

**МИНОБРНАУКИ РОССИИ**  
**Санкт-Петербургский государственный**  
**электротехнический университет**  
**«ЛЭТИ» им. В.И. Ульянова (Ленина)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по учебной практике**  
**Тема: Генетические алгоритмы**

Студент гр. 3384

Питеров А. В.

Преподаватель

Жангиров Т. Р.

Санкт-Петербург

2025

## **Цели учебной практики**

- Ознакомиться с генетическими алгоритмами на примере решения оптимизационной задачи.
- Разработать полноценное приложение с GUI для работы над задачей, экспериментов с генетическими алгоритмами.

## **Задача (Вариант 16)**

Задано  $N$  фигур тетрамино (фигуры из игры тетрис) различных форм. Необходимо разместить их на поле шириной  $L$ , так, чтобы количество рядов, занимаемых фигурами, было минимально.

## **Содержание**

Итерация №1.....	3
Цели итерации.....	3
Постановка работы.....	3
Компоненты генетического алгоритма.....	4
Работа над GUI.....	6
Итерация №2.....	9
Цели итерации.....	9
Попытка реализации ГА.....	9
Знания для GUI.....	9
Вывод.....	10
Итерация №3.....	11
Цели итерации.....	11
Прототип ГА.....	11
Выводы.....	12
итерация №4.....	13
Цель.....	13
Рефакторинг, отладка и исследование ГА.....	13
Функции GUI.....	17
Вывод.....	17
Выводы.....	18

## **ИТЕРАЦИЯ №1**

### **Цели итерации**

- Определить как реализовывать ГА.
- Разработать прототип GUI.
- Срок до 29 июня.

### **Постановка работы**

Отчитаны требования к итоговой программе.

Установлено, что задачами программы является: решение пользовательской задачи с помощью ГА (в текущей работе — одну задачу данную вариантом) с возможностью настройки ГА (выбор параметров, конструирование из готовых блоков), демонстрировать поиск решения (графиками и визуализацией решения).

Отчитана теория касающаяся тетриса. Задача уточняется тем, что: фигуры бывают 7-ми типов (O, I, T, Z, S, L, J); их можно располагать на поле и вращать на  $90^\circ$ , они должны быть «прижаты к нижней части» поля.

Поле прямоугольное имеет один установленный параметр — ширину, длину предложено оценить, чтобы не работать с бесконечностью.

Целевая функция — количество занимаемых фигурами рядов для конкретного решения, может быть вычислена алгоритмически обходом поля.

Отчитана предложенная методическое пособие.

## **Компоненты генетического алгоритма**

Так как задача состоит из фигур и поля, решение задачи можно представить в двух тривиальных видах, в табл. 1 приведено их сравнение.

Таблица 1 — Сравнение представлений

Представление 1	Представление 2
<b>Суть</b>	
Назначение каждой фигуре своей позиции на поле и ориентации.	Назначение каждой клетке поля принадлежности какой-либо фигуре.
<b>Параметры для кодировки</b>	
Позиция на поле и ориентация (4 поворота) — целые числа.	Принадлежность (каждой фигуре выдать свой номер) — целое число.
<b>Представление кодировки, затраты на память</b>	
Каждой фигуре сопоставить пару параметров и объединить в строку по некоторому принципу (последовательно, случайно).  Память = количество фигур * вес параметров	Для каждой ячейки хранить параметр.  Память = количество ячеек в поле * вес параметра
<b>Кодировка данных параметров влечёт наличие некорректных решений — проблемы кодировок</b>	
Выход фигуры за границы поля.  Наложение фигур друг на друга.	Образование несуществующего типа фигуры.  Фигура несвязна (разделена на части).  Неправильное количество фигур.
<b>Как реагировать на некорректное решение?</b>	
Предполагается, что подавляющее количество решений будет некорректным в обоих представлениях. Поэтому отбрасывание приведёт к случайному поиску, исправление равноценно разработке переборных алгоритмов, а другие кодировки если и есть, то	

нетривиальны. Выбран подход со штрафами. Соотношения между штрафами — есть гиперпараметр, определяющий траекторию работы ГА.

