**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Кафедра “фундаментальная информатика и информационные технологии”**

**ОТЧЕТ**

**по лабораторной работе №4**

**по дисциплине «Алгоритмы и структуры данных»**

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

**Вариант – 5**

**Студент гр. 22Б16-пу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Олизько С.С.**

**Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дик А.Г.**

**Санкт-Петербург**

**2023 г**

Оглавление

[**Цель работы:** 3](#_Toc153215487)

[**Задача:** 3](#_Toc153215488)

[**Теоритическая часть:** 3](#_Toc153215489)

[**Алгоритм метода:** 5](#_Toc153215490)

[**Описание программы:** 6](#_Toc153215491)

[**Рекомендации для пользователя:** 6](#_Toc153215492)

[**Контрольный пример:** 8](#_Toc153215493)

[**Анализ результатов работы алгоритма и вводных условий:** 8](#_Toc153215494)

[**Вывод:** 10](#_Toc153215495)

[**Источники:** 10](#_Toc153215496)

[**Листинг программы с детальными комментариями:** 11](#_Toc153215497)

# **Цель работы:**

Исследование двух основных способов кодирования генотипа хромосом в генетическом алгоритме и проверка их эффективности.

# **Задача:**

Изучить особенности кодирования генетических алгоритмов Написать программу поиска минимума функции, выбрав вариант тестовой функции из таблицы. Протестировать программу на выбранной тестовой функции. Написать отчет.

# **Теоритическая часть:**

Генетический алгоритм (ГА) представляет собой эвристический оптимизационный метод, вдохновленный процессами биологической эволюции. Разработанный в области искусственного интеллекта, ГА эффективно применяется для решения задач оптимизации и поиска.

**Основные принципы Генетического Алгоритма**

1. Популяция и Генотипы:

Популяция: Начальное множество потенциальных решений, известных как "особи", формирует популяцию. Каждая особь представляет собой кандидата на оптимальное решение.

Генотип: Представление решения в виде генов или строк, который может быть изменен в процессе эволюции.

1. Приспособленность:

Оценка, насколько успешно особь решает заданную задачу. Чем лучше решение, тем выше приспособленность.

1. Селекция:

Определение того, какие особи будут участвовать в создании следующего поколения. Особи с более высокой приспособленностью имеют больший шанс быть выбранными.

1. Скрещивание (Кроссовер):

Обмен генетической информацией между двумя особями для создания потомства. Эмулирует биологический процесс рекомбинации ДНК.

1. Мутация:

Случайные изменения генетической информации в особи. Позволяет внести разнообразие в популяцию и избежать застревания в локальных оптимумах.

1. Эволюция:

После создания нового поколения путем селекции, скрещивания и мутации, процесс повторяется. Алгоритм эволюционирует в поиске более приспособленных решений.

**Применение в Оптимизации**

Генетические алгоритмы успешно применяются в различных областях оптимизации, таких как:

* Функциональная оптимизация: Поиск глобального экстремума для функций.
* Комбинаторная оптимизация: Решение задач, связанных с комбинаторикой, таких как задачи о рюкзаке и задачи о планировании.
* Нейронные сети: Обучение параметров нейронных сетей.

# **Алгоритм метода:**

**1. Инициализация:** Задание параметров алгоритма: размер популяции, количество поколений, вероятность мутации, размер турнира для селекции, тип кодировки, границы поиска. Создание начальной популяции особей с использованием функции initialize\_population.

**2. Основной цикл эволюции:**

**2.1 Вычисление приспособленности:** Для каждой особи в популяции вычисляется значение функции приспособленности с использованием функции calculate\_fitness.

**2.2 Турнирная селекция:** Отбор особей для скрещивания с использованием функции tournament\_selection.

**2.3 Скрещивание (Кроссовер):** Создание потомства из выбранных особей с использованием функции crossover.

**2.4 Мутация:** Процесс мутации генотипов с использованием функции mutate.

**2.5 Обновление популяции:** Замена старой популяции новой, созданной в результате селекции, скрещивания и мутации.

**2.6 Обновление интерфейса:** Периодическое обновление визуального интерфейса с использованием функции update\_population\_table для отображения текущей популяции.

**2.7 Пауза и обновление лучшего результата:** Введена пауза для обновления интерфейса. Обновление лучшего результата с использованием функции update\_best\_result\_label.

