

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования «Национальный исследовательский

университет ИТМО»

Факультет Программной инженерии и компьютерной техники

02.05.2024

Лабораторная работа №6

Методы оптимизации

Вариант 9

Раевский Григорий, группа Р3221

Содержание

Задание	2
Решение	2
Выводы	6
Код	6

Задание

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

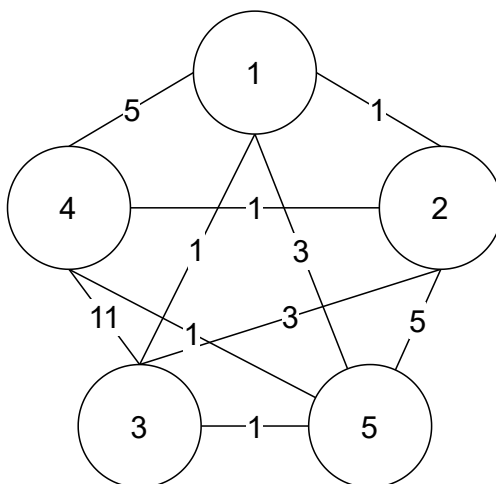
За целевую функцию принять сумму расстояний между городами. Размер популяции $N = 4$. Оператор мутации представляет собой случайную перестановку двух чисел в геноме, которые выбираются случайно. Вероятность мутации 0.01.

Решение

Значение целевой функции — стоимость всей поездки. Одно из оптимальных решений - 1-2-4-5-3-1 (длиной 5).

При этом ищем *min* целевой функции.

Стоимости перемещения между городами:



Исходная популяция (случайно сгенерированные последовательности):

№	Последовательность	Значение функции
1	1-2-3-4-5-1	19
2	3-2-1-4-5-3	11
3	3-4-1-2-5-3	23
4	1-4-3-2-5-1	27

Среднее значение $\frac{19+11+23+27}{4} = 20$.

Для первого скрещивания выберем 4 и 2 последовательности (14325, 32145). Первая граница будет идти после 1 числа, а вторая — после 3 (1|43|25, 3|21|45). Произведем замену середин: *|21|** и *|43|**. Произведем вставку чисел из родительской перестановки. Для первого числа: берем 3, 2 пропускаем (есть внутри границ), берем 5, 1 пропускаем (есть внутри границ), берем 4. Для второго числа действуем аналогично. В итоге получим 32154 и 14352.

Для последовательности 3-2-1-5-4-3 значение функции - 19. Для последовательности 1-4-3-5-2-1 - 23. Важно отметить, что здесь и далее последовательность формируется на основе числа: 12345 "то же самое, что и" 1-2-3-4-5-1, просто где-то удобнее записывать последовательность, а где-то число.

Для второго скрещивания выберем 1 и 3 последовательности (12345, 34125). Первая граница будет идти после 2 числа, а вторая — после 4 (12|34|5, 34|12|5). Произведем замену середин: *|12|** и *|34|**. Произведем вставку чисел из родительской перестановки. В итоге получим 45123 и 25341.

Для последовательности 4-5-1-2-3-4 значение функции - 19. Для последовательности 2-5-3-4-1-2 - 23.

Для каждого из потомков "число мутаций" 0.84, 0.02, 0.45, 0.11 - поэтому мутацию не проводим. Мутация проводится лишь тогда, когда число от 0 до 1 (с шагом 0.01) меньше или равно вероятности мутации из условия (0.01).

Тогда получаем таблицу родителей и потомков:

№	Родители	Потомки	Значение целевой функции потомка
4	1-4-3-2-5-1	3-2-1-5-4-3	19
2	3-2-1-4-5-1	1-4-3-5-2-1	23
1	1-2-3-4-5-1	4-5-1-2-3-4	19
3	3-4-1-2-5-3	2-5-3-4-1-2	23

Популяция первого поколения после отсеивания худших особей:

№	Последовательность	Значение функции
1 (2)	3-2-1-4-5-3	11
2 (1)	1-2-3-4-5-1	19
3 (н)	3-2-1-5-4-3	19
4 (н)	4-5-1-2-3-4	19

Среднее значение $\frac{11+19+19+19}{4} = 17$.

Теперь перейдем ко второму поколению.

Для первого скрещивания выберем 3 и 1 последовательности (32154, 32145). Первая граница будет идти после 3 числа, а вторая — после 4 (321|5|4, 321|4|5). Произведем замену середин: ***|4|* и ***|5|*. Произведем вставку чисел из родительской перестановки. В итоге получим 32145 и 32154.

