Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №3 по дисциплине "Основы искусственного интеллекта"

Тема Генетические и эволюционные алгоритмы

Студент Варламова Е. А.

Группа ИУ7-13М

Оценка (баллы) _____

Преподаватели Строганов Ю.В.

Москва — 2023 г.

СОДЕРЖАНИЕ

Введение

Генетические и эволюционные алгоритмы представляют собой набор методов оптимизации, вдохновленных принципами естественного отбора и генетики. Они широко используются для решения сложных задач оптимизации, в моделировании, в финансовой математике, инженерии, компьютерной графике и дизайне.

Целью данной лабораторной работы является создание программы, которая с использованием алгоритмов оптимизации (генетического и роя частиц), аппроксимирует функцию, задаваемую пользователем в виде кривых Безье. Для этого необходимо решить следующие задачи:

— описать общие этапы функционирования системы;

— описать предлагаемый генетический алгоритм (выбранные функции мутаций, скрещивания и пр.);

— описать алгоритм роя частиц;

— привести особенности реализации ПО, решающего поставленную задачу;

— провести сравнение результатов работы генетического алгоритма и алгоритма роя частиц (среднеквадратичная ошибка, время вычислений) в зависимости от количества итераций.

1 Аналитическая часть

1.1 Общие этапы функционирования системы

Пользователь задаёт графически M точек на холсте. По этим точкам строится кривая Безье. Далее на кривой Безье выбирается N точек, которые подаются на вход алгоритмам аппроксимации (генетическому и роя частиц) и используются в них для вычисления ошибки.

Для аппроксимации использовались полиномы степени не выше 5. Алгоритмы аппроксимации подбирали наилучшие коэффициенты в смысле минимизации ошибки. Функция для вычисления ошибки:

$$E = \frac{1}{N} \sum_{i=1}^{N} |c_i \cdot x_i - y_i|, \tag{1.1}$$

где

— E – значение ошибки для коэффициентов c_i, $i \in [1; N]$, N – количество точек на кривой Безье;

— x_i, y_i, $i \in [1; N]$ – координаты точек кривой Безье.

Формализация задачи в виде IDEF0-диаграммы изображена на рисунке 1.1.

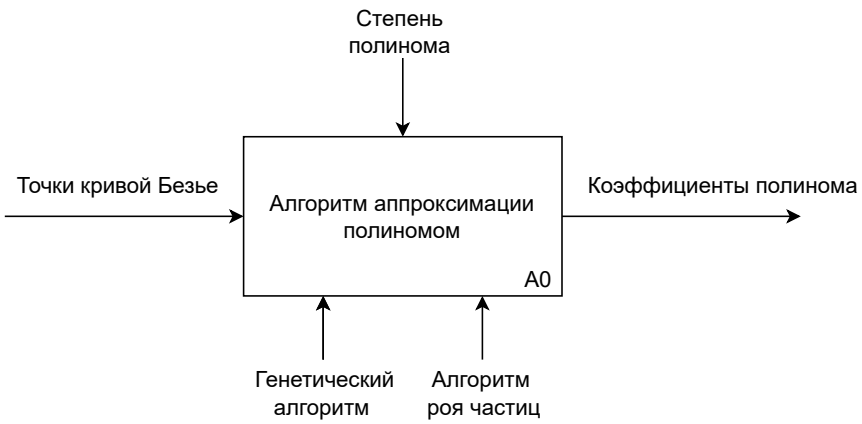

Рис. 1.1: IDEF0-диаграмма задачи

1.2 Генетический алгоритм

Генетический алгоритм является метаэвристическим методом оптимизации, вдохновленным естественным отбором и генетикой. Он используется для решения задач оптимизации и поиска, особенно в случаях, когда пространство поиска сложно или многомерно.

Опишем основные понятия, используемые в генетическом алгоритме.

— Популяция: Генетический алгоритм работает с популяцией индивидуумов (потенциальных решений). Популяция состоит из множества особей, представляющих потенциальные решения задачи оптимизации.

— Генотип: Генотип представляет генетическое кодирование индивидуума в терминах хромосом, генов и аллелей (фактически – размерность решения).

— Функция приспособленности: Каждый индивидуум оценивается по тому, насколько хорошо он соответствует целевой функции или заданной цели. При работе с оптимизационными задачами, функция приспособленности разрабатывается для оценки того, насколько хорошо решение соответствует желаемому результату.

— Количество поколений: определяет время жизни популяции и представляет собой количество итераций прохода по основным этапам алгоритма.

