

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

ОТЧЕТ
по учебной практике
Тема: Нахождение экстремальных точек полиномиальной функции на
заданном интервале

Студенты гр. 0304	_____	Гурьянов С.О. Свечников И.В.
Руководитель	_____	Жангиров Т.Р.

Санкт-Петербург
2022

ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Студенты Гурьянов С.О., Свечников И.В.

Группа 0304

Тема практики: Нахождение экстремальных точек полиномиальной функции (степень не выше 9) на заданном интервале

Задание на практику:

Для заданного полинома не выше 9 степени необходимо найти глобальный минимум.

Входные данные:

- Коэффициенты полинома 9 степени $a_0, a_1, a_2, \dots, a_{10}$
- Интервал поиска $[l, r]$

Сроки прохождения практики: 29.06.2022 – 12.07.2022

Дата сдачи отчета: 05.07.2022

Дата защиты отчета: 05.07.2022

Студенты

Гурьянов С.О.
Свечников И.В.

Руководитель

Жангиров Т.Р.

АННОТАЦИЯ

Данная работа подразумевает реализацию генетического алгоритма с использованием графического интерфейса. Она разделена на несколько итераций. Под каждую итерацию выделяется один пункт отчёта.

СОДЕРЖАНИЕ

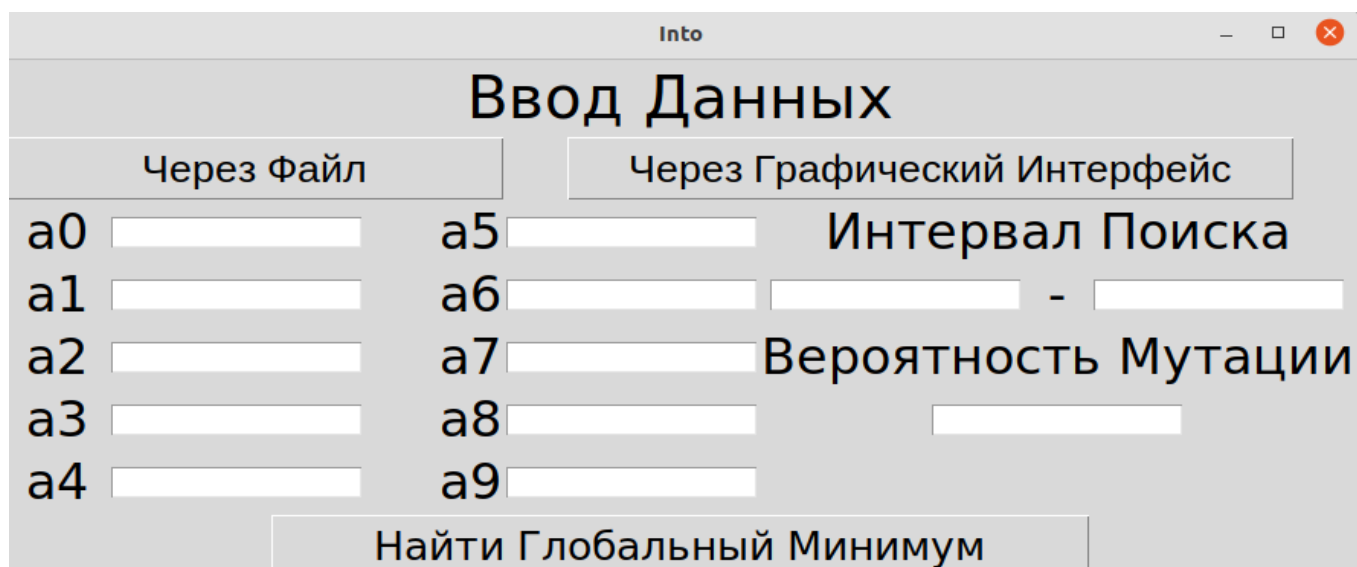
	Введение	5
1	Итерация 2	6
1.1	Скетч с GUI, который планируется реализовать.	6
1.2.	Описание сценариев взаимодействия пользователя с программой	6
1.3	Определение и обоснование параметров модификации ГА для решения задачи.	7
2.1	Итерация 3	7
2.1.	Скриншоты начальной реализации GUI.	7
2.2.	Описание ввода данных (через терминал).	7
2.3.	Описание реализации алгоритма.	8
	Заключение	9
	Список использованных источников	10

ВВЕДЕНИЕ

Целью работы является программная реализация решения поставленной оптимизационной задачи на языке Python с использованием ГА. Основными задачами выполнения работы являются: формирование прототипа GUI и выбор метода решения задачи, частичная реализация программы, в которой присутствует GUI и реализовано хранения данных и основные элементы ГА. Также создана инструкция по сборке и запуску программы. На конечной итерации должна быть выполнена цель работы, а именно программа должна полностью работать вместе с её графической частью, ГА должен гарантированно находить решения.

