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

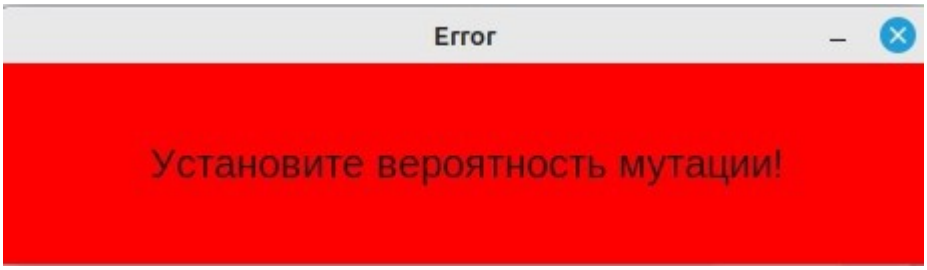

Рисунок 3 — Ошибка при запуске

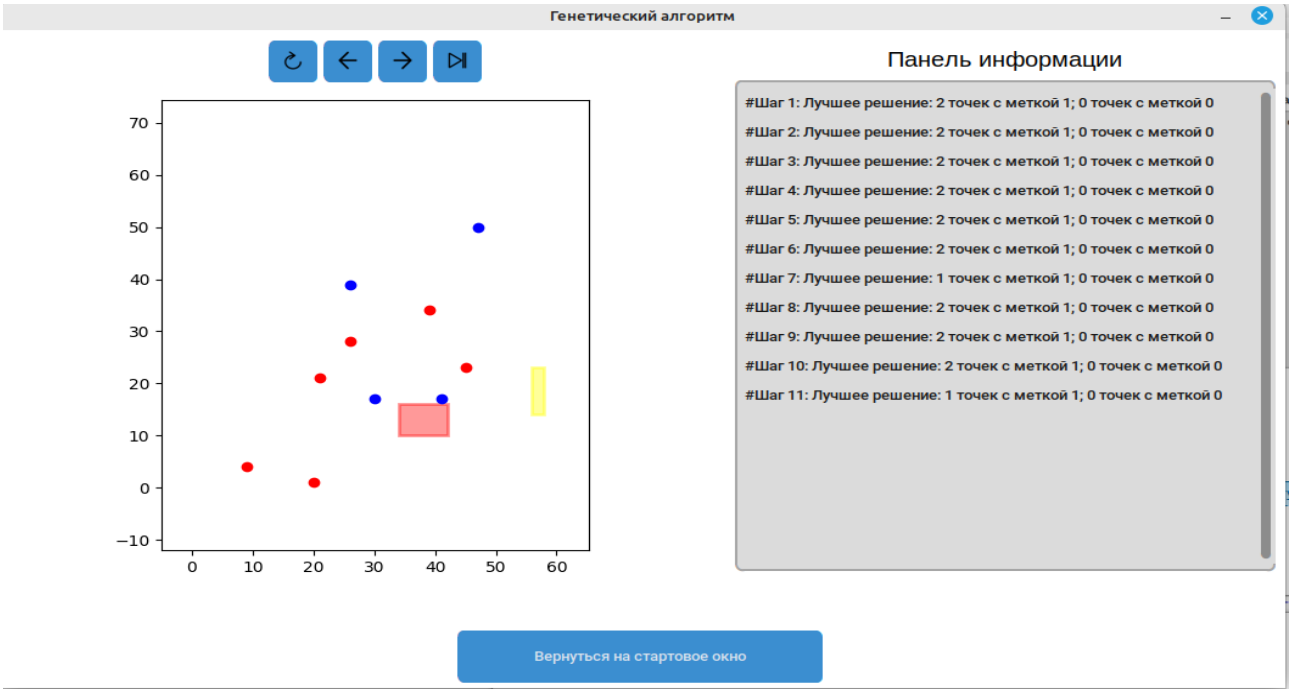

Рисунок 4 — Отображение работы алгоритма

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.

RectangleInfo содержит прямоугольник, значение его приспособленности, а также количество нулевых и единичных точек, которые находятся в этом прямоугольнике.

Объединение Func отвечает за ключи словаря функций. Mutation — функция мутации. Crossing — функция скрещивания. Fitness — функция приспособленности.

