Домашнее задание № 10

Задача 1. Найти экстремум функции $f(x_1, \dots, x_n) = \sum_{i=1}^{n} x_i^2$ на множестве $\sum_{i=1}^{n} x_i^4 \le 1$,

то есть решить задачу:

$$\begin{cases} x_1^2 + \dots + x_n^2 \to \text{extr}, \\ x_1^4 + \dots + x_n^4 - 1 \le 0. \end{cases}$$

1. Составляем функцию Лагранжа

$$L(x, \lambda) = \sum_{i=0}^{m+r} \lambda_i \varphi_i(x):$$

$$L(x, \lambda) = \lambda_0 \left(\sum_{i=1}^{n} x_i^2 \right) + \lambda_1 \left(\sum_{i=1}^{n} x_i^4 - 1 \right)$$

2. Вычислим частные производные:

$$\frac{\partial L(x, \lambda)}{\partial x_i} = 2\lambda_0 x_i + 4\lambda_1 x_i^3, \quad i = 1, \dots, n$$

3. Запишем необходимое условие экстремума:

$$\begin{cases} 2\lambda_0 x_i + 4\lambda_1 x_i^3 = 0, \quad i = 1, \dots, n. \\ \lambda_1 \left(\sum_{i=1}^{n} x_i^4 - 1 \right) = 0 \end{cases}$$

4. Рассмотрим 2 случая:

а) $\lambda_0 = 0$

б) $\lambda_0 \neq 0$

а) При $\lambda_0 = 0$ система примет вид:

$$\begin{cases} 4\lambda_1 x_i^3 = 0, & i = 1, \dots, n \\ \lambda_1 \left(\sum_{i=1}^{n} x_i^4 - 1 \right) = 0 \end{cases}$$

Очевидно, что при $\lambda_1 \neq 0$ она не совместна. При $\lambda_1 = 0$ весь вектор стационарных условий и подозрительных точек на экстремум не возникает.

б) Рассмотрим случай, когда $\lambda_0 \neq 0$. Тогда удобно положить $\lambda_0 = 2$:

$$\begin{cases} 4x_i + 4\lambda_1 x_i^3 = 0, & i = 1, \dots, n \\ \lambda_1 \left(\sum_{i=1}^{n} x_i^4 - 1 \right) = 0 \end{cases}$$

Из последнего уравнения следует, что либо $\lambda_1 = 0$, либо $\sum_{i=1}^{n} x_i^4 = 1$. Из первого случая следует, что $x_1 = \dots = x_n = 0$ — имеем 1 подозрительную точку $(0, \dots, 0)$, подставку в нее целевой функции принимает нулевое значение, в остальных же — положительное, следовательно в данной точке достигается глобальный минимум. Во втором случае система примет вид:

$$\begin{cases} x_i (1 + \lambda_1 x_i^2) = 0, & i = 1, \dots, n \\ \sum_{i=1}^{n} x_i^4 = 1 \end{cases}$$

Тогда возможно 2 случая:

1) $\lambda_1 \geq 0$ — из n уравнений следует, что $x_1 = \dots = x_n = 0$, что противоречит последнему уравнению.

2) $\lambda_1 < 0$ — из n уравнений следует, что существует 3 решения: $x_i = 0$, $x_i = \pm \sqrt{-\frac{1}{\lambda_1}}$. Исходя из этого для $x_i = \pm \sqrt{-\frac{1}{\lambda_1}}$ обозначим множество индексов x_i как Ω, мощность Ω обозначим как k. Тогда: $\sum_{i=1}^{n} x_i^4 = \sum_{i \in \Omega} x_i^4 + \sum_{i \notin \Omega} x_i^4 = \sum_{i \in \Omega} \frac{1}{\lambda_1^2} = \frac{k}{\lambda_1^2} = 1$

Отсюда $\lambda_1 = -\sqrt{k}$ и соответственно:

$x_i = \pm \sqrt[4]{\frac{1}{k}}$, $i \in \Omega$, $x_i = 0$, $i \notin \Omega$ — все точки удовлетворяющие данному

Критерию подозрительности на экстремум. В них целевая функция принимает значение $\sum_{i=1}^{n} x_i^2 = \sum_{i \in \Omega} x_i^2 = \frac{k}{\sqrt{k}} = \sqrt{k}$ – максимуму при данном значении достигает-ся, когда $I = n$, т.е. все координаты подозрительной на экстремум точки ис-ключены. Отсюда следует, что целевая функция достигает глобального макси-мума в тех точках x, у которых все координаты x_i равны $\pm \sqrt[4]{\frac{1}{n}}$, с максиму-мом $\sqrt{n}$.

