Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О.В. Непомнящий
подпись инициалы, фамилия
« ____» _____ 20 __ г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Программная система оптимизации комплектации технических систем на основе генетического алгоритма
тема

Руководитель	_____	доцент, канд.тех.наук	Н.Ю. Сиротинина
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		Э.О. Оюн
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	доцент, канд.тех.наук	В.И. Иванов
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2019

РЕФЕРАТ

Выпускная квалификационная работа по теме «Программная система оптимизации комплектации технических систем на основе генетического алгоритма» содержит 48 страниц текстового документа, 14 используемых источников, 20 иллюстраций, 3 таблицы, 1 приложение.

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ, ОПТИМИЗАЦИЯ КОМПЛЕКТАЦИИ, ИСКУССТВЕНЫЙ ИНТЕЛЛЕКТ, МАШИННОЕ ОБУЧЕНИЕ, ЭВОЛЮЦИЯ, КРОССИНГОВЕР, МУТАЦИЯ, ВИЗУАЛИЗАЦИЯ.

Цель работы: разработать приложение для оптимизации комплектации технических систем на основе генетического алгоритма.

Задачи:

1. Аналитический обзор и сравнение аналогов.
2. Выбор средств разработки.
3. Проектирование базы данных.
4. Разработка структуры приложения.
5. Реализация программного продукта.

Генетический алгоритм актуален в тех сферах, где необходимо найти оптимальное решение с небольшими затратами времени и ресурсов. Для того чтобы лучше узнать ход работы генетического алгоритма необходимо приложение, которое могло бы наглядно продемонстрировать процесс изменения популяций в ходе процесса эволюции.

В ходе выполнения работы были проанализированы различные программы оптимизации на основе генетического алгоритма. Была спроектировано и разработано приложение для оптимизации комплектации системного блока персонального компьютера на основе генетического алгоритма, позволяющая изменять настройки генетического алгоритма, выбирать характеристики технических систем, а также реализующая визуализацию процесса эволюции.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Темой бакалаврской работы является оптимизация комплектации технических систем на основе генетического алгоритма.

Генетический алгоритм является одним из способов машинного обучения, таким образом генетический алгоритм тесно переплетается с развитием искусственного интеллекта.

На сегодняшний день, развитие искусственного интеллекта является очень актуальной темой. Искусственный интеллект может применяться в любой сфере деятельности без особого вмешательства человека. Уже сегодня существуют различные интеллектуальные системы, однако они имеют узкую область применения. Одной из особенностей искусственного интеллекта является его способность к самообучению, которая позволяет самостоятельно совершенствоваться.

Помимо машинного обучения, генетический алгоритм актуален в тех сферах, где необходимо найти оптимальное решение с небольшими затратами времени и ресурсов.

Оптимизация комплектации технических систем необходима для выбора оптимального набора технических систем, удовлетворяющего заданным критериям. Оптимизация комплектации применяется практически во всех областях человеческой деятельности, связанных с использованием различных аппаратных устройств: медицине, военной промышленности, космических исследованиях и других.

При большом количестве доступных технических систем, поиск оптимального набора возможен только путем перебора всех возможных сочетаний технических систем, но такой подход займет большое количество времени.

Для компромисса между временем поиска набора технических систем и его оптимальностью могут использоваться различные методы, например,

динамическое программирование, метод ветвей и границ, генетический алгоритм и другие.

Целью дипломной работы является разработка приложения для оптимизации комплектации технических систем на основе генетического алгоритма.

В данной работе, в качестве примера выбрана задача комплектации системного блока персонального компьютера, с целью обучения студентов основам генетического алгоритма.

Задачи, которые необходимо решить для достижения цели:

1. Аналитический обзор и сравнение аналогов.
2. Выбор средств разработки.
3. Проектирование базы данных.
4. Разработка структуры приложения.
5. Реализация программного продукта.

Исходя из того, что приложение будет использоваться в обучении, необходимо чтобы оно имело следующий функционал:

- выбор характеристик технических систем;
- выбор методов селекции и скрещивания;
- обработка фактора совместимости технических систем;
- изменяемая вероятность мутации;
- изменяемое количество особей в популяции;
- визуальное отображение процесса эволюции.

Приложение должно иметь графический интерфейс и поддерживаться на операционных системах Windows и Linux актуальных версий.

1 Анализ предметной области

1.1 Описание классического генетического алгоритма

Генетические алгоритмы – процедуры поиска, основанные на механизмах естественного отбора и наследования [1, с. 124].

Определения, используемые в области генетического алгоритма:

Популяция – конечное множество особей [1, с. 126].

Особи – набор хромосом с закодированными в них множествами параметров задачи, т.е. решений (точки в пространстве поиска).

Хромосома – упорядоченная последовательность генов [1, с. 126].

Ген – атомарный элемент генотипа [1, с. 126].

Генотип или структура – набор хромосом данной особи. Следует учесть, что особь может быть представлена как генотипом, так и единичной хромосомой.

Фенотип – набор значений, соответствующих данному генотипу, т.е. декодированная структура или множество параметров задачи [1, с. 126].

Аллель – значение конкретного гена, также определяемое как значение свойства или вариант свойства [1, с. 126].

Локус – место размещения данного гена в хромосоме [1, с. 126]. Множество позиций генов – локи.

Генетический алгоритм широко используется в областях поиска оптимальных решений. В качестве задач, которые решаются при помощи генетического алгоритма, можно привести [2, с. 20]:

- задача об одномерном ранце;
- задача покрытия множества;
- задача дихотомического разбиения графа;
- задача о назначениях;
- задача коммивояжера.

Достоинства генетического алгоритма [3]:

- широкая область применения;
- возможность использования с неэволюционными алгоритмами;
- пригодность для поиска в сложном пространстве решений большой размерности;
- отсутствие ограничений на вид целевой функции;
- интегрируемость с искусственными нейросетями.

Недостатки генетического алгоритма [3]:

- отсутствие гарантии оптимальности полученного решения;
- относительно высокая вычислительная трудоемкость;
- невысокая эффективность на заключительных этапах моделирования.

Классический генетический алгоритм включает в себя следующие операции:

1. Формирование начальной популяции.
2. Вычисление функций приспособленности особей.
3. Проверка условия остановки алгоритма.
4. Селекция. Выбор самых приспособленных особей.
5. Скрещивание особей. Обмен генами между особями.
6. Мутация. Замена случайного гена у определенной части поколения.
7. Формирование новой популяции.
8. Повторяем шаги 2 – 7 до тех пор, пока не будет достигнуто условие.

Блок-схема классического генетического алгоритма представлена на рисунке 1.

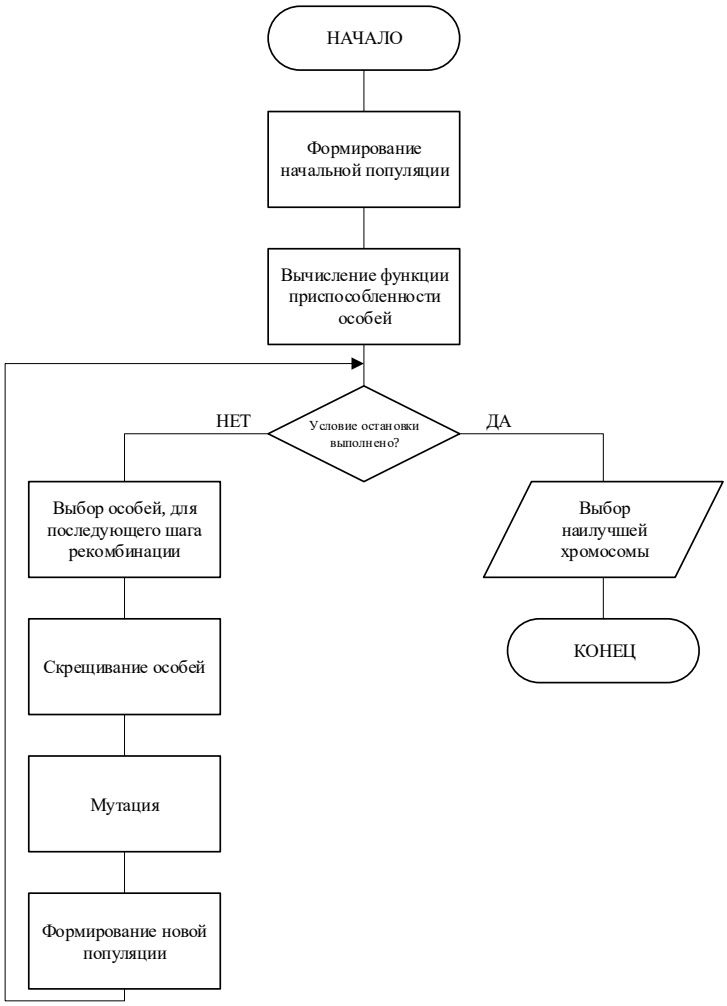

Рисунок 1 – Блок-схема классического генетического алгоритма

1.1.1 Формирование начальной популяции

Формирование начальной популяции является первым этапом в реализации генетического алгоритма. Первоначальная популяция создается путем случайной генерации хромосом, причем приспособленность хромосом не является важным показателем на данном этапе. Таким образом, начальная популяция может быть совершенно не конкурентоспособна, однако в ходе работы алгоритма популяция станет более приспособленной [1].

Выбор исходной популяции связан с представлением параметров задачи в форме хромосом особи. Это представление определяется способом кодирования. В классическом генетическом алгоритме обычно используется

двоичное кодирование [4, с. 8], то есть аллели генов представлены в виде 0 или 1. Длина хромосом зависит от условия задачи. Помимо двоичного кодирования генов также существуют логарифмическое и вещественное кодирования [4, с. 46].

Логарифмическое кодирование применяется в генетических алгоритмах для уменьшения длины хромосом. Оно используется в задачах многомерной оптимизации с большими пространствами поиска решений.

При вещественном кодировании аллели гена представлены в виде вещественных чисел. Такое кодирование позволяет уменьшить объем вычислительных процедур на каждом шаге эволюции за счет отсутствия двоично-десятичных преобразований при расчете значений функций приспособленности и уменьшения размеров хромосом.

