**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по учебной практике**

**по теме «Генетические алгоритмы»**

Тема: **Задача о кратчайшем пути.**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 0382 |  | Злобин А.С. Андрющенко К.С. |
| Преподаватель |  | Жангиров Т. Р. |

Санкт-Петербург

2022

## Цель работы

 Разработать и реализовать программу, решающую одну из оптимизационных задач (файл “Варианты”) с использованием генетических алгоритмов (ГА), а также визуализирующая работу алгоритма.

**Задача**

## Вариант 9

Задача о кратчайшем пути (задача поиска самого короткого [пути](https://ru.wikipedia.org/wiki/%D0%9F%D1%83%D1%82%D1%8C_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)) (цепи) между двумя точками (вершинами) на [графе](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), в которой минимизируется сумма [весов рёбер](https://ru.wikipedia.org/wiki/%D0%92%D0%B7%D0%B2%D0%B5%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84), составляющих путь).

Необходимо реализовать вариант о нахождении кратчайшего пути между всеми парами вершин в графе, т.е. вывести матрицу кратчайших путей  
Входные данные:

- Список вершин

- Список ребер

## Выполнение работы

Используемый язык программирования: Python 3.

## Введение

**Выполнение работы**

1. Формирование прототипа GUI и выбор метода решения задачи.

Для реализации графического интерфейса будет использоваться фреймворк Kivy.

В качестве входных данных используется список вершин и список рёбер. Пользователь будет иметь возможность задать граф, для которого будут вычисляться кратчайшие пути, с помощью матрицы смежности или же «рисуя» граф в представленном окне с помощью инструментов «создать вершину», «создать направленное ребро» и «создать ненаправленное ребро». Так же будет возможность выполнить элемент по эпохам или же перейти сразу к конечному результату. Каждая эпоха будет отображаться в виде графа, на котором цветом выделены возможные пути (особи поколения). Наиболее оптимальный маршрут (лучшая особь) будет отмечаться другим цветом. При нажатии на кнопку настроек параметров алгоритма будет возможность изменить количество эпох и численность популяции.

Прототип графического интерфейса представлен в приложении 1.

Описание генетического алгоритма.

Первым делом нам нужно формализовать задачу и определиться со способом хранения информации в хромосомах. Здесь могут быть самые разные варианты, однако остановимся на следующем. Одна хромосома описывает все возможные маршруты от исходной точки до всех остальных вершин графа.

Узлы, для удобства, пронумерованы от 0 до n - 1 (всего n узлов). Каждый маршрут представлен отдельным списком. Так как длина пути изначально неизвестна, то он заканчивается тогда, когда встречается вершина с номером назначения.

Для хранения графа будем использовать матрицу смежности, которая на пересечении строк и столбцов содержит значения весов дуг между соответствующими вершинами. Те клетки, что не содержат значений, означают отсутствие связей. По идее, здесь можно прописать бесконечные величины, означающие бесконечно длинные маршруты.

Следующим шагом в программе нужно определить, как вычислять приспособленность отдельных особей, как выполнять скрещивание и мутации. Начнем с функции приспособленности. Она должна возвращать кортеж значений, которые являются суммой длин путей до вершин графа.

В функции мутации перебираем списки маршрутов особи и для каждого выполняем перемешивание индексов с некоторой заданной вероятностью.

В качестве функции скрещивания будем использовать алгоритм упорядоченного скрещивания.

1. Начало реализации

## Вывод