Рассмотрим остальные точки подозри-тельные на экстремум: поскольку $x_1 = -\sqrt{k} < 0$ – по условию согласования знаков, то в этих точках либо мак-симум максимум, либо экстремума нет. Проверим это, вычислив частные про-изводные и полный дифференциал вто-рого порядка от функции Лагранжа:

$$\frac{\partial^2 L(x,\lambda)}{\partial x_i^2} = 2\lambda_0 + 12\lambda_i x_i^2,$$

$$\frac{\partial^2 L(x,\lambda)}{\partial x_i \partial x_j} = 0, \quad i \neq j;$$

$$L''_{xx}(a,\lambda)dx^2 = \sum_{i=1}^{n}(4 - 12\sqrt{\lambda}\, a_i^2)dx_i^2 =$$

$$= 4\sum_{i\in\Omega}^{n-c} dx_i^2 - 8\sum_{i\in\Omega}^{c} dx_i^2$$

Пусть $\varphi(x) = x_1^4 + \dots + x_n^4 - 1$ и $\varphi(a)dx = 0$:

$$\varphi'(a)dx = \sum_{i=1}^{n} 4a_i^2 dx_i = 4\sum_{i\in\Omega} a_i^3 dx_i = 0$$

Рассмотрим вектор dx, который имеет только одну ненулевую координату, отвечающую индексу $i \in \Omega$. Тогда будут выполняться условия:

$\varphi'(a)dx = 0$, $L''_{xx}(a,\lambda)dx^2 > 0$, следовательно в точке a максимума нет, либо и экстремума в общем случае.

Ответ: максимум минимум достигается в точке $x=0$ и равен, глобального

максимум достигается в точках, все координаты которых имеют вид $\pm\sqrt{\frac{1}{n}}$ и равен $\sqrt{n}$.

Задача 1

Проверим полученное аналитически значения экстремумов при помощи оптимизатора scipy. Возьмем значение n = 100.

In [1]:
```
import numpy as np
from scipy.optimize import minimize
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# Определяем целевую функцию
def objective(x):
    return np.sum(np.square(x))

# Определяем ограничения
def constraint(x):
    return -(np.sum(np.power(x, 4)) - 1)

n = 100

# Начальное приближение
x0 = np.ones(n)

# Определяем ограничения типа неравенства как словарь
constraint_eq = {'type': 'ineq', 'fun': constraint}

# Запускаем оптимизацию с использованием метода SLSQP
result = minimize(objective, x0, constraints=constraint_eq, method='SLSQP')

# Выводим результат
print(f'n = {n}')
print("Минимум функции:", -result.fun)
print("Аргументы минимума:", result.x)
```

```
n = 100
Минимум функции: -0.0
Аргументы минимума: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0.]
```

In [3]:
```python
# Определяем целевую функцию для оптимизации с отрицательным знаком
def objective(x):
    return -np.sum(x**2)

# Определяем ограничения
def constraint(x):
    return -(np.sum(np.power(x, 4)) - 1)

# Задаем начальное приближение
n = 100
x0 = np.ones(n)

# Определяем ограничения типа неравенства как словарь
constraint_eq = {'type': 'ineq', 'fun': constraint}

# Запускаем оптимизацию с использованием метода SLSQP
result = minimize(objective, x0, constraints=constraint_eq, method='SLSQP')

# Выводим результат (с отрицательным знаком)
print(f'n = {n}')
print("Максимум функции:", -result.fun)
print("Аргументы максимума:", result.x)
```

```
n = 100
Максимум функции: 10.000000670334469
Аргументы максимума: [0.31622788 0.31622768 0.31622812 0.31622768 0.316227
88 0.31622769
 0.31622788 0.31622788 0.31622788 0.31622768 0.31622812 0.31622768
 0.31622788 0.31622769 0.31622788 0.31622788 0.31622788 0.31622768
 0.31622812 0.31622768 0.31622788 0.31622769 0.31622788 0.31622788
 0.31622788 0.31622768 0.31622795 0.31622768 0.31622788 0.31622769
 0.31622788 0.31622788 0.31622788 0.31622768 0.31622812 0.31622768
 0.31622788 0.31622769 0.31622788 0.31622788 0.31622788 0.31622768
 0.31622812 0.31622768 0.31622788 0.31622769 0.31622788 0.31622788
 0.31622788 0.31622768 0.31622812 0.31622768 0.31622788 0.31622769
 0.31622788 0.31622788 0.31622788 0.31622768 0.31622795 0.31622768
 0.31622788 0.31622764 0.31622788 0.31622788 0.31622768 0.31622768
 0.31622812 0.31622768 0.31622788 0.31622764 0.31622788 0.31622788
 0.31622788 0.31622768 0.31622812 0.31622768 0.31622769 0.31622764
 0.31622788 0.31622788 0.31622768 0.31622768 0.31622788 0.31622768
 0.31622764 0.31622764 0.31622788 0.31622788 0.31622768 0.31622768
 0.31622788 0.31622788 0.31622788 0.31622764 0.31622788 0.31622788
 0.31622737 0.31622622 0.31622576 0.31622769]
```