### Как проверить корректность решения?

Построение решения с фиксацией выхода за пределы поля и наложения.	Представлять исследуемую фигуру графом. Использовать обходы графов, изоморфизм, подсчитывать корректные фигуры.
--	---

Первый подход технически легче, поэтому будет реализован в первую очередь. Ожидается долгая сходимость, поэтому выбрано несколько операторов ГА для экспериментов, см. табл. 2.

Таблица 2 — Характеристики и операторы ГА

Популяция	Фиксированной длины с повторениями
<b>Оператор</b>	<b>Выбор (В порядке реализации)</b>
Отбор родителей	Панмиксия (случайно) Ин(аут)бридинг (метрика - расстояние Хэмминга)
Скрещивание	Дискретная рекомбинация k-точечный кроссинговер
Мутация	Мутация инверсией (выборки, участков) Мутация обменом (генов, участков)
Отбор в новую популяцию	Отбор усечением (с/без элитаризма)

Как мера сохранения решений: банк на основе очереди с приоритетом фиксированной длины, использующийся при «плохих» преобразованиях популяции.

## Работа над GUI

Выделены следующие подходы к модификации поведения ГА: изменение исходного кода программы; выбор готовых решений для операторов ГА (встроенных в код программы, подключаемых извне — плагины); описание операторов как скриптов; конструирование потока исполнения ГА в визуальной форме. Решено реализовать второй подход как доступный.

При разработке интерфейса положено, что все касающиеся настроек должно быть убрано в отдельные меню-окна, а основное окно посвящено исключительно работе алгоритма. Так продуман интерфейс, перечисление всех окон и требований к ним см. табл. 3.

Таблица 3 — элементы GUI

Элемент	Описание
<b>Основное окно</b>	Содержит в себе ниже следующие панели
Панель управления	Содержит в себе кнопки для перехода к настройкам входящих данных и ГА, секции для управления ГА (следующий шаг, переход/пауза), для управления популяцией (сохранить/загрузить, сбросить), для управления графиками (сброс, сохранение).
Панель процесса	Содержит две вкладки: область просмотра и таблица популяции. Таблица агрегирует всю информацию рассчитываемую программой по каждому решению и позволяет вывести конкретное в область просмотра, которая позволяет увидеть его в графическом, кодированном и декодированном виде.
Панель информации	Содержит в себе секцию с требуемой информацией: номер популяции, средние и лучшие значения приспособленности и штрафа; секцию с графиками

	приспособленности и штрафов за весь процесс работы.
<b>Всплывающие окно настройки входных данных</b>	Вызывается при нажатии кнопки настройки данных на панели управления: позволяет настроить источники данных (ручной, файл и случайно) и выбрать конкретный для текущей работы.
<b>Всплывающие окно настройки ГА</b>	Вызывается при нажатии кнопки настройки ГА на панели управления: позволяет настроить и выбрать ГА для текущей работы. Настройка разбита по вкладкам, в соответствие с операторами ГА, в которых можно выбрать оператор из реализованных в коде программы.
<b>Всплывающие окно перехода</b>	Вызывается при нажатии кнопки переход на панели управления: позволяет выбрать ограничение в количестве популяций как критерий остановки процесса и запустить его.

Определены глобальные данные: выбранные для текущей работы данные и ГА (настройка и внутренние состояние), текущая популяция, накопленная информация (построение графиков).

Определены связанные с ними потоки данных, что сведено в табл. 4.

Таблица 4 — Потоки данных

<b>Действие</b>	<b>Источник</b>	<b>Формат</b>
<b>Входные данные</b>		
Загрузить	Пользователь	Текстовый
<b>ГА (Настройка)</b>		
Загрузить/Сохранить	Пользователь/Программа	JSON
<b>Популяция (Решение)</b>		
Загрузить/Сохранить	Пользователь/Программа	Бинарный

Данные для графиков		
Сохранение	Пользователь	CSV

Определён порядок работы с программой, см. на рис. 1.