1.1.2 Оценка приспособленности

Оценка приспособленности особей в популяции заключается в расчете функции приспособленности каждой отдельной особи. Форма функции приспособленности зависит от характера решаемой задачи. Предполагается что значение функции приспособленности всегда принимает неотрицательные значения и что для решения оптимизационной задачи необходимо максимизировать или минимизировать эту функцию. Если исходная форма функции приспособленности не удовлетворяет условиям задачи, то выполняется соответствующее преобразование [1, с. 130].

Например, имеется популяция численностью в 2 особи, закодированные в двоичном виде. Обозначим эти особи x и y. Особь x представлена генотипом – [10110], а особь y – [10001]. Следовательно, фенотип $x=10110_{10}=22$, $y=10001_{10}=17$. Необходимо найти максимум функции $f(x) = 2x^2 + 1$ для целочисленной переменной x, принимающей значения [0,

31]. Тогда для особи x функция приспособленности будет $F(x) = 2 * 22^2 + 1 = 969,$ а для особи y $F(y) = 2 * 17^2 + 1 = 579.$

1.1.3 Проверка условия остановки алгоритма

Условие остановки генетического алгоритма зависит от его конкретного применения. В оптимизационных задачах, если известно максимальное или минимальное значение функции приспособленности, то остановка алгоритма может произойти после достижения ожидаемого оптимального значения. Остановка алгоритма также может произойти в случае, когда его выполнение не приводит к улучшению уже достигнутого значения. Алгоритм может быть остановлен по истечении определенного времени выполнения либо после выполнения заданного количества итераций. Если условие остановки выполнено, то производится переход к завершающему этапу выбора «наилучшей хромосомы». В противном случае на следующем шагу выполняется селекция [1, с. 130].

1.1.4 Селекция

Селекция заключается в выборе особей, которые будут участвовать в создании потомков для следующей популяции. Отбор особей ведется исходя из рассчитанных ранее функций приспособленности. Существуют различные методы селекции:

- рулеточная селекция;
- турнирная селекция;
- ранговая селекция.

Также существуют особые процедуры репродукции, такие как элитарная стратегия и генетический алгоритм с частичной заменой популяции.

Рулеточная селекция является основным методом отбора особей для родительской популяции. В данном методе вероятность выбора особи тем вероятнее, чем лучше ее значение функции приспособленности:

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \quad (1)$$

где p_i – вероятность выбора особи;

f_i – значение функции приспособленности для i особи;

N – количество особей в популяции.

В турнирной селекции популяция разбивается на несколько групп, каждая группа содержит в себе несколько особей. Затем, из каждой такой группы выбирается особь с наилучшим показателем функции приспособленности, она и попадает в родительский пул. Схема турнирной селекции для групп, состоящих из двух особей представлена на рисунке 2.

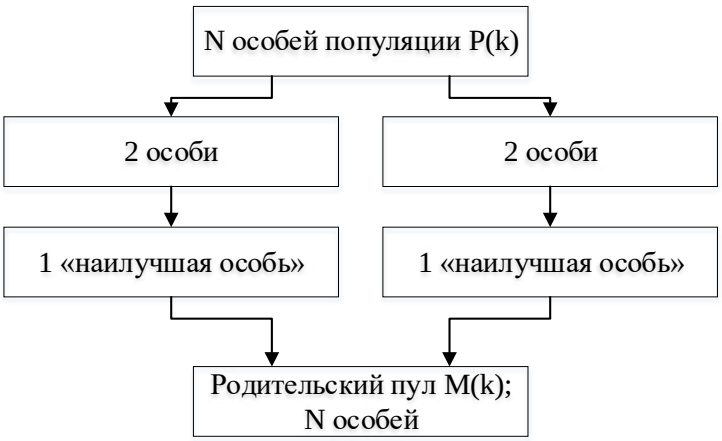

Рисунок 2 – Схема турнирной селекции

При ранговой селекции особи популяции ранжируются по значениям их функции приспособленности. Это можно представить, как отсортированный список особей, упорядоченных по направлению от наиболее приспособленных к наименее приспособленным или наоборот, в

котором каждой особи приписывается число, определяющее ее место в списке и называемое рангом. Количество копий каждой особи, введенных в родительскую популяцию, рассчитывается по заданной функции в зависимости от ранга особи [5]. Пример такой функции может быть график, представленный на рисунке 3.

Рисунок 3 – График функции для ранговой селекции

В силу того, что в основе генетического алгоритма лежит случайный поиск, высока вероятность потери лучших особей в популяции, так как они могут не попасть в родительский пул. Для того, чтобы избежать подобного случая можно использовать генетический алгоритм с элитарной стратегией. Его суть заключается в том, что определенное количество особей, имеющих лучшие показатели функции приспособленности, гарантировано переносятся в следующую популяцию.

Генетический алгоритм с частичной заменой популяции, характеризуется тем, что часть популяции переходит в следующее состояние без каких-либо изменений.

1.1.5 Скрещивание

На первом этапе скрещивания выбираются пары особей из родительской популяции. Это временная популяция, состоящая из особей, отобранных в результате селекции [1, с. 134]. Выбор родительской пары может производится несколькими методами:

- панмиксия;
- инбридинг;
- аутбридинг.

При панмиксии оба родителя выбираются случайным образом, следовательно, все особи в родительской популяции имеют одинаковые шансы быть выбранными.

При инбридинге первый родитель выбирается случайным образом, а вторым выбирается такой, который наиболее похож на первого родителя.

При аутбридинге первый родитель выбирается случайным образом, а вторым выбирается такой, который наименее похож на первого родителя

Инбридинг и аутбридинг бывают двух форм: фенотипный и генотипный. В случае фенотипной формы похожесть измеряется в зависимости от значений функций приспособленности, а в случае генотипной формы похожесть измеряется в зависимости от представления генотипа.

На следующем этапе производится обмен генами между родительскими парами. Обмен генами производится случайным образом в соответствии с вероятностью $p_{c.}$. Данный этап называется кроссинговером. Кроссинговер может быть представлен в виде одноточечного скрещивания, двухточечного скрещивания, многоточечного скрещивания и равномерного скрещивания.

При одноточечном скрещивании выбираются пары хромосом из родительской популяции. Далее для каждой пары отобранных таким образом родителей разыгрывается позиция гена (локус) в хромосоме, определяющая

так называемую точку скрещивания – *l(k)*. Если хромосома каждого из родителей состоит из L генов, то очевидно, что точка скрещивания *l(k)* представляет собой натуральное число, меньшее L. Поэтому фиксация точки скрещивания сводится к случайному выбору числа из интервала [1, L-1]. В результате скрещивания пары родительских хромосом получается следующая пара потомков:

- потомок, хромосома которого на позициях от 1 до *l(k)* состоит из генов первого родителя, а на позициях от *l(k)*+1 до L – из генов второго родителя;

- потомок, хромосома которого на позициях от 1 до *l(k)* состоит из генов второго родителя, а на позициях от *l(k)*+1 до L – из генов первого родителя.

Действие одноточечного кроссинговера представлено на рисунке 4.

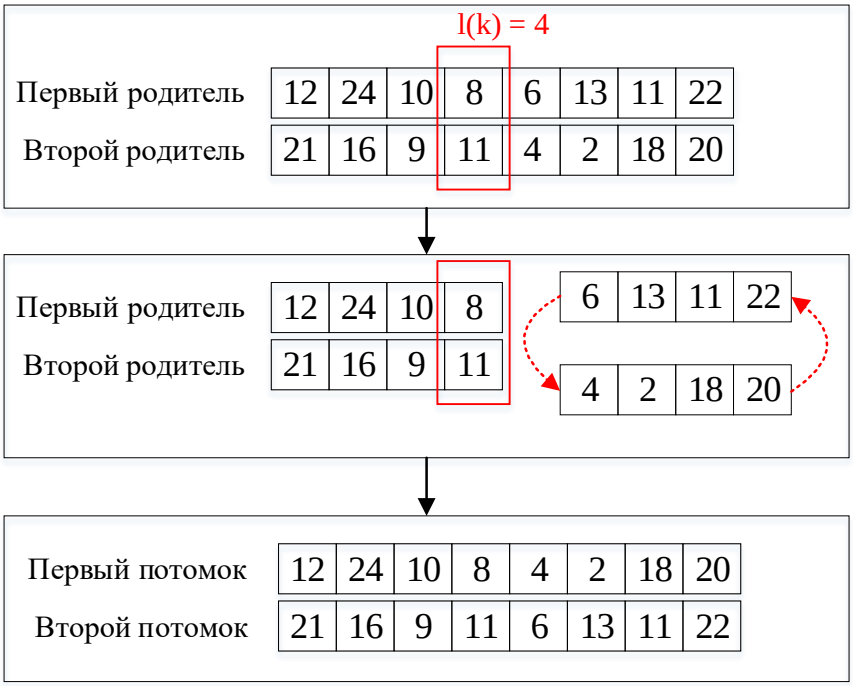

Рисунок 4 – Одноточечный кроссинговер

Двухточечное скрещивание отличается от одноточечного скрещивания тем, что родительские хромосомы обмениваются участком генетического

кода, который находится между двумя случайно выбранными точками скрещивания.

Многоточечное скрещивание представляет собой обобщение предыдущих операций и характеризуется соответственно большим количеством точек скрещивания.

Равномерное скрещивание, иначе называемое монолитным или одностадийным, выполняется в соответствии со случайно выбранным эталоном, который указывает, какие гены должны наследоваться от первого родителя, а какие от второго. Допустим, что выбран эталон 010110111011, в котором 1 означает принятие гена на соответствующей позиции от первого родителя, а 0 – от второго родителя. Таким образом, сформируется первый потомок. Для второго потомка эталон необходимо считывать аналогично, причем 1 означает принятие гена на соответствующей позиции от второго родителя, а 0 – от первого родителя. Пример равномерного скрещивания представлен на рисунке 5.