Вывод: полученные значения совпадают с аналитическими расчетами: fmin = 0, при xi = 0, fmax = $\sqrt{n}$, для xi = +- 1 / $\sqrt[4]{n}$

Задача 2. Решить задачу коммивояжера методом ветвей и границ:

	1	2	3	4	5
1	∞	4	5	7	5
2	8	∞	5	6	6
3	3	5	∞	9	6
4	3	5	6	∞	2
5	6	2	3	8	∞

1. Возьмем в качестве произвольного маршрута: $X_0 = (1,2)(2,3)(3,4)(4,5)(5,1)$, тогда: $F(X_0) = 4+5+9+2+6 = 26$

Для определения нижней границы множества решений, для этого найдем в каждой строке минимальный элемент. Матрица примет следующий вид:

	1	2	3	4	5	min
1	∞	4	5	7	5	4
2	8	∞	5	6	6	5
3	3	5	∞	9	6	3
4	3	5	6	∞	2	2
5	6	2	3	8	∞	2

Затем вычтем минимальный элемент из рассматриваемой строки:

	1	2	3	4	5
1	∞	0	1	3	1
2	3	∞	0	1	1
3	0	2	∞	6	3
4	1	3	4	∞	0
5	4	0	1	6	∞

Проведем операцию редукции по столбцам, для этого в каждом столбце найдем минимальный элемент

	1	2	3	4	5
1	∞	0	1	3	1
2	3	∞	0	1	1
3	0	2	∞	6	3
4	1	3	4	∞	0
5	4	0	1	6	∞
min	0	0	0	1	0

После вычитания минимальных элементов получаем полностью редуцированную матрицу:

	1	2	3	4	5
1	∞	0	1	2	1
2	3	∞	0	0	1
3	0	2	∞	5	3
4	1	3	4	∞	0
5	4	0	1	5	∞

Значения минимальных элементов строк и столбцов называются константами приведения. Сумма констант приведения определяет минимальную длину H:

$$H = 4+5+3+2+2+0+0+0+1+0 = 17$$

Элемент матрицы соответствует расстоянию от пункта i до j. Поскольку в матрице 5 городов, то матрица удовлетворяет множеству допустимых маршрутов

2. Определим ребро ветвления и разобьем все множество маршрутов относительно этого ребра на 2 подмножества (i,j) и (i,j). Для этого для всех клеток матрицы с нулевыми элементами кроме ∞ и определим для них сумму образовавшихся констант

приведения, которые указаны в скобках:

	1	2	3	4	5	min
1	∞	0(1)	1	2	1	1
2	3	∞	0(1)	0(2)	1	0
3	0(3)	2	∞	5	3	2
4	1	3	4	∞	0(0)	1
5	4	0(1)	1	5	∞	1
min	1	0	1	2	1	0

$d(1,2) = 1+0 = 1$; $d(2,3) = 0+1 = 1$; $d(2,4) = 0+2 = 2$;

$d(3,1) = 2+1 = 3$; $d(4,3) = 1+1 = 2$; $d(5,2) = 1+0 = 1$;

Наибольшая оценка достигает ребро равная 3. Следовательно, множество разбивается на 2 подмножества $(3,1)$ и $(3^*,1^*)$