**3. Завершение работы алгоритма:** Вывод лучшего решения, его координат и значение функции приспособленности.

**4. Опциональное выполнение итераций:** Возможность выполнения дополнительных итераций по запросу пользователя.

**5. Остановка алгоритма:** Возможность прервать выполнение алгоритма по запросу пользователя с использованием функции stop\_algorithm.

**6. Обработка ошибок:** Обработка и вывод ошибок, которые могут возникнуть в процессе выполнения.

**7. Завершение программы:** Завершение работы программы после выполнения всех итераций или прерывания пользователем.

# **Описание программы:**

Программа реализована на языке python 3.10 с использованием следующих

пакетов: tkinter, functools, threading, time, random. В генетическом алггоритме используются 10 функций. В таблицах описание функций генетического алгоритма.

*Таблица 1: функции*

|  |  |  |
| --- | --- | --- |
| **Имя функции** | **Тип возвращаемого значения** | **Описание функции** |
| initialize\_population | list (Individual) | Инициализация начальной популяции для генетического алгоритма. Возвращает список особей (экземпляров класса Individual). |
| generate\_random\_binary\_for\_objects | str | Генерация случайного бинарного числа для кодирования указанного числа объектов. Возвращает строку с бинарным числом. |
| is\_point\_within\_bounds | bool | Проверяет, находится ли точка в заданных границах. Возвращает True, если точка в границах, иначе False. |
| generate\_genotype | list | Генерация генотипа в соответствии с выбранной кодировкой. Возвращает сгенерированный генотип. |
| calculate\_fitness | None | Вычисление значения функции приспособленности для особи. Не возвращает значения, а присваивает его атрибуту fitness. |
| decode\_genotype | tuple | Декодирование генотипа в соответствии с выбранной кодировкой. Возвращает кортеж с координатами точки. |
| tournament\_selection | tuple | Турнирная селекция для выбора особей для скрещивания. Возвращает кортеж выбранных особей для скрещивания. |
| crossover | Individual | Скрещивание двух особей. Возвращает потомка после скрещивания. |
| mutate | None | Мутация генотипа особи. Не возвращает значения, а изменяет генотип переданной особи. |
| genetic\_algorithm | tuple | Генетический алгоритм для оптимизации функции приспособленности. Возвращает кортеж с лучшим результатом и его значением. |

# **Рекомендации для пользователя:**

**Запуск программы:**

Убедитесь, что на вашем компьютере установлен Python.

Установите необходимые библиотеки: tkinter (для графического интерфейса), pandas, random и datetime. Выполните команду pip install pandas.

Запустите код, чтобы активировать графический интерфейс и начать использование генетического алгоритма.

**Настройка параметров генетического алгоритма:**

Введите параметры алгоритма, такие как размер популяции, количество поколений, вероятность мутации, размер турнира для селекции, тип кодировки и границы поиска.

Удостоверьтесь, что значения параметров корректны перед запуском.

**Запуск генетического алгоритма:**

Нажмите кнопку "Старт", чтобы начать выполнение генетического алгоритма с заданными параметрами.

**Остановка алгоритма:**

В любой момент вы можете нажать кнопку "Стоп", чтобы прервать выполнение алгоритма.

**Выполнение дополнительных итераций:**

Используйте поле для ввода "Выполнить итерации", чтобы указать количество дополнительных итераций, которые вы хотите выполнить.

Нажмите "Выполнить итерации", чтобы выполнить указанное количество итераций.

**Отображение результатов:**

После завершения выполнения алгоритма, вы увидите лучший результат (координаты точки и значение функции) в соответствующем разделе интерфейса.

**Таблица популяции:**

В разделе "Результаты" отображается таблица с информацией о текущей популяции: номер особи, геном особи и значение приспособленности.

**Исходный код программы доступен по ссылке ниже:**

<https://github.com/StephanOlizko?tab=repositories>

# **Контрольный пример:**

*Рис 1: пример окна программы*

# **Анализ результатов работы алгоритма и вводных условий:**

Тесты проводились с 15% вероятностью мутации и с размером турнира в размере 30% от размера популяции.

