ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(СПБГУ)

Факультет прикладной математики – процессов управления

Лабораторная работа 2

# **Исследование генетического алгоритма. Изучение различных кодировок**

# **генотипа**

Вариант 8

Работу выполнил Студент 2 курса Бакалавриата

Карпенко Виктор Андреевич

Преподаватель: Дик Александр Генадьевич

Санкт-Петербург 2023

**Цель работы:** изучить и реализовать генетический алгоритм, изучить различные кодировки генотипа и провести тестирование работы алгоритма на наборе целевых функций.

**Ход работы**

Генетический алгоритм (ГА) – алгоритм поиска, моделирующий процесс естественного отбора в биологии. Он состоит из популяции особей, которые представляют собой решения задачи. Генотип особи представляет собой набор генов, которые определяют ее свойства. Операторы селекции, скрещивания и мутации взаимодействуют в популяции, создавая новые решения и обновляя их популяцию.

Обобщенная схема работы алгоритма:

1. Выбрать популяцию особей  
2. Оценить каждую особь в популяции с помощью целевой функции  
3. Провести операции селекции, скрещивания и мутации, чтобы создать новые особи  
4. Оценить новые особи и очистить популяцию от худших особей  
5. Повторять шаги 3-4 до тех пор, пока не будет достигнуто заданное количество итераций или не будет найден оптимальный результат

При формализации задачи будем считать генами координаты точки по осям. Тогда мутация будет происходить, варьируя значение генов, а новые особи в результате кроссинговера будут находиться в прямоугольнике, ограниченном точками родителей.

Программа написана на языке Python. В основе лежат классы индивида и популяции. Для них проработаны функции селекции, скрещивания(для популяции) и мутации, целевой функции (для индивида). В нашем случае целевая функция совпадает с заданной нам, минимум которой необходимо найти. Пользовательский интерфейс сделан с помощью библиотеки PySimpleGui. Листинг кода находится в приложении.

Для работы с алгоритмом было реализовано два класса – Individ и Genetic, поля и методы которых описаны в таблицах ниже.

Таблица 1, поля класса *Individ*

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| pos | np.ndarray((1, 2), dtype = int) | Положение |
| (start, end) | Пределы поиска |
| score | float | Индивидуальное решение |

Таблица 2, методы класса *Individ*

|  |  |  |  |
| --- | --- | --- | --- |
| Возвращаемый тип | Имя | Входные данные | Описание |
| None | calculateFunction | - | Обновление значения частицы |
| mutate | - | Мутация |

Таблица 3, поля класса *Genetic*

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| numberOfIndividums | int | Размер популяции |
| crossoverRate | float | Доля родителей в популяции |
| mutationSteps | Int | Количество шагов мутации |
| chanceMutations | float | Шанс мутации особи |
| BestScore | Float | Лучший результат |

Таблица 4, методы класса *Genetic*

|  |  |  |  |
| --- | --- | --- | --- |
| Возвращаемый тип | Имя | Входные данные | Описание |
| None | startGenetic | - | Выполнение итераций |
| reset | – | Сброс состояния роя |
| (Individ, Individ) | crossover | Выполнения кроссинговера |

**Контрольный пример**

При запуске программы пользователь вводит необходимые ему параметры с помощью переключателей(стрелочек).

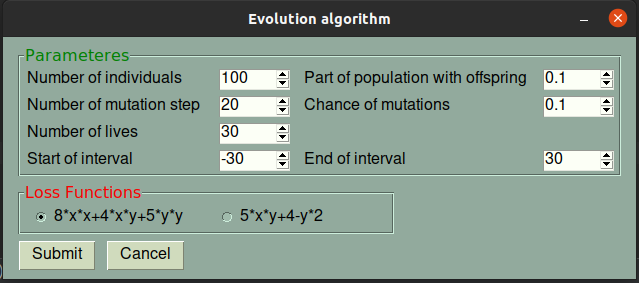


Рис.1 Старт приложения

После применения параметров и выполнения генетического алгоритма открывается панель анализа, где можно посмотреть на график функции, таблицу по поколениям, график средней и максимальной приспособленности, и анимацию «смещения» популяций.