Генетический алгоритм

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_random_change() - функция мутации одной особи, принимает в себя прямоугольник и список точек, а также объект класса ParamMutation. С вероятностью из ParamMutation происходит выбор, будет ли прямоугольник увеличен или уменьшен. Далее случайным образом для каждой стороны отдельно выбирается, на сколько она будет увеличена или уменьшена. Не гарантируется изменение количества точек внутри прямоугольника.

mutation_random_point() - функция мутации одной особи, принимает в себя прямоугольник и список точек, а также объект класса ParamMutation. Координаты прямоугольника меняются относительно случайной точки из списка points. Если было выбрано расширять прямоугольник, то выбирается точка, которая не входит в прямоугольник. Иначе выбирается входящая в прямоугольник точка. Далее случайным образом выбирается, будет ли прямоугольник расширен относительно своей левой верхней координаты или правой нижней. Гарантируется изменение количества точек внутри прямоугольника.

mutation_hybrid() - функция мутации одной особи, принимает в себя прямоугольник и список точек, а также объект класса ParamMutation. Случайным образом выбирается функция мутации — mutation_random_change() и mutation_random_point().

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.

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

Рисунок 5 — Вычисление u.

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

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

Рисунок 6 — Вычисление offsping.

Функция возвращает нового прямоугольника-потомка.

count_points_in_rectangle() - вспомогательная функция, принимает себя список точек и прямоугольник. Проходит по всем точкам и определяет, какие из них лежат внутри прямоугольника. Возвращает пару — количество нулевых точек внутри прямоугольника и количество единичных точек внутри прямоугольника.

fitness() - принимает в себя список точек и прямоугольник, а также объект класса ParamFitness. Возвращает приспособленность. С помощью функции count_points_in_rectangle() подсчитывается количество нулевых и еденичных точек внутри прямоугольника, затем вычисляется значение приспособленности для этого прямоугольника. Функция возвращает объект класса RectangleInfo.

get_new_generation() - вспомогательная функция получения нового поколения. Принимает в себя словарь, содержащий функции скрещивания, вычисления приспособленности и мутации, список точек, список прямоугольников и объект класса ParamGeneticAlgorithm. Новое поколение

формируется либо скрещиванием 2-х особей с определенным шансом, либо оставлением особи предыдущего поколения. Функция возвращает массив прямоугольников нового поколения. В случае скрещивания с определенным шансом происходит мутация потомка. Функция возвращает список прямоугольников нового поколения.

select_parent() - вспомогательная функция, принимает список прямоугольников, возвращает случайный элемент из списка прямоугольников.

truncation_selection() - функция отбора усечением. Принимает в себя те же параметры, что и функция get_new_generation(). Формирует список отобранных для размножения особей selected следующим образом: остается половина лучших по приспособленности прямоугольников, которые участвуют в генерации следующего поколения. Функция возвращает результат работы функции get_new_generation(), в которую в качестве массива прямоугольников передается список selected.

roulette_selection() - аналог функции runcation_selection(). Сохраняет принцип работы предыдущей функции, за исключением способа формирования списка selected. Вычисляется значение минимальной приспособленности особи текущего поколения, и если оно отрицательное, то отсчет веса для выбора особей для формирования следующего поколения начинается с этого значения (вес только положительный). Случайным образом по правилу рулетки выбираются особи текущего поколения. Больший шанс быть выбранным у особи с большей приспособленностью.

tournament_selection() - турнирный отбор. Работает также, как и предыдущие методы отбора. Список selected формируется следующим образом: для турнира выбирается половина особей и из них выбирается лучшая, которая и добавляется в список selected.

elite_selection() - элитарный отбор. Выбирается 20% лучших особей текущего поколения, которые автоматически переходят в следующее поколение. Затем по правилу рулетки (roulette_selection) формируется нового остаток поколения.

Связь GUI и алгоритма.

Реализован класс State отвечающий за текущее стояние класса Executor. Включает в себя набор три лучших решения. Метод get_value возвращает три лучший прямоугольника в порядке убывания.. Метод __str__ отвечает за логирование данного объекта. Нужен в последующем для консоли в GUI. Для удобства были реализованы методы __iter__ и __next__. Они позволяют итерироваться по объектов.

Класс StoreData отвечает за восстановление данных стартового окна.

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

Эксперимент.

Установим параметры:

- Шанс мутации: 40%
- Шанс скрещивания: 70%
- Шанс увеличения прямоугольника: 60%
- Максимальное количество эпох: 600
- Количество индивиов в эпохе: 100

Для теста со всеми красными (метка 0) точками все методы справились.