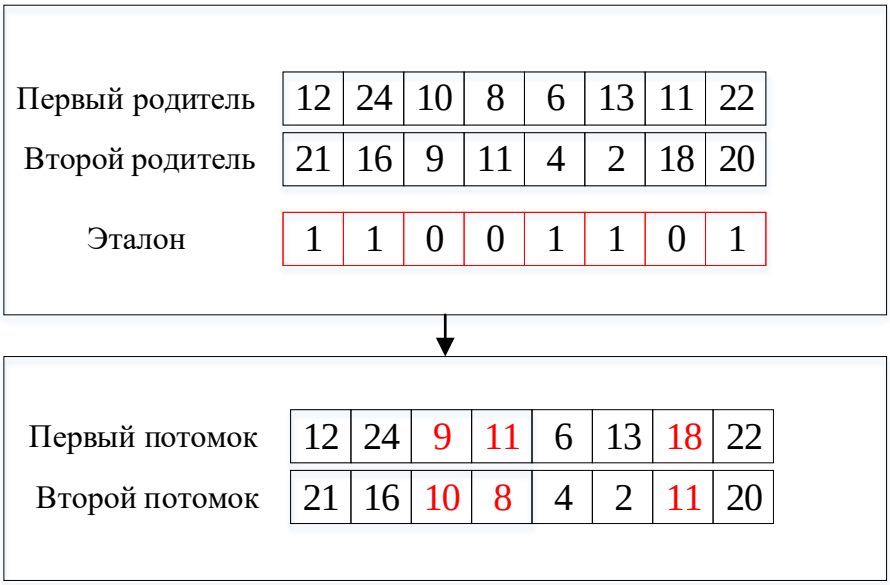

Рисунок 5 – Равномерное скрещивание

1.1.6 Мутация

Оператор мутации с вероятностью p_m изменяет значение гена в хромосоме. Например, если в хромосоме [10011] мутации подвергается ген на позиции 3, то его значение, равное 0, изменяется на 1. Это приводит к образованию новой хромосомы [10111]. Вероятность мутации обычно очень мала. Вероятность мутации может эмулироваться, например, случайным выбором числа из интервала [0, 1] для каждого гена и отбором для выполнения этой операции тех генов, для которых разыгранное число оказывается меньшим или равным значению p_m.

1.1.7 Формирование новой популяции

Особи, полученные в результате применения генетических алгоритмов к особям временной родительской популяции, включаются в состав новой популяции. Она становится так называемой текущей популяцией для данной итерации генетического алгоритма. На каждой итерации рассчитываются значений функций приспособленности для всех особей этой популяции, после чего проверяется условие остановки алгоритма и либо фиксируется результат в виде хромосомы с наилучшим значением функции приспособленности, либо осуществляется переход к следующему шагу генетического алгоритма, то есть к селекции.

1.1.8 Заключение

Генетический алгоритм является эффективным методом для решения задач поиска оптимального решения. Операции, используемые в генетическом алгоритме, обеспечивают быстрое схождение популяции к максимальному или минимальному решению для заданной функции. Однако, в виду того, что в основе генетического алгоритма лежит метод случайного

поиска, нельзя определить точное время нахождения оптимального решения. Для того чтобы уменьшить время поиска используются различные виды операций генетического алгоритма, а иногда и различные модификации генетического алгоритма, такие как генетический алгоритм с частичной заменой популяции и генетический алгоритм с элитарной стратегией.

1.2 Аналитический обзор аналогов

На сегодняшний день существует большое количество программ, использующих генетические алгоритмы. Эти программы разработаны для поиска оптимального решения в области той или иной задачи. Таковыми, например, являются FlexTool [1, с. 170] и Evolver [1, с. 214]. Помимо этого, существуют программы, реализованные частными разработчиками, например, CyberBiology [6]. Рассмотрим каждую из них подробнее.

Целью аналитического обзора является сравнение аналогов с разрабатываемым приложением. В качестве критериев для сравнения используются:

- наличие интерфейса;
- гибкость настроек;
- простота в использовании;
- наличие визуализации процесса;
- цена.

1.2.1 FlexTool

FlexTool – это модульный программный инструмент, предоставляющий среду для применения генетического алгоритма в различных областях с минимальным взаимодействием с пользователем [7].

Flextool M2.2 является расширением для MATLAB. Объекты генетических алгоритмов предоставляются в виде файлов с расширением m.

Программа разработана с упором на модульность, гибкость, возможность обновления и надежность.

FlexTool имеет большое количество настраиваемых параметров:

– вид генетического алгоритма (с полной сменой поколения, с устойчивым состоянием, микро ГА);

– кодирование (бинарное, логарифмическое, вещественное);

– селекция (турнирная, рулеточная, рейтинговая);

– кроссовер (одноточечный, двухточечный, многоточечный).

Демоверсия FlexTool представлена на рисунке 6.

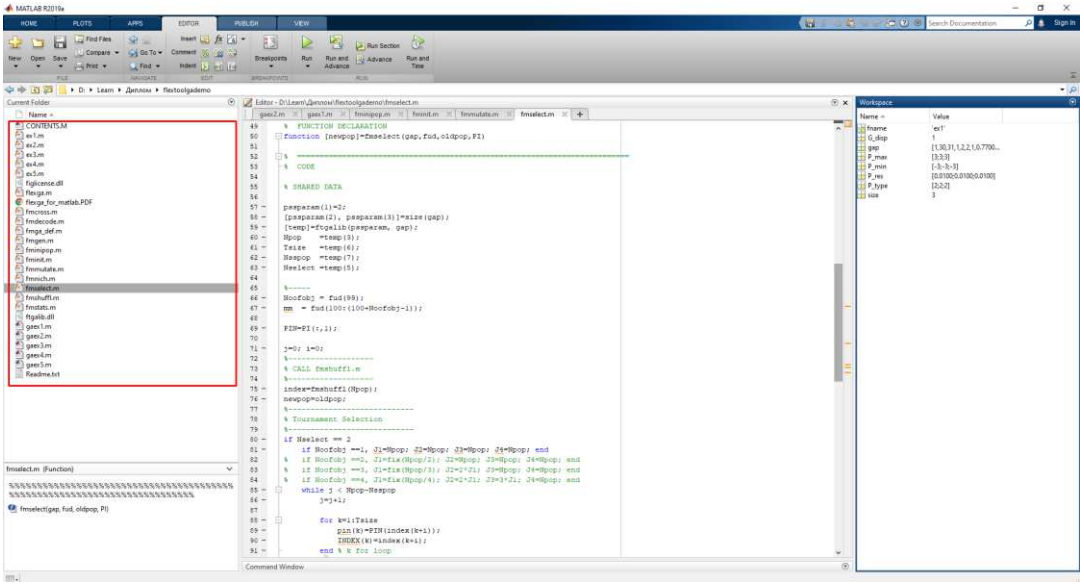

Рисунок 6 – Демоверсия FlexTool

Несмотря на все преимущества FlexTool, стоит учесть, что это модульный программный инструмент для MATLAB. Настройки параметров производятся прямо в коде, в виду отсутствия интерфейса, нет возможности наблюдать за ходом работы генетического алгоритма, а также цена полной версии FlexTool находится в районе 80$. На данный момент инструмент FlexTool недоступен, так как поддержка была прекращена в 2009 году.

1.2.2 Evolver

Evolver – это программный пакет, который позволяет пользователям решать широкий спектр задач оптимизации с использованием генетического алгоритма. Запущенный в 1989 году, это был первый коммерческий пакет генетических алгоритмов для персональных компьютеров. Программа изначально была разработана Axcelis, Inc. и в настоящее время принадлежит Palisade Corporation [8].

Evolver способна оптимизировать функции нескольких переменных, при этом она может применятся для поиска максимума или минимума функции двух и даже одной переменной. Примечательно что в случае одной переменной скрещивание практически не производится, и выполняется только функция мутации, что характерно для эволюционного программирования [1, с. 224].

Результаты работы генетического алгоритма, реализованного в программе Evolver, представлен на рисунке 7.

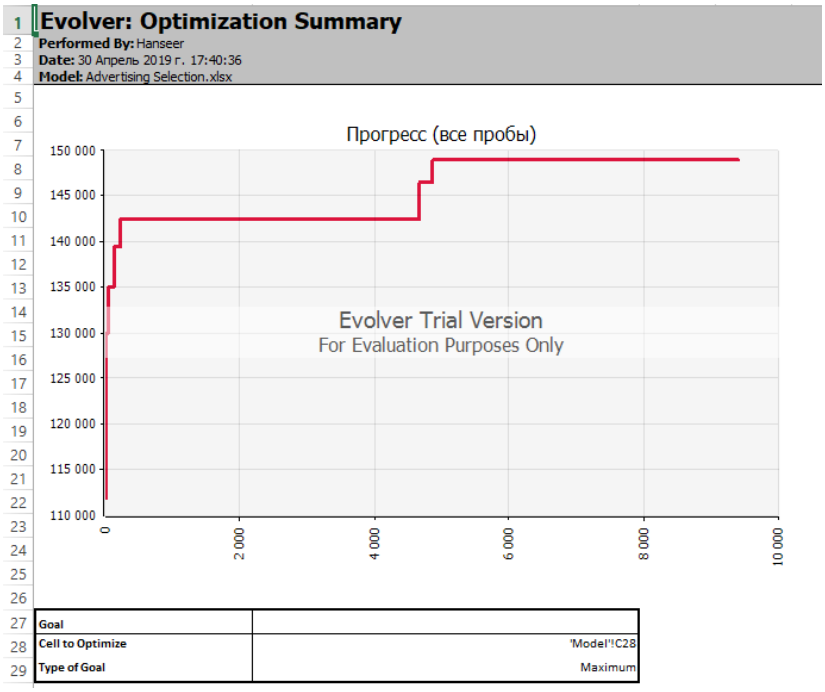

Рисунок 7 – Результат оптимизации в Evolver

Evolver является расширением для Excel, следовательно у программы отсутствует свой интерфейс, а пользователю не доступны настройки параметров генетического алгоритма, также цена продукта составляет 1055£ или 88000 рублей.

1.2.3 CyberBiology

CyberBiology является частной разработкой Михаила Царькова. реализованной в 2016 году. Данная программа предназначена для наглядного отображения хода работы генетического алгоритма.

Идеей разработки является среда, в которой особи (боты) должны находить пищу, чтобы продлить время своей жизни. Помимо особей и пищи, в среде присутствуют ядовитые клетки, которые мгновенно уничтожают бота, однако бот в состоянии «обезвредить» такую клетку. Генами особей в программе являются команды.

