Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Лабораторные работы №6
по дисциплине «Методы оптимизации»

Вариант: **11**

Преподаватель:
Селина Елена Георгиевна

Выполнил:
Поленов Кирилл Александрович
Группа: P3213

Санкт-Петербург, 2025

Варианты 7-12

Дано множество из n городов и матрица расстояний между ними. Требуется объехать все города по кратчайшему пути, причем в каждом городе необходимо побывать один раз и вернуться в город, из которого был начат маршрут. Задачу необходимо решить с помощью генетического алгоритма.

	Город 1	Город 2	Город 3	Город 4	Город 5
Город 1	0	1	1	5	3
Город 2	1	0	3	1	5
Город 3	1	3	0	11	1
Город 4	5	1	11	0	1
Город 5	3	5	1	1	0

За целевую функцию следует принять сумму расстояний между городами.
Размер популяции N = 4.
Оператор мутации представляет собой случайную перестановку двух чисел в геноме, которые выбираются случайно. Вероятность мутации 0.01.

1. Решение вручную

Исходная популяция

№ строки	Код	Значение целевой функции
1	45312	5
2	42153	17
3	54312	17
4	43215	19

Пусть выбраны пары: (1, 4), (2, 4)
В результате были получены потомки:

№ строки	Родители	Потомки	Значение целевой функции для потомков
1	|45|312	|43|512	17
2	|43|215	|45|321	11
3	4|215|3	5|321|4	11
4	4|321|5	4|215|3	17

Популяция первого поколения после отсечения худших особей в результате работы оператора редукции:

№ строки	Код	Значение целевой функции	Вероятность участия в процессе размножения
1	45312	5	0.45
2	45321	11	0.21
3	53214	11	0.21
4	42153	17	0.13

Пусть выбраны пары: (1, 3), (2, 4)
В результате были получены потомки:

№ строки	Родители	Потомки	Значение целевой функции для потомков
1	|45|312	|53|124	5
2	|53|214	|45|321	11
3	45|32|1	24|15|3	13
4	42|15|3	54|32|1	19

Популяция второго поколения после отсечения худших особей в результате работы оператора редукции:

№ строки	Код	Значение целевой функции
1	45312	5
2	53124	5
3	45321	11
4	53214	11

Пусть выбраны пары: (3, 4), (1, 2)
В результате были получены потомки:

№ строки	Родители	Потомки	Значение целевой функции для потомков
1	|453|21	|532|14	11
2	|532|14	|453|21	11
3	453|12|	531|24|	5
4	531|24|	453|12|	5

Популяция второго поколения после отсечения худших особей приняла вид:

№ строки	Код	Значение целевой функции
1	45312	5
2	53124	5
3	45321	11
4	53214	11

Таким образом после 3 итераций значение целевой функции для лучшего решения изменилось с 5 на 5, среднее значение изменилось с 15.0 до 8.0, а общее качество с 60 до 32.