**Вещественный**

*Таблица 2: тесты вещественного генома*

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 10 | 50 | (5.787321315647901, 5.731907718693096) | 2.551372887590508 |
| 20 | 50 | (5.067180382506338, 5.140771885603176) | 0.7563257677447138 |
| 30 | 50 | (5.02622939404128, 5.699598538094829) | 0.09032970086907145 |
| 40 | 50 | (5.002082159690984, 6.00684823988011) | 0.00006423994537056 |

*Таблица 3: тесты вещественного генома*

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 50 | 10 | (4.544841536798366, 5.255701069413931) | 1.3826578045678604 |
| 50 | 20 | (5.001977371199578, 5.70279257751691) | 0.0883478919664854 |
| 50 | 30 | (5.000364610578188, 6.000010248648917) | 0.0000531868529711 |
| 50 | 40 | (5.000413516655275, 6.000549078439268) | 0.0000009854712292 |

**Бинарный**

*Таблица 4: тесты бинарного генома*

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 10 | 50 | (5, 7) | 1 |
| 20 | 50 | (5, 6) | 0 |
| 30 | 50 | (5, 6) | 0 |
| 40 | 50 | (5, 6) | 0 |

*Таблица 5: тесты бинарного генома*

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 50 | 10 | (4, 4) | 8 |
| 50 | 20 | (5, 7) | 1 |
| 50 | 30 | (5, 6) | 0 |
| 50 | 40 | (5, 6) | 0 |

**Вещественный геном:**

При исследовании влияния размера популяции на качество решения видно, что увеличение числа особей до 50 приводит к улучшению результатов. С увеличением числа итераций генетический алгоритм сходится к оптимальному решению, и значение функции в лучшей точке стремится к нулю.

**Бинарный геном:**

В отличие от вещественного генома, бинарный способ кодирования ограничивает представление только целыми числами. Результаты тестов показывают, что даже с небольшим количеством итераций (10), бинарный геном способен предоставлять высококачественные решения. Однако, важно отметить, что значения в лучшей точке ограничены целочисленными значениями.

# **Вывод:**

Проведено исследование двух основных способов кодирования генотипа хромосом в генетическом алгоритме.

Генетический алгоритм эффективен для оптимизации как вещественных, так и бинарных геномов. Увеличение размера популяции и числа итераций оказывает положительное влияние на качество решения и для вещественного и для бинарного генома. Бинарный геном обеспечивает стабильные результаты, даже при ограниченном количестве итераций.

Важно учитывать, что бинарный способ кодирования ограничивает точность представления результатов, что делает вещественный геном предпочтительным в случаях, где требуется высокая точность в оптимальной точке.

# **Источники:**

<https://habr.com/ru/articles/128704/>

<https://docs.python.org/3/library/tkinter.html>

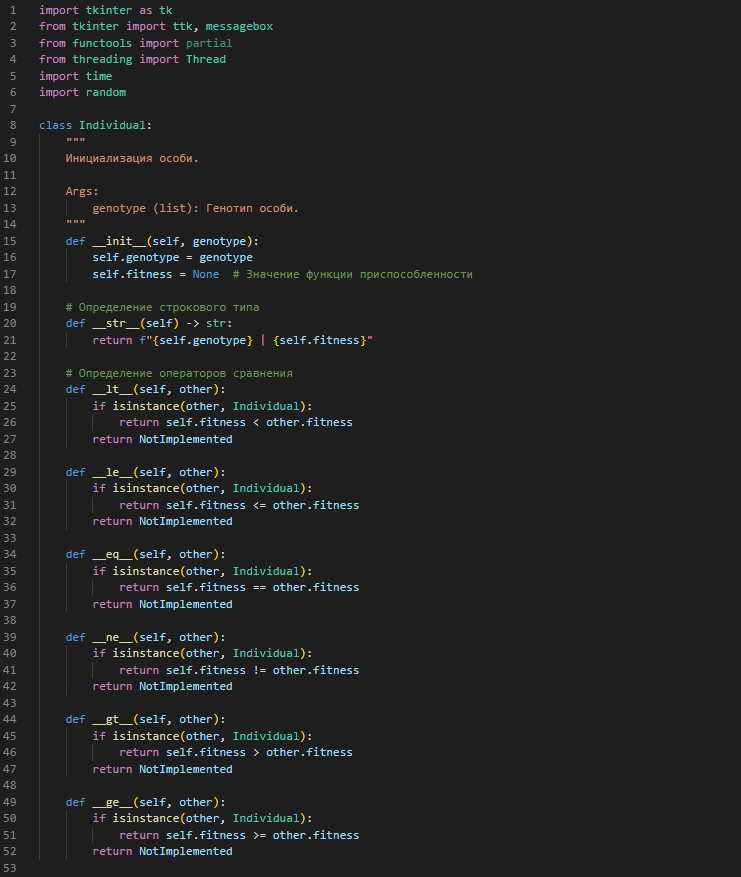
<https://docs.python.org/3/library/functools.html>