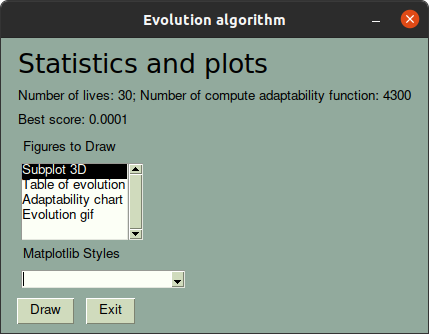


Рис 2. Выбор объекта для просмотра

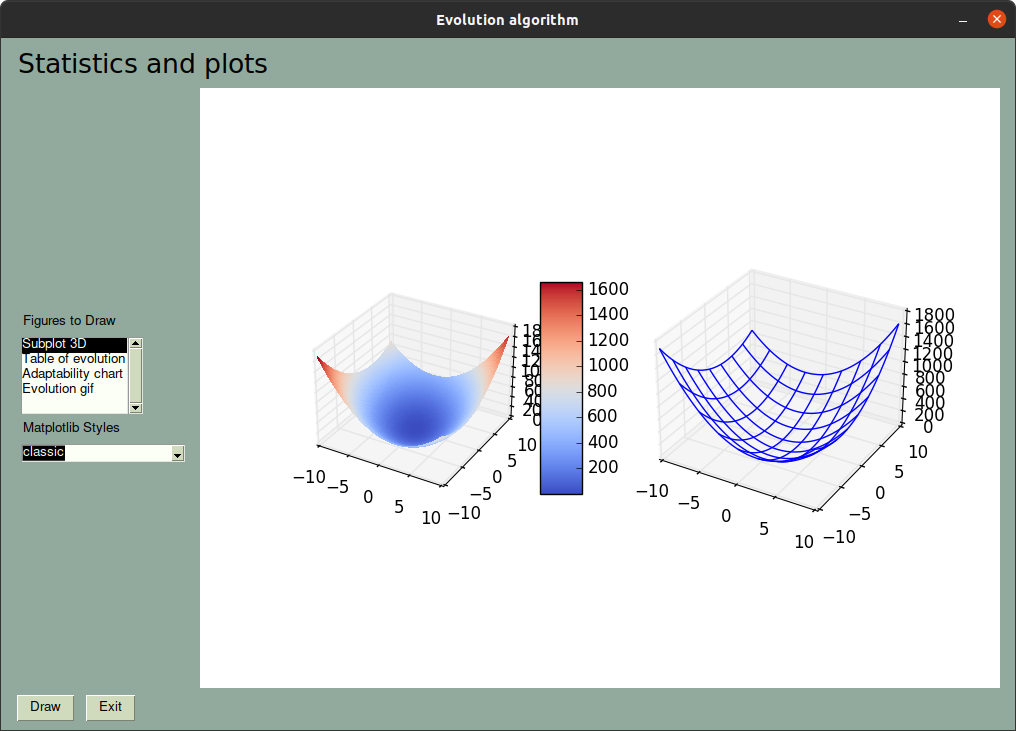
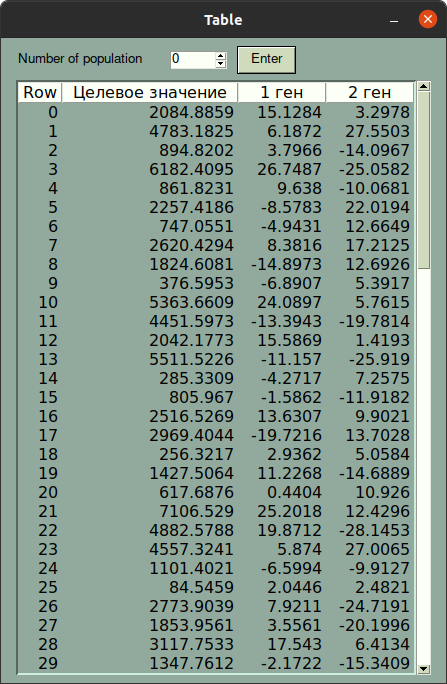