2. Программное решение

main.py

```
import itertools
import random
from numpy.random import choice

MUT_PROB = 0.01

def route_length(route, matrix):
    l = 0
    for i in range(len(route) - 1):
        l += matrix[route[i]][route[i + 1]]
    l += matrix[route[-1]][route[0]]
    return l


def make_child(p1, p2, splits):
    child = [None] * splits[0] + p2[splits[0]:splits[1]] + [None] * (c -
splits[1])
    i = 0
    j = splits[0] + 1
    stop = False
    while not stop:
        if child[i] is not None:
            i += 1
            if i >= c:
                stop = True
            continue
        while not stop:
            if p1[j] in child:
                j += 1
                if j >= c:
                    j = 0
                if j == splits[0] + 1:
                    stop = True
                continue
            child[i] = p1[j]
            break
    return child


def mutate_child(child, prob=MUT_PROB):
    if random.random() < prob:
        splits = list(choice(range(c), size=2, replace=False))
        child[splits[0]], child[splits[1]] = child[splits[1]],
child[splits[0]]
        return True
    return False


def make_children(p1, p2):
    while True:
        splits = sorted(list(choice(range(c + 1), size=2, replace=False)))
        if 2 <= splits[1] - splits[0] < c-1:
            break
    c1 = make_child(p1, p2, splits)
    c2 = make_child(p2, p1, splits)
    par1 = ''.join(map(lambda x: str(x + 1), p1[:splits[0]])) + '|' +
''.join(
        map(lambda x: str(x + 1), p1[splits[0]:splits[1]])) + '|' +
''.join(map(lambda x: str(x + 1), p1[splits[1]:]))
```

```
    par2 = ''.join(map(lambda x: str(x + 1), p2[:splits[0]])) + '|' +
''.join(
        map(lambda x: str(x + 1), p2[splits[0]:splits[1]])) + '|' +
''.join(map(lambda x: str(x + 1), p2[splits[1]:]))
    ch1 = ''.join(map(lambda x: str(x + 1), c1[:splits[0]])) + '|' + ''.join(
        map(lambda x: str(x + 1), c1[splits[0]:splits[1]])) + '|' +
''.join(map(lambda x: str(x + 1), c1[splits[1]:]))
    ch2 = ''.join(map(lambda x: str(x + 1), c2[:splits[0]])) + '|' + ''.join(
        map(lambda x: str(x + 1), c2[splits[0]:splits[1]])) + '|' +
''.join(map(lambda x: str(x + 1), c2[splits[1]:]))

    for i in range(1, 3):
        print(f"{locals()[f'par{i}']}  | {locals()[f'ch{i}']} |
{route_length(locals()[f'c{i}'], matrix)}")
    if mutate_child(c1):
        print("Потомок 1 мутировал: " + ''.join(map(lambda x: str(x + 1),
c1)))
    if mutate_child(c2):
        print("Потомок 2 мутировал: " + ''.join(map(lambda x: str(x + 1),
c2)))
    return c1, c2


def generation(c, matrix, p, g):
    global first_length, first_average, first_sum, final_length,
final_average, final_sum
    og_cities = list(range(c))
    population = sorted([random.sample(og_cities, len(og_cities)) for _ in
range(p)], key=lambda route: route_length(route, matrix))
    for i in range(g):
        print(f"Поколение {i + 1}")
        lengths = [route_length(route, matrix) for route in population]
        if i == 0:
            first_length = lengths[0]
            first_sum = sum(lengths)
            first_average = first_sum / len(lengths)
        probabilities = [1 / length for length in lengths]
        total_probability = sum(probabilities)
        probabilities = [prob / total_probability for prob in probabilities]
        if i != 0 and i < g - 1:
            print("Код   | Значение целевой функции | Вероятность участия в
размножении")
            for i, (code, length, prob) in enumerate(zip(population, lengths,
probabilities), 1):
                print(f"{''.join(map(lambda x: str(x + 1), code))} | {length}
| {prob}")
        else:
            print("Код   | Значение целевой функции")
            for i, (code, length) in enumerate(zip(population, lengths), 1):
                print(f"{''.join(map(lambda x: str(x + 1), code))} |
{length}")
        print()

        all_pairs = list(itertools.combinations(range(p), 2))
        pairs = random.sample(all_pairs, p // 2)
        pairs_str = ", ".join([f"({pair[0] + 1}, {pair[1] + 1})" for pair in
pairs])
        print(f"Пусть выбраны пары: {pairs_str}")
        print("Родители | Потомки | Значение целевой функции для потомков")
        for j, pair in enumerate(pairs):
            p1 = population[pair[0]]
            p2 = population[pair[1]]
            children = make_children(p1, p2)
            unique_children = []
```

```
            for child in children:
                if child not in population:
                    unique_children.append(child)

            population += unique_children
        print()
        population.sort(key=lambda route: route_length(route, matrix))
        population = population[:p]
        print()
    print(f"Финальное поколение")
    lengths = [route_length(route, matrix) for route in population]
    final_length = lengths[0]
    final_sum = sum(lengths)
    final_average = final_sum / len(lengths)
    print("Код   | Значение целевой функции")
    for i, (code, length) in enumerate(zip(population, lengths), 1):
        print(f"{''.join(map(lambda x: str(x + 1), code))} | {length}")
    print()
    return population[0], route_length(population[0], matrix)


if __name__ == "__main__":
    c = 5
    matrix = [[0, 1, 1, 5, 3], [1, 0, 3, 1, 5], [1, 3, 0, 11, 1], [5, 1, 11,
0, 1], [3, 5, 1, 1, 0]]
    p = 4
    g = 3
    print()
    result, length = generation(c, matrix, p, g)
    print(f"Таким образом после {g} итераций значение целевой функции для
лучшего решения изменилось с {first_length} на {final_length}, среднее
значение изменилось с {first_average} до {final_average}, а общее качество с
{first_sum} до {final_sum}.")
```

Вывод программы

Поколение 1
Код | Значение целевой функции
45321 | 11
13524 | 13
45123 | 19
34521 | 19

Пусть выбраны пары: (1, 3), (1, 2)
Родители | Потомки | Значение целевой функции для потомков
45|32|1 | 45|12|3 | 19
45|12|3 | 45|32|1 | 11
|45|321 | |13|524 | 13
|13|524 | |45|321 | 11

Поколение 2
Код | Значение целевой функции | Вероятность участия в размножении
45321 | 11 | 0.33288409703504046
13524 | 13 | 0.2816711590296496
45123 | 19 | 0.192722371967655
34521 | 19 | 0.192722371967655

Пусть выбраны пары: (1, 4), (2, 4)
Родители | Потомки | Значение целевой функции для потомков
|453|21 | |345|21 | 19
|345|21 | |453|21 | 11
Потомок 1 мутировал: 35421
13|52|4 | 41|52|3 | 27
34|52|1 | 13|52|4 | 13

Поколение 3
Код | Значение целевой функции
35421 | 5
45321 | 11
13524 | 13
45123 | 19

Пусть выбраны пары: (1, 2), (1, 3)
Родители | Потомки | Значение целевой функции для потомков
|354|21 | |453|21 | 11
|453|21 | |354|21 | 5
|354|21 | |135|42 | 5
|135|24 | |354|21 | 5

Финальное поколение
Код | Значение целевой функции
35421 | 5
13542 | 5

45321 | 11
13524 | 13

Таким образом после 3 итераций значение целевой функции для лучшего решения изменилось с 11 на 5, среднее значение изменилось с 15.5 до 8.5, а общее качество с 62 до 34.