<https://docs.python.org/3/library/threading.html>

<https://docs.python.org/3/library/time.html>

<https://docs.python.org/3/library/random.html>

# **Листинг программы с детальными комментариями:**



*Рис 2: листинг*

**

*Рис 3: листинг*

**

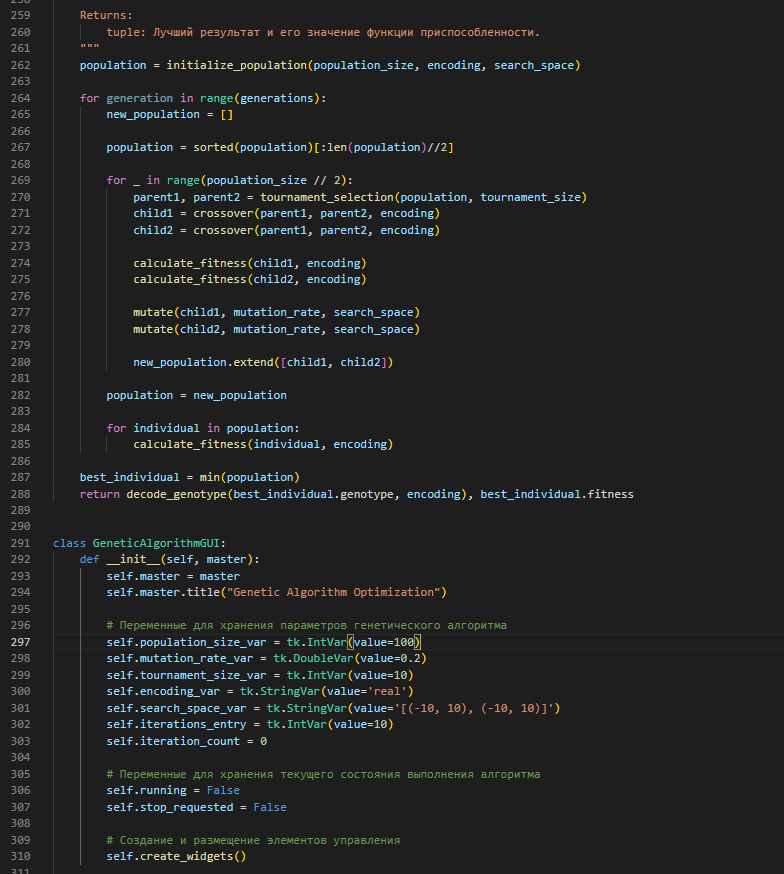
*Рис 4: листинг*

