МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Название кафедры

ОТЧЕТ

по учебной практике

Тема: Генетические алгоритмы

Студенты гр. 2381 Дудкин М.В.
Слабнова Д.А.
Мавликаев И.

Руководитель _____ Жангиров Т.Р.

Санкт-Петербург
2024

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студенты Дудкин М. Слабнова Д.А. Мавликаев И.

Группа 2381

Тема практики: генетические алгоритмы

Задание на практику:

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 вводятся пользователем.

Сроки прохождения практики: 26.06.2024 – 09.07.2024

Дата сдачи отчета: 09.07.2024

Дата защиты отчета: 09.07.2024

Студенты		Дудкин М. Слабнова Д.А. Мавликаев И.
Руководитель		Жангиров Т.Р.

АННОТАЦИЯ

Цель практики - изучить работу генетических алгоритмов, различные способы их реализации. Разработать приложение с графическим интерфейсом, позволяющее находить ступенчатую функцию, аппроксимирующую заданный пользователем полином. Поиск решения будет происходить посредством генетического алгоритма.

SUMMARY

The purpose of the practice is to study the work of genetic algorithms, various ways of their implementation. To develop an application with a graphical interface that allows you to find a step function approximating a user-defined polynomial. The search for a solution will take place through a genetic algorithm.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель практики - получить базовые знания о генетических алгоритмах и способах их реализации и закрепить их при выполнении практической работы на данную тему.

1. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

1.1. Общие сведения

- Вектор — упорядоченный набор чисел, называемых компонентами вектора.
- Булев вектор — вектор, компоненты которого принимают значения из двухэлементного (булева) множества, например, {0, 1} или {−1, 1}.
- Хеммингово расстояние — используется для булевых векторов и равно числу различающихся в обоих векторах компонент.
- Хромосома — вектор (или строка) из каких-либо чисел. Если этот вектор представлен бинарной строкой из нулей и единиц, например, 1010011 то он получен либо с использованием двоичного кодирования, либо кода Грея. Каждая позиция (бит) хромосомы называется геном.
- Хеммингово пространство — пространство булевых векторов, с введенным на нем расстоянием (метрикой) Хемминга. В случае булевых векторов размерности n рассматриваемое пространство представляет собой множество вершин n-мерного гиперкуба с хеммингговой метрикой. Расстояние между двумя вершинами определяется длиной кратчайшего соединяющего их пути, измеренной вдоль ребер.
- Индивидуум (генетический код, особь) — набор хромосом (вариант решения задачи). Обычно особь состоит из одной хромосомы, поэтому в дальнейшем особь и хромосома идентичные понятия.
- Расстояние — хеммингово расстояние между бинарными хромосомами.
- Кроссинговер ′ (кроссовер) — операция, при которой две хромосомы обмениваются своими частями. Например, 1100&1010 −→ 1110&1000.
- Мутация — случайное изменение одной или нескольких позиций в хрмосоме. Например, 1010011 −→ 1010001.
- Инверсия — изменение порядка следования битов в хромосоме или в ее фрагменте. Например, 1100 −→ 0011.
- Популяция — совокупность индивидуумов.

- Пригодность (приспособленность) — критерий или функция, экстремум которой следует найти.
- Локус — позиция гена в хромосоме
- Аллель — совокупность подряд идущих генов.

1.2. Рекомбинация

Оператор рекомбинации применяют сразу же после оператора отбора родителей для получения новых особей-потомков. Смысл рекомбинации заключается в том, что созданные потомки должны наследовать генную информацию от обоих родителей. Различают дискретную рекомбинацию и кроссинговер.

Дискретная рекомбинация в основном применяется к хромосомам с вещественными генами. Основными способами дискретной рекомбинации являются собственно дискретная рекомбинация, промежуточная, линейная и расширенно линейная рекомбинации.

Дискретная рекомбинация применима для любого типа генов (двоичные, вещественные и символьные).

1.3. Мутация

После рекомбинации, т.е. появления потомков, происходят мутации. Цель мутации - выбить конкретную особь из локального оптимума, путем случайного (то есть не обязательно оптимального) изменения ген в хромосоме. Самый простой вариант мутации - с не меняющимся шагом, то есть диапазон изменения всегда одинаков.

$$new\ gen = rand(old\ gen - step,\ old\ gen + step)$$

Другой способ мутации:

$$new\ gen = old\ gen +- \alpha \cdot \delta$$

$$\delta = \sum_{i=1}^{m} a(i) \cdot$$

1.4. Отбор родителей и скрещивание

Существует несколько подходов к выбору родительской пары.