Пример работы CyberBiology представлен на рисунке 8.

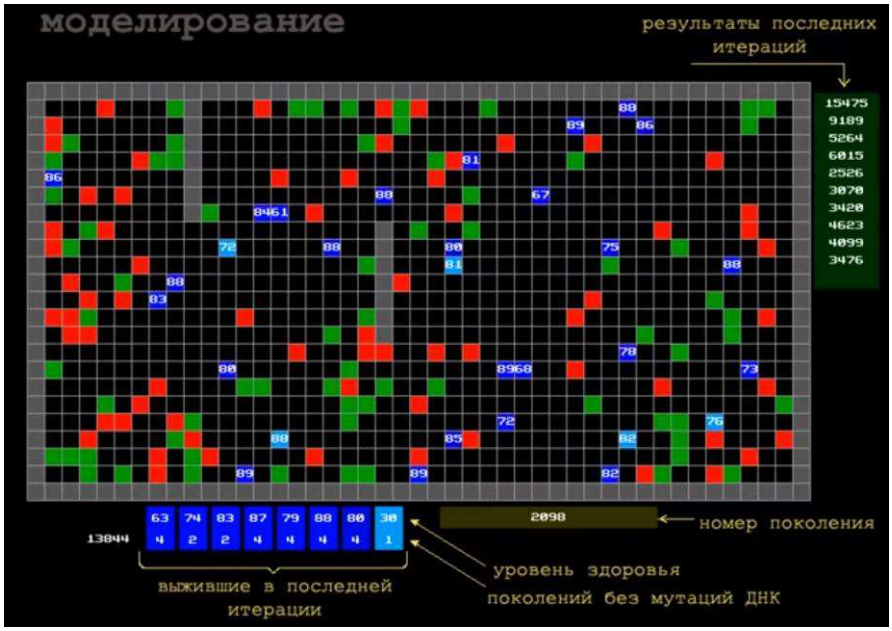

Рисунок 8 – Пример работы CyberBiology

Данная программа идеально подходит для наблюдения за ходом генетического алгоритма. Однако в ней отсутствует возможность настройки генетического алгоритма. Программа хранится в свободном доступе на GitHub.

1.2.4 Заключение

Сравнение аналогов представлено в таблице 1.

Таблица 1 – Сравнение аналогов

Критерий	FlexTool	Evolver	CyberBiology	Разрабатываемое приложение
Наличие интерфейса	Нет	Нет	Нет	Да
Гибкость настроек	Да	Нет	Нет	Да
Простота в использовании	Нет	Да	Да	Да
Визуализации процесса	Нет	Нет	Да	Да
Цена	~5000р.	~73000 р.	Бесплатно	Бесплатно

В результате аналитического обзора, можно сделать вывод, что ни одна из рассмотренных программ не соответствует всем критериям сразу. Программы просты в использовании только если отсутствует гибкость настроек, в виду того, что все настройки прописаны в коде. Прикладные программы имеют высокую цену, при этом пользователь не может наблюдать за ходом процесса.

2 Проектирование и выбор средств разработки

2.1 Выбор средств разработки

Грамотно выбранные средства на начальном этапе разработки имеют большое значение для успешного завершения проекта. Поскольку корректно осуществленный выбор средств разработки оказывает значительное влияние на сроки и стоимость разработки необходимого ПО.

2.1.1 Выбор языка программирования для реализации программы

На сегодняшний день существует множество языков программирования для разработки приложений, однако ни один из языков не превосходит остальные по всем критериям. Превосходство какого-либо языка программирования может проявляться только в контексте решения какой-либо конкретной задачи. Однако многие задачи могут быть эффективно решены с помощью практически любого современного языка программирования. Ниже приведен список с кратким описанием некоторых языков программирования, на которых может быть реализован генетический алгоритм

Java – сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). В настоящее время проект принадлежит OpenSource и распространяется по лицензии GPL. Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре с помощью виртуальной Java-машины.

Также в Java есть библиотека с набором функций для реализации генетического алгоритма. Данная библиотека носит название Jenetics, что позволило бы ускорить написание программы, однако пришлось отказаться

от идеи реализации на языке Java, по причине отсутствия опыта разработки на данном языке, а также из-за плохой производительности, связанной с программой автоматической очистки памяти [9].

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование [10, с. 4]. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Python имеет большое количество библиотек для работы с генетическим алгоритмом: Pyvolution, DEAP, pyStep, Pyevolve, pyRobot, PonyGE, inspired, DRP.

C++ - компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование [11]. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

C++ имеет библиотеки для работы с генетическим алгоритмом, например GAlib.

Было принято решение разрабатывать программу на языке Python в силу его быстродействия и наличия опыта разработки на этом языке, но без

использования библиотеки для работы с генетическим алгоритмом, в силу ухудшения читабельности кода и его понимания.

Для написания графического интерфейса был выбран инструмент Qt.

Qt - кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++ [12]. Есть также «привязки» ко многим другим языкам программирования: Python — PyQt, PySide; Ruby — QtRuby; Java — Qt Jambi; PHP — PHP-Qt и другие.

2.1.2 СУБД SQLite

Для хранения данных было принято решение использовать реляционную базу данных под управлением СУБД SQLite, поскольку это компактная легко встраиваемая в другие системы реляционная база данных. В данном контексте слово «встраиваемая» означает, что данная БД является файловой, а не клиент-серверной, что в свою очередь означает, что движок SQLite – это не отдельно работающий процесс, а библиотека, которая впоследствии будет являться частью программы.

Следует отметить, что для работы с другой системой управления базами данных потребовалось бы скачать и установить соответствующий драйвер, а с SQLite никакой установки не потребуется, так как язык Python уже имеет встроенную по умолчанию поддержку SQLite [13].

2.1.3 Вспомогательные инструменты для разработки

В ходе разработки приложения были использованы следующие инструменты:

PyCharm - интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает

веб-разработку на Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

Qt Designer - кроссплатформенная свободная среда для разработки графических интерфейсов (GUI) программ использующих библиотеку Qt. Входит в состав Qt framework.

DB Browser for SQLite - высококачественный визуальный инструмент с открытым исходным кодом для создания, проектирования и редактирования файлов базы данных, совместимых с SQLite.

Notepad++ - свободный текстовый редактор с открытым исходным кодом для Windows с подсветкой синтаксиса большого количества языков программирования и разметки, а также языков описания аппаратуры VHDL и Verilog. Базируется на компоненте Scintilla, написан на C++ с использованием STL, а также Windows API и распространяется под лицензией GNU General Public License.

2.2 Структура приложения

Согласно требуемому функционалу, пользователь должен иметь возможности изменять характеристики технических систем, изменять настройки генетического алгоритма и визуально наблюдать за ходом процесса. Исходя из этого была разработана диаграмма прецедентов, представленная на рисунке 9.

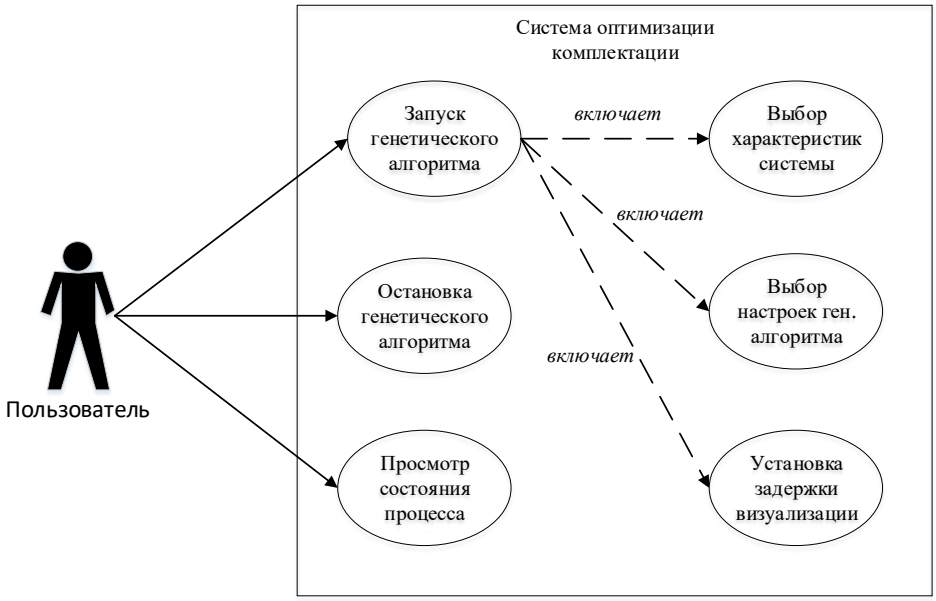

Рисунок 9 – Диаграмма прецедентов системы

Запуск работы генетического алгоритма, подразумевает под собой последовательный вызов подсистем, реализующих процесс эволюции. Диаграмма последовательности при выборе прецедента «Запуск генетического алгоритма», представлена на рисунке 10.

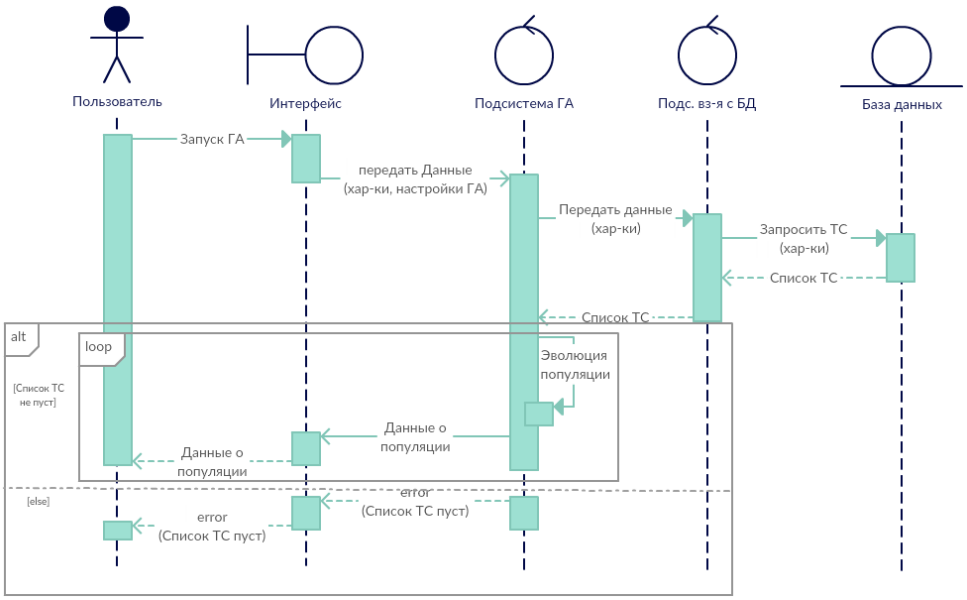

Рисунок 10 – Диаграмма последовательности «Запуск генетического алгоритма»

При запуске генетического алгоритма, происходит запрос данных из базы данных.