**

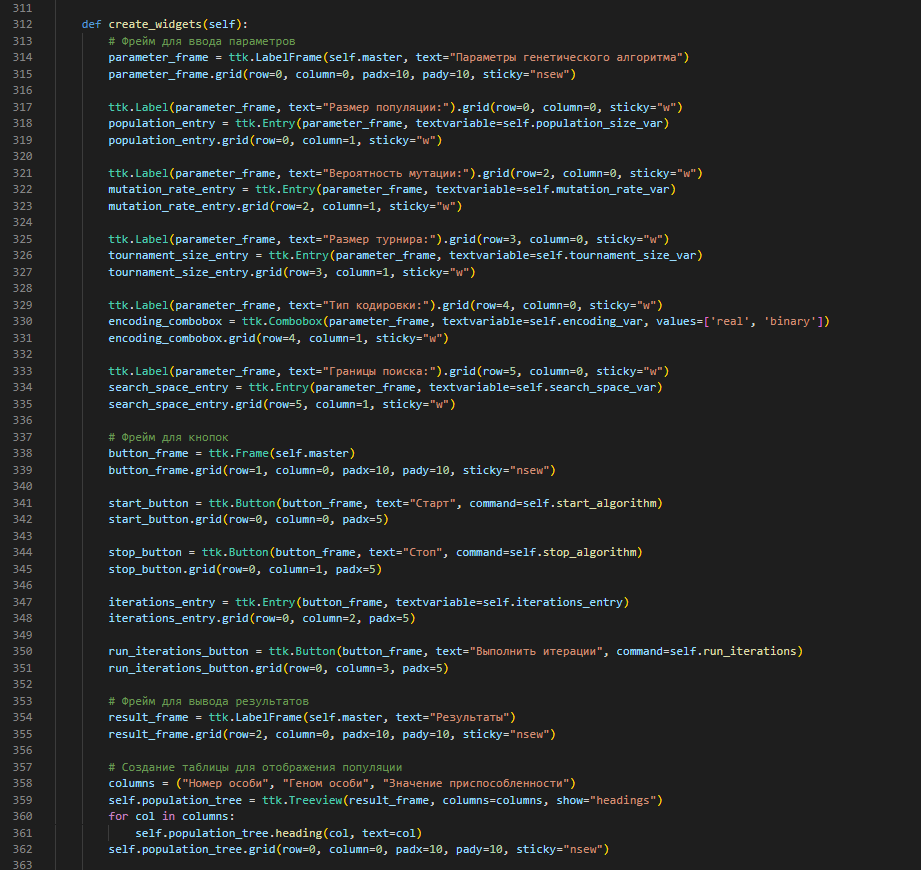
*Рис 5: листинг*

**

*Рис 6: листинг*

**

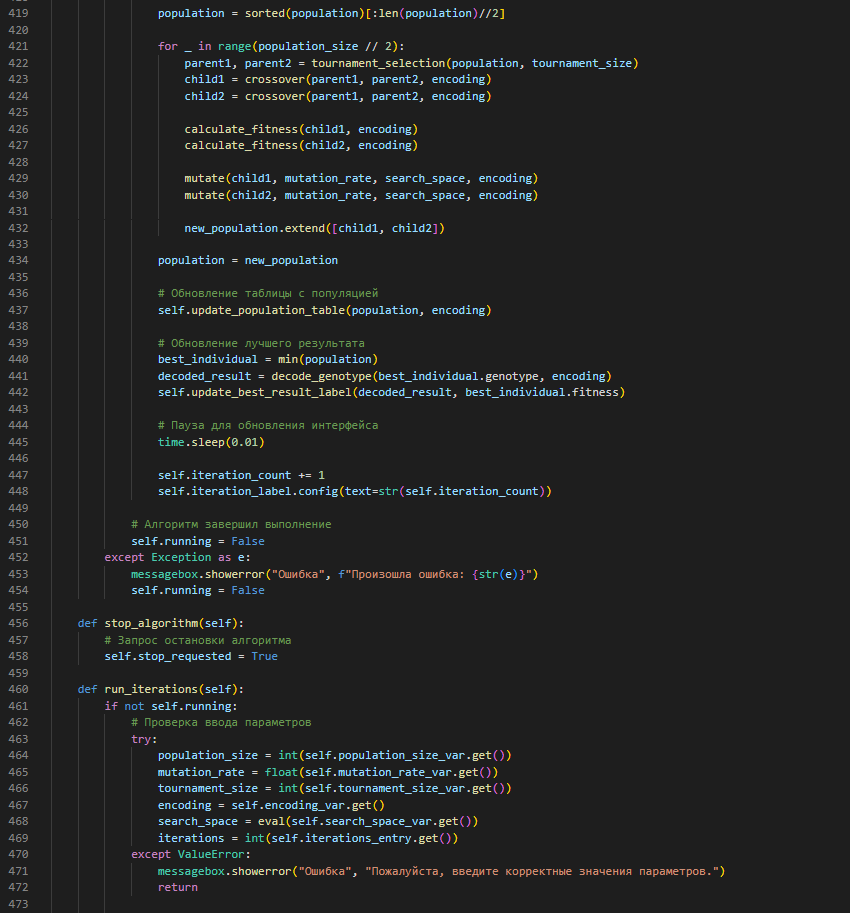
*Рис 7: листинг*

**

*Рис 8: листинг*

**

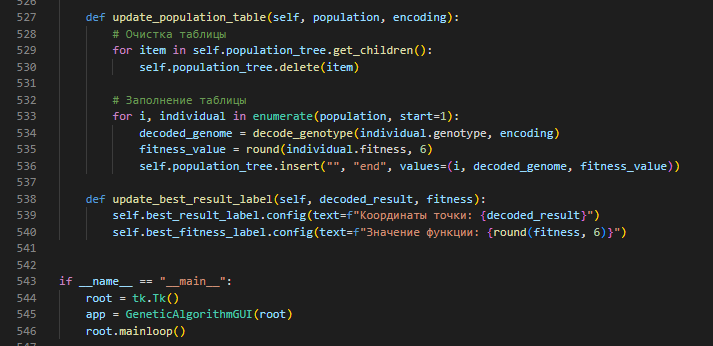
*Рис 9: листинг*

**

*Рис 10: листинг*

**

*Рис 11: листинг*

**

*Рис 12: листинг*