Отбор родителей происходит случайно при помощи вероятности скрещивания в случае гибридного алгоритма. В случае алгоритма с нефиксированной популяцией отбора родителей не происходит вовсе.

1.5 Отбор в новую популяцию.

Пользователю предоставляется выбор - гибридный алгоритм с элитарным отбором или алгоритм с нефиксированной популяцией, где отбора в новую популяцию не происходит.

1.6 Гибридный алгоритм.

Идея гибридных алгоритмов заключается в сочетании генетического алгоритма с некоторым другим классическим методом поиска, подходящим в данной задаче. В каждом поколении все сгенерированные потомки оптимизируются выбранным методом и затем заносятся в новую популяцию. Тем самым получается, что каждая особь в популяции достигает локального оптимума, вблизи которого она находится. Далее производятся обычные для ГА действия: отбор родительских пар, кроссинговер и мутации. На практике гибридные алгоритмы оказываются очень удачными. Это связано с тем, что вероятность попадания одной из особей в область глобального максимума обычно велика. После оптимизации такая особь будет являться решением задачи.

Известно, что генетический алгоритм способен быстро найти во всей области поиска хорошие решения, но он может испытывать трудности в получении из них наилучших. Обычный оптимизационный метод может быстро достичь локального максимума, но не может найти глобальный.

Сочетание двух алгоритмов позволяет использовать преимущества обоих.

1.7 Алгоритм с нефиксированной популяцией.

В генетическом алгоритме с нефиксированным размером популяции каждой особи приписывается максимальный возраст, то есть число поколений, по прошествии которых особь погибает. Внедрение в алгоритм нового параметра — возраста — позволяет исключить оператор отбора в новую популяцию. Возраст каждой особи индивидуален и зависит от ее

приспособленности. В каждом поколении t на этапе воспроизведения обычным образом создается дополнительная популяция из потомков. Размер дополнительной популяции (AuxPopsize(t)) пропорционален размеру основной популяции (Popsize(t)) и равен

$$AuxPopsize(t) = [Popsize(t)Pc],$$

где Pc — вероятность воспроизведения. Для воспроизведения особи выбираются из основной популяции с равной вероятностью независимо от их приспособленности. После применения мутации и кроссинговера потомкам приписывается возраст согласно значению их приспособленности. Возраст является константой на протяжении всей эволюции особи (от рождения до гибели). Затем из основной популяции удаляются те особи, срок жизни которых истек, и добавляются потомки из промежуточной популяции. Таким образом, размер после одной итерации алгоритма вычисляется по формуле:

$$P\ opsize(t + 1) = P\ opsize(t) + AuxP\ opsize(t) - D(t),$$

где D(t) — число особей, которые умирают в поколении t.

2. РЕАЛИЗАЦИЯ КЛАССОВ И GUI

2.1. Класс хромосомы

Класс Chromosome.

Поля:

- double probMutation - вероятность мутации каждого гена
- std::vector genes - вектор генов
- int length - длина хромосомы
- double max_mutation_step - шаг мутации, нужен для примитивной мутации
- double down_border - найденный минимум (в начале значение функции рассчитывается в нескольких точках, из них выбирается минимум и максимум. Эта информация нужна для того, чтобы вычислить шаг мутации)
- double up_border - верхняя граница
- double estimate - оценка, наскоолько данная хромосома апроксимирует функцию.
- int method_mul, int method_recomb - какие методы мутации и рекомбинации использовать.
- int age, int birthDate - возраст (продолжительность жизни) и итерация, на которой появилась особь. Не используется в своих методах, но данные поля важны в одном из методов популяции (создания и удаления особей при изменении).