Опишем основные этапы генетического алгоритма с указанием функций, используемых в них в данной работе:

1. Инициализация: На этапе инициализации создается начальная популяция индивидуумов. Индивидуумы обычно инициализируются случайным образом или с помощью некоторых эвристик, зависящих от конкретной задачи.

2. Оценка приспособленности: Каждый индивидуум из начальной популяции оценивается с помощью функции приспособленности. Функция приспособленности используется для определения того, насколько хорошо решение соответствует цели оптимизации.

   В данной работе используется функция, описанная в 1.1.

3. Отбор (селекция): На этом этапе выбираются индивидуумы, которые будут иметь право размножаться и передавать свои гены в следующее поколение. Чем выше приспособленность индивидуума, тем выше вероятность его отбора.

В данной работе используется метод селекции «стохастически равный отбор», так как он гарантирует отбор не только индивида с большой приспособленностью, но и других особей с меньшей приспособленностью, что обеспечивает более равномерный отбор и сохраняет разнообразие популяции.

4. Скрещивание: Выбранные родительские индивидуумы комбинируют свои генотипы с использованием операторов скрещивания (кроссинговера), создавая потомство, которое, в свою очередь, составит новое поколение.

   В данной работе используется оператор скрещивания «одноточечное скрещивание», так как он позволяет частично сохранить структуру геномов родителей, при этом эффективно объединив генетическую информациию.

5. Мутация: На этом этапе случайным образом изменяются гены у некоторых индивидуумов популяции. Мутация помогает сохранить генетическое разнообразие в популяции, предотвращая сходимость к локальным оптимумам.

   В данной работе используется случайная мутация, что позволяет достичь более разноообразных результатов. Процент генов, которые подвергаются мутации – 40%.

6. Оценка нового поколения: Новое поколение индивидуумов оценивается с использованием функции приспособленности. Этот этап определяет, насколько успешно индивидуумы нового поколения решают задачу оптимизации.

7. Выбор лучших особей для формировнаия нового поколения: Выбераются лучшие особи из нового поколения для создания новой популяции. При этом в новую популяцию также могут быть помещены лучшие представители старого поколения.

   В данной работе лучшие родители помещаются в новую популяцию.

8. Завершение: Работа генетического алгоритма может завершиться по истечении определенного количества поколений, достижении желаемого критерия останова, либо при достижении определенной структуры популяции.

1.3 Алгоритм роя частиц

Алгоритм роя частиц (PSO) – это метаэвристический алгоритм оптимизации, основанный на моделировании движения роя частиц в пространстве поиска решений. Алгоритм роя частиц является эффективным способом оптимизации и

поиска, который моделирует поведение роя частиц в поисках оптимального решения в многомерном пространстве поиска.

Опишем основные этапы алгоритма роя частиц:

1. Инициализация: Начальная инициализация происходит путем задания начального положения и скорости каждой частицы в пространстве поиска решений случайным образом. Также определяются глобальное и личное лучшие решения.

2. Оценка приспособленности: Каждая частица в рое присваивает своей текущей позиции значение приспособленности с учетом целевой функции, которую необходимо оптимизировать.

   В данной работе используется функция, описанная в 1.1.

3. Обновление скорости и позиции: На этом этапе каждая частица обновляет свою скорость и позицию на основе своего личного опыта и опыта взаимодействия с окружающими частицами. Это обновление включает в себя использование двух основных компонентов: личная лучшая позиция частицы и глобально лучшая позиция, найденная в рое.

   Используется следующая формула пересчёта скорости:

$$V_j(t+1) = w \cdot V_j(t) + c_1 \cdot r_{1j}(t) \cdot [P_j(t)X_j(t)] + c_2 \cdot r_{2j}(t) \cdot [G(t)X_j(t)] \quad (1.2)$$

   где

   — $V_j(t)$ – скорость частицы j в момент времени t;
   — $X_j(t)$ – координаты частицы j в момент времени t;
   — $P_j(t)$ – личная лучшая позиция частицы j в момент времени t;
   — $G(t)$ – глобально лучшая позиция, найденная в рое на момент времени t;
   — r_{1j}, r_{2j} – случайные величины в интервале от [0,1);
   — w – показатель инерции движения роя
   — c_1, c_2 – коэффициенты, определяющие значимость своего лучшего решения и значимость лучшего решения роя соответственно.

   Для пересчёта позиции используется формула:

$$x_j(t+1) = x_j(t) + v_j(t+1) \quad (1.3)$$

4. Выбор лучших решений: На этом этапе происходит обновление личного и глобального лучших решений в рое. Если текущая позиция частицы лучше ее предыдущего лучшего положения, то это положение становится новым личным лучшим. Также происходит обновление лучшего решения для всего роя на основе лучших решений индивидуальных частиц.