Для теста со всеми синими точками:

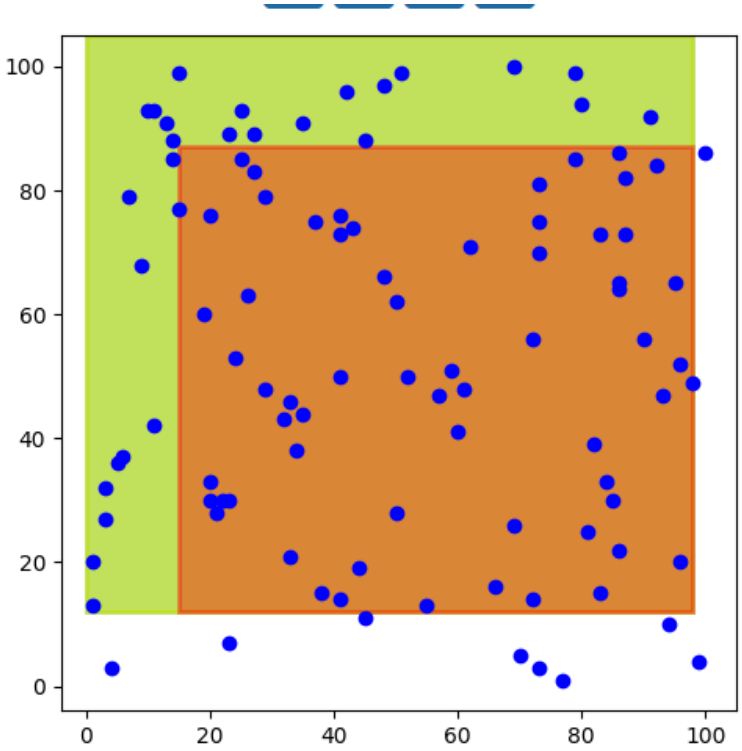

Рисунок 7 — Метод отбора: Отсечение.

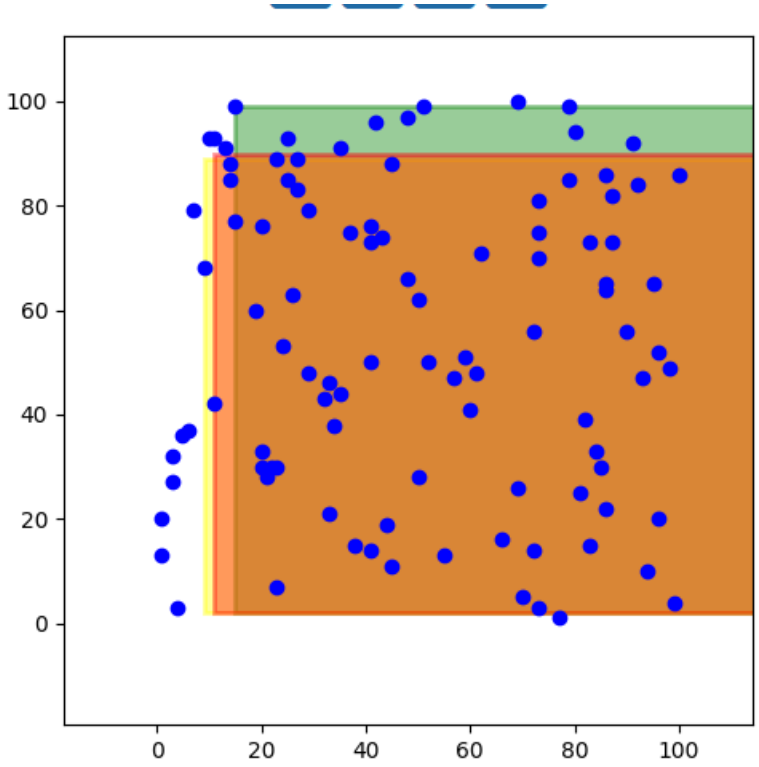

Рисунок 7 — Метод отбора: Рулетка.