Рис 3. График функции

Рис 4. Таблицы поколений

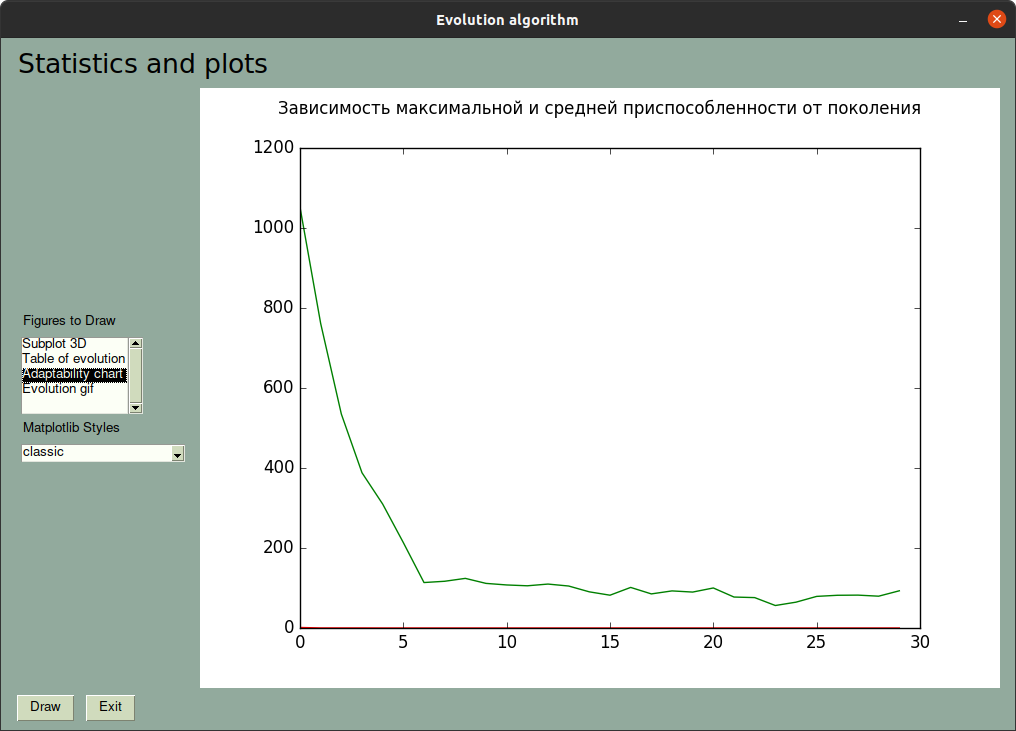


Рис 5. График приспособленности

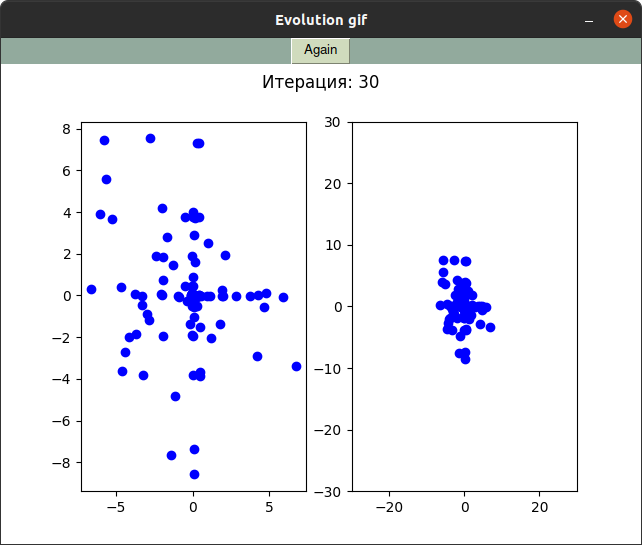


Рис 6. Анимация эволюции

**Анализ результатов**

Почти всегда генетический алгоритм сходится к правильному ответу на моей целевой функции. Поэтому весь анализ будет по вариации параметров алгоритма.

Так как каждый раз начальная популяция выбирается случайно, то прямой зависимости количества жизней от какого-то параметра наблюдать не можем.

Рис.7

Рис. 8

Рис. 9

На предыдущих графиках можно заметить, что не наблюдается никакой зависимости количество шагов эволюции от данных параметров.

Рис. 10

На последнем графике заметно, что слишком большое количество шагов мутации увеличивает количество шагов эволюции. Это можно объяснить тем, что точки, близкие к решению в результате мутации отдаляются от решения.

**Вывод**

В ходе выполнения работы был реализован генетический алгоритм на Python для решения задачи оптимизации, изучены различные параметры алгоритма и их влияние на результат. Тестирование работы алгоритма на наборе целевых функций показало его способность находить оптимальные решения.