1. Итерация 2

1.1. Скетч с GUI, который планируется реализовать.



Ввод Данных

Через Файл Через Графический Интерфейс

a0 a5 Интервал Поиска -

a1 a6

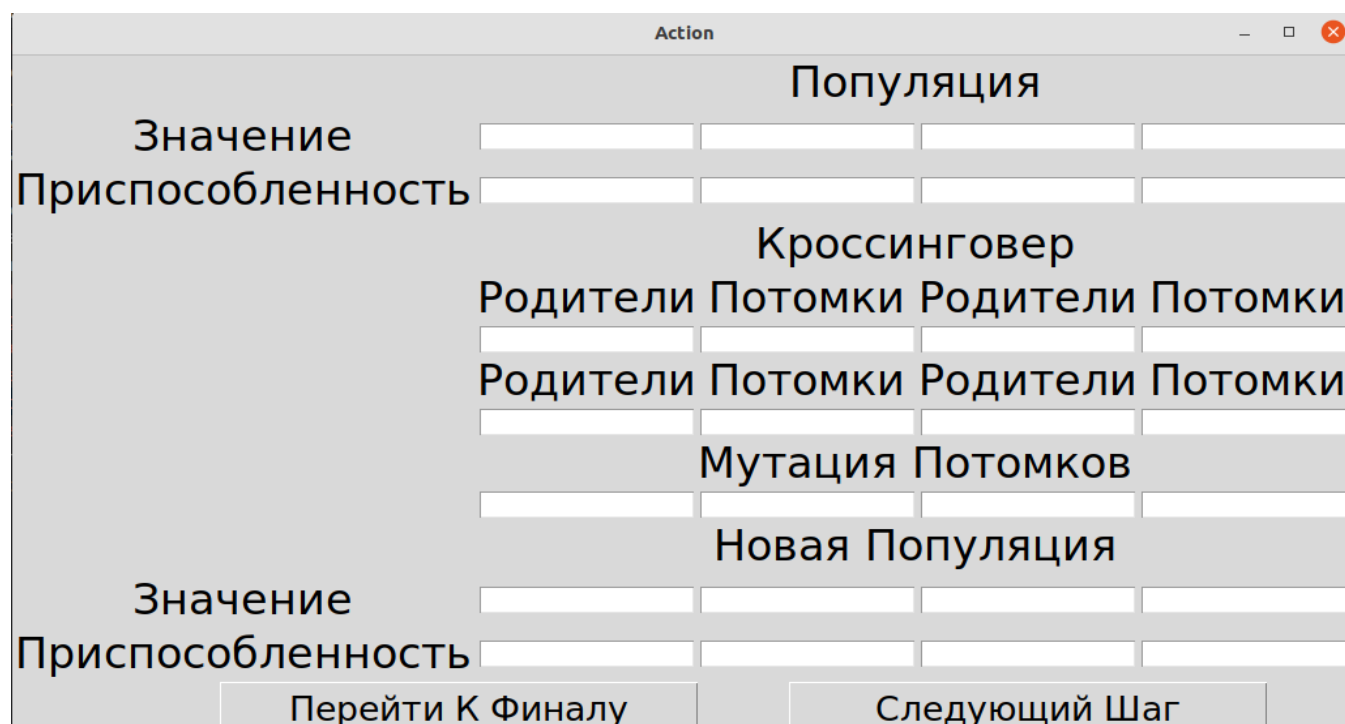
a2 a7 Вероятность Мутации

a3 a8

a4 a9

Найти Глобальный Минимум

Рис. 1 — начало взаимодействия с программой



Популяция

Значение	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Приспособленность	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Кроссинговер				
Родители	Потомки	Родители	Потомки	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
Родители	Потомки	Родители	Потомки	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
Мутация Потомков				
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
Новая Популяция				
Значение	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Приспособленность	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Перейти К Финалу Следующий Шаг

Рис. 2 - шаг алгоритма

1.2. Описание сценариев взаимодействия пользователя с программой

Сценарий взаимодействия пользователем с программой:

- 1) Запуск программы.
- 2) Выбор способа ввода данных.