Методы:

- Chromosome(double probMutation, double down, double up, int number, int len, int method_mut, int method_pro, int age, int birthDate) - инициализация. Создает вектор рандомных высот длины N между down и up и записывает в поле genes

- static std::vector recombination(Chromosome parent1, Chromosome parent2, int method = 0) - интерфейс рекомбинации. Реализовано два способа рекомбинации.
- void mutate(); - интерфейс мутации. Реализовано два способа мутации.
- double new_gene(double old_gene) - генерирует новый ген для простой мутации
- static void discr_recomb(Chromosome parent1, Chromosome parent2, std::vector& answer); - дискретная рекомбинация
- static void inter_recomb(Chromosome parent1, Chromosome parent2, std::vector& answer); - промежуточная рекомбинация
- void mutate_dumb() - мутация, изменяющая ген на случайное число в всегда одинаковом диапазоне
- void mutate_better() - мутация с непостоянным диапазоном (разным шагом). Для данной мутации была создана вспомогательная функция calc_d.

2.2. Класс популяции

Поля класса Population:

- int countIndivids - количество хромосом
- std::vector<Chromosome> chromosomes - вектор хромосом
- double threshold - пороговое значение, по котороромому обрезается популяция при элитарном отборе.
- double probReproduction - вероятность скрещивания
- Population(Polynomial &polynom, double down = 0, double up = 100, int countIndivids = 50, int countSteps = 10, double probMutation = 0.2, double probReproduction = 0.6, int method_mut, int method_recomb);
- void updatePopulation(std::vector<Chromosome> chromosomes);
- 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).

- 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 updateNonfixed(Polynomial &polynom, int count) - добавляет детей в популяцию с нефиксированным размером.
- void cutOldIndivids(int curIteration) - удаляет слишком старые особи.
- void addAge(int maxAge) - добавляет возраст новым особям. Возраст рассчитывается зависимости от приспособленности

2.3. Класс полинома

Поля класса Polynom:

- double left - левая граница, на которой определен полином
- double right - правая граница

Методы:

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

2.4. Класс алгоритма

Класс генетического алгоритма. Поля класса:

- Polynomial polynom - хранится полином.
- Population population - хранится популяция.
- int iteration - количество итераций.
- int time - номер текущей итерации.
- int method - метод ГА (0 - гибридный алгоритм, 1 - алгоритм с нефиксированной популяцией)
- double criterion - критерий завершения ГА

- Algorithm(int count_individs, int count_steps, double probMutation, std::vector<double> coefs_polynom, int iteration, double criterion, double left, double right, int method, int method_mut, int method_recomb, double probReproduction) - инициализация.
- std::vector<Chromosome> stepAlgorithm(); - шаг алгоритма, в зависимости от выбранного метода.
- std::vector<Chromosome> top(); - возвращает 3 лучше хромосомы популяции.
- void printStep(); - метод, который использовался до того, как был написан GUI.

2.5 GUI

- 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, esteem_graph - экземпляр EsteemGraph, three_graph - экземпляр ThreeGraph.

    А также поля с требуемыми для инициализации и шага алгоритма, поля для кнопок и полей ввода.
  - __init__(self): инициализирует все поля
  - the_button_was_clicked, starter, go_on_without_stop - методы кнопок (активируются при нажатии). Кнопки отвечают за ввод параметров, выполнение шага и запуск алгоритма без остановки.
  - do_step и do_step_without_draw - методы шага алгоритма. Первый используется для пошагового выполнения алгоритма (шаг происходит при нажатии кнопки пользователем), на каждом шаге отрисовывается график. Второй - при запуске алгоритма без остановки (не тратит ресурсы на отрисовку графиков).

3. РЕАЛИЗАЦИЯ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

3.1. Гибридный алгоритм.

Алгоритм использует элитарный отбор в новую популяцию, родители выбираются случайно, для каждой особи пользователем задана вероятность скрещивания. Алгоритм работает быстро, нечувствителен к входным данным. Методом подбора параметров было выявлено, что оптимальный размер популяции находится в диапазоне от 50 до 100 особей.

3.2. Алгоритм с нефиксированной популяцией.

В алгоритме с нефиксированной популяцией особь живёт заданное количество итераций. В реализованной версии программы максимальное количество итераций, которое может прожить индивид, равно трём. В зависимости от приспособленности особи ей присваивается возраст - 2, 1 или 0, в зависимости от того, в какой трети отсортированной популяции находится хромосома.