5. Проверка условия останова: В процессе алгоритма роя частиц, данное условие проверяется для определения, достигнуто ли желаемое решение, либо для определения количества итераций, после которых алгоритм должен завершить свою работу.

6. Завершение: По истечении определенного количества итераций или при достижении желаемого критерия останова, алгоритм завершает свою работу и возвращает наилучшее найденное решение или его приближение.

Вывод

В данном разделе были описаны общие этапы функционирования системы, приведена формализация задачи, также описаны генетический алгоритм и алгоритм роя частиц.

2 Конструкторская часть

2.1 Генетический алгоритм

Схема генетического алгоритма представлена на рисунке 2.1.

Рис. 2.1: Схема генетического алгоритма

2.2 Алгоритм роя частиц

Схема алгоритма роя частиц представлена на рисунке 2.2.

Рис. 2.2: Схема алгоритма роя частиц

Вывод

В данном разделе были приведены схемы генетического алгоритма и алгоритма роя частиц, используемых для аппроксимации кривых безье.

3 | Технологическая часть

3.1 Выбор средств разработки

В качестве языка программирования был использован язык Python, поскольку этот язык кроссплатформенный и для него разработано огромное количество библиотек и модулей, решающих разнообразные задачи.

В частности, имеются библиотеки, включающие в себя алгоритмы оптимизации: генетический алгоритм и алгоритм роя частиц.

В данной работе были использованы библиотеки «pyswarms» [1], в которой реализован алгоритм роя частиц, и библиотека «pygad» [2], в которой реализован генетический алгоритм.

Для создания графиков была выбрана библиотека matplotlib [3], доступная на языке Python, так как она предоставляет удобный интерфейс для работы с данными и их визуализации.

3.2 Листинги ПО

В листинге 3.1 представлена инициализация параметров алгоритма роя частиц и реализация функции подсчёта ошибок.

Листинг 3.1: Инициализация параметров алгоритма роя частиц

```
def fitness_func(values):
    res = [np.mean(np.abs(np.polyval(coeffs, x) - y)) for coeffs in values]
    return res
def RunSwarm(xv, yv, dimension, iters = 100):
    x = list(xv)
    y = list(yv)
    options = {'c1': 0.5, 'c2': 0.3, 'w': 0.9}
    optimizer = ps.single.GlobalBestPSO(n_particles=100, dimensions=
       dimension, options=options)

    cost, pos = optimizer.optimize(fitness_func, iters=iters,verbose= False)
    return pos
```

В листинге 3.2 представлена инициализация параметров генетического алгоритма и реализация функции подсчёта ошибок.

Листинг 3.2: Инициализация параметров генетического алгоритма

```
def fitness_func(ga_instance, solution, solution_idx):
    coefficients = np.array(solution)
    predicted_values = np.polyval(coefficients, x)
    fitness = np.mean(np.abs(predicted_values - y))
    return 1 / fitness

def RunGA(x, y, dimension, iters = 500):
    x = xv
    y = yv

    num_generations = iters
    num_parents_mating = 10
    fitness_function = fitness_func
    sol_per_pop = 20
    num_genes = dimension

    parent_selection_type = "sss"
    crossover_type = "single_point"
    mutation_type = "random"
    mutation_percent_genes = 40
    parent_selection_type = "sss"
    keep_parents = 1

    ga_instance = pygad.GA(num_generations=num_generations,
                           num_parents_mating=num_parents_mating,
                           fitness_func=fitness_function,
                           sol_per_pop=sol_per_pop,
                           num_genes=num_genes,
                           parent_selection_type=parent_selection_type,
                           crossover_type=crossover_type,
                           mutation_type=mutation_type,
                           mutation_percent_genes=mutation_percent_genes,
                           keep_parents=keep_parents)

    ga_instance.run()
    solution, solution_fitness, solution_idx = ga_instance.best_solution()
    return solution
```

3.3 Примеры работы

На рисунках 3.1-3.4 приведены примеры работы ПО для кривой Безье, построенной на 3, 4, 5 и 6 точках. Синим цветом отображена ввёденная пользователем кривая Безье (построенная на введённых точках). Красным и зелёным цветом показаны аппроксимации этой кривой алгоритмами генетическим и роя частиц соответственно.