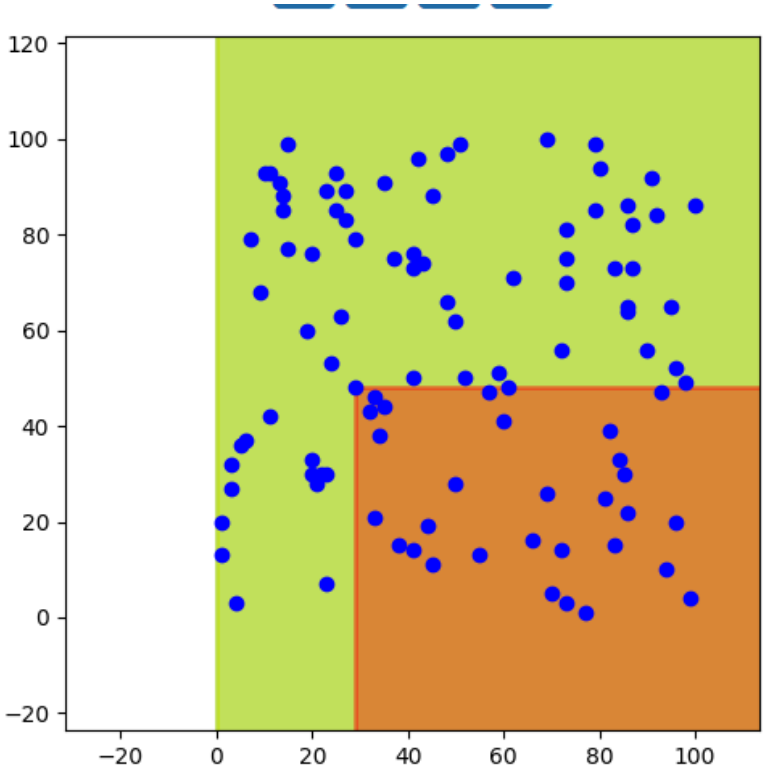

Рисунок 7 — Метод отбора: Турнир.

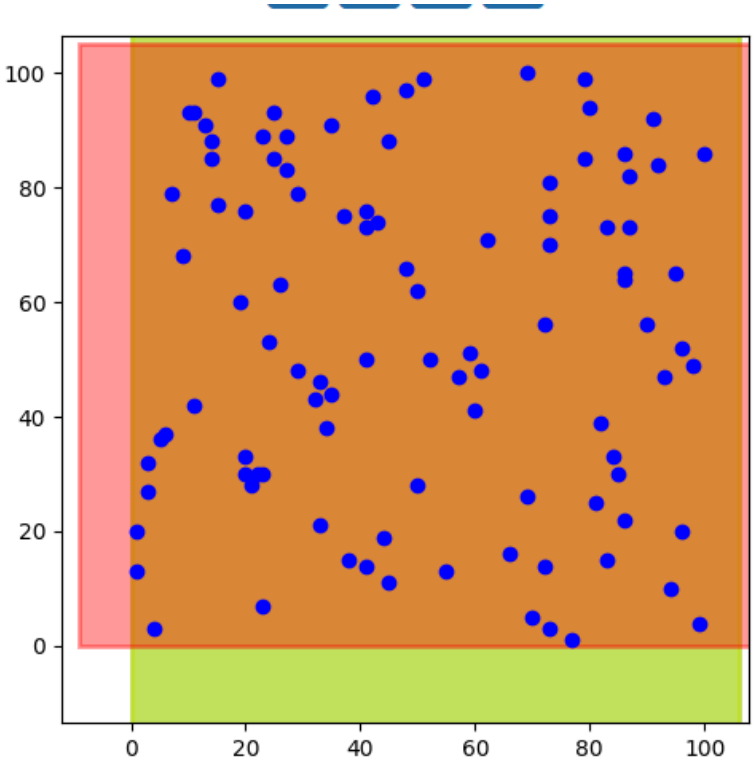

Рисунок 7 — Метод отбора: Элита.

Из эксперимента видно, что лучшее методы для этого случая элита и турнир, а рулетка и отсечение не выдали нужный ответ.

Проведём эксперимент над tests/test1.csv:

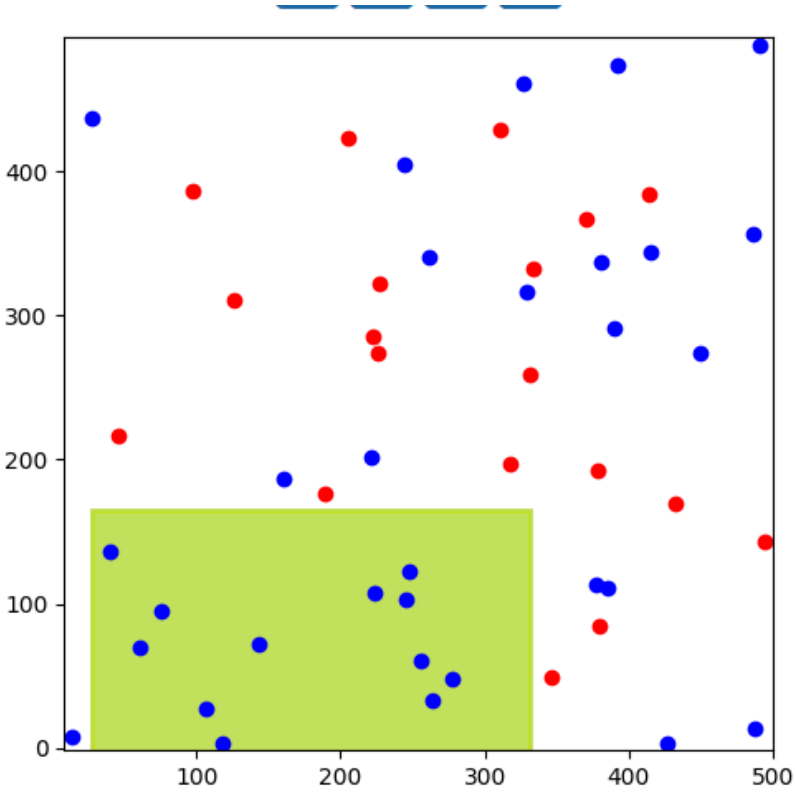

Рисунок 7 — Метод отбора: Отсечение.

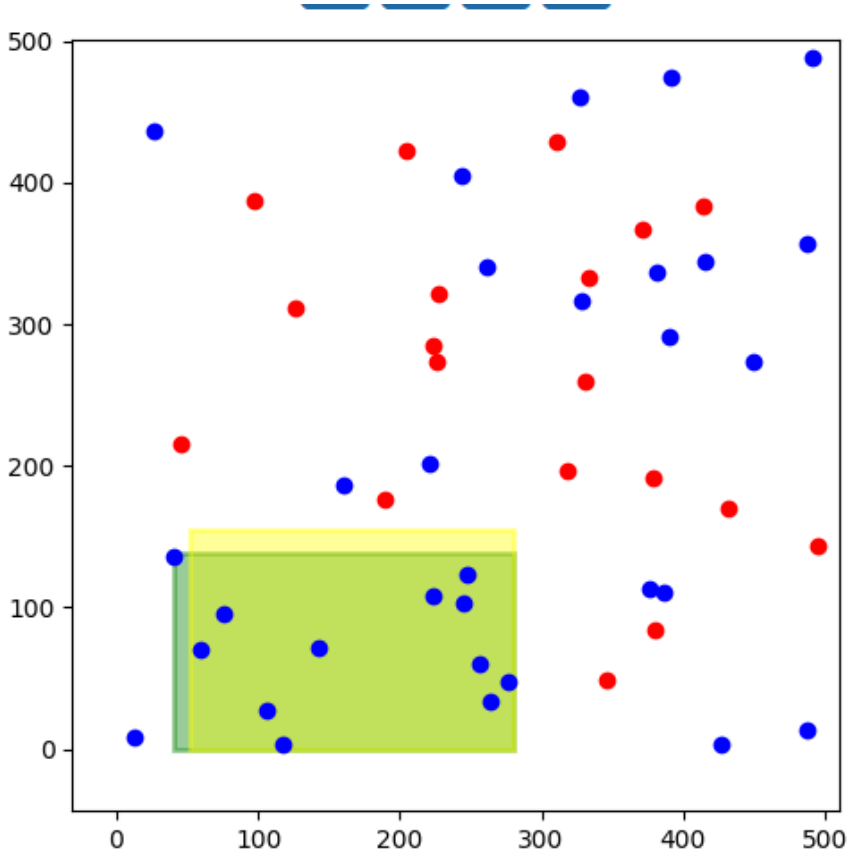

Рисунок 7 — Метод отбора: Рулетка.

