

ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

профессор

должность, уч. степень, звание

подпись, дата

Ю.А. Скобцов

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

Оптимизация многомерных функций с помощью эволюционной стратегии

По дисциплине: Эволюционные методы проектирования программно-
информационных систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.

4134к

подпись, дата

Костяков Н.А.

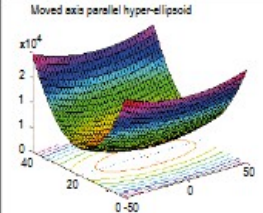
инициалы, фамилия

Санкт-Петербург
2024

Цель работы:

Оптимизация функций многих переменных модификация методом эволюционной стратегии. Графическое отображение результатов оптимизации

Вариант 4:

№ вв.	Название	Оптимум	Вид функции	График функции
4	Moved axis parallel hyper-ellipsoid function	global minimum $f(x)=0$; $x(i)=5*i$, $i=1:n$	$f_{1c}(x) = \sum_{i=1}^n 5i \cdot x_i^2$ $-5.12 \leq x_i \leq 5.12$ $f1c(x)=\text{sum}(5*i \cdot x(i)^2),$ $i=1:n;$	

Задание:

1. Создать программу, использующую ЭС для нахождения оптимума функции согласно таблице вариантов, приведенной в приложении А.

Для всех Benchmark-ов оптимумом является минимум. Программу выполнить на встроенном языке пакета Matlab - Python (или любом, доступным вам, языке программирования).

2. Для $n=2$ вывести на экран график данной функции с указанием найденного экстремума, точек популяции. Для вывода графиков использовать стандартные возможности пакета Matlab - Python. Предусмотреть возможность пошагового просмотра процесса поиска решения.

3. Исследовать зависимость времени поиска, числа поколений (генераций), точности нахождения решения от основных параметров генетического алгоритма:

- число особей в популяции
- вероятность мутации.

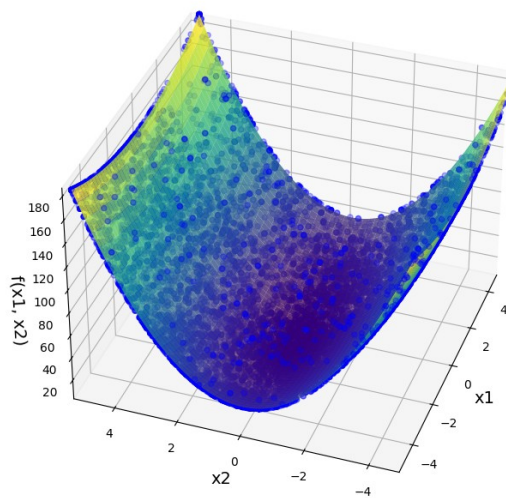
Критерий останова вычислений – повторение лучшего результата заданное количество раз или достижение популяцией определенного возраста (например, 100 эпох).

4. Повторить процесс поиска решения для $n=3$, сравнить результаты, скорость работы программы.

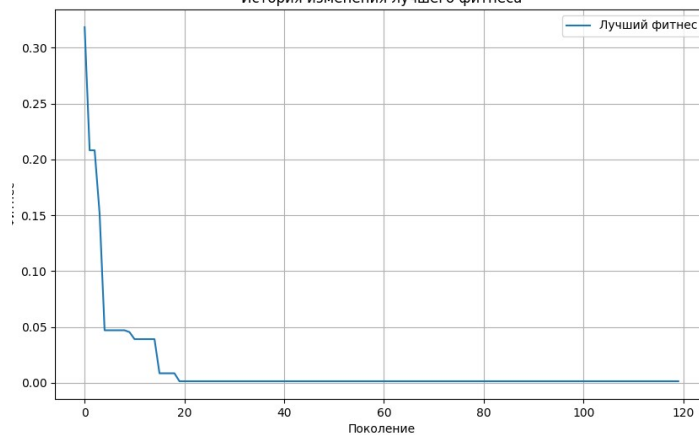
Выполнение:

Для $n = 2$:

Оптимизация `fitness_function` с помощью ЭС

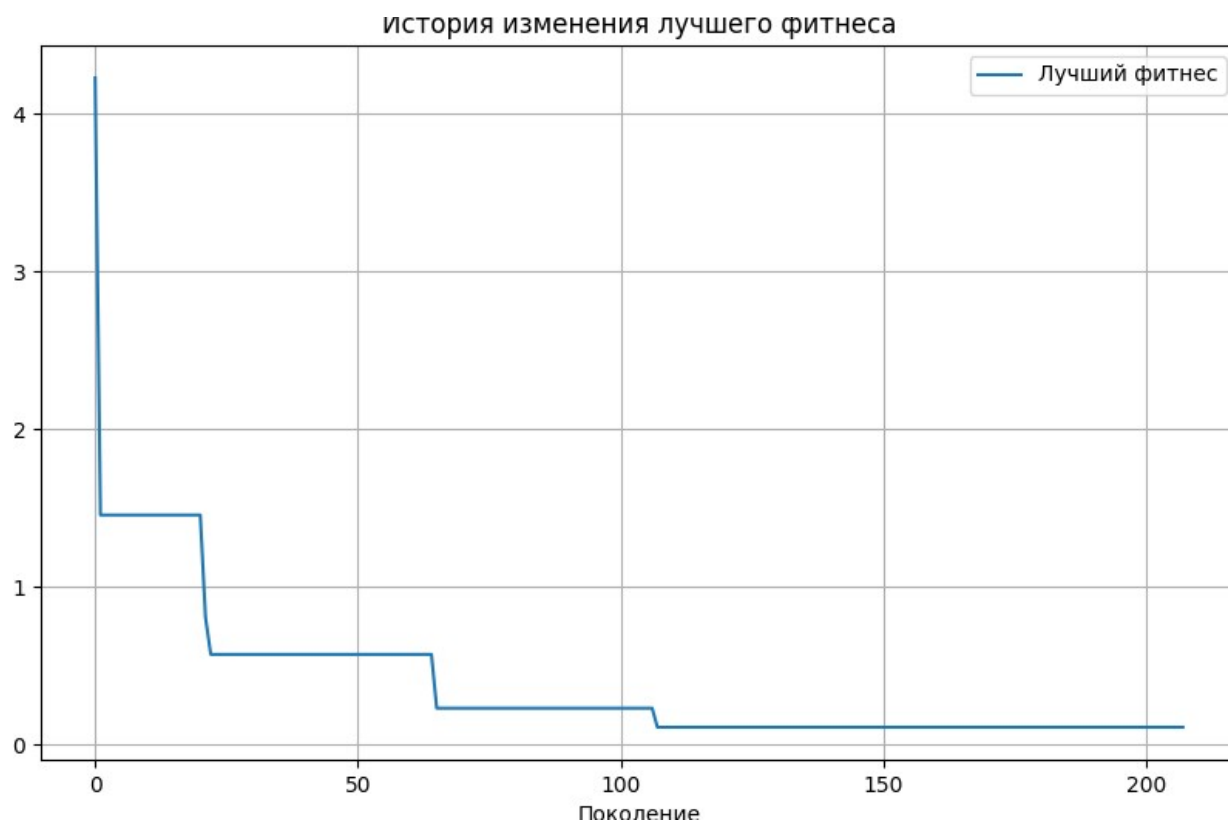


История изменения лучшего фитнеса



```
PS D:\Vyzovskoe3-4\7 сем\ЭМПИС\лаб5> python main.py
Остановка на поколении 119 из-за отсутствия улучшений за 100 поколений.
Лучшее найденное решение (ЭС):  $x_1 = 0.013446$ ,  $x_2 = -0.014198$ 
Значение функции в этой точке (ЭС): 0.001390
Время выполнения программы: 21.68 секунд
PS D:\Vyzovskoe3-4\7 сем\ЭМПИС\лаб5> 
```

Для $n=3$:



```
PS D:\Vyzovskoe3-4\7 сем\ЭМППИС\лаб5> python main3.py
Остановка на поколении 207 из-за отсутствия улучшений за 100 поколений.
Лучшее найденное решение (ЭС): x1 = 0.160145, x2 = 0.060178, x3 = -0.076682
Значение функции в этой точке (ЭС): 0.112057
Время выполнения программы: 0.68 секунд
```

Выводы:

В данной работе была реализована программа для оптимизации многомерной функции Эасома с использованием эволюционной стратегии. В результате экспериментов были получены оптимальные значения функции, визуализированные на графиках, а также проведено исследование влияния параметров алгоритма, таких как размер популяции и вероятность мутации, на время поиска и точность нахождения решения. Для трехмерного случая был проведен аналогичный анализ, что позволило сравнить эффективность алгоритма в зависимости от размерности задачи.