Исключим ребро $(3,1)$ на ∞ и осуществим приведение матрицы расстояний для образовавшегося подмножества $(3^*,1^*)$:

	1	2	3	4	5	min
1	∞	0	1	2	1	0
2	3	∞	0	0	1	0
3	∞	2	∞	5	3	2
4	1	3	4	∞	0	0
5	4	0	1	5	∞	0
	1	0	0	0	0	3

Нижняя граница гамильтоновых циклов этого подмножества:

$H(3^*, 1^*) = 17 + 3 = 20$

Включение ребра (3,1) приводится путём исключения всех элементов 3 строки и 1 столбца, в которой элемент (1,3) на ∞, для исключения образования негамильтоновых цикла. В результате получим матрицу, уменьшем её в приведённой форме:

	2	3	4	5	min
1	0	∞	2	1	0
2	∞	0	0	1	0
4	3	4	∞	0	0
5	0	1	5	∞	0
min	0	0	0	0	0

Сумма констант уменьшения равна 0, нижняя граница подмножества равна:

$H(3,1) = 17 + 0 = 17 \leq 20$ – поскольку нижняя граница подмножества (3,1) меньше, чем подмножества $(3^*, 1)$, то

ребро (3,1) включаем в маршрут с новой границей $H = 17$.

3. Определяем ребро ветвления:

	2	3	4	5	min
1	0(1)	∞	2	1	1
2	∞	0(1)	0(2)	1	0
4	3	4	∞	0(4)	3
5	0(1)	1	5	∞	1
	0	1	2	1	0

$d(1,2) = 1+0 = 1$; $d(2,3) = 0+1 = 1$; $d(2,4) = 0+2 = 2$; $d(4,5) = 3+1 = 4$; $d(5,2) = 1+0 = 1$

Наибольшая сумма констант приведения 4. Проводим операцию описание вне дуги ребра (4,5):

	2	3	4	5	min
1	0	∞	2	1	0
2	∞	0	0	1	0
4	3	4	∞	∞	3
5	0	1	5	∞	0
min	0	0	0	1	4

$H(4^{*}, 5^{*}) = 17 + 4 = 21$

Включаем ребро (4,5):

	2	3	4	min
1	0	∞	2	0
2	∞	0	0	0
5	0	1	∞	0
min	0	0	0	0

Сумма констант приведения равна 0.

$H(4,5) = 17 + 0 = 17 \leq 21$ – ребро (4,5) включаем в новой границей $H = 17$

4. Определяем ребро ветвления:

	2	3	4	min
1	0(2)	∞	2	2
2	∞	0(1)	0(2)	0
5	0(1)	1	∞	1
min	0	1	2	0

$d(1,2) = 2+0 = 2$; $d(2,3) = 0+1 = 1$; $d(2,4) = 0+2 = 2$; $d(5,2) = 1+0 = 1$

Максимальная $d = 2$ для ребра (2,4)

Исключаем ребро (2,4):

	2	3	4	min
1	0	∞	2	0
2	∞	0	∞	0
5	0	1	∞	0
min	0	0	2	2

$H(2^*,4^*) = 17 + 2 = 19$

Включаем ребро (2,4)

	2	3	min
1	0	∞	0
5	0	1	0
min	0	1	1

Сумма констант приведения равна 1.

$H(2,4) = 17 + 1 \leq 19$ – ребро (2,4) включено в маршрут с новой границей $H = 18$.

В соответствии с полученной матрицей включаем в гамильтонов маршрут ребра (1,2) и (5,3); в итоге, по удобству ветвления гамильтонов цикл образуют ребра – (3,1)(1,2)(2,4)(4,5)(5,3), с длиной маршрута 18 ($F(M_1) = 18$).

Ответ: (3,1)(1,2)(2,4)(4,5)(5,3).
$F(M_1) = 18$.

Задача 2

Проверим при помощи оптимизатора ORTools описанное выше аналитическое решение.