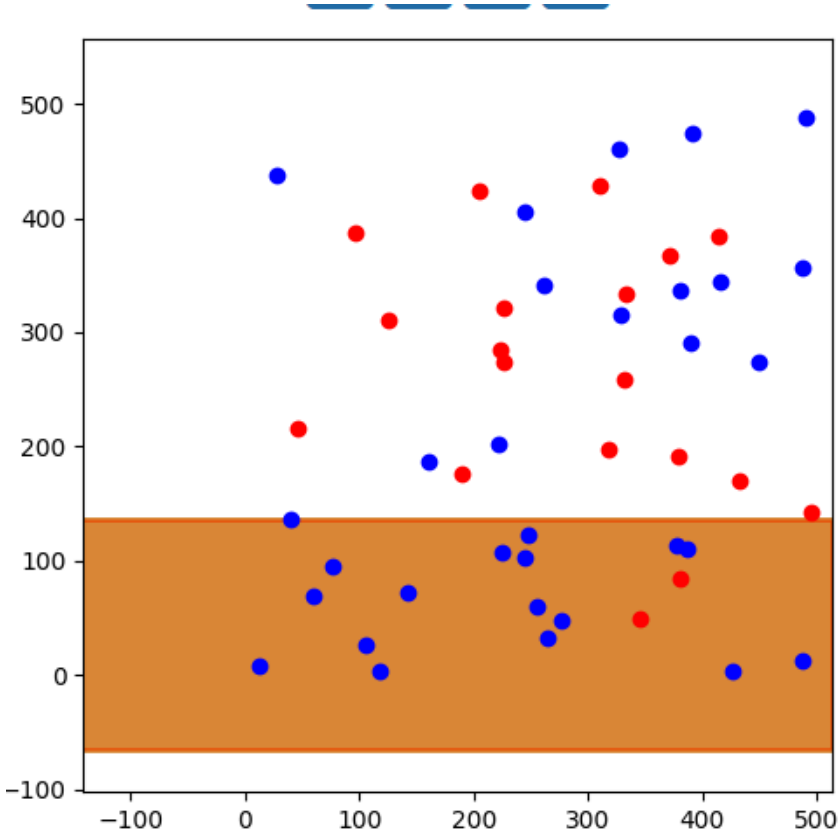

Рисунок 7 — Метод отбора: Турнир.

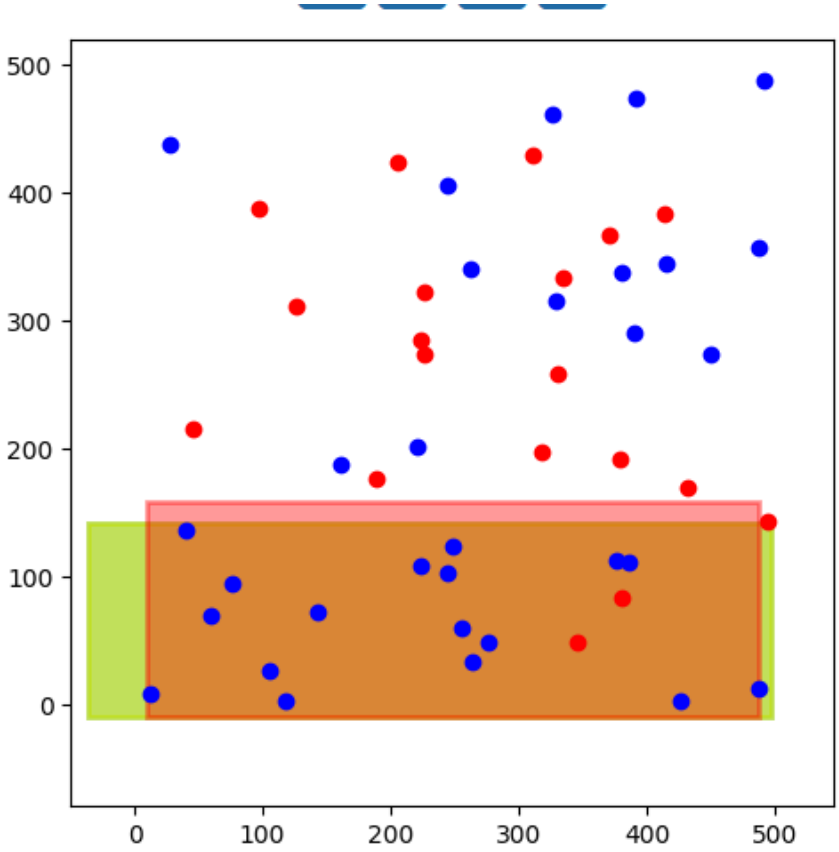

Рисунок 7 — Метод отбора: Элита.

Из эксперимента видно, что лучшее методы для этого случая элита и турнир, а рулетка и отсечение не выдали нужный ответ.

Проведём эксперимент над tests/test4.csv:

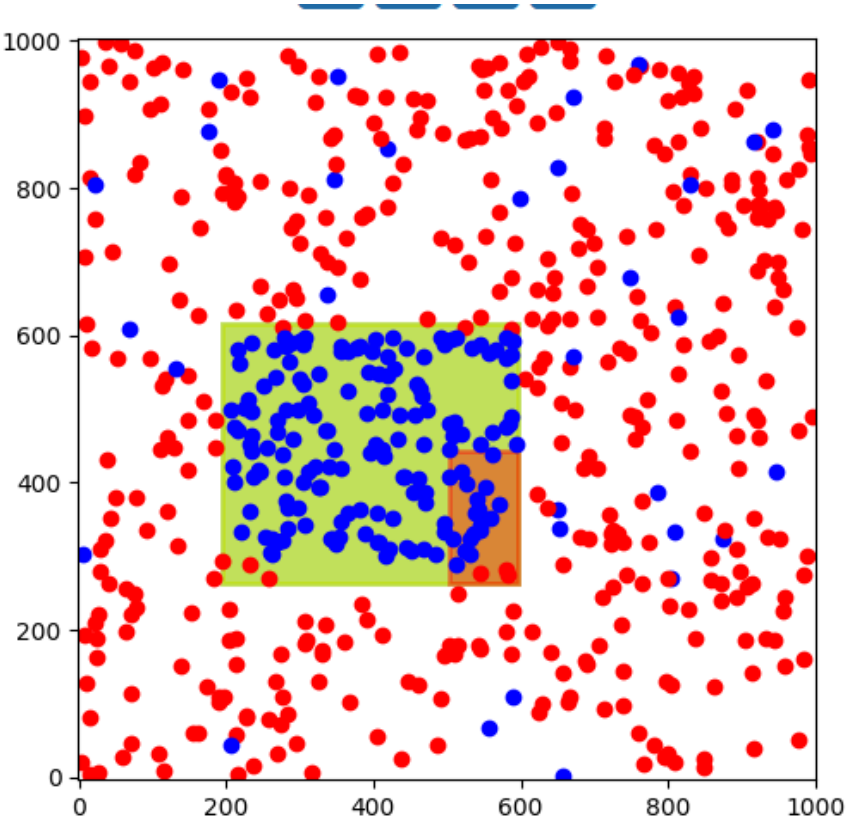

Рисунок 7 — Метод отбора: Отсечение.

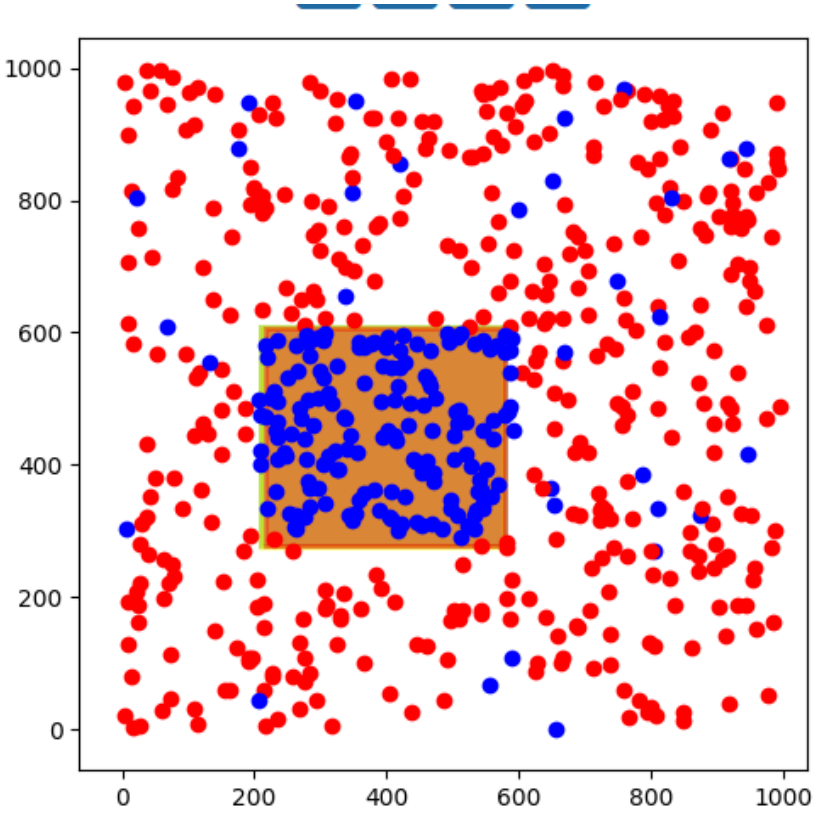

Рисунок 7 — Метод отбора: Рулетка.