Для последовательности 3-2-1-4-5-3 значение функции - 11. Для последовательности 3-2-1-5-4-3 - 19.

Для второго скрещивания выберем 2 и 4 последовательности (12345, 45123). Первая граница будет идти после 1 числа, а вторая — после 2 (1|2|345, 4|5|123). Произведем замену середин: *|5|*** и *|2|***. Произведем вставку чисел из родительской перестановки. В итоге получим 35412 и 12345.

Для последовательности 3-5-4-1-2-3 значение функции - 11. Для последовательности 1-2-3-4-5-1 - 19.

Для каждого из потомков "число мутаций" 0.36, 0.93, 0.11, 0.40 - поэтому мутацию не проводим.

Тогда получаем таблицу родителей и потомков:

№	Родители	Потомки	Значение целевой функции потомка
3	3-2-1-5-4-3	3-2-1-4-5-3	11
1	3-2-1-4-5-3	3-2-1-5-4-3	19
2	1-2-3-4-5-1	3-5-4-1-2-3	11
4	4-5-1-2-3-4	1-2-3-4-5-1	19

Популяция второго поколения после отсека худших особей:

№	Последовательность	Значение функции
1 (н)	3-2-1-4-5-3	11
2 (н)	3-5-4-1-2-3	11
3 (1)	3-2-1-4-5-3	11
4 (н)	3-2-1-5-4-3	19

Среднее значение $\frac{11+11+11+19}{4} = 13$.

Теперь перейдем к третьему поколению.

Для первого скрещивания выберем 2 и 3 последовательности (35412, 32145). Первая граница будет идти после 1 числа, а вторая — после 3 (3|54|12, 3|21|45). Произведем замену середин: *|21|** и *|54|**. Произведем вставку чисел из родительской перестановки. В итоге получим 42135 и 15432.

Для последовательности 4-2-1-3-5-4 значение функции - 5. Для последовательности 1-5-4-3-2-1 - 19.

Для второго скрещивания выберем 1 и 4 последовательности (32145, 32154). Первая граница будет идти после 1 числа, а вторая — после 2 (3|2|145, 3|2|154). Произведем замену середин: *|2|*** и *|2|***. Произведем вставку чисел из родительской перестановки. В итоге получим 12453 и 12543.

Для последовательности 1-2-4-5-3-1 значение функции - 5. Для последовательности 1-2-5-4-3-1 - 19.

Для каждого из потомков "число мутации 0.61, 0.09, 0.46, 0.70 - поэтому мутацию не проводим.

Тогда получаем таблицу родителей и потомков:

№	Родители	Потомки	Значение целевой функции потомка
2	3-5-4-1-2-3	4-2-1-3-5-4	5
3	3-2-1-4-5-3	1-5-4-3-2-1	19
1	3-2-1-4-5-3	1-2-4-5-3-1	5
4	3-2-1-5-4-3	1-2-5-4-3-1	19

Популяция третьего поколения после отсеечения худших особей:

№	Последовательность	Значение функции
1 (н)	4-2-1-3-5-4	5
2 (н)	1-2-4-5-3-1	5
3 (1)	3-2-1-4-5-3	11
4 (н)	3-5-4-1-2-3	11

Среднее значение $\frac{5+5+11+11}{4} = 8$.

Как мы видим, в популяции 3 поколения присутствуют 2 оптимальных особи. Ни в одной из итераций генетического алгоритма не произошло мутаций.

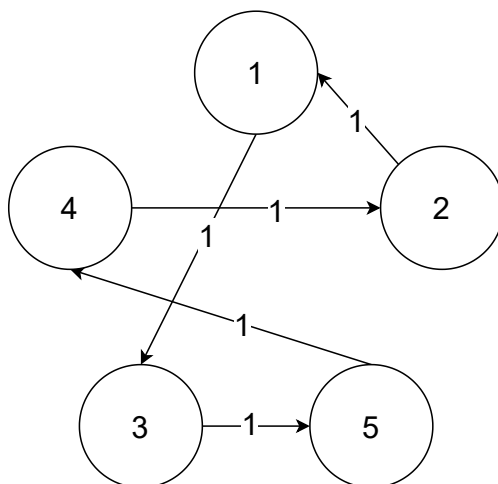
Для **ПРИМЕРА** работы мутации возьмем особь 4-2-1-3-5-4 (42135). Тогда если для нее сработает оператор мутации, мы переставим вторую и четвертую цифры местами и получим особь 43125 (ее значение функции для обхода 4-3-1-2-5-4 - 19).