In [1]:
```
import random
import matplotlib.pyplot as plt
import numpy as np
from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp
from deap import base, creator, tools, algorithms
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
def create_data_model():
    data = {}
    data['distance_matrix'] = [
            [0, 4, 5, 7, 5],
            [8, 0, 5, 6, 6],
            [3, 5, 0, 9, 6],
            [3, 5, 6, 0, 2],
            [6, 2, 3, 8, 0]
        ]
    data['num_vehicles'] = 1
    data['depot'] = 2
    return data

def main():
    data = create_data_model()
    manager = pywrapcp.RoutingIndexManager(len(data['distance_matrix']), data['num_vehic
    routing = pywrapcp.RoutingModel(manager)

    def distance_callback(from_index, to_index):
        from_node = manager.IndexToNode(from_index)
        to_node = manager.IndexToNode(to_index)
        return data['distance_matrix'][from_node][to_node]

    transit_callback_index = routing.RegisterTransitCallback(distance_callback)
    routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)

    search_parameters = pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.first_solution_strategy = (routing_enums_pb2.FirstSolutionStrategy

    solution = routing.SolveWithParameters(search_parameters)
    if solution:
        print('Objective: {} miles'.format(solution.ObjectiveValue()))
        index = routing.Start(0)
        plan_output = 'Route for vehicle 0:\n'
        route_distance = 0
        while not routing.IsEnd(index):
            plan_output += ' {} ->'.format(manager.IndexToNode(index))
            previous_index = index
            index = solution.Value(routing.NextVar(index))
            route_distance += routing.GetArcCostForVehicle(previous_index, index, 0)
        plan_output += ' {}\n'.format(manager.IndexToNode(index))
        print(plan_output)
        plan_output += 'Route distance: {}miles\n'.format(route_distance)
    else:
        print('No solution found !')

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

```
Objective: 18 miles
Route for vehicle 0:
 2 -> 0 -> 1 -> 3 -> 4 -> 2
```

Решение полностью совпало с аналитическим, единственным отличие в данном случае является то, что нумерация вершин начинается с 0.

Также дополнительно для проверки результатов помимо оптимизатора используем метаэвристический алгоритм. В качестве метаэвристического алгоритма выберем генетический алгоритм. Реализуем его, используя фрэймворк DEAP.

In [3]:
```python
# Определяем функцию расчета длины маршрута
def evaluate(individual, distance_matrix):
    total_distance = 0
    for i in range(len(individual) - 1):
        total_distance += distance_matrix[individual[i]][individual[i + 1]]
    total_distance += distance_matrix[individual[-1]][individual[0]]  # Замыкаем маршрут
    return (total_distance,)

# Задаем параметры и запускаем генетический алгоритм
random.seed(42)
distance_matrix = [
    [0, 4, 5, 7, 5],
    [8, 0, 5, 6, 6],
    [3, 5, 0, 9, 6],
    [3, 5, 6, 0, 2],
    [6, 2, 3, 8, 0]
]

# Создаем инструменты для генетического алгоритма
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)
toolbox = base.Toolbox()
toolbox.register("indices", random.sample, range(len(distance_matrix)), len(distance_mat
toolbox.register("individual", tools.initIterate, creator.Individual, toolbox.indices)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("mate", tools.cxOrdered)
toolbox.register("mutate", tools.mutShuffleIndexes, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", evaluate, distance_matrix=distance_matrix)

pop = toolbox.population(n=100)
hof = tools.HallOfFame(1)
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("avg", np.mean)
stats.register("min", np.min)
result, log = algorithms.eaSimple(pop, toolbox, cxpb=0.5, mutpb=0.2, ngen=10, stats=stat

# Выводим результаты
best_individual = hof[0]
best_distance = best_individual.fitness.values[0]
best_route = best_individual
print("Лучший маршрут:", best_route)
print("Длина лучшего маршрута:", best_distance)
```

```
gen     nevals  avg     min
0       100     25.55   18
1       54      23.01   18
2       52      20.65   18
3       57      19.52   18
4       64      19.27   18
5       64      19.08   18
```