Если полученный список пуст, то пользователь видит сообщение об ошибке на интерфейсе, в противном случае запускается процесс оптимизации.

Рассмотрим процесс взаимодействия студента с приложением на примере выполнения лабораторной работы.

Цель лабораторной работы – описать влияние значения вероятности мутации на процесс эволюции.

Ход работы:

1. Ввести значение вероятности мутации в поле «Вероятность мутации» и запустить работу генетического алгоритма.

2. Проанализировать на каких итерациях происходили мутации, сколько генов мутировали во время итерации, каким образом мутация повлияла на значение средней цены популяции на данной итерации.

3. Ввести значение вероятности мутации, равное 0.

4. Повторить пункт 2.

5. Ввести значение вероятности мутации, равное 100.

6. Повторить пункт 2.

7. Сделать выводы, определить какое значение является оптимальным и при каком значении популяции не сходятся к минимуму.

По окончанию процесса эволюции, студент может проанализировать следующие данные:

- номер итерации, на которой было найдено решение;

- состав хромосомы полученного решения (какие гены наследовались, мутировали, были заменены для достижения совместимости и.т.д.);

- график среднего значения цены (на каких итерациях происходило увеличение или уменьшение значения цены).

Так как приложение имеет гибкие настройки, студент может увидеть не только влияние типа скрещивания или селекции, но и влияние вероятности

мутации, размера популяции, характеристик технических систем, максимально допустимого количества итераций, значения оптимальной цены.

Общее представление процесса взаимодействия студента с приложением в целях обучения отображено на диаграмме последовательности, представленной на рисунке 11.

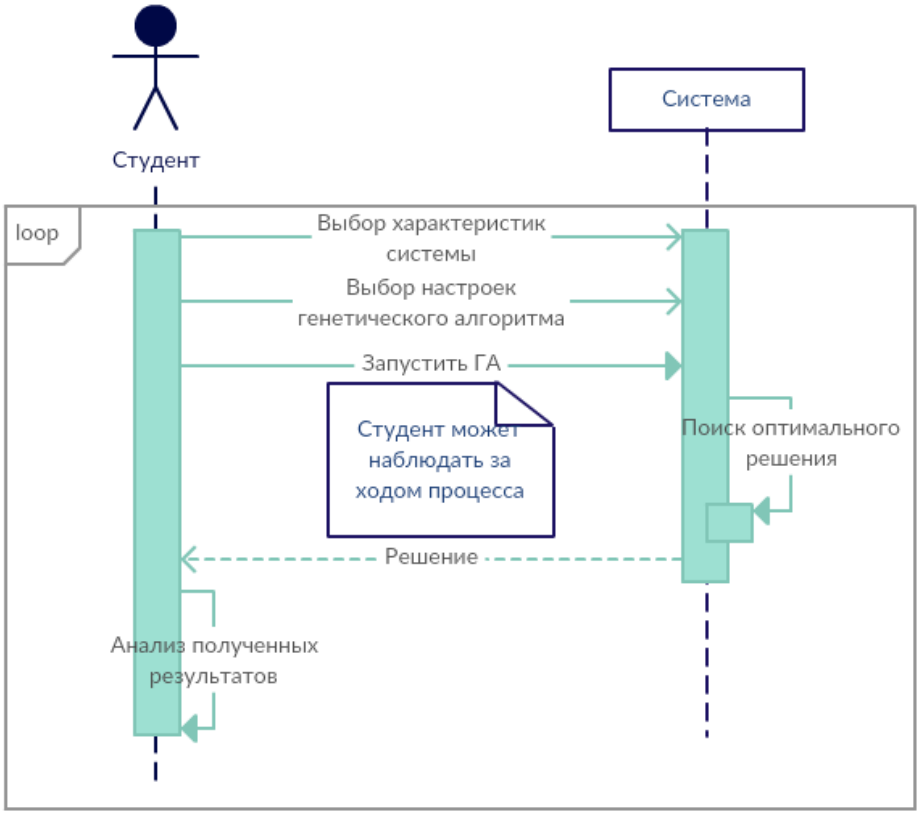

Рисунок 11 – Диаграмма последовательности взаимодействия студента с приложением

Таким образом студент может изменять различные настройки генетического алгоритма и наблюдать каким образом это отображается на ходе процесса эволюции.

В том случае, если пользователь выбирает лишь часть характеристик технических систем, то остальные характеристики принимают значение «Любой». Таким образом, при выполнении процесса эволюции, технические

системы, которые не были сконфигурированы пользователем, автоматически подбираются подсистемой генетического алгоритма. Если же пользователь не выбирает ни одной характеристики технических систем, то процесс эволюции будет производиться на всем наборе систем имеющихся в базе данных, а их совместимость будет контролироваться подсистемой генетического алгоритма. Также все настройки генетического алгоритма имеют значение по умолчанию, тем самым позволяют избежать ошибок при выполнении.

Структура приложения представлена в виде пяти основных элементов: интерфейса пользователя, подсистемы визуализации, подсистемы генетического алгоритма, подсистемы взаимодействия с базой данных и самой базой данных. Наглядно архитектура приложения представлена на рисунке 12.

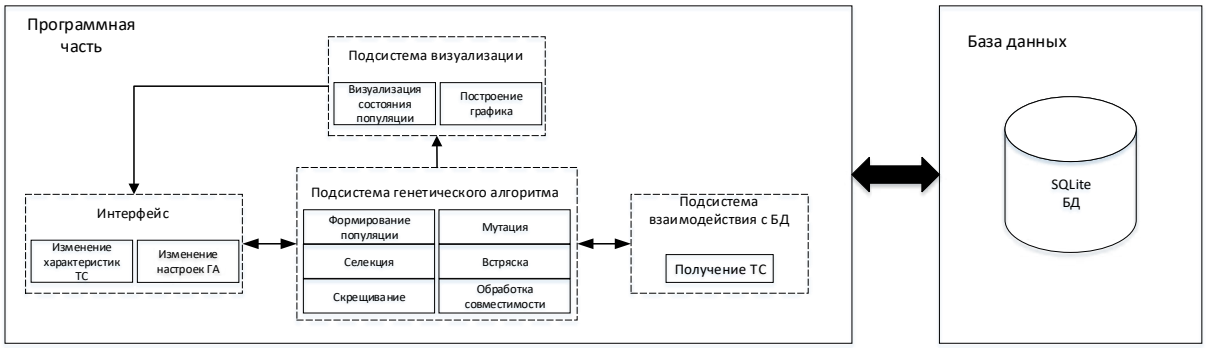

Рисунок 12 – Схема архитектуры приложения

Интерфейс пользователя – предназначен для взаимодействия пользователя с программой, позволяет пользователю установить параметры для работы генетического алгоритма, выбрать характеристики технических систем, а также наблюдать за ходом работы.

Подсистема визуализации – подсистема, предназначенная для визуализации данных полученных в ходе работы генетического алгоритма. Данная подсистема получает данные из подсистемы генетического алгоритма, которые были получены в результате процесса эволюции и

создает графический объект, которые затем отображается на интерфейсе. Визуализация происходит в реальном времени.

Подсистема генетического алгоритма – подсистема, в которой происходит сам процесс эволюции. Данная подсистема воспроизводит все операции классического генетического алгоритма, изменение настроек генетического алгоритма производится на интерфейсе.

Подсистема взаимодействия с базой данных – подсистема, предназначения для получения записей, соответствующих характеристикам, выбранным пользователем. Данная подсистема отправляет запросы к базе данных, и создает собственные списки технических систем, которые соответствуют характеристикам, выбранными пользователем.

База данных – база данных, содержащая 6 таблиц, каждая из которых содержит записи о технических системах определенного вида.

3 Разработка приложения

Каждая подсистема приложения и интерфейс описаны отдельным классом. Диаграмма классов представлена на рисунке 13.

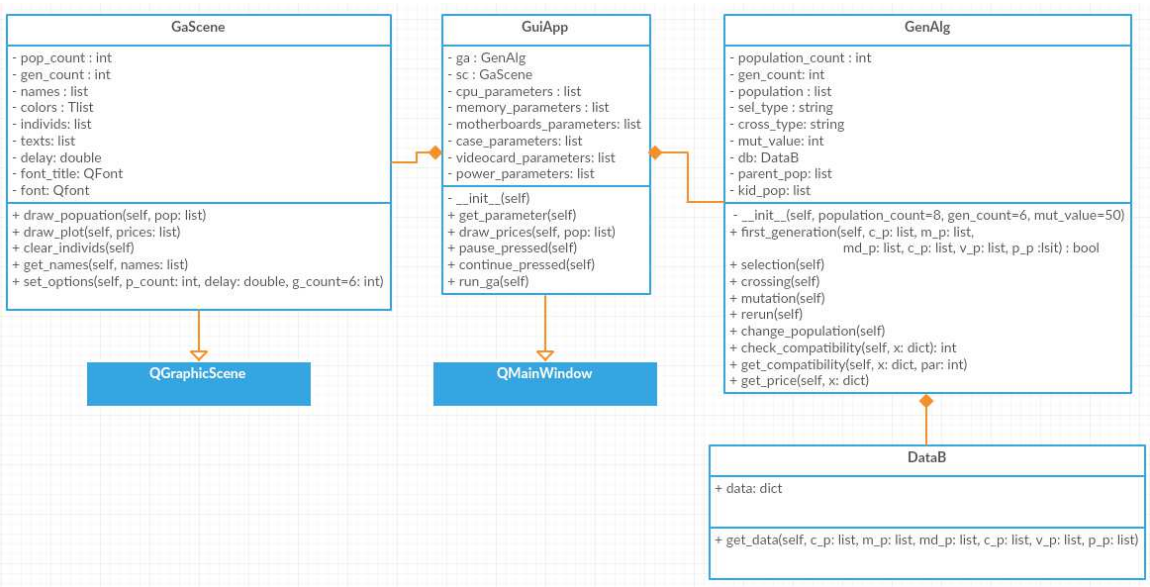

Рисунок 13 – Диаграмма классов

Таким образом подсистема генетического алгоритма описана классом GenAlg, подсистема визуализации классом GaScene, подсистема взаимодействия с базой данных классом DataB и интерфейс приложения классом GuiApp.

