МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по учебной практике

Тема: Генетические алгоритмы и PSA

Студенты гр. 2382 _____ Федоров М.В.
Бухарин М.А.
Муравин Е.Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2024

Цель работы.

Изучить генетические алгоритмы.

Задание.

Задача о выделении области:

Дано N точек в двумерном евклидовом пространстве. Для каждой i-й точки задана метка: 0 или 1. Необходимо найти координаты левого нижнего угла, высоту и ширину прямоугольника, чтобы полученный прямоугольник содержал как можно больше точек с меткой 1, и как можно меньше точек с меткой 0.

Основные теоретические положения.

Задача ГА - найти оптимальное решение некоторой задачи, путем развития популяции потенциальных решений называемых индивидуумами. Решения итеративно оцениваются и используются для создания следующего поколения.

Генотип - описание одного индивидуума, набор генов сгруппированных в хромосому. При скрещивании хромосома содержит гены своих родителей.

Популяция - множество индивидуумов, то есть потенциальных решений, которые хранит генетический алгоритм. Популяция всегда отображает текущее поколение.

Функция приспособленности/целевая функция - функция, которую необходимо оптимизировать в рамках решаемой задачи. Является функцией от индивидуума, и показывает качество решения представленным хромосомой. На каждой итерации ГА рассчитывает приспособленность индивидуума для формирования нового поколения.

Отбор - формирование множества индивидуумов, которые будут использоваться для формирования следующего поколения. Отбор основывается на приспособленности индивидуума, и чем он лучше, тем больше вероятность его отобрать. Причем хромосомы дающие низкое значение приспособленности не исключают возможность отбора. Таким образом приспособленность популяции увеличивается.

Выполнение работы.

Для представления точек будет использоваться класс Pointer. Это класс содержит три поля, которые соответствуют характеристикам точки: координаты и метка. Так же реализованы сетеры для каждого поля.

Класс прямоугольника Rectangle задаётся двумя точками: левый верхний угол, правый нижний. Так же реализованы сетеры для каждого поля.

Целевая функция будет подсчитывать количество точек внутри прямоугольника. И возвращать кортеж значений: (num_point1 * n — num_point0 * m, s) — где num_point1, num_point0, это количество точек с определёнными марками; n, m, некоторые числа(определим позже); s — площадь прямоугольника(этот параметр нужен, чтоб новый индивид не увеличивался в размерах).

Метод отбора не был выбран, поскольку хочется сравнить разные методы и найти эффективный. Но предположительно будет использоваться отбор усечением.

В качестве функции скрещивания будет использоваться одноточечное скрещивание. Новые прямоугольники будут получаться путём обмена точками, которые их задают.

В качестве мутации будет выбрана функция, которая увеличивает размер прямоугольника на n(позже определим точное значение).

Графический интерфейс

Графический интерфейс был разработан на языке программирования Python с использованием библиотек *customtkinter*, *matplotlib*, *PIL*, *tkinter*.

Основная часть интерфейса реализована с помощью библиотеки *customtkinter*, с помощью библиотеки *matplotlib* реализовано пошаговое отображение покрывающих плоскость прямоугольников и множество точек на графике. Библиотека *PIL* использовался для обработки изображений, установленных на кнопках взаимодействия. С помощью *tkinter* реализован функционал открытия **.csv* файла, для считывания данных.

Примеры работы *GUI* представлены на рис. 1 - рис. 4.