```
6       69      18.98   18
7       65      18.71   18
8       52      18.52   18
9       61      18.62   18
10      57      18.59   18
Лучший маршрут: [4, 2, 0, 1, 3]
Длина лучшего маршрута: 18.0
```

Дадим оценку устойчивости метаэвристики в зависимости от начальной точки и от количества итераций.

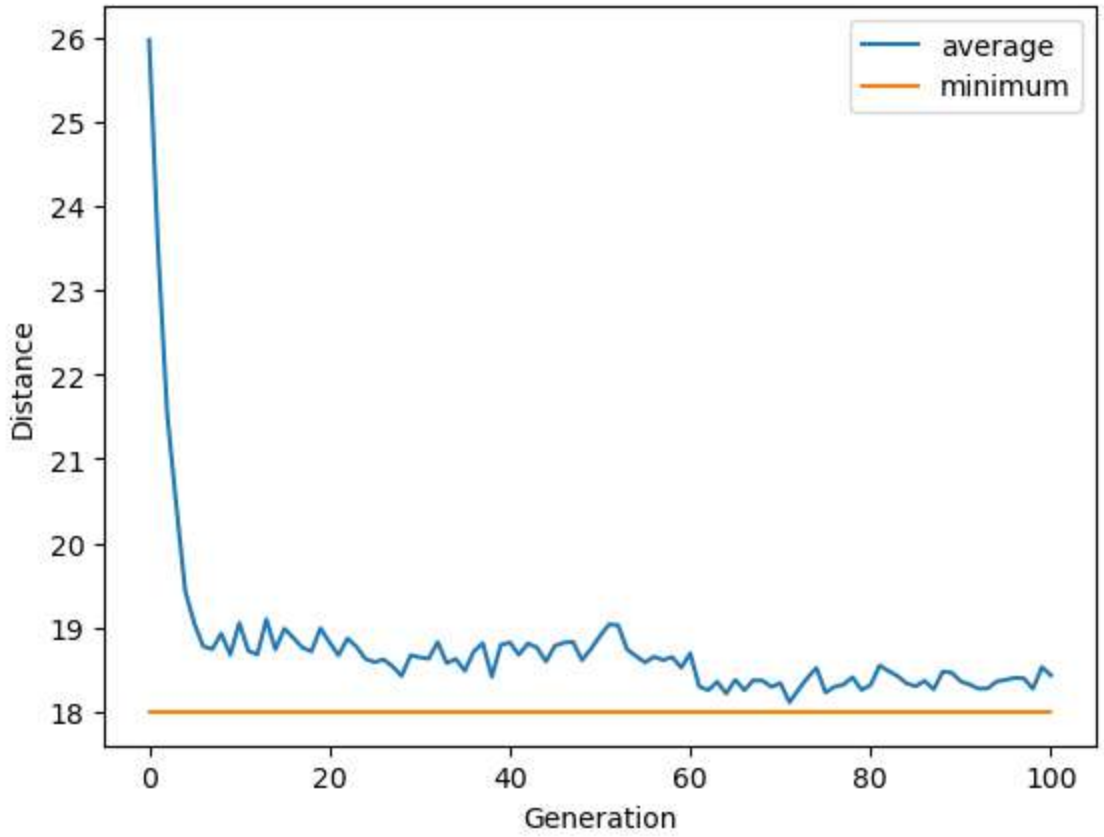
In [4]:

```python
import array
import random
import numpy as np
import matplotlib.pyplot as plt
from deap import base, creator, tools, algorithms

# Создаем объект класса DistanceMatrix для хранения матрицы расстояний
class DistanceMatrix:
    def __init__(self, distance_map):
        self.distance_map = distance_map

    def __getitem__(self, indices):
        return self.distance_map[indices[0]][indices[1]]

    def __len__(self):
        return len(self.distance_map)

# Создание эволюционной стратегии
def evolutionary_strategy(distance_matrix, population_size=300, num_generations=100):
    # Настройка особей (в данном случае, перестановки городов)
    creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
    creator.create("Individual", array.array, typecode='i', fitness=creator.FitnessMin)
    toolbox = base.Toolbox()
    toolbox.register("indices", np.random.permutation, len(distance_matrix))
    toolbox.register("individual", tools.initIterate, creator.Individual, toolbox.indice
    toolbox.register("population", tools.initRepeat, list, toolbox.individual)

    # Оценка функции (в данном случае, сумма расстояний)
    def eval_func(individual):
        indices = list(individual) + [individual[0]]
        return (sum(distance_matrix[indices[i], indices[i + 1]] for i in range(len(indiv

    # Регистрация функций
    toolbox.register("mate", tools.cxOrdered)
    toolbox.register("mutate", tools.mutShuffleIndexes, indpb=0.05)
    toolbox.register("select", tools.selTournament, tournsize=3)
    toolbox.register("evaluate", eval_func)

