

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики-процессов управления

Программа бакалавриата

“Большие данные и распределенная цифровая платформа”

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

**на тему «Исследование генетического алгоритма. Изучение различных
кодировок генотипа»**

Вариант – 2

**Студент гр. 23Б15-пу
Антонян А. А.**

**Преподаватель
Дик А.Г.**

Санкт-Петербург

2024 г.

Оглавление

1. Цель работы	3
2. Теоретическая часть	3
3. Описание задачи	4
4. Основные шаги программы	4
5. Описание программы	6
6. Рекомендации пользователя.....	7
7. Рекомендации программиста	7
8. Исходный код программы:.....	7
9. Контрольный пример	8
10. Исследование	9
11. Вывод.....	10
12. Источники	10

Цель работы

Целью лабораторной работы является исследование двух основных способов кодирования генотипа хромосом в генетическом алгоритме и проверка их эффективности. Этот алгоритм должен решить задачу нахождения минимума функции.

Теоретическая часть

Генетический алгоритм (ГА) — это метод оптимизации, вдохновленный механизмами естественного отбора и эволюции. Основная цель ГА — найти оптимальное решение для заданной функции путем эволюции популяции возможных решений. Алгоритм работает следующим образом:

1. **Инициализация популяции:** Создается начальная популяция из случайных решений (хромосом).
2. **Оценка пригодности:** Каждый индивид (хромосома) оценивается с помощью целевой функции, которая определяет его пригодность. Чем лучше значение функции, тем более "приспособленным" считается индивид.
3. **Селекция:** Из популяции выбираются индивиды для размножения. Обычно используются методы, как турнирный отбор, где случайные пары сравниваются, и более приспособленный индивид проходит в следующий этап.
4. **Кроссинговер:** Выбранные индивиды "скрещиваются" для создания потомков. Происходит обмен генами (параметрами) между родителями с некоторой вероятностью, что способствует разнообразию решений.
5. **Мутация:** Случайным образом изменяются некоторые гены потомков с небольшой вероятностью, чтобы избежать преждевременной сходимости и улучшить исследование пространства решений.

6. **Обновление популяции:** Потомки заменяют предыдущую популяцию, и процесс повторяется для заданного числа поколений.
7. **Выходное условие:** Алгоритм останавливается, когда достигается заданное число поколений или когда изменения в значениях становятся незначительными.

Также в алгоритме была выполнена модификация в части создания новой популяции посредством элитарного отбора. **Элитарный отбор** — это модификация генетического алгоритма, направленная на сохранение наиболее приспособленных индивидов на протяжении всех поколений. Данная техника обеспечивает перенос наилучших решений, найденных на каждом этапе, в следующее поколение без изменений. Основная цель элитарного отбора — улучшить эффективность и результативность генетического алгоритма, минимизируя потерю уже найденных хороших решений из-за случайного кроссовера или мутаций.

Описание задачи

Задача состоит в нахождении глобального минимума функции с помощью ГА. Алгоритм направлен на нахождение такой пары значений (x, y) , при которой значение функции Розенброка: $f(x,y)=100 \times (y-x^2)^2 + (1-x)^2$ становится минимально возможным. Глобальный минимум для этой функции находится в точке $(1,1)$, где значение функции равно 0.

Основные шаги программы

1. **Инициализация популяции:** создается начальная популяция случайных особей (пар значений x и y).
2. **Селекция:** отбор лучших особей из текущего поколения для создания следующего поколения, основываясь на их «приспособленности».

3. **Кроссинговер:** скрещивание выбранных пар особей для генерации потомков с некоторыми признаками обоих родителей.
4. **Мутация:** случайное изменение значений потомков для поддержания генетического разнообразия и предотвращения преждевременной сходимости.
5. **Элитарный отбор:** добавление в следующее поколение лучших особей из текущего поколения для гарантии, что оптимальные решения не потеряются.
6. **Оценка результатов:** на каждом этапе вычисляется значение функции для лучшей особи текущего поколения, чтобы отслеживать прогресс к оптимальному решению.