Класс GuiApp находится в состоянии композиции с классами GaScene и GenAlg, в то время как класс GenAlg находится в композиции с классом DataB.

Класс GaScene наследуется от объекта QGraphicScene. QGraphicScene – графический объект, при помощи которого можно отображать двумерные фигуры.

Класс GuiApp наследуется от объекта QMainWindow. QMainWindow – объект, используемый для построения интерфейса.

3.1 Подсистема генетического алгоритма

Подсистема генетического алгоритма представлена классом GenAlg.

В данном классе хранятся переменные и списки, используемые в ходе генетического алгоритма, методы, реализующие операции генетического алгоритма, а также методы, позволяющие достичь совместимости технических систем и избежать копирования сильнейших особей.

В данном классе создается объект класса DataB, который представляет подсистему взаимодействия с базой данных. Таким образом связь между этими двумя классами является композицией.

Описание переменных и списков класса GenAlg представлено в таблице 2.

Таблица 2 – Описание переменных и списков класса GenAlg

Имя	Тип	Описание
population_count	int	Количество особей в популяции
gen_count	int	Количество генов в хромосоме

Окончание таблицы 2

Имя	Тип	Описание
mut_value	int	Вероятность мутации в процентах
sel_type	str	В зависимости от значения данной переменной, используется тот или иной метод селекции
cross_type	str	В зависимости от значения данной переменной используется тот или иной метод скрещивания
population	list	Список, в котором хранятся особи текущей популяции
parent_pop	list	Список, в котором хранятся особи родительского пула
kid_pop	list	Список, в котором хранятся особи потомки
db	DataB	Объект класса DataB, представление подсистемы взаимодействия с базой данных

Описание методов класса GenAlg представлено в таблице 3.

Таблица 3 – Описание методов класса GenAlg

Имя	Аргументы	Описание
__init__	population_count = 4: int; gen_count = 6: int; mut_value = 50: int	Конструктор класса, вызывается при создании объекта. На вход принимает количество особей в популяции, количество генов в хромосоме, вероятность мутации в процентах
first_generation	c_p: list; m_p: list; md_p: list; c_p: list; v_p: list; p_p: list	Метод, формирующий начальную популяцию. На вход подаются выбранные пользователем характеристики, представленные в виде списков
selection	-	Метод, выполняющий процесс селекции, тип селекции выбирается исходя из значения переменной sel_type
crossing	-	метод, выполняющий процесс скрещивания, тип скрещивания выбирается исходя из значения cross_type

Окончание таблицы 3

Имя	Аргументы	Описание
check_compatibility	x: dict	Метод, производящий проверку на совместимость генов в особи. Принимает на вход особь, возвращает локус несовместимого гена
get_compatibility	x: dict; par: int	Метод, производящий замену несовместимого гена. Принимает на вход особь и локус гена
get_price	x: dict	Метод, вычисляющий цену отдельного решения. Принимает на вход особь, возвращает значение цены
mutation	-	Метод, выполняющий процесс мутации одной из особей, количество мутирующих генов выбирается случайно. Мутация происходит с вероятностью, заданной в переменной mut_value
rerun	-	Метод, выполняющий процесс «встряски», данный метод заменяет половину текущей популяции на случайно сгенерированные. «Встряска» происходит только в том случае, если большинство особей в популяции являются копиями.
change_population	-	Метод, выполняющий процесс смены популяций. Данный метод, заменяет текущую популяцию на популяцию потомков и очищает родительский пул

В виду того, что присутствует фактор совместимости, добавляется операция достижения совместимости. Также, для того чтобы снизить вероятность провала в локальный минимум, добавлена операция «встряски».

Блок-схема модифицированного генетического алгоритма представлена на рисунке 14.

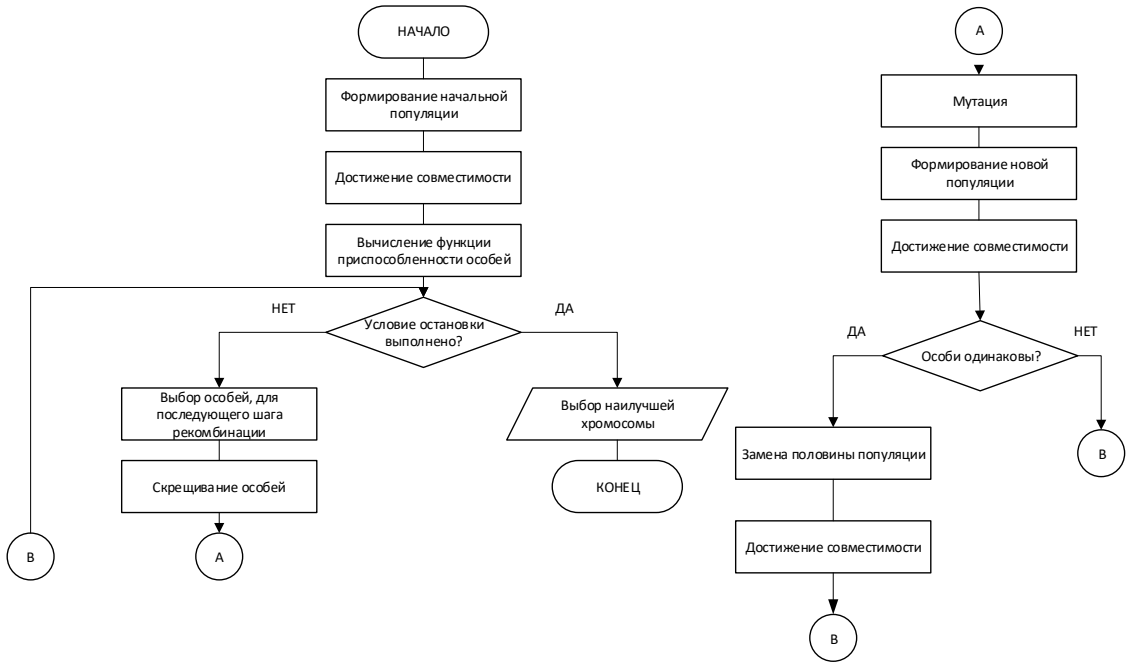

Рисунок 14 – Блок-схема модифицированного генетического алгоритма

Совместимость достигается следующим образом, для каждой особи вызывается метод get_compatibility(self, x, par) до тех пор, пока все гены не будут совместимы. Также при замене несовместимых генов, существует определенная иерархия. У каждой технической системы есть свой приоритет, чем выше приоритет, тем больше у технической системы привилегий остаться «на месте».

Приоритеты технических систем:

1. Процессор.
2. Материнская плата.
3. Оперативная память.
4. Видеокарта.
5. Блок питания.
6. Корпус.

Таким образом, процессор имеет самый высокий приоритет, в то время как корпус имеет самый низкий приоритет. То есть, в том случае если окажется что несовместимы материнская плата и оперативная память, то заменяться будет память, так как ее приоритет ниже.

Блок-схема алгоритма достижения совместимости представлена на рисунке 15.

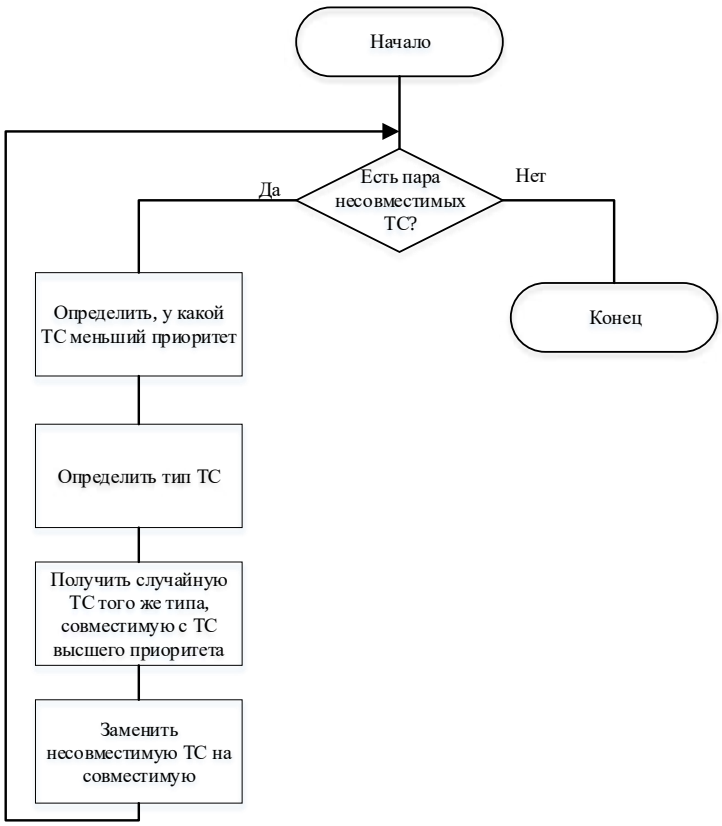

Рисунок 15 – Блок-схема алгоритма достижения совместимости

Данный алгоритм выполняется для каждой особи, при формировании начальной популяции, и смене популяций.

3.1 База данных

База данных состоит из 6 таблиц:

1. cpus – содержит записи о процессорах.
2. motherboards – содержит записи о материнских платах.
3. memory – содержит записи о оперативной памяти.
4. videocards – содержит записи о видеокартах.
5. power – содержит записи о блоках питания.
6. cases – содержит записи о корпусах.