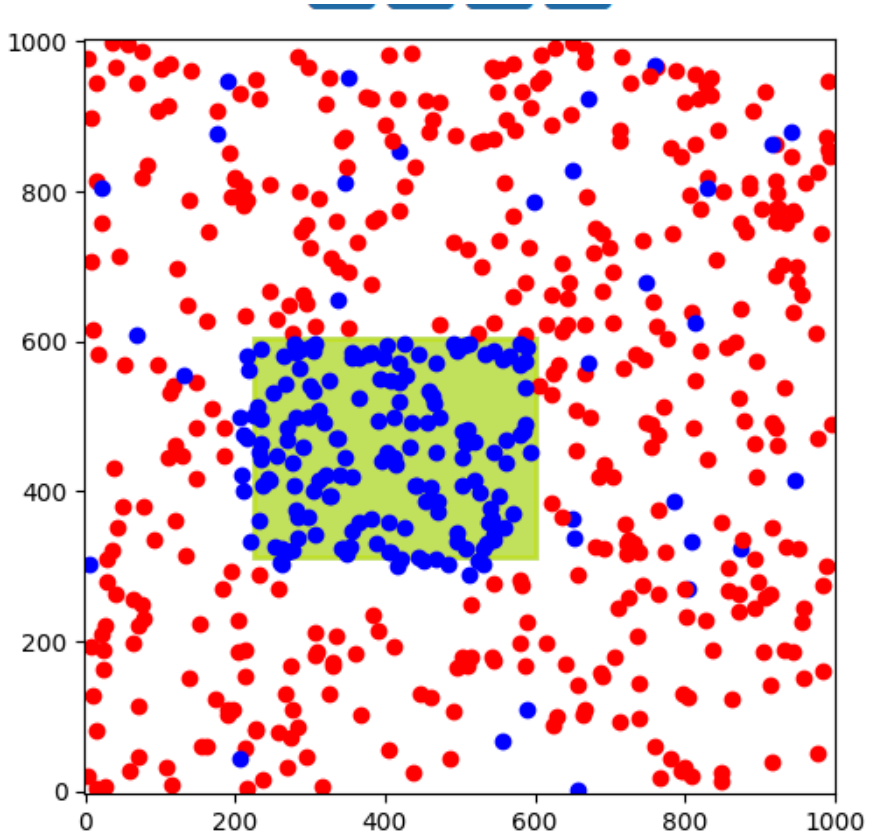

Рисунок 7 — Метод отбора: Турнир.

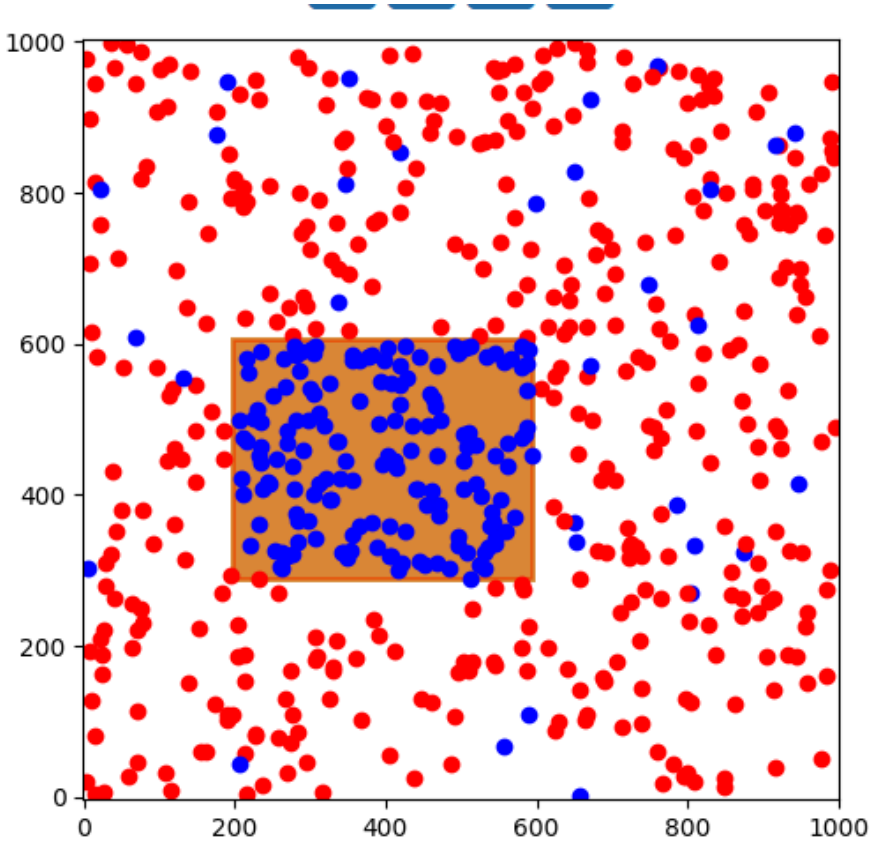

Рисунок 7 — Метод отбора: Элита.

Из эксперимента видно, что лучшее методы для этого случая элита и турнир, а рулетка и отсечение не выдали нужный ответ.

Из всех экспериментов видно, что лучшие методы элита и турнир.

Выводы.

В ходе проведения данной работы удалось создать генетический алгоритм, который ищет прямоугольник, в который входит минимальное количество точек с меткой 0 и максимальное с меткой 1. Также был реализован интерфейс GUI для более удобного использования данного алгоритма. Был проведены эксперименты. Выяснилось, что лучшее методы для нашей задачи элитарный и турнирный.