Выполнила Касьяненко Вера (P3220)

Лабораторная №6

Вариант 5

Условие:

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	4	5	3	8
Город 2	4	0	7	6	8
Город 3	5	7	0	7	9
Город 4	3	6	7	0	9
Город 5	8	8	9	9	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.

Решение вручную:

Исходная популяция

№ строки	Код	Значение целевой функции
1	31254	33
2	32451	35
3	45132	35
4	12345	35

Пусть для скрещивания были выбраны следующие пары: (1,2) и (2,3)
В результате были получены потомки:

№ строки	Родители	Потомки	Значение целевой функции для потомков
1	|312|54	|324|15	33
2	|324|51	|312|45	33
3	32|45|1	52|13|4	33
4	45|13|2	32|45|1	35

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

№ строки	Код	Значение целевой функции	Вероятность участия в процессе размножения
1	31254	33	0.25
2	32415	33	0.25
3	31245	33	0.25
4	52134	33	0.25

Пусть для получения второго поколения были выбраны следующие пары строк: (2,3) и (2,4).
В результате были получены потомки:

№ строки	Родители	Потомки	Значение целевой функции для потомков
1	32|415|	13|245|	35
2	31|245|	32|415|	33
3	32|41|5	52|13|4	33
4	52|13|4	35|41|2	32

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

№ строки	Код	Значение целевой функции
1	35412	32
2	31254	33
3	32415	33
4	31245	33

Пусть для получения третьего поколения были выбраны следующие пары строк: (1,4) и (2,3).
В результате были получены потомки:

№ строки	Родители	Потомки	Значение целевой функции для потомков
1	35|41|2	13|24|5	35
2	31|24|5	53|41|2	31
3	312|54|	432|15|	35
4	324|15|	321|54|	35

Популяция второго поколения после отсечении худших особей приняла вид:

№ строки	Код	Значение целевой функции
1	53412	31
2	35412	32
3	31254	33
4	32415	33

Таким образом после 3 поколений значение целевой функции для лучшего решения изменилось с 33 на 31, среднее значение изменилось с 34.5 до 32.25, а общее качество с 138 до 129.

Код:

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

31254 | 33

32451 | 35

45132 | 35

12345 | 35

Пусть выбраны пары: (1, 2), (2, 3)

Родители | Потомки | Значение целевой функции для потомков

|312|54| |324|15 | 33

|324|51| |312|45 | 33

32|45|1 | 52|13|4 | 33

45|13|2 | 32|45|1 | 35

Поколение 2

Код | Значение целевой функции | Вероятность участия в размножении

31254 | 33 | 0.25

32415 | 33 | 0.25

31245 | 33 | 0.25

52134 | 33 | 0.25

Пусть выбраны пары: (2, 3), (2, 4)

Родители | Потомки | Значение целевой функции для потомков

32|415| | 13|245| | 35

31|245| | 32|415| | 33

32|41|5 | 52|13|4 | 33

52|13|4 | 35|41|2 | 32

Поколение 3

Код | Значение целевой функции

35412 | 32

31254 | 33

32415 | 33

31245 | 33

Пусть выбраны пары: (1, 4), (2, 3)

Родители | Потомки | Значение целевой функции для потомков

35|41|2 | 13|24|5 | 35

31|24|5 | 53|41|2 | 31

312|54| | 432|15| | 35

324|15| | 321|54| | 35

Финальное поколение

Код | Значение целевой функции

53412 | 31

35412 | 32

31254 | 33

32415 | 33

Таким образом после 3 итераций значение целевой функции для лучшего решения изменилось с 33 на 31, среднее значение изменилось с 34.5 до 32.25, а общее качество с 138 до 129.