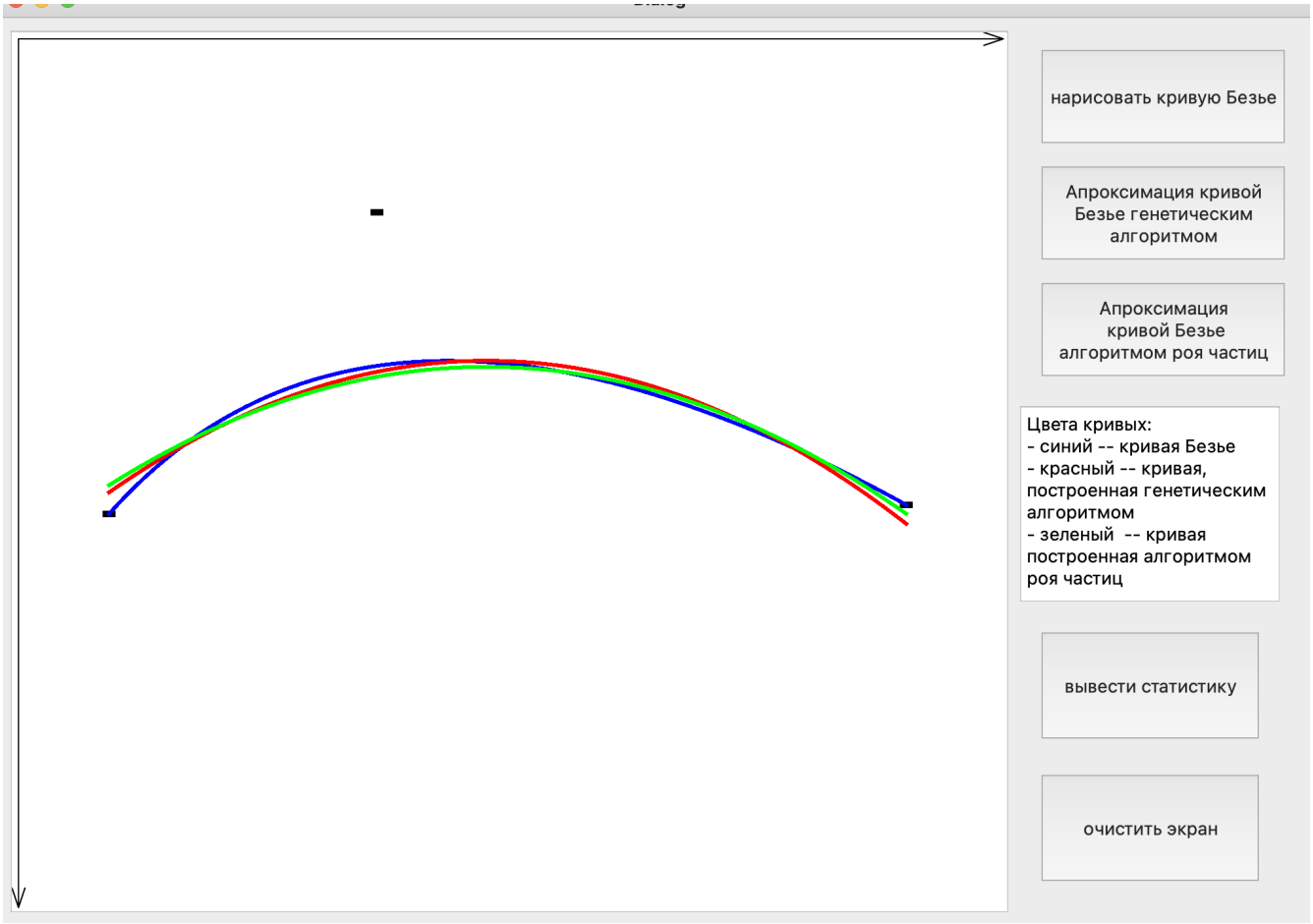

Рис. 3.1: Пример для кривой Безье, построенной на 3 точках

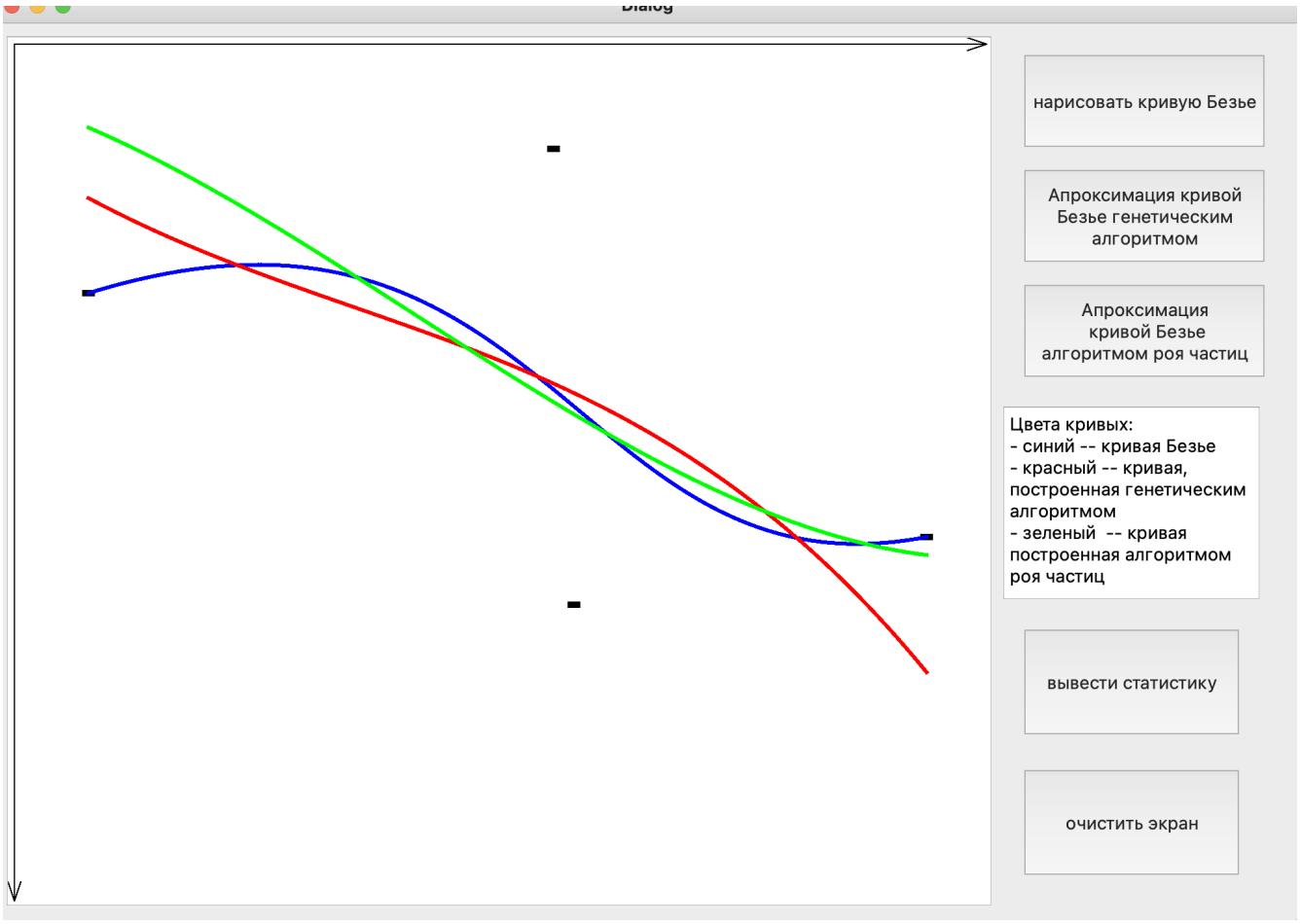

Рис. 3.2: Пример для кривой Безье, построенной на 4 точках

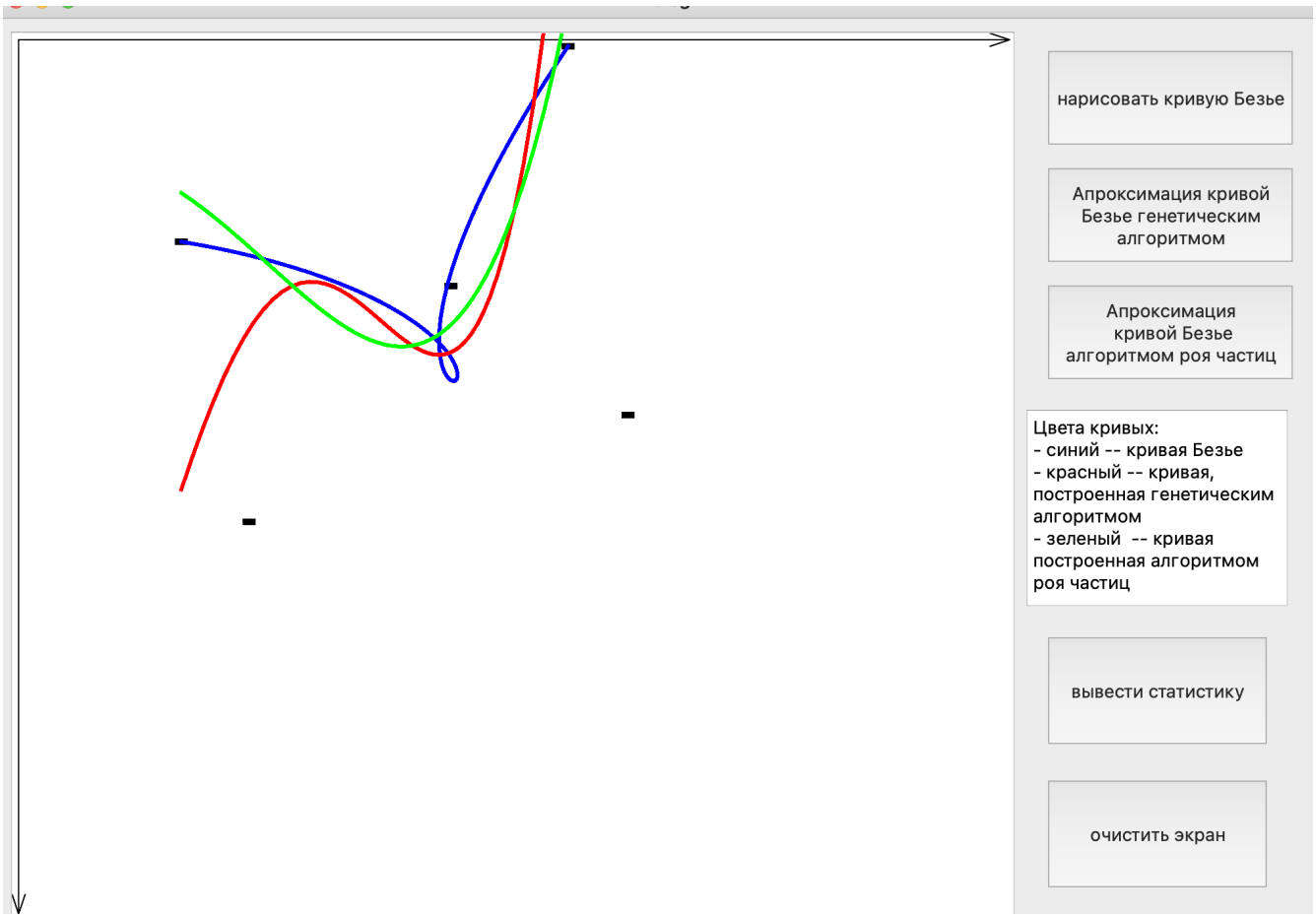

