Отчёт по лабораторной работе №2 по фундаментальным концепциям искусственного интеллекта на тему: “Глобальная оптимизация и метаэвристические алгоритмы”

Выполнил студент группы М80-114СВ-24 Сипкин Владислав.

1. Функция Химмельблау:

а) Этот код включает:

Листинг 1 – Программа реализации тестирования данной функция Химмельблау на поиск оптимума 3 разными алгоритмами доступными в библиотеке pygmo

```
import pygmo as pg
import pandas as pd

# задание алгоритма тестирования функции
def test_func(problem, algorithm):
    pop = pg.population(problem, 20)    # создание популяции
    pop = algorithm.evolve(pop)     # эволюция популяции с помощью алгоритма
    best_x = pop.champion_x[0]    # выбор лучших результатов для x, y и f(x, y)
    best_y = pop.champion_x[1]
    best_f = pop.champion_f[0]
    return best_x, best_y, best_f

# задание функции Химмельбау
class Himmelbau:
    def fitness(self, x):
        return [(x[0]**2 + x[1] - 11)**2 + (x[0] + x[1]**2 - 7)**2]

    def get_bounds(self):
        return ([-5,-5],[5,5])

prob = pg.problem(Himmelbau())      # определение задачи оптимизации в качестве функции Химмельбау
results = []      # список результатов тестирования

PSO = pg.algorithm(pg.pso(gen=100))     # алгоритм роя частиц
CMAES = pg.algorithm(pg.cmaes(gen=100))     # алгоритм стратегии эволюции адаптации ковариационной матрицы
DE = pg.algorithm(pg.de(gen=100))       # алгоритм дифференциальной эволюции

results.append(["Particle_Swarm_Optimization"] + list(test_func(prob, PSO)))       # запись результатов тестирования через алгоритм роя частиц
results.append(["Covariance_Matrix_Adaptation_Evolution_Strategy"] + list(test_func(prob, CMAES)))     # запись результатов тестирования через алгоритм стратегии эволюции адаптации ковариационной матрицы
results.append(["Differential_Evolution"] + list(test_func(prob, DE)))        # запись результатов тестирования через алгоритм дифференциальной эволюции
```

```
# вывод результатов тестирования функции на поиск оптимума 3 разными
алгоритмами доступными в библиотеке
visualisation_results = pd.DataFrame(results, columns=[
    'Algorithm',
    'x',
    'y',
    'f(x, y)',
])

print(visualisation_results)
```

Алгоритмы, используемые для тестирования:

- PSO - алгоритм роя частиц;
- CMAES - алгоритм стратегии эволюции адаптации ковариационной матрицы;
- DE - алгоритм дифференциальной эволюции.

б) Результат реализации тестирования данной функции на поиск оптимума 3 разными алгоритмами доступными в библиотеке pygmo:

	Algorithm	x	y	f(x, y)
0	Particle_Swarm_Optimization	3.000000	2.000005	3.652240e-10
1	Covariance_Matrix_Adaptation_Evolution_Strategy	3.000003	2.000001	5.109080e-10
2	Differential_Evolution	3.000003	1.999998	2.395466e-10

Рис. 1. Результат тестирования функции Химмельбау на поиск оптимума 3 разными алгоритмами доступными в библиотеке pygmo

2. Функция трехгорбого верблюда:

а) Этот код включает:

Листинг 2 – Программа реализации тестирования функции трехгорбого верблюда на поиск оптимума 3 разными алгоритмами доступными в библиотеке pygmo

```
import pygmo as pg
import pandas as pd
from itertools import product

# задание алгоритма тестирования функции
def test_func(problem, algorithm):
    pop = pg.population(problem, 20)    # создание популяции
    pop = algorithm.evolve(pop)     # эволюция популяции с помощью алгоритма
    best_x = pop.champion_x[0]    # выбор лучших результатов для x, y и f(x, y)
    best_y = pop.champion_x[1]
    best_f = pop.champion_f[0]
    return best_x, best_y, best_f

# задание функции трехгорбого верблюда
class ThreeHumpedCamel:
```

```
    def fitness(self, x):
        return [(x[0]**2 + x[1] - 11)**2 + (x[0] + x[1]**2 - 7)**2]

    def get_bounds(self):
        return ([-5,-5],[5,5])

prob = pg.problem(ThreeHumpedCamel())      # определение задачи оптимизации в
качестве функции трехгорбого верблюда
results = []      # список результатов тестирования

PSO = pg.algorithm(pg.pso(gen=100))     # алгоритм роя частиц
CMAES = pg.algorithm(pg.cmaes(gen=100))     # алгоритм стратегии эволюции
адаптации ковариационной матрицы
DE = pg.algorithm(pg.de(gen=100))       # алгоритм дифференциальной эволюции

results.append(["Particle_Swarm_Optimization"] + list(test_func(prob,
PSO)))       # запись результатов тестирования через алгоритм роя частиц
results.append(["Covariance_Matrix_Adaptation_Evolution_Strategy"] +
list(test_func(prob, CMAES)))     # запись результатов тестирования через
алгоритм стратегии эволюции адаптации ковариационной матрицы
results.append(["Differential_Evolution"] + list(test_func(prob, DE)))       #
запись результатов тестирования через алгоритм дифференциальной эволюции

# вывод результатов тестирорования функции на поиск оптимума 3 разными
алгоритмами доступными в библиотеке
visualisation_results = pd.DataFrame(results, columns=[
    'Algorithm',
    'x',
    'y',
    'f(x, y)',
])

print(visualisation_results)
```

Алгоритмы, используемые для тестирования:

- PSO - алгоритм роя частиц;
- CMAES - алгоритм стратегии эволюции адаптации ковариационной матрицы;
- DE - алгоритм дифференциальной эволюции.

б) Результат реализации тестирования данной функции на поиск оптимума 3 разными алгоритмами доступными в библиотеке pygmo:

	Algorithm	x	y	f(x, y)
0	Particle_Swarm_Optimization	3.000000	2.000005	3.652240e-10
1	Covariance_Matrix_Adaptation_Evolution_Strategy	3.000003	2.000001	5.109080e-10
2	Differential_Evolution	3.000003	1.999998	2.395466e-10

Рис. 2. Результат тестирования функции трехгорбого верблюда на поиск оптимума 3 разными алгоритмами доступными в библиотеке pygmo