Каждая таблица содержит записи о технической системе определенного вида. Для каждой технической системы есть несколько полей, в которых хранится информация о характеристиках этой технической системы. При помощи этих полей, происходит достижение совместимости, а также фильтрация технических систем по критериям, выбранных пользователем.

Табличная схема базы данных представлена на рисунке 16.

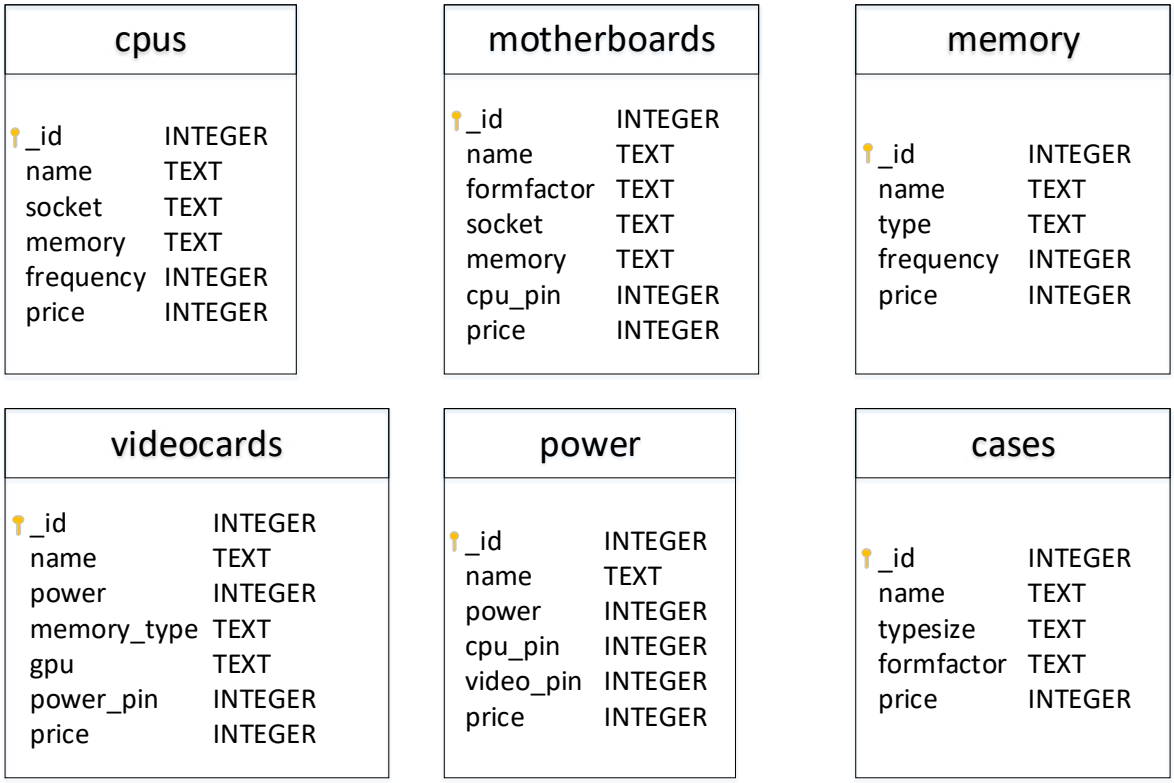

Рисунок 16 – Табличная схема базы данных

Отсутствие связей между таблицами, объясняется тем, что все проверки, а также поиск необходимых технических систем проводятся непосредственно внутри приложения при помощи запросов.

База данных хранится в файле «perifery.db».

3.2 Подсистема взаимодействия с базой данных

Подсистема генетического алгоритма представлена классом DataB.

В данном классе хранится список технических систем, удовлетворяющих характеристикам, выбранных пользователем.

Класс содержит один метод get_data(self, cpu_p, memory_p, motherboard_p, case_p, video_p, power_p) – метод принимает на вход характеристики технических систем, далее выполняет запросы к базе данных исходя из значений характеристик.

Данные, полученные из базы данных, помещаются в отдельный список и хранятся в оперативной памяти до закрытия приложения или до перезапуска генетического алгоритма с другими характеристиками технических систем. Перед помещением в список, происходит преобразование данных в объекты типа «словарь». Таким образом образуются 6 списков:

- cpus;
- motherboards;
- memorys;
- cases;
- videocards;
- powers.

Ключи и значения словарей технических систем, полностью соответствуют ключам и значениям записей, представленных в базе данных.

3.3 Интерфейс пользователя

Интерфейс пользователя представлен на рисунке 17.

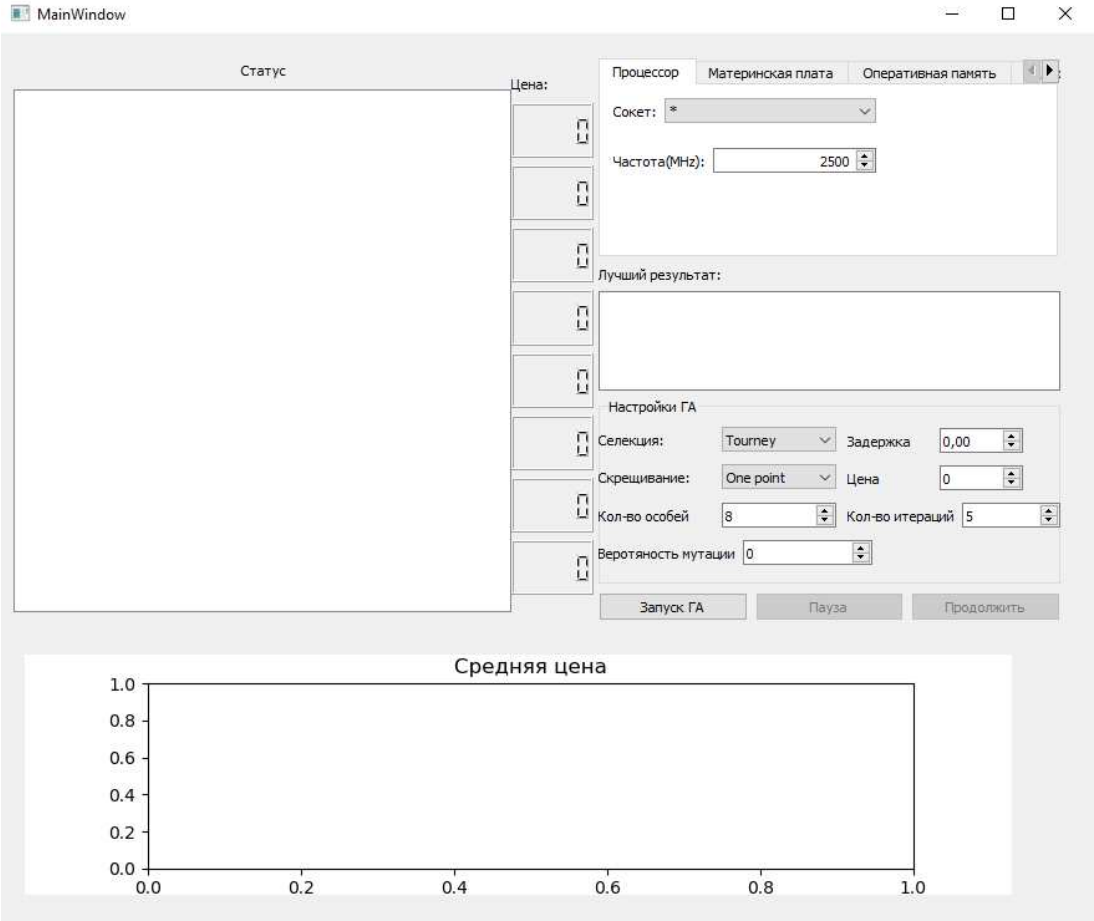

Рисунок 17 – Интерфейс пользователя

Слева находится объект класса QGraphicsView, в котором происходит визуализация процесса генетического алгоритма. Рядом с ним находятся поля, в которых отображаются цены отдельных комплектаций.

В правом верхнем углу находится объект класса QTabWidget, он имеет 6 вкладок: Процессор, Материнская плата, Оперативная память, Видеокарта, Блок питания, Корпус. На каждой из этих вкладок пользователь может выбирать характеристики той или иной технической системы. Открытые вкладки представлены на рисунке 18 и рисунке 19.

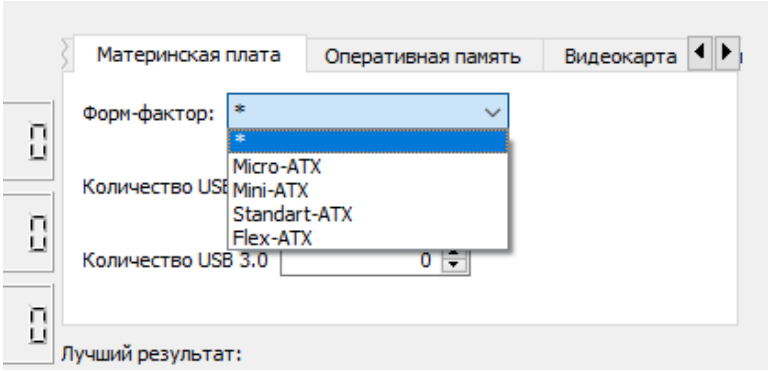

Рисунок 18 – Вкладка «Материнская плата»

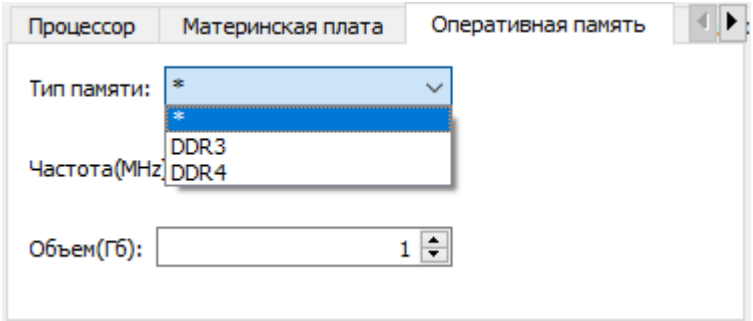

Рисунок 19 – Вкладка «Оперативная память»

Ниже находится текстовое поле «Лучший результат», в котором выводятся данные о особи с лучшей приспособленностью, в нашем случае комплектации с меньшей стоимостью. Поле обновляется каждую итерацию, то есть пользователь может видеть «лучшую» особь на каждой итерации.