Рисунок 1 - порядок работы с программой

Выделены следующие способы реализации GUI:

- С использованием библиотеки на основном языке (C++)
- Связка со скриптовым языком (Python)
- web-интерфейс

Выбран первый подход, для чего сделан неглубокий обзор средств, в который попали, в скобках — причина отказа: Qt, wxWidgets, GTK (cairo для отрисовки), FLTK, ImGUI (необходимость в GPU) и некоторые другие. Из них для данной работы выбран Qt, поскольку он де-факто стандарт и обладает готовыми средствами для отрисовки графиков.

## **ИТЕРАЦИЯ №2**

### **Цели итерации**

- Разработать по плану элементы ГА, оформив в самостоятельную программу с возможностью выгрузки результатов работы (данные, популяция).
- Разработать часть программного интерфейса отвечающую за ввод и просмотр популяции.
- Срок до 2 июля.

### **Попытка реализации ГА**

Обдумана программная реализация каждого компонента на C++. Предположено, что следует осторожно отнестись к памяти: заранее выделить достаточно для популяции и её производных из операторов ГА, переиспользовать после отбора в новую популяцию.

Предпринята попытка реализации — неудачно. Возникли проблемы в архитектуре, для решения которых требуется больше времени.

### **Знания для GUI**

Поскольку пункт с ГА не был реализован, нет «кода», чтобы выполнить второй пункт.

Предпринята попытка реализации доступной части, что привело к осознанию слабых знаний в Qt, полученных ранее.

Так освоен принцип построения GUI с помощью Qt Widgets, связь кода и формы составленной графически в QtDesigner; работа с QIcon, создано представление о ресурсной системе Qt — файлах .qrc; подключение сигналов и открытие дополнительных окон; работа с графиками посредством QtCharts и QCustomPlot — для работы выбран первый как достаточный, использование

QPainter. Составлены примеры. Данные знания непосредственно нужны для достижения цели.

## **Вывод**

Необходимо закончить работу над компонентами ГА, после чего будет содержание для заполнения GUI.

Начальная проработка была слишком объёмной: можно было разделить на две итерации, закончив вторую с отлаженными основными структурами данных, методами их загрузки и выгрузки, генерации — перенеся конечную реализацию ГА на третью итерацию.

## ИТЕРАЦИЯ №3

### Цели итерации

- Закончить работу, начатую в прошлой итерации
- Срок до 4 июля

### Прототип ГА

Написана консольная программа применяющая ГА к задаче и выводящая метрики в файл CSV для построения графиков. Проведены эксперименты: ГА достигает решений не выходящих за поле, без наложений, но сильно разбросанных по полю.

Для программы реализованы примитивы задачи, такие как фигура, решение в обычной и кодированной формах, поле и функции-оценки использующие его для построения решения, что применяется для подсчёта штрафов, вспомогательные функции.

Решение представляет из себя список требуемых фигур с параметрами позиции и ориентации на поле. Для кодировки из задачи вычисляется необходимое количество бит на каждый параметр.

Кодированное решение, решение и метрики собраны в структуру особи.

Далее была реализована первая версия ГА, которая имела проблемы с фрагментацией памяти, что приводило к исключению типа `bad_alloc` при работе на 30-31 шаге работы ГА: исследовалось подбором параметров и инструментом `valgrind-massif`.

Для решения проблемы реализована ранее предложенная схема, попутно исправлены ошибки с генераторами случайных чисел и другие.

Программа разбита на модули.

## **Выводы**

- Необходимо объединить коды GUI и ГА, например подключив к QtCreator исходники или предоставив библиотеку с заголовками.
- Необходимо связать GUI и ГА, для этого отрефакторить и обобщить прототип.
- Необходимо провести эксперименты с ГА, чтобы улучшить результаты, рассмотреть следующие операторы.

## ИТЕРАЦИЯ №4

### Цель

- Провести рефакторинг кода ГА, провести отладку и исследование.
- После заполнить прототип GUI функционально для работы с ГА.
- Срок до 6 июля

### Рефакторинг, отладка и исследование ГА

Прототип ГА разрабатывался в одном файле для того, чтобы не отвлекаться на разбивку, систему сборки. В данной итерации код разбит на модули, добавлена система сборки cmake.

