МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

ОТЧЕТ

по учебной практике

Тема: Генетические алгоритмы

Студенты гр. 2381 Мавликаев И.А.

Слабнова Д.А.

Дудкин М.В.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2024

Цель работы.

Разобраться в теме генетических алгоритмов, их видов и тонкостей реализации, а также создать проект, на них основанный.

Задание.

Для заданного полинома f(x) (степень не больше 8) необходимо найти параметры ступенчатой функции g(x) (высота ступеней), которая приближает полиномиальную функцию, то есть минимизировать расстояние |f(x) - g(x)| между функциями на заданном интервале [l, r]. Количество ступеней вводится пользователем.

Выполнение работы.

Изменения в классе полинома

Добавлены функции:
double Polynomial:: getValue(double x) – Возвращает значение полинома в точке x.

void Polynomial::print() – Выводит коэффициенты полинома в консоль для отладки.

double Polynomial::Evaluation(Chromosome& chromosome) – Функция качества индивида. Рассчитывает интеграл |f(x) - g(x)| на заданном интервале. Используется интеграл, а не накопленная разность для соразмерности значений.

Изменения в классе алгоритма

Добавлены функции:
std::vector<Chromosome> stepHybridAlgorithm() - шаг алгоритма.
std::vector<Chromosome> top() - возвращает 3 лучшие хромосомы популяции
void printStep() - печать 3 лучших хромосом.

Пока что программа работает так, что часто 3 лучших значения популяции становятся одинаковыми, это будет исправлено.

Изменения в классе популяции

Метод алгоритма elite_selection разбит на несколько методов: добавлены методы:
void sortPopulation() - сортирует поле chromosomes (вектор хромосом) по estimate (оценка).
void addChildren(std::vector<Chromosome> &children, Polynomial &polynom) - добавляет в вектор хромосом детей данных хромосом. Используется в элитарной селекции.
void elite_selection(Polynomial &polynom) - метод элитарной селеции. Популяция хромосом обрезается до ¾ популяции, в неё добавляются дети и она сортируется при помощи вышеописанных методов. Затем она обрезается до исходного количества хромосом (остаются лучшие).

Изменения в классе хромосомы

Убрано ограничение по высоте столбцов (по значению отдельного гена). Метод мутации переписан так, чтобы потом можно было легко переключиться на другие методы мутации.

Изменения в GUI

Было создано простое приложение, состоящее из кнопки и двух окон графика, а также “функция-заглушка” для функции алгоритма. Функция-заглушка возвращает три массива из double и одно число double (по сути, конечная функция самого алгоритма должна возвращать три лучших особи и наилучшую оценку). При нажатии кнопки графики обновляются. В первом графике добавляются точки (будущий график оценки приближения в зависимости от количества итераций). Во втором графике, которые потом будет отвечать за визуализацию самой функции и трёх лучших решений, будут менять ломанные, отвечающие за потомков. Был создан файл main.py, к которому подключена библиотека PyQt6, и были реализованы классы:

- ThreeGraph - класс графика визуализации кривой и текущих решений
    - __init__(self, start, fin, polynome_data):
      инициализирует граф, по polynome_data отрисовывает сам полином (на интервале start - fin). Создает три пока пустых подграфа - будущие три лучших решения.
    - update(self, val1, val2, val3):
    - val* - новые значения лучших решений. Обновляет изображения их на графике. Т.к. размерность (длина) массива-решения меньше длины вектора polynome_data, то перед тем, как отображать эти данные, их надо “растянуть” на всю ось OX.
    - widen_val(self, val):
      Расширяет данные массива - решения так, чтобы они отображались на всём графике

- EsteemGraph - график оценки
    - __init__(self, num_of_iter):

      Инициализирует график, num_of_iter - максимальное количество итераций, задаваемое пользователем (для *красивой* отрисовки графика длина оси OX сразу задаётся, а не увеличивается с увеличением количества шагов).
    - update(self, val)

      Добавляет в график следующую точку (x, y) = (номер следующей итераци, val)

- MainWindow - само приложение
    - Имеет поля, button - кнопка, esteem_graph - экземпляр EsteemGraph, three_graph - экземпляр ThreeGraph.

      А также поля coeff_for_pol, step, start, finish, num_of_iter - которые потом будут передаваться пользователем в программу

      Поле counter считает количество нажатий кнопки (проделанных шагов)
    - __init__(self):
    - Инициализирует вышеописанные поля и создает единое окно со всеми виджетами.
    - the_button_was_clicked(self):
    - Событие, которое происходит при нажатии кнопки. С помощью заглушки pretend_alg() вычисляются “следующие значения”. Для полей esteem_graph, three_graph вызываются соответствующие им методы update. Переменная counter

увеличивается на 1. Если counter >= заданному количеству итераций - кнопка перестаёт быть доступной.

Цели по GUI до следующей итерации: превратить, с помощью библиотеки python ctype, основной метод алгоритма в библиотеку для питона и заменить “заглушку” на настоящую функцию. Добавить поля ввода данных(аргументов) в приложение. Нормально оформить интерфейс приложения.