Рис 1. Блок-схема основной программы

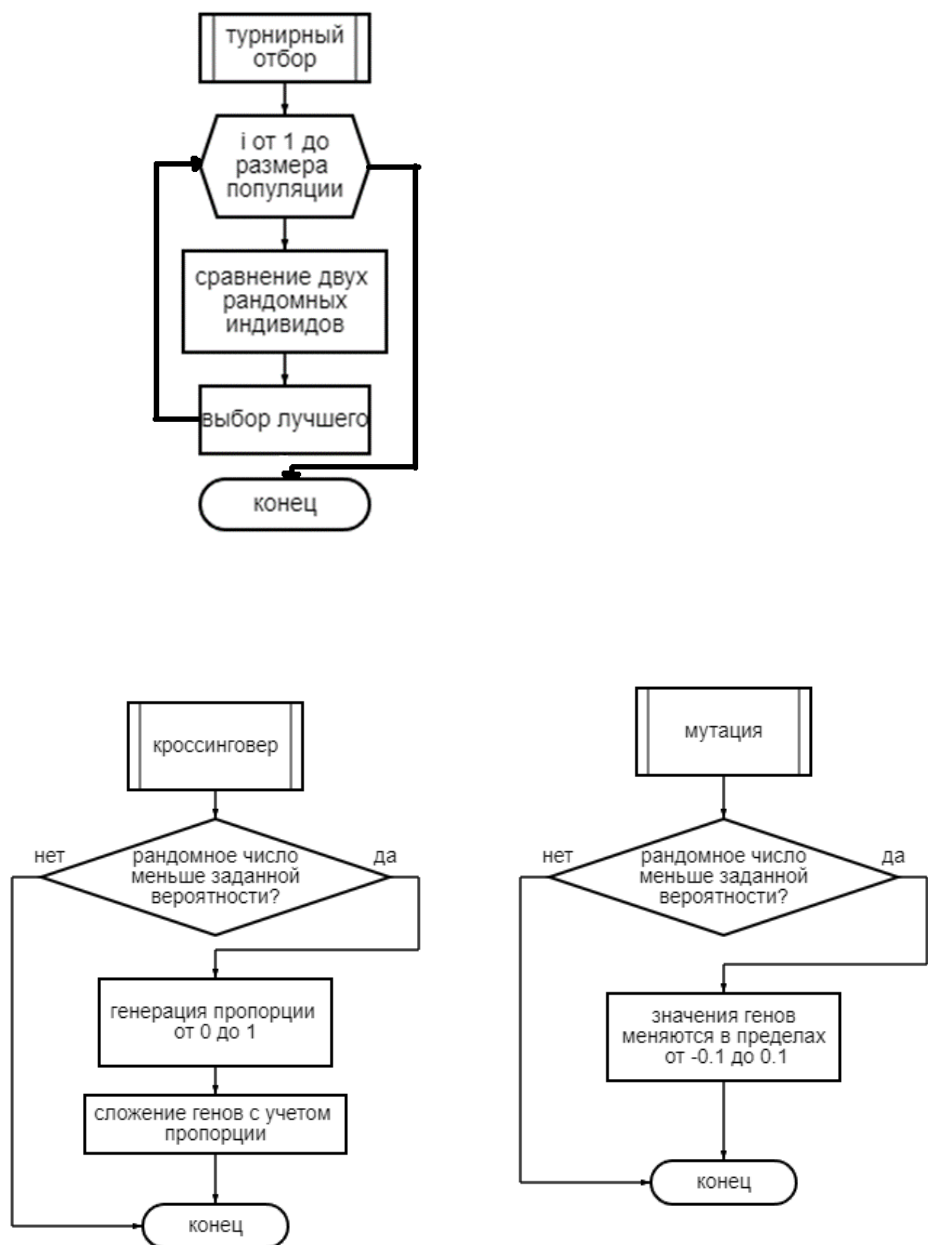


Рис 2. Блок-схемы подпрограмм

Описание программы

Программная реализация написана на языке Python 3.12.2 с использованием библиотеки numpy[1] и встроенного модуля tkinter[2]. В процессе разработки программы использовался следующий модуль:

Функция	Описание	Возвращаемое значение
initialize_population()	Создает популяцию.	list
select()	Проводит турнирный отбор особей	list
crossover()	Скрещивает особи и создает кроссинговер	tuple
mutate()	Создает мутацию в генах индивида	tuple
genetic_algorithm()	Главная функция, включающая в себя все остальные. Реализует ГА	tuple

Рекомендации пользователя

Для запуска программы убедитесь, что у вас установлен Python и необходимые библиотеки, такие как numpy [1]. Код можно запустить в среде разработки или через командную строку, используя консоль для настройки параметров и генерации данных. Запуск программы производится через файл genetic_algorithm.py.

При запуске программы вам будет предложено выбрать параметры запускаемого алгоритма. Вводите ответы соответствующие поля в GUI. Результаты работы будут представлены в таблице.

Рекомендации программиста

Для поддержания актуальности и работоспособности программы используйте последние версии библиотек, особенно numpy[1]. Применяйте практики надлежащего именования переменных и функций для улучшения читаемости кода.

Исходный код программы:

<https://github.com/ArseniiAntonin/spbu-algorithms-and-data-structures>

Контрольный пример

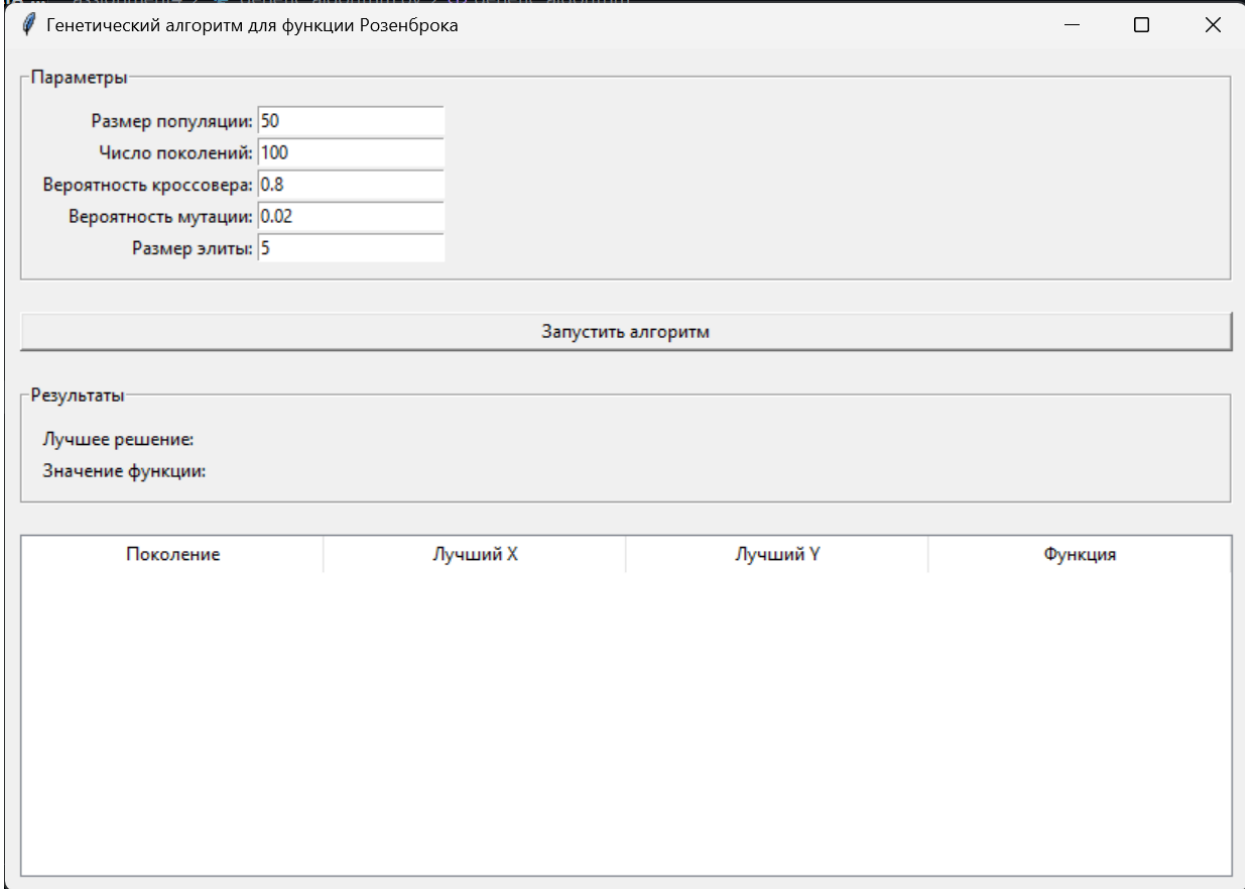
1. Запуск программы

Для запуска программы используйте файл `genetic_algorithm.py`.

Программа загружает GUI и позволяет пользователю настроить параметры ГА.

2. Выбор параметров ГА

После запуска программы пользователю будет предложено выбрать параметры (Рис. 3).



Поколение	Лучший X	Лучший Y	Функция
-----------	----------	----------	---------

Рис 3. Пример выбора параметров

3. Обработка данных и вывод результатов

После выбора параметров программа запускает алгоритм, а затем выводит результаты в таблице (Рис. 4).

Результаты			
Лучшее решение: X: 1.0027, Y: 1.0068			