- 3) Проверка корректности данных.
- 4) Вывод результата в виде таблицы.

1.3 Определение и обоснование параметров модификации ГА для решения задачи.

Для решения данной задачи будет использоваться метод прерывистого равновесия. Обосновано это тем, что данный метод позволяет эффективно выходить за пределы локальных ям и формировать глобальный минимум. Оператором выбора родителей будет являться панмиксия. Получившиеся в результате кроссинговера потомки и наиболее пригодные родители случайным образом смешиваются. Из общей массы в новое поколение попадут лишь те особи, пригодность которых выше средней. Тем самым достигается управление размером популяции в зависимости от наличия лучших особей. В качестве оператора кроссинговера будет выбран одноточечный кроссинговер, а в качестве оператора мутации — одноточечная мутация, поскольку метод прерывистого равновесия предполагает их использование. Таким образом, необходимо учитывать вероятность мутации, заданную пользователем. Оператор отбора в новую популяцию - элитарный отбор, поскольку необходимо случайным образом смешать родителей и получившихся потомков.

2. Итерация 3

2.1. Реализация GUI.

GUI реализован при помощи библиотеки Tkinter языка Python. Первое окно (стартовое) нужно для выбора ввода данных и возможности ввода параметров через графический интерфейс (файл *interface.py*). Второе окно (*interface2.py*) используется для демонстрации итерации алгоритма. Существует возможность перейти к следующей итерации или в конец алгоритма (когда формируется конечное решение).

2.2. Описание ввода данных (через терминал).

На этом этапе поддерживается работа с программой только через терминал. При запуске программы вначале требуется через пробел ввести коэффициенты полинома, начиная от x^0 , заканчивая x^9 (10 чисел). Далее вводятся 2 числа — это интервал нахождения минимума полинома. Далее вводится параметр гу-

стота — данный параметр показывает, сколько целых чисел промежутка будет в начальной популяции (густота 1 означает, что в начальной популяции будут все целые числа в данном интервале, густота 0.1 означает, что в начальной популяции будет 10% всех целых чисел данного промежутка). Следующий параметр — вероятность мутации хромосомы при одной итерации.

2.3 Описание реализации алгоритма.

Для реализации алгоритма используются функции *find_suitability(value, coeffs)*, *make_descendants(generation)*, *mutate(sequence, credibility)*. Функция *find_suitability(value, coeffs)* позволяет определить приспособленность решения. Она принимает на вход точку *value* и список коэффициентов в том порядке, в каком они подавались на вход через терминал. Данная функция возвращает значение полинома в данной точке. Функция *make_descendants(generation)* принимает на вход поколение (последовательность точек) и выполняет формирование потомков. Для этого каждому элементу списка *generation* ставится в соответствие случайное число — индекс этого же списка. В каждой такой паре формируется 2 потомка (то есть общее число потомков в 2 раза больше общего числа родителей). Формирование происходит по следующему принципу: выбирается случайная точка в хромосомах родителей, первый потомок принимает все гены до этой точки у первого родителя и после этой точки — у второго родителя. Второй потомок, наоборот, принимает все гены первого родителя после данной точки и все гены второго родителя до этой точки. Такое действие продлевается для каждой пары родителей и формируется список потомков. Функция *mutate(sequence, credibility)* принимает последовательность хромосом *sequence* и вероятность возникновения мутации *credibility*. Выполняется обход последовательности, для каждого её элемента генерируется случайное число от 0 до 1, если это число меньше *credibility*, то происходит мутация. Случайным образом инвертируется какой-либо ген хромосомы. Для этого выполняется операция «исключающее или». Реализация алгоритма находится в файле *algorithm.py*.

ЗАКЛЮЧЕНИЕ

Созданы скетчи графического интерфейса и выбрана модификация генетического алгоритма. Её выбор был обоснован. Разработана программа, реализующая поставленную задачу на целых числах. Разработка осуществлена при помощи генетического алгоритма. Данный алгоритм даёт не всегда точное решение, однако его точность оказывается высокой (в большинстве случаев даёт правильный результат). Конечный результат, при этом, зависит от входных параметров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Панченко Т.В. Учебно-методическое пособие “Генетический алгоритмы”.