Выводы

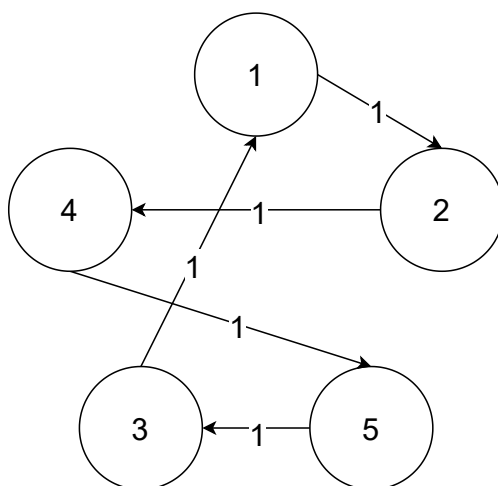
Генетический алгоритм постепенно улучшал среднее значение функции и наконец нашел 2 оптимальных решения после отсеечения худших особей в 3 поколении.

В итоге генетический алгоритм на 3 итерации получил 2 способа обхода:

1. Обход по маршруту 1 из 3 поколения, 4-2-1-3-5-4;



2. Обход по маршруту 2 из 3 поколения, 1-2-4-5-3-1:



Код

```
1 import random
2
3 cities = [
4     [0, 1, 1, 5, 3],
```

```

5     [1, 0, 3, 1, 5],
6     [1, 3, 0, 11, 1],
7     [5, 1, 11, 0, 1],
8     [3, 5, 1, 1, 0]
9 ]
10
11
12 def calculate_cost(path):
13     cost = 0
14     for i in range(len(path)):
15         cost += cities[path[i] - 1][path[(i + 1) % len(path)] - 1]
16     return cost
17
18
19 def create_initial_population(size, num_cities):
20     population = []
21     for _ in range(size):
22         path = list(range(1, num_cities + 1))
23         random.shuffle(path)
24         population.append((path, calculate_cost(path), "Init population"))
25     return population
26
27
28 def crossover(parent1, parent2):
29     n = len(parent1[0])
30     point1, point2 = sorted(random.sample(range(1, n - 1), 2))
31     middle1 = parent1[0][point1:point2]
32     middle2 = parent2[0][point1:point2]
33
34     def make_child(middle, parent):
35         child = [-1] * n
36         child[point1:point2] = middle
37         added = set(middle)
38         fill_idx = point2
39         for city in parent[0]:
40             if city not in added:
41                 while child[fill_idx] != -1:

```

```

42         fill_idx = (fill_idx + 1) % n
43         child[fill_idx] = city
44         added.add(city)
45     return child
46
47     child1 = make_child(middle2, parent1)
48     child2 = make_child(middle1, parent2)
49     return (child1, calculate_cost(child1), f"Child of {parent1[0]}, {parent2[0]}"), \
50           (child2, calculate_cost(child2), f"Child of {parent1[0]}, {parent2[0]}")
51
52
53 def mutate(path, mutation_rate=0.01):
54     if random.random() < mutation_rate:
55         idx1, idx2 = random.sample(range(len(path[0])), 2)
56         path[0][idx1], path[0][idx2] = path[0][idx2], path[0][idx1]
57         path = (path[0], calculate_cost(path[0]), f"Mutated from {path[0]}")
58     return path
59
60
61 def genetic_algorithm(cities, population_size=4, optimal_cost=5):
62     num_cities = len(cities)
63     population = create_initial_population(population_size, num_cities)
64     print("Initial Population:")
65     for path in population:
66         print(path[0], path[1], path[2])
67
68     generation = 0
69     while True:
70         print("\nGeneration", generation)
71         new_population = []
72         costs = []
73
74         for i in range(0, len(population), 2):
75             parent1, parent2 = population[i], population[i + 1]
76             child1, child2 = crossover(parent1, parent2)
77             child1 = mutate(child1)
78             child2 = mutate(child2)

```



```

79         new_population.extend([child1, child2])
80         costs.append(child1[1])
81         costs.append(child2[1])
82
83     print("Children and Costs:")
84     for child in new_population:
85         print(child[0], child[1], child[2])
86
87     all_individuals = population + new_population
88     all_individuals = sorted(all_individuals, key=lambda x: x[1])[:population_size]
89
90     print("Updated Population:")
91     for ind in all_individuals:
92         print(ind[0], ind[1], ind[2])
93
94     population = all_individuals
95     if any(ind[1] == optimal_cost for ind in population):
96         break
97
98     generation += 1
99
100
101 genetic_algorithm(cities)

```