Рис. 3.3: Пример для кривой Безье, построенной на 5 точках

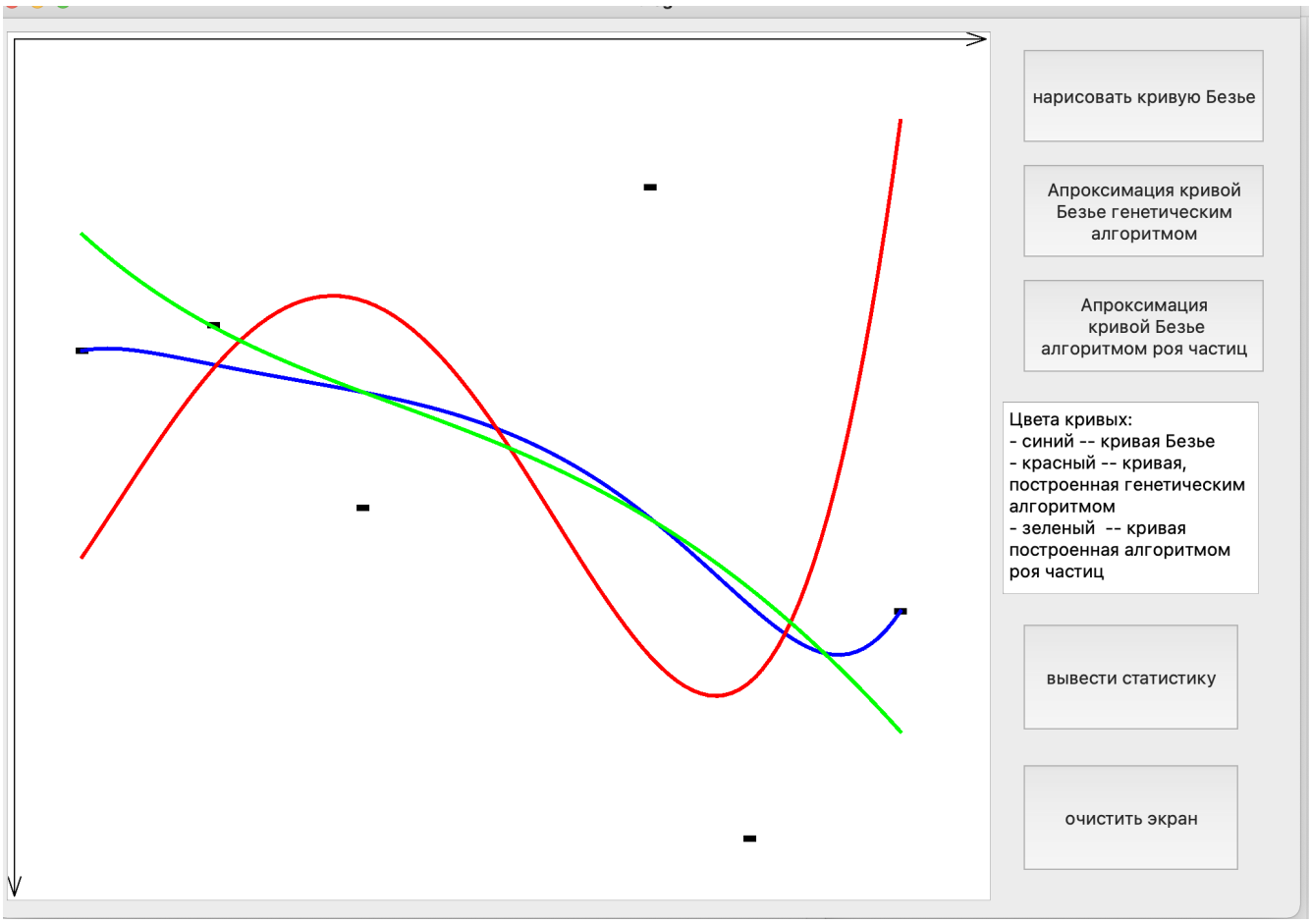

Рис. 3.4: Пример для кривой Безье, построенной на 6 точках

Видно, что аппроксимация алгоритмом роя частиц в большинстве случаев работает точнее.

Вывод

В данном разделе были обоснованы средства реализации ПО, реализовано ПО, решающее поставленную задачу и приведены детали реализации алгоритма роя частиц и генетического алгоритма.

4 Исследовательская часть

4.1 Сравнение алгоритмов аппроксимации кривых Безье

Цель: провести сравнение результатов работы генетического алгоритма и алгоритма роя частиц (среднеквадратичная ошибка, время вычислений) в зависимости от количества итераций (поколений в случае генетического алгоритма).

Результаты исследования приведены на рисунках 4.1 и 4.2 .