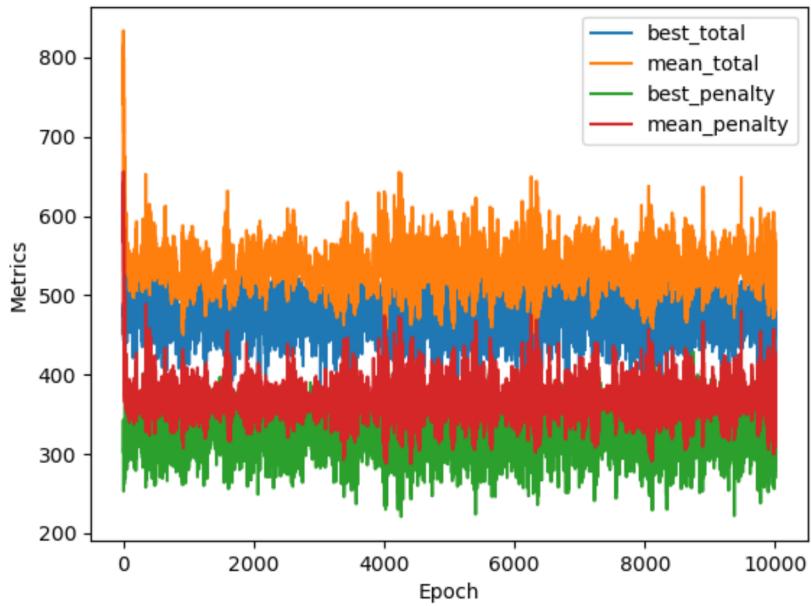
Для проверки корректности и улучшения работы проведён анализ работы ГА посредством интерактивных вычислений в python на некоторых входных данных.

Для проверки корректности был построен график следующих метрик: приспособленность лучшей особи и в среднем, штраф лучшей особи и в среднем, см. рис. 2а. Из него выявлено, что алгоритм работает неправильно: совершенно нет сходимости популяции к лучшим решениям, нет улучшения решений — колебания.

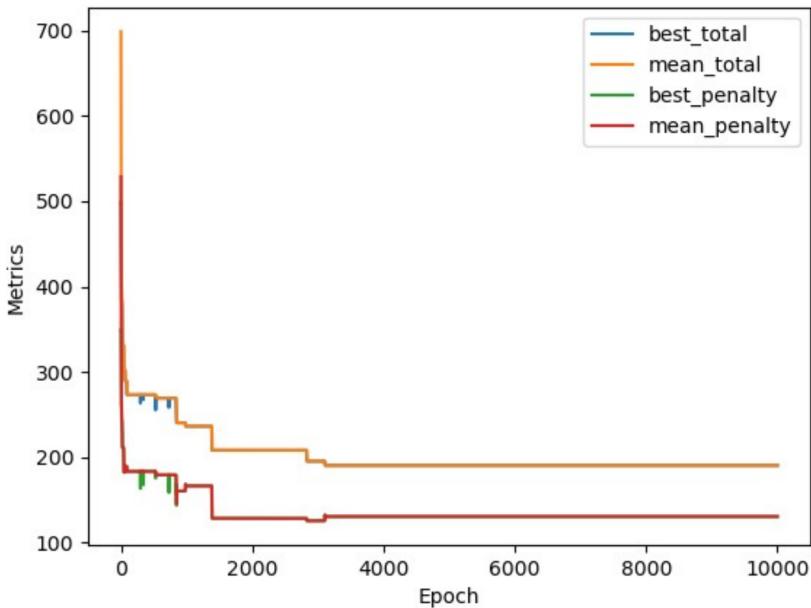
В ходе выполнения разбивки обнаружен ряд ошибок, исправление которых значительно повлияло на корректность работы ГА — см. рис. 2б.

Следующей целью стало исследование алгоритма на «постоянных» участках. Предположено, что на них популяция имеет низкое разнообразие, из-за чего страдает качество скрещивания.

Данное предположение подтверждено анализом битовой схожести решений см. рис. 3. Так выходит, что лучшее решение распространяется на все особи — популяция вырождается только в одно решение.