Алгоритм очень чувствителен к входным данным. В случае, если коэффициент скрещивания высокий, то популяция слишком быстро растет, если низкий, то сворачивается до менее чем 3 особей и программа завершается.

ЗАКЛЮЧЕНИЕ

Были реализованы два метода генетических алгоритмов и гибридный алгоритм оказался гораздо эффективнее алгоритма с нефиксированной популяцией и менее требовательным к входным данным. В алгоритме с нефиксированной популяцией либо очень быстро начинается увеличиваться количество хромосом, если взять вероятность скрещивания близкой к единице, либо слишком быстро убывает, если взять близкой к нулю. Экспериментальным путём было выявлено, что наиболее оптимальными значениями для него являются около 100 для количества особей в популяции и около 0.4 для вероятности скрещивания.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Т.В. Панченко “Генетические алгоритмы”. Издательский дом «Астраханский университет» 2007. 88 с.

ПРИЛОЖЕНИЕ А

ТЕСТИРОВАНИЕ

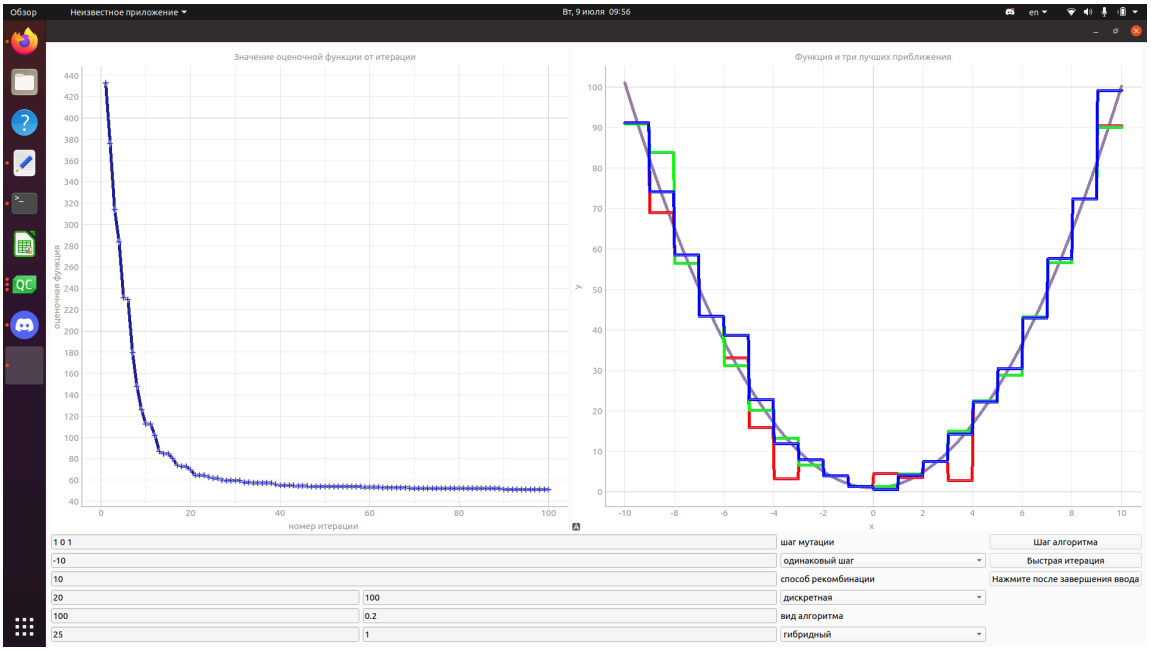

рис. 1. результат работы гибридного алгоритма.