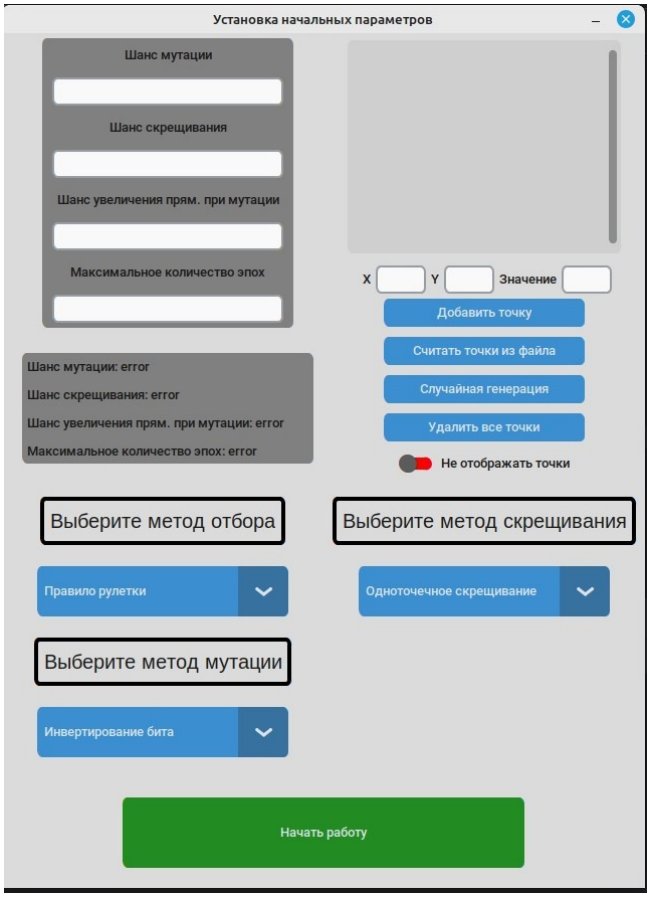

Рисунок 1 — Установка начальных параметров

Рисунок 2 — Множество точек

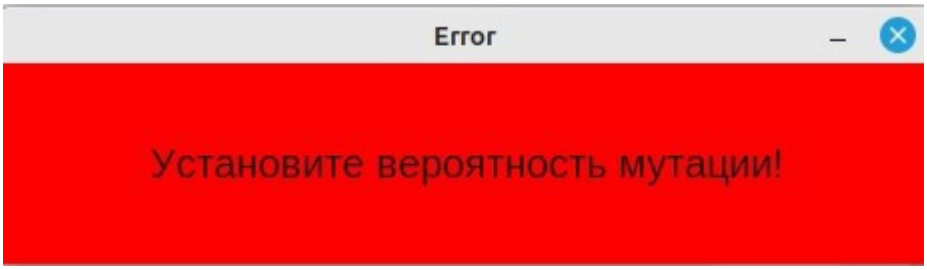

Рисунок 3 — Ошибка при запуске

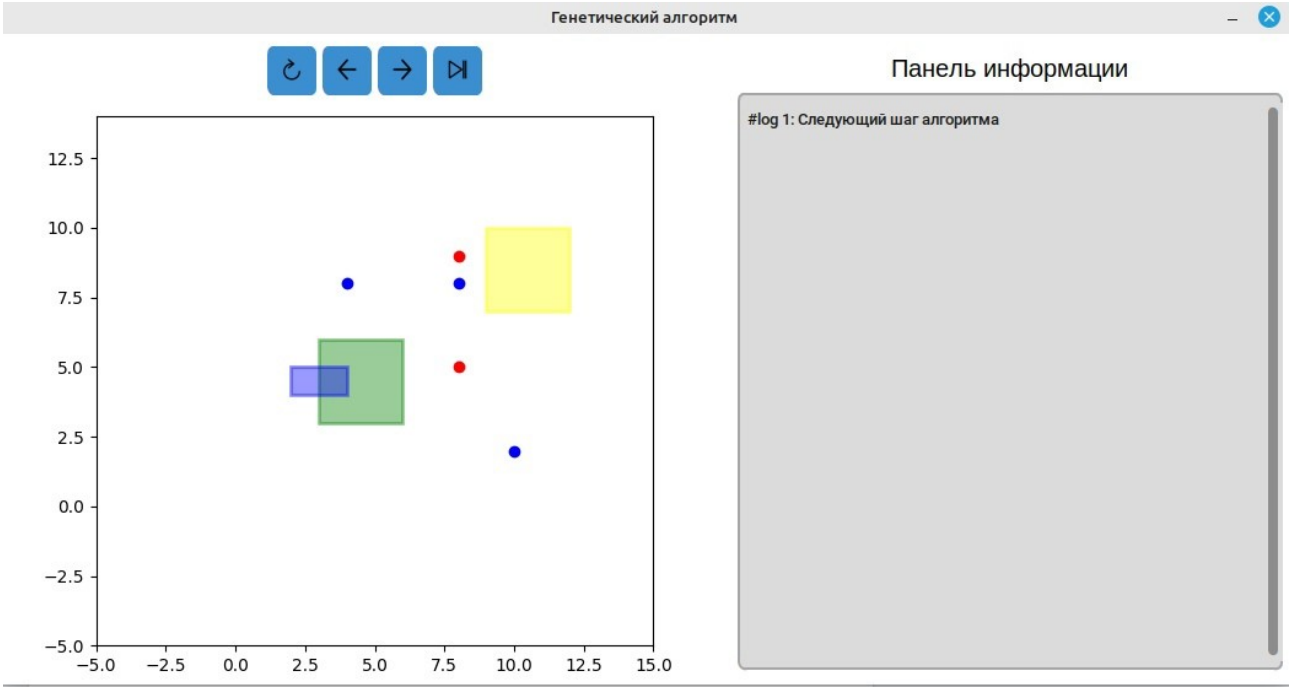

Рисунок 4 — Отображение работы алгоритма

Основные классы.