Значение функции: 0.0002			
Поколение	Лучший X	Лучший Y	Функция
0	-6.9728	52.4035	1494.8709
1	-6.9728	52.4035	1494.8709
2	-6.9728	52.4035	1494.8709
3	-1.4513	1.8334	13.4551
4	-1.4513	1.8334	13.4551
5	-1.4513	1.8334	13.4551
6	-1.4513	1.8334	13.4551
7	-1.4513	1.8334	13.4551
8	-1.4513	1.8334	13.4551
9	-1.4513	1.8334	13.4551

Рис 4. Результат работы алгоритма

Исследование

В рамках данной лабораторной работы было проведено сравнение работы алгоритма с модификацией или без. ГА – эвристический алгоритм, поэтому 100% сходимость гарантировать нельзя. На основе проведенных тестов можно сделать вывод, что модификация значительно улучшает сходимость алгоритма, а модифицированный алгоритм быстрее приходит к наиболее точному ответу. Имея следующие параметры (Рис.5), алгоритм без элитарного отбора пришел к верному ответу 5 раз из 10.

Параметры	
Размер популяции:	50
Число поколений:	100
Вероятность кроссовера:	0.8
Вероятность мутации:	0.5
Размер элиты:	5

Рис. 5 Экспериментальные параметры

Модифицированный алгоритм с теми же параметрами сошелся 8 раз из 10 на случайных запусках. Можно сделать вывод о увеличении эффективности алгоритма с такой модификацией.

Вывод

В рамках данной работы был разработан алгоритм для нахождения глобального минимума функции. Программа была протестирована на функции Розенбаума. Реализованный алгоритм обеспечивает возможность оптимизации различных функций, например поиск экстремумов или решение NP полных задач. В части создания новой популяции была реализована модификация с помощью элитарного отбора, что помогло сделать каждое новое поколение не хуже предыдущего.

Источники

1. Numpy documentation // Numpy URL: <https://numpy.org> (дата обращения: 27.10.2024).
2. Tkinter documentation // Tkinter URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения: 28.10.2024)