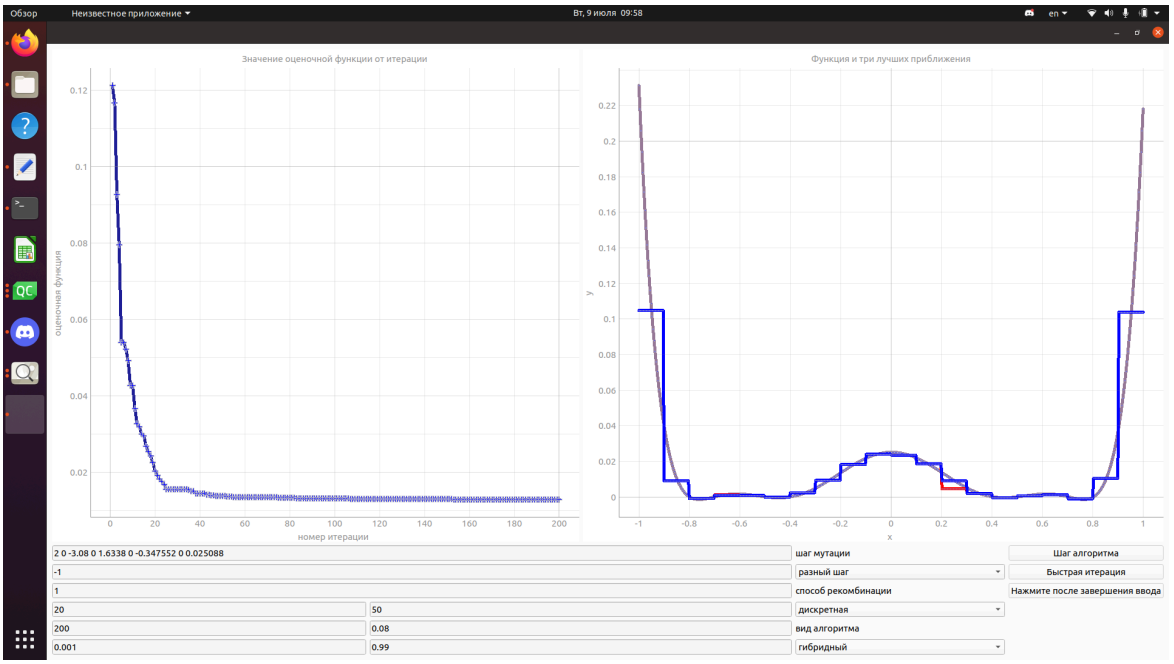

рис. 2 - результат работы гибридного алгоритма

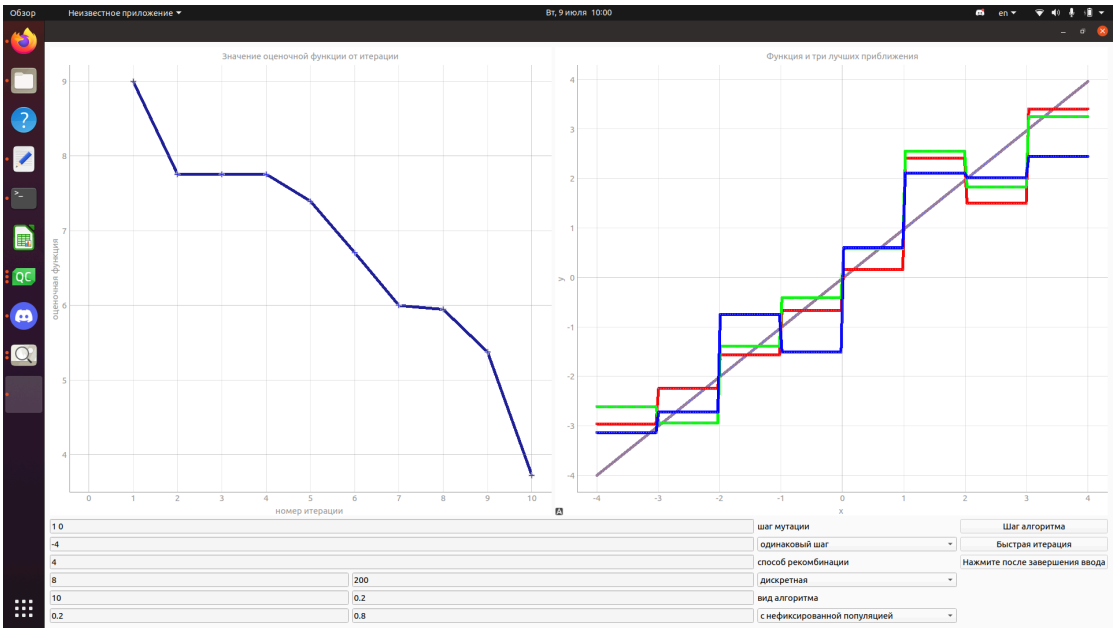

рис 3 - результат работы алгоритма с нефиксированной популяцией.