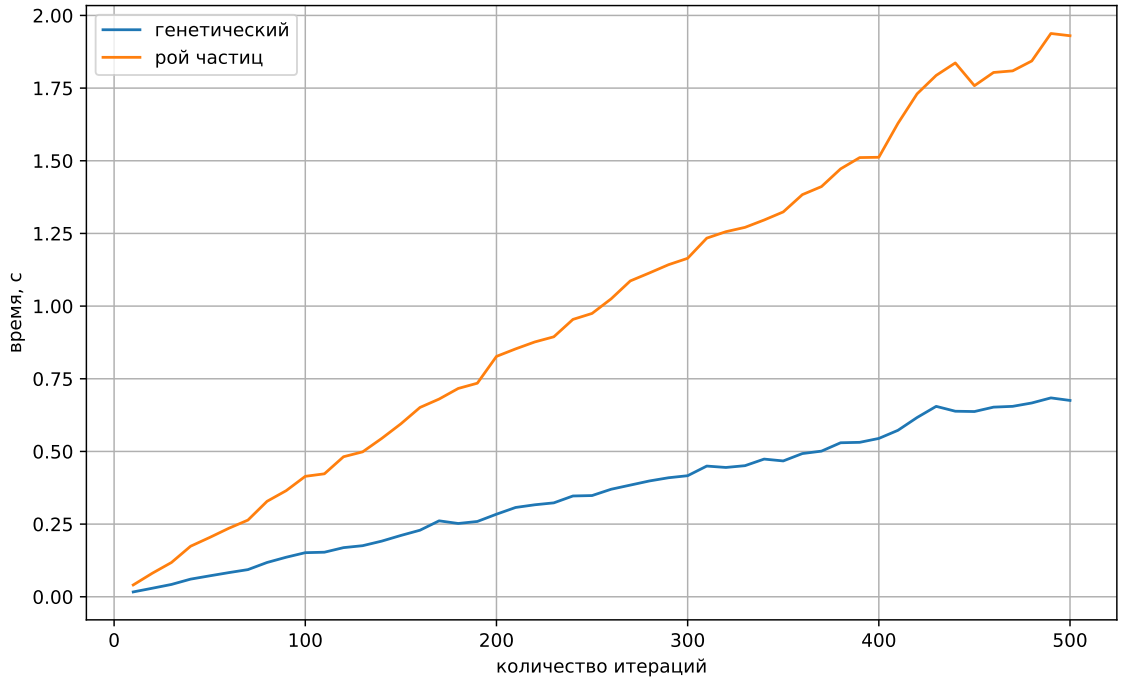

Рис. 4.1: Результат по времени работы

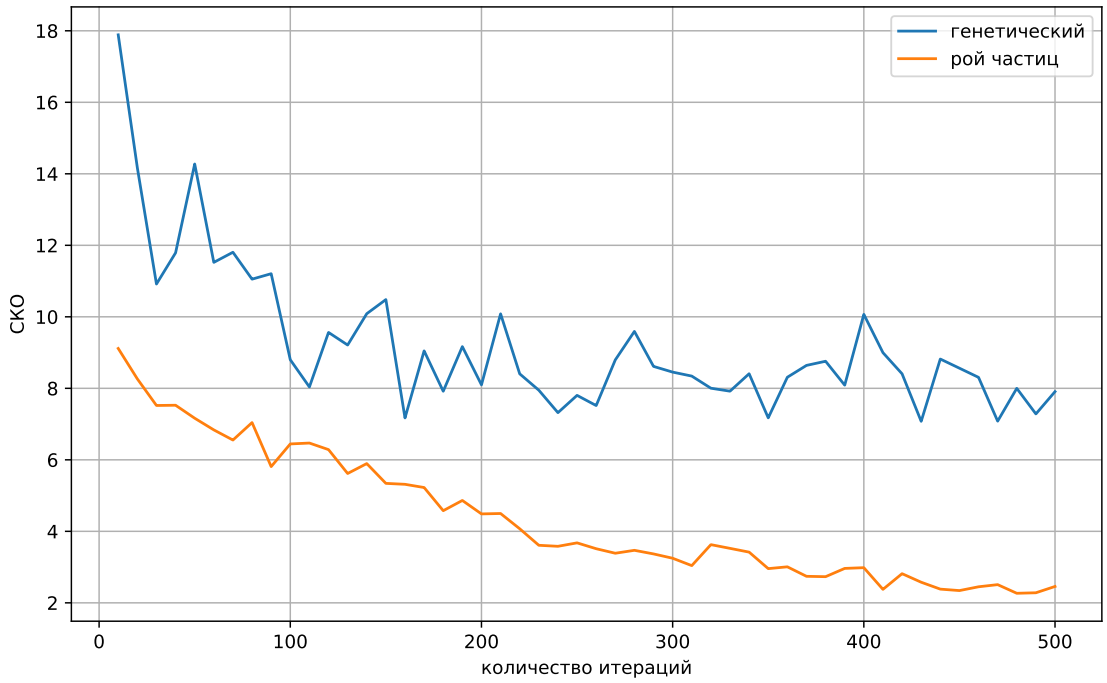

Рис. 4.2: Результат по СКО

Видим, что алгоритм роя частиц имеет меньшую ошибку, однако работает дольше генетического.

Вывод

В данном разделе было проведено сравнение результатов работы генетического алгоритма и алгоритма роя частиц (среднеквадратичная ошибка, время вычислений) в зависимости от количества итераций: было выяснено, что алгоритм роя частиц имеет меньшую ошибку, однако работает дольше генетического.

ЗАКЛЮЧЕНИЕ

Целью данной лабораторной работы являлось создание программы, которая с использованием алгоритмов оптимизации (генетического и роя частиц), аппроксимирует функцию, задаваемую пользователем в виде кривых Безье. Цель работы была достигнута. Для этого были решены следующие задачи:

— описаны общие этапы функционирования системы;

— описан предлагаемый генетический алгоритм (выбранные функции мутаций, скрещивания и пр.);

— описан алгоритм роя частиц;

— приведены особенности реализации ПО, решающего поставленную задачу;

— проведено сравнение результатов работы генетического алгоритма и алгоритма роя частиц (среднеквадратичная ошибка, время вычислений) в зависимости от количества итераций.

В результате сравнения алгоритмов было выяснено, что алгоритм роя частиц имеет меньшую ошибку, однако работает дольше генетического.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Miranda L. J. V.* PySwarms, a research-toolkit for Particle Swarm Optimization in Python // Journal of Open Source Software. — 2018. — Т. 3, вып. 21. — DOI: 10.21105/joss.00433. — URL: https://doi.org/10.21105/joss.00433.
2. *Gad A. F.* PyGAD: An Intuitive Genetic Algorithm Python Library. — 2021. — arXiv: 2106.06158 [cs.NE].
3. Библиотека визуализации данных matplotlib [Электронный ресурс]. — Режим доступа: URL: https://matplotlib.org (дата обращения: 13.12.2023).