Для представления точек будет использоваться класс Pointer. Это класс содержит три поля, которые соответствуют характеристикам точки: координаты и метка. Так же реализованы сетеры для каждого поля.

Класс прямоугольника Rectangle задаётся двумя точками: левый верхний угол, правый нижний. Так же реализованы сетеры для каждого поля.

Добавлены структуры ParamFitness, ParamMutation, ParamProbability, ParamGeneticAlgorithm, PairRectangleInt.

ParamFitness отвечает за параметры функции приспособленности, штрафы и поощрения за вхождение точек в прямоугольник.

ParamMutation отвечает за параметры мутации, шанс мутации, шаня увеличить прямоугольник при мутации, шанс уменьшить прямоугольник.

ParamProbability включает в себя структуру ParamMutation и вероятность скрещивания.

ParamGeneticAlgorithm отвечает за параметры алгоритма и влючает структуры ParamProbability, ParamFitness и количество особей в популяции.

PairRectangleInt пара прямоугольника и значения его функции приспособленности.

Генетический алгоритм

Имеется класс Point и Rectangle, в которых заданы параметры точки и прямоугольника соответственно. Прямоугольник задается точками левого верхнего и правого нижнего углов.

Также имеются классы ParamFitness, ParamProbability и ParamMutatuion, которые содержат соответствующие параметры. Класс ParamGeneticAlgorithm содержит в себе все эти классы.

Класс PairRectangleInt содержит прямоугольник и значение его приспособленности.

Был написан ряд основных функций, которые реализуют генетический алгоритм.

first_generation() - принимает в себя список точек и количество особей в первом поколении. Далее случайным образом для каждого прямоугольника выбираются по 2 координаты x и y точек из списка точек, также случайно выбирается смещение от этих точек в диапазоне от -5 до 5, и по этим точкам строится очередной прямоугольник. Функция возвращает список построенных прямоугольников.

mutation() - функция мутации одной особи, принимает в себя прямоугольник. Случайным образом выбирается значение от 0 до половины длины стороны прямоугольника + 1 для каждой координаты (x и y). Затем с определенным шансом происходит увеличение, либо уменьшение прямоугольника на выбранное ранее значение для каждой координаты. Функция возвращает новый прямоугольник.

crossing() - функция скрещивания. Был выбран метод скрещивания — имитация двоичного скрещивания. Вычисляется значение beta по следующей формуле: случайным образом от 0 до 1 выбирается вещественное число u и

- если $u \leqslant 0.5$: $\beta = (2u)^{\frac{1}{\eta+1}}$,
- иначе: $\beta = \left[\frac{1}{2}(1-u)\right]^{\frac{1}{\eta+1}}$

Далее по следующей формуле отдельно вычисляются координаты x и y прямоугольника-потомка.

$$offsping_1 = ½\ [(1 + \beta]parent_1 + (1 - \beta)parent_2]);$$
$$offsping_2 = ½\ [(1 - \beta]parent_1 + (1 + \beta)parent_2]),$$

Функция возвращает нового прямоугольника-потомка.

Fitness() - принимает в себя список точек и прямоугольник. Возвращает приспособленность. Проходим по всем точкам и проверяем, лежит ли данная точка в прямоугольнике. Если да, то смотрим ее метку. Если метка — 1, то увеличиваем рейтинг приспособленность текущей особи. Иначе уменьшаем. Функция возвращает значение приспособленности прямоугольника int.

get_func_next_generation() - функция получения нового поколения. Принимает в себя функции мутации, скрещивания и измерения приспособленности. Имеется встроенная функция select_parent(), случайным образом выбирает родителя. Родители с большей приспособленностью имеют больший шанс быть выбранными. Новое поколение формируется либо скрещиванием 2-х особей с определенным шансом, либо оставлением особи предыдущего поколения. Функция возвращает массив прямоугольников нового поколения.

Также были написаны функции для визуализации некоторых функций алгоритма. Это нужно для правки и проверки работоспособности отдельных частей алгоритма.

Связь GUI и алгоритма.

Реализован класс State отвечающий за текущее стояние класса Executor. Включает в себя набор точек и три лучших решения. Метод get_value пока не реализован. Метод __str__ отвечает за логирование данного объекта. Нужен в последующем для консоли в GUI.

Класс Executor отвечает за сам генетический алгоритм. Его состояние характеризует каждый шаг алгоритма. Метод get_state возращает текущее состояние алгоритма. Метод update_solution обновляет состояние, даже в обратную сторону.

Выводы.

На текущий момент удалось создать почти полностью рабочий интерфейс GUI, а также написать прототипы основных функций генетического алгоритма, которые могут быть откорректированы в будущем.