(а) — первый вариант ГА



(б) — второй (исправлений)

Рисунок 2 — График метрик

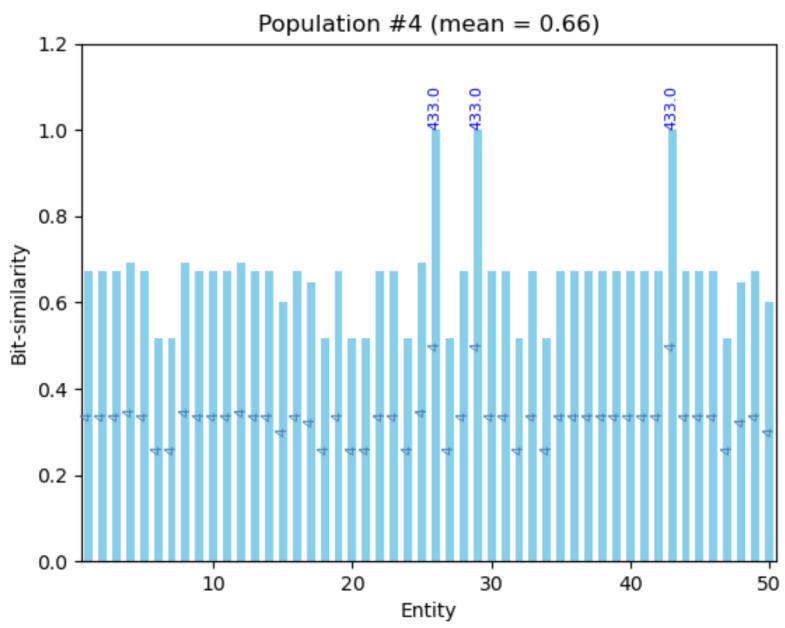
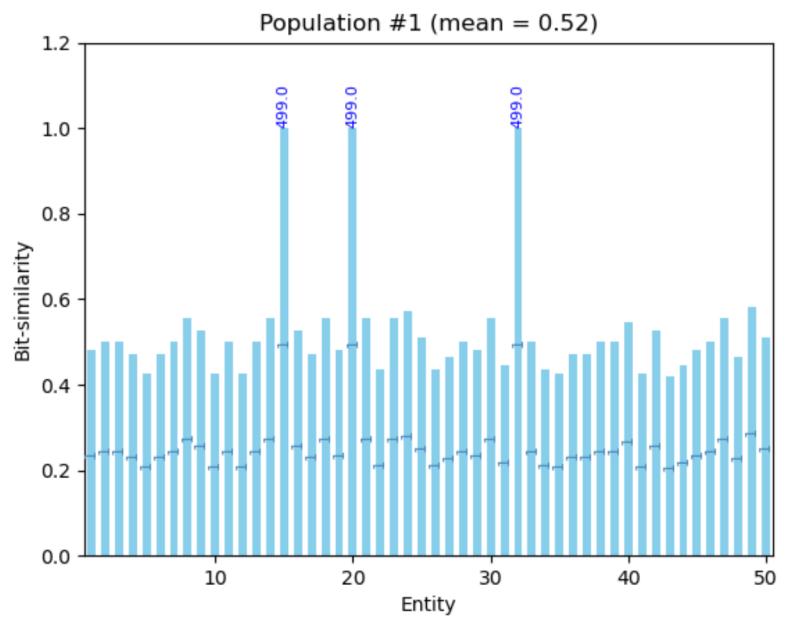


Рисунок 3 — Битовая сходимость популяции на эпохе  
(Числа в столбцах — эпоха появления генома, числа над столбцами — только  
над лучшими особями, их приспособленность)