    # Создание начальной популяции и эволюция
    population = toolbox.population(n=population_size)
    stats = tools.Statistics(lambda ind: ind.fitness.values)
    stats.register("min", np.min)
    stats.register("avg", np.mean)
    logbook = tools.Logbook()
    population, logbook = algorithms.eaSimple(population, toolbox, cxpb=0.5, mutpb=0.2,

    # Получение лучшей особи
    best_ind = tools.selBest(population, k=1)[0]
    best_distance = eval_func(best_ind)[0]

    return best_ind, best_distance, logbook

# Определение матрицы расстояний
distance_matrix = DistanceMatrix([
```

```
    [0, 4, 5, 7, 5],
    [8, 0, 5, 6, 6],
    [3, 5, 0, 9, 6],
    [3, 5, 6, 0, 2],
    [6, 2, 3, 8, 0]
])

# Получение решения
best_individual, best_distance, logbook = evolutionary_strategy(distance_matrix)

# Построение графика
gen, avg, min_ = logbook.select("gen", "avg", "min")
plt.plot(gen, avg, label="average")
plt.plot(gen, min_, label="minimum")
plt.xlabel("Generation")
plt.ylabel("Distance")
plt.legend()
plt.show()

print('Best individual:', best_individual)
print('Best distance:', best_distance)
```

```
gen     nevals  min     avg
0       300     18      25.9667
1       174     18      23.47
2       196     18      21.54
3       208     18      20.46
4       179     18      19.4333
5       173     18      19.05
6       168     18      18.7767
7       163     18      18.7433
8       193     18      18.9267
9       178     18      18.6767
10      179     18      19.0533
11      207     18      18.7233
12      177     18      18.68
13      183     18      19.0967
14      194     18      18.74
15      184     18      18.9867
16      195     18      18.8767
17      193     18      18.76
18      163     18      18.7167
19      207     18      18.99
20      161     18      18.8233
21      179     18      18.67
22      176     18      18.87
23      189     18      18.77
24      191     18      18.6267
25      189     18      18.59
26      187     18      18.62
27      175     18      18.5367
28      169     18      18.4267
29      180     18      18.67
30      186     18      18.6467
31      171     18      18.63
32      186     18      18.8267
33      160     18      18.5767
34      183     18      18.6267
35      183     18      18.4867
36      187     18      18.7167
37      178     18      18.8133
38      191     18      18.4167
39      176     18      18.7933
40      186     18      18.8233
41      176     18      18.6767
42      189     18      18.81
```

43	183	18	18.7633
44	175	18	18.5967
45	178	18	18.7833
46	175	18	18.82
47	184	18	18.83
48	174	18	18.6133
49	182	18	18.7467
50	167	18	18.9
51	188	18	19.0367
52	198	18	19.0267
53	172	18	18.74
54	176	18	18.66
55	165	18	18.5867
56	163	18	18.6533
57	183	18	18.6133
58	189	18	18.65
59	182	18	18.5233
60	175	18	18.6933
61	192	18	18.3
62	188	18	18.2567
63	184	18	18.3567
64	188	18	18.2167
65	167	18	18.38
66	171	18	18.2567
67	170	18	18.3733
68	179	18	18.37
69	185	18	18.2967
70	175	18	18.34
71	171	18	18.1133
72	187	18	18.2633
73	163	18	18.4033
74	167	18	18.52
75	178	18	18.23
76	189	18	18.2967
77	200	18	18.3233
78	209	18	18.41
79	193	18	18.26
80	178	18	18.3167
81	181	18	18.55
82	201	18	18.4833
83	171	18	18.4233
84	188	18	18.3367
85	187	18	18.3033
86	180	18	18.3667
87	175	18	18.2667
88	190	18	18.4767
89	186	18	18.4633
90	182	18	18.36
91	181	18	18.3233
92	183	18	18.2733
93	184	18	18.28
94	171	18	18.36
95	185	18	18.38
96	190	18	18.4033
97	194	18	18.3967
98	187	18	18.2733
99	191	18	18.53
100	194	18	18.4333




```
Best individual: Individual('i', [2, 0, 1, 3, 4])
Best distance: 18
```

Вывод: аналитическое решение задачи коммивояжера было проверено с помощью оптимизатора ORTools, и решения полность совпали. С помощью фреймворка DEAP был реализован генетический алгоритм: полученные значения совпадают с полученными аналитически и с помощью оптимизатора. Дана оценка устойчивости генетического алгоритма - с увеличением количества генераций среднее растояние сходится к минимуму.