance']+dist
            results_df.at[j,'Function counts']=count[j]
            #print("d")
for i in range(48):
    results_df.at[i,"AvgDistance"]=results_df.at[i,"AvgDistance"]/100
    results_df.at[i,"Stability"]=results_df.at[i,"Stability"]/100
print(results_df)
results_df.to_excel("EMO3_rezal.xlsx", index=False)
```