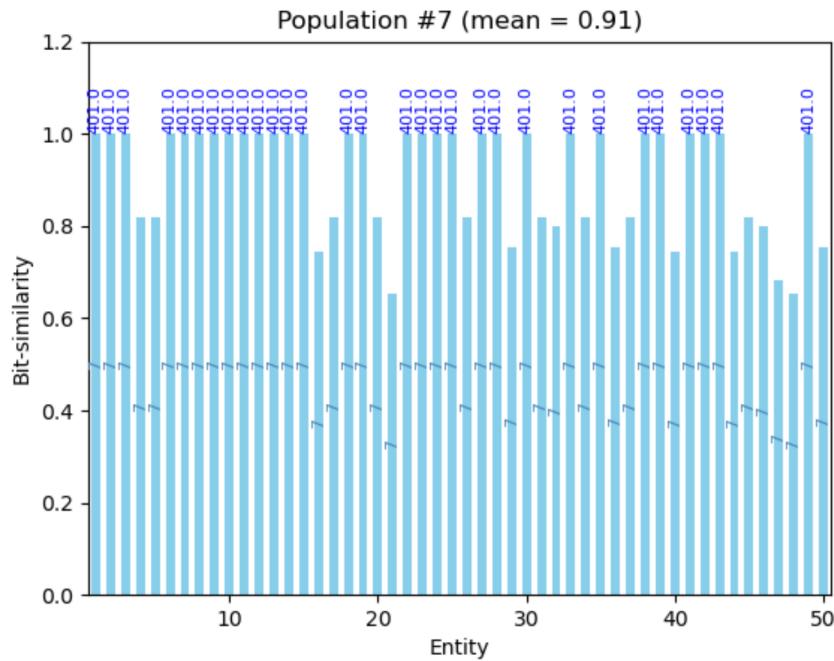


Рисунок 3 — Битовая сходимость популяции на эпохе (продолжение)

Проблему настройкой параметров ГА и реализованной части операторов решить не удалось.

Исследование проводилось на задаче:  
 $(Len=10; O=3, I=1, T=1, S=2, Z=0, L=1, J=2)$ . Код построения графиков см. в файле exploring.ipynb.

Отлаженный код ГА был собран в единый класс для предоставления коду GUI. Были объедены проект QtCreator-а и собственный код для дальнейшей работы над GUI в IDE.

## **Функции GUI**

Для данного блока осталось немного времени в итерации, поэтому он не выполнен целиком: разработаны прототипы-примеры для всех компонентов GUI, но они не собраны в едино и не представлены в репозитории.

## **Вывод**

- Отрефакторен и исправлен код ГА.
- Требуются следующие операторы ГА, для чего его необходимо обобщить.
- GUI необходимо собрать из меньших отработанных примеров.

## ВЫВОДЫ

Работа проведена в итеративной форме. В начале каждой итерации формулировались локальные цели.

Выявлена необходимость командой работы и субъективные пределы возможностей при реализации крупных проектов.

Проведена большая работа касающаяся теории генетических алгоритмов - в начале работы составлено подробное описание проектируемого алгоритма.

В ходе реализации генетического алгоритма с помощью исследования его поведения и ключевых метрик выявлены и исправлены ошибки в коде.

Выявлено, что задача работает с достаточно большими данными, из-за чего уделено внимание представлению решений и работе с памятью. Так решения кодируются наименьшим возможным числом бит, основной объем памяти выделяется один раз, и дальше используется повторно. Освоен новый инструмент для профилирования кучи программы.

Реализована консольная демонстрационная программа для протоколирования и взаимодействия с ГА.

Так освоены принципы работы генетических алгоритмов, способ их построения, получен опыт о возникающих при этом проблемах и способах их решения.

Проработан прототип графического интерфейса, разобраны примеры реализации всех его компонентов, составлены примеры для дальнейшего объединения. Итоговая программа остается в неполноценном виде.

Так освоены принципы разработки графических интерфейсов с использованием библиотеки Qt6 и его стандартных инструментов, модулей. Разобраны вопросы связки стороннего кода с данными инструментами.

**Отзыв**  
на учебную практику  
студента гр. 3384 Питерова А.

В ходе учебной практики студентом решалась задача о размещения фигур тетрамино при помощи генетических алгоритмов. В результате прохождения учебной практики было разработано ПО с графическим интерфейсом на языке C++. Питеров А. продемонстрировал навыки по проектированию приложений с графическим интерфейсом и понимание работы генетических алгоритмов. Все поставленные задачи были выполнены в полном объеме.

За учебную практику студент Питеров А. заслуживает оценку «**Отлично**»

Руководитель:

старший преподаватель каф. МОЭВМ Жангиров Т.Р.



06.07.2025