МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика, искусственный интеллект и системы управления»
Кафедра «Системы обработки информации и управления»

ОТЧЕТ

Лабораторная работа № 4

по курсу «Методы поддержки принятия решений»

Решение оптимизационных задач с помощью
генетических алгоритмов

ИСПОЛНИТЕЛЬ: Терентьев В.О.
ФИО

группа ИУ5-73Б _____
подпись

"__"_____2021 г.

ПРЕПОДАВАТЕЛЬ: Терехов В.И.
ФИО

_____
подпись

"__"_____2021 г.

Москва - 2021

1. Цель работы

Целью лабораторной работы является углубление и закрепление теоретических знаний, полученных на лекциях, приобретение практических навыков самостоятельной работы при решении оптимизационных задач больших размерностей с помощью генетических алгоритмов.

В процессе выполнения лабораторной работы по теме «Решение оптимизационных задач с помощью генетических алгоритмов» на примере задачи поиска кратчайшего пути для информационного пакета (сообщения) в компьютерной сети студенты решают следующие задачи (задания):

- описывают предметную область;
- определяют исходные данные задачи;
- формулируют задачу и исходные данные в терминах генетических алгоритмов;
- определяют последовательность работы генетического алгоритма;
- разрабатывают компьютерную программу;
- исследуют работу генетического алгоритма и полученное решение.

2. Задание

1. Сформулировать задачу и описать исходные данные в терминах генетических алгоритмов.
2. Разработать программу, которая осуществляет поиск кратчайшего пути для информационного пакета (сообщения) в компьютерной сети с помощью генетического алгоритма.
3. При проведении серии экспериментов (не меньше 10) по исследованию работы генетического алгоритма программа должна позволять пользователю задавать топологию сети (пропускные способности каналов связи), содержащей не менее 10 компьютеров (серверов), а также указывать компьютер-отправитель и компьютер-получатель. Должны отображаться все решения (хромосомы) одного поколения до и после применения каждого оператора (скрещивания, селекции, редукции и мутации). Переход к следующему поколению должен осуществляться: в автоматическом режиме в соответствии с заданным критерием; в ручном режиме.

3. Используемые технологии

Язык программирования Python, библиотека NumPy, библиотека для визуализации данных Matplotlib, Jupyter Notebook.

4. Блок-схема генетического алгоритма

5. Выполнение работы

In [1]:
```
import numpy as np
import random
import matplotlib.pyplot as plt
```

Функция приспособленности:

In [2]:
```
def fitness(chromosome):
    y = network[departure][chromosome[0]]
    y += network[destination][chromosome[-1]]
    for i in range(1, len(chromosome)):
        y += network[chromosome[i-1]][chromosome[i]]
    return y
```

Селекция:

In [3]:
```
def selection(population):
    return sorted(population, key=lambda x:fitness(x))[:population_size//2]
```

Скрещивание:

In [4]:
```
def crossover(parent1, parent2, k=2):
    child1 = parent2[:]
    child2 = parent1[:]
    for i in range(k):
        tmp = np.random.randint(1, len(child1)-1)
        for i in range(tmp):
            child1[i] = parent1[i]
            child2[i] = parent2[i]
    return child1, child2
```

Мутация:

In [5]:
```
def mutation(chromosome, prob=0.01):
    for i in range(len(chromosome)):
        if np.random.random() < prob:
            mut_gen = chromosome[i]
            while chromosome[i] == mut_gen:
                mut_gen = np.random.randint(n_computers)
            chromosome[i] = mut_gen
    return chromosome
```

Исходные данные:

In [6]:
```
departure = 0
destination = 9
n_computers = 10
population_size = 100
n_generations = 50
mutation_probability = 0.01

user_network = [[0,                 110, 78, 71, 67, 139, 29, 234, 65, 1000],
                [0,0,               155, 72, 1000, 87, 1000, 1000, 72, 1000],
                [0,0,0,             1000, 1000, 1000, 1000, 1000, 1000, 51],
                [0,0,0,0,           345,   58, 1000,   90,  145, 1000],
                [0,0,0,0,0,         1000,    4, 1000,   32,  160],
                [0,0,0,0,0,0,       1000, 1000, 1000,   87],
                [0,0,0,0,0,0,0,     1000,   67, 1000],
                [0,0,0,0,0,0,0,0,   140,  159],
                [0,0,0,0,0,0,0,0,0, 87],
                [0,0,0,0,0,0,0,0,0,0,0]]
```

Построение топологии сети (пропускные способности каналов связи):

In [7]:
```
def build_network():
    network = []
    for i in range(n_computers):
        network.append([])
        for j in range(n_computers):
            network[i].append(0)
    for i in range(n_computers):
        for j in range(i+1, n_computers):
            network[i][j] = np.random.randint(200)
    for i in range(n_computers):
        for j in range(i):
            network[i][j] = network[j][i]
    for i in range(len(network)):
        for j in range(len(network[i])):
            if network[i][j] >= 100:
                network[i][j] = 1000
                network[j][i] = 1000
    return network
```

Начальная популяция:

In [8]:
```
def initial_population():
    chromosomes = []
    for k in range(population_size):
        chromosomes.append([])
        for i in range(n_computers-2):
            chromosomes[k].append(np.random.randint(n_computers))
    return chromosomes
```

Пропускные способности каналов связи:

In [9]:
```
network = build_network()
network
```

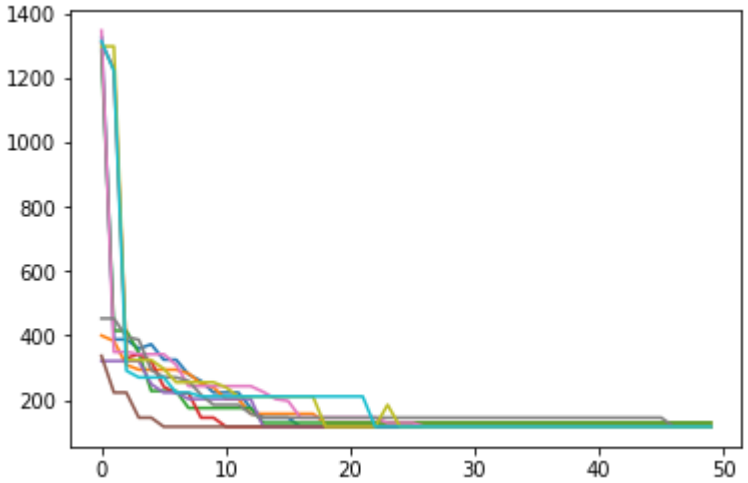
Out[9]:
```
[[0, 47, 50, 1000, 92, 1000, 1000, 1000, 1000, 82],
 [47, 0, 77, 1000, 85, 69, 1000, 1000, 14, 1000],
 [50, 77, 0, 1000, 1000, 61, 1000, 1000, 23, 1000],
 [1000, 1000, 1000, 0, 1000, 80, 1000, 41, 1000, 1000],
 [92, 85, 1000, 1000, 0, 1000, 1000, 1000, 1000, 53],
 [1000, 69, 61, 80, 1000, 0, 65, 1000, 1000, 1000],
 [1000, 1000, 1000, 1000, 1000, 65, 0, 1000, 94, 80],
 [1000, 1000, 1000, 41, 1000, 1000, 1000, 0, 48, 80],
 [1000, 14, 23, 1000, 1000, 1000, 94, 48, 0, 56],
 [82, 1000, 1000, 1000, 53, 1000, 80, 80, 56, 0]]
```

Генетический алгоритм:

In [10]:
```
for i in range(10):
    print('\n\tЭксперимент: ', i+1)
    chromosomes = initial_population()
    best_start = min(chromosomes, key=lambda chromosome: fitness(chromosome))
    print('Поколение 0 Лучший представитель {} Результат {}'.format(best_start, fitness(
    res = []
    log = []
    for k in range(n_generations):
        selected = selection(chromosomes)

        for i in range((population_size-len(selected))//2):
            parents = random.sample(selected, 2)
            child1, child2 = crossover(parents[0], parents[1])
            selected.append(child1)
            selected.append(child2)

        for i in range(len(selected)):
            selected[i] = mutation(selected[i])

        chromosomes = selected

        best_chromosome = min(chromosomes, key=lambda chromosome: fitness(chromosome))
        log.append(fitness(best_chromosome))
        #print('Поколение {} Лучший представитель {} Результат {}'.format(k, best_chromo
        res.append(best_chromosome)
    plt.plot(log)
    best_exp = min(res, key=lambda chromosome: fitness(chromosome))
    print('Поколение {} Лучший представитель {} Результат {}'.format(n_generations, best
```

```
        Эксперимент:  1
Поколение 0 Лучший представитель [6, 8, 2, 8, 2, 0, 9, 7] Результат 1455
Поколение 50 Лучший представитель [1, 8, 8, 8, 8, 8, 8, 8] Результат 117

        Эксперимент:  2
Поколение 0 Лучший представитель [1, 5, 5, 1, 1, 5, 8, 9] Результат 1310
Поколение 50 Лучший представитель [0, 0, 2, 2, 8, 8, 8, 9] Результат 129

        Эксперимент:  3
Поколение 0 Лучший представитель [0, 2, 8, 1, 5, 0, 2, 8] Результат 1285
Поколение 50 Лучший представитель [0, 2, 2, 2, 2, 2, 2, 8] Результат 129

        Эксперимент:  4
Поколение 0 Лучший представитель [1, 8, 2, 2, 5, 5, 9, 0] Результат 1309
Поколение 50 Лучший представитель [0, 1, 1, 8, 8, 8, 8, 8] Результат 117

        Эксперимент:  5
Поколение 0 Лучший представитель [1, 5, 2, 0, 2, 8, 9, 9] Результат 356
Поколение 50 Лучший представитель [1, 1, 1, 1, 8, 8, 9, 9] Результат 117

        Эксперимент:  6
Поколение 0 Лучший представитель [2, 8, 8, 9, 8, 1, 4, 4] Результат 337
Поколение 50 Лучший представитель [0, 0, 0, 1, 8, 9, 9, 9] Результат 117

        Эксперимент:  7
Поколение 0 Лучший представитель [1, 0, 1, 1, 5, 2, 1, 6] Результат 1428
Поколение 50 Лучший представитель [0, 0, 1, 1, 1, 8, 8, 8] Результат 117

        Эксперимент:  8
Поколение 0 Лучший представитель [9, 7, 9, 3, 5, 5, 1, 8] Результат 1461
Поколение 50 Лучший представитель [1, 8, 8, 8, 8, 8, 8, 9] Результат 117

        Эксперимент:  9
Поколение 0 Лучший представитель [1, 8, 3, 5, 1, 0, 2, 8] Результат 1386
Поколение 50 Лучший представитель [0, 1, 1, 1, 1, 1, 8, 8] Результат 117
```

```
        Эксперимент:  10
Поколение 0 Лучший представитель [2, 2, 8, 1, 9, 4, 0, 9] Результат 1314
Поколение 50 Лучший представитель [0, 0, 0, 0, 1, 1, 8, 9] Результат 117
```



In [ ]:

6. Выводы

В результате выполнения лабораторной работы были получены навыки решения оптимизационных задач больших размерностей с помощью генетических алгоритмов.