Листинг

```
import numpy as np
import matplotlib.pyplot as plt
import time

# Новая функция оценки
def fitness_function(f):
    return sum((5 * i + 1) * (f[i] ** 2) for i in range(len(f)))

# Параметры эволюционной стратегии
population_size = 300          # Размер популяции
max_generations = 300          # Максимальное количество поколений
mutation_probability = 0.5     # Вероятность мутации
mutation_sigma = 0.5           # Стандартное отклонение для мутации
no_improvement_limit = 100     # Лимит поколений без улучшений для остановки

# Диапазоны для визуализации и ограничений популяции
x_min_vis, x_max_vis = -5.12, 5.12

# Инициализация начальной популяции в диапазоне [-5.12, 5.12]
initial_population = np.random.uniform(x_min_vis, x_max_vis, (population_size, 2))

# Построение сетки для визуализации функции
x1 = np.linspace(x_min_vis, x_max_vis, 200)
x2 = np.linspace(x_min_vis, x_max_vis, 200)
x1, x2 = np.meshgrid(x1, x2)
z = np.array([fitness_function([x1_val, x2_val]) for x1_val, x2_val in
zip(x1.flatten(), x2.flatten())])
z = z.reshape(x1.shape)

# Создание фигуры
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(x1, x2, z, cmap='viridis', edgecolor='none', alpha=0.8)

# Установка границ осей
ax.set_xlim([x_min_vis, x_max_vis])
ax.set_ylim([x_min_vis, x_max_vis])
ax.set_zlim([0, np.max(z)]) # Для лучшей видимости
ax.view_init(elev=30, azim=240)
ax.set_title('Оптимизация fitness_function с помощью ЭС', fontsize=16)
ax.set_xlabel('x1', fontsize=14)
ax.set_ylabel('x2', fontsize=14)
ax.set_zlabel('f(x1, x2)', fontsize=14)

# Начало замера времени
start_time = time.time()

best_fitness_history = []
best_solution = initial_population[0]
```

```

best_fitness = fitness_function(best_solution)

# Основной цикл эволюционной стратегии
no_improvement_count = 0

for generation in range(max_generations):
    # Оценка популяции
    fitness_values = np.array([fitness_function(ind) for ind in initial_population])

    # Поиск лучшего решения
    current_best_fitness = np.min(fitness_values)
    best_idx = np.argmin(fitness_values)

    if current_best_fitness < best_fitness:
        best_fitness = current_best_fitness
        best_solution = initial_population[best_idx]
        no_improvement_count = 0 # Сброс при улучшении
    else:
        no_improvement_count += 1 # Увеличиваем счетчик без улучшения

    best_fitness_history.append(best_fitness)

    # Проверка условия остановки
    if no_improvement_count >= no_improvement_limit:
        print(f"Остановка на поколении {generation} из-за отсутствия улучшений за {no_improvement_limit} поколений.")
        break

    # Создание новой популяции
    new_population = []
    for _ in range(population_size):
        # Выбор родителя случайным образом
        parent = initial_population[np.random.choice(population_size)]

        # Мутация с вероятностью
        if np.random.rand() < mutation_probability:
            child = parent + np.random.normal(0, mutation_sigma, 2)
            child = np.clip(child, x_min_vis, x_max_vis)
        else:
            child = parent
        new_population.append(child)

    initial_population = np.array(new_population)

    # Отображение текущей популяции на графике
    ax.scatter(initial_population[:, 0], initial_population[:, 1],
               [fitness_function(ind) for ind in initial_population],
               color='blue', alpha=0.3)

    plt.pause(0.1) # Пауза для пошагового просмотра

```

```
# Отображение найденного экстремума (ЭС)
ax.scatter(best_solution[0], best_solution[1], best_fitness,
           color='red', s=100, label='Найденный минимум (ЭС)')

# Добавление условных обозначений
ax.legend(loc='upper right')

# Окончание замера времени
end_time = time.time()
execution_time = end_time - start_time

# Вывод результатов
print(f'Лучшее найденное решение (ЭС): x1 = {best_solution[0]:.6f}, x2 = {best_solution[1]:.6f}')
print(f'Значение функции в этой точке (ЭС): {best_fitness:.6f}')
print(f'Время выполнения программы: {execution_time:.2f} секунд')

# История изменения лучшего фитнеса
plt.figure(figsize=(10, 6))
plt.plot(best_fitness_history, label='Лучший фитнес')
plt.title('История изменения лучшего фитнеса')
plt.xlabel('Поклоение')
plt.ylabel('Фитнес')
plt.legend()
plt.grid()
plt.show()
```