Ниже поля «Лучший результат» находятся поля, при помощи которых пользователь может изменить настройки генетического алгоритма: тип селекции, тип скрещивания, размер популяции, шанс мутации, задержку между итерациями, оптимальную стоимость и максимальное количество итераций. Задержка между итерациями нужна для того, чтобы пользователь мог увидеть сам процесс эволюции, так как без задержки визуализация будет происходить мгновенно.

В самом низу находится объект QWidget, на котором строится график среднего значения цены всей популяции на данной итерации.

Также на интерфейсе присутствуют три кнопки: Запуск ГА, Пауза, Продолжить. При нажатии на кнопку «Запуск ГА», данные с интерфейса передаются подсистеме генетического алгоритма, подсистема создает новую популяцию и происходит процесс эволюции. При нажатии на кнопку «Пауза», подсистема генетического алгоритма приостанавливается, и пользователь может внимательно рассмотреть популяцию на данной итерации. При нажатии на кнопку «Продолжить», подсистема генетического алгоритма продолжает свою работу.

3.4 Подсистема визуализации

Подсистема визуализации представлена классом GaScene и наследуется от QGraphicsScene. Данный класс содержит один основной метод draw_population(self, pop). Этот метод вызывается каждый раз, когда происходит любое изменение в популяции. Метод получает текущие данные о популяции, затем рассчитывает размеры и координаты квадратов (генов), и добавляет их на холст.

Для того, чтобы можно было отличать гены по принадлежности, они окрашиваются в определенный цвет. За каждой особью закреплен свой цвет. Далее на каждый квадрат добавляется надпись – название технической системы, которую хранит данный ген. После того как все гены были добавлены на холст, метод возвращает готовый холст, который в свою очередь отображается на интерфейсе.

График строится при помощи метода draw_plot(self, price). Метод принимает среднее значение цены на данной итерации, добавляет его в массив и строит по этому массиву график, при помощи средств библиотеки matplotlib. Данный метод вызывается каждый раз, когда происходит смена популяции.

Результат работы подсистемы визуализации представлен на рисунке 20.

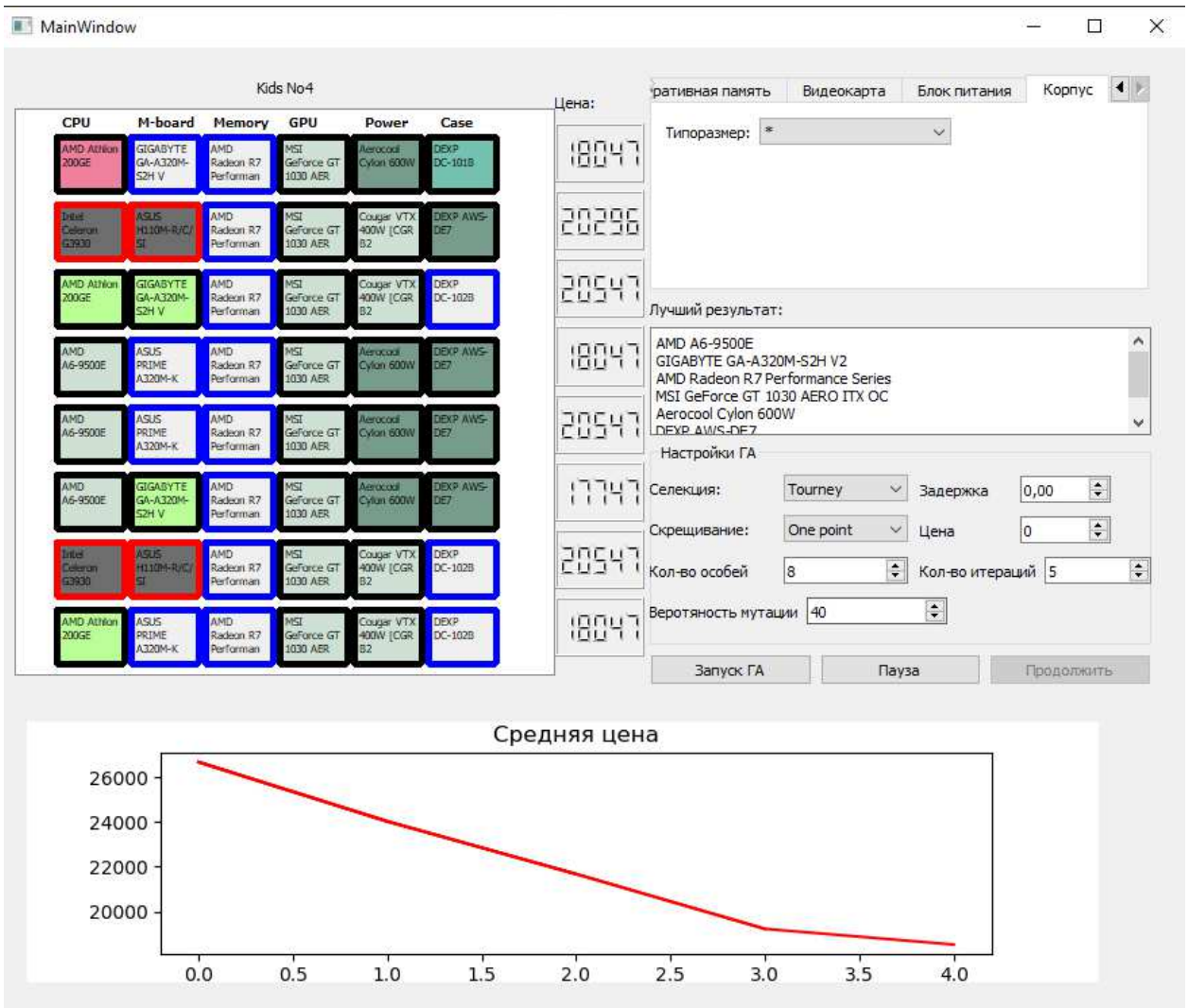

Рисунок 20 – Визуализация процесса

Квадраты белого цвета, обведенные синей рамкой – это гены, которые заменяют несовместимые. Квадраты серого цвета, обведенные красной рамкой – мутированные гены.

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы была реализована программа на основе генетического алгоритма, выполняющая оптимизацию по одному критерию. Написаны различные методы селекции и скрещивания. Программа имеет гибкие настройки, визуализацию процесса эволюции.

В ходе работы были выполнены задачи:

1. Аналитический обзор и сравнение аналогов
2. Выбор средств разработки
3. Проектирование базы данных
4. Разработка структуры приложения
5. Реализация программного продукта

Таким образом, все задачи решены, поставленная цель - разработка приложения для оптимизации комплектации технических систем на основе генетического алгоритма – достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы: науч. изд. / Д. Рутковская, М. Пилиньский, Л. Рутковский; под общ. ред. А. С. Попова - М. : Горячая линия–Телеком. – 2006. – 452 с.

2. Батищев, Д. И. Применение генетических алгоритмов к решению задач дискретной оптимизации : науч. изд. / Д. И. Батищев, Е. А. Неймарк, Н. В. Старостин. – Н. Новгород: Изд-во Нижегор. госуниверситета, 2006. – 88 с.

3. Генетические алгоритмы на примерах решения задач раскроя. Проблемы управления : отчет о НИР / Подлазова А. В. – М.: Московский государственный институт стали и сплавов (технологический университет), 2008. – 7 с.

4. Пантелеев, А. В. Метаэвристические алгоритмы поиска глобального экстремума : учебное пособие / А. В. Пантелеев. – М.: МАИ-ПРИНТ, 2009. – 160 с.

5. Курейчик, В. М. Генетические алгоритмы: науч. изд. / В.М. Курейчик, Л. А. Гладков, В.В. Курейчик ; под ред. В.М. Курейчика. – 2-е изд., испр. и доп. –М.: ФИЗМАЛИТ, 2006. – 320 с.

6. CyberBiology [Электронный ресурс] // хостинг IT-проектов Github. - Режим доступа: https://github.com/CyberBiology/CyberBiology

7. FlexTool [Электронный ресурс] // Астрофизический институт Потсдама. – Режим доступа: https://www.aip.de/Q20_FLEXTOOL.htm

8. Evolver [Электронный ресурс] // Компания Palisade Corporation. - Режим доступа: https://www.palisade.com/evolver/

9. Васильев, А. Н. Java. Объектно-ориентированное программирование : учебное пособие / А. Н. Васильев. - Санкт-Петербург : Питер, 2012. – 400 с.

10. Сузи, Р. А. Язык программирования Python : учебное пособие / Р. А. Сузи. – М. : Бином. Лаборатория знаний. – 2006. – 206 с.

11. Уильямс, Э. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ : учебное пособие / Э. Уильямс; пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2012. – 672с.

12. Боровский, А. Н. Qt 4.7+ Практическое программирование : учебное пособие / Боровский А. Н. – Санкт-Петербург : БХВ-Петербург, 2012. – 496 с.

13. What is SQLite? [Электронный ресурс]: статья // SQLite – Режим доступа: https://www.sqlite.org/index.html.

14. СТО 4.2–07–2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – Введ. 09.01.2014. – Красноярск: ИПК СФУ, 2014. – 60 с.

ПРИЛОЖЕНИЕ А

Листинг методов генетического алгоритма

```
def first_generation(self, cpu_p, memory_p, motherboard_p, case_p, video_p, power_p):
    self.price_list.clear()
    self.db.get_data(cpu_p, memory_p, motherboard_p, case_p, video_p, power_p)
    if any(not self.db.data[key] for key in self.db.data):
        return False
    else:
        for i in range(self.population_count):
            for j, key in enumerate(self.db.data):
                self.population[i]['gen'][j] = randint(0, len(self.db.data[key]) - 1)
            self.population[i]['index'] = i
        for x, i in enumerate(self.population):
            while self.check_compatibility(i):
                self.get_compatibility(i, self.check_compatibility(i))
            self.get_price(i)
            for j in range(self.gen_count):
                i['gen_parent'][j] = x
        return True
def selection(self):
    if self.sel_type == 'Tourney':
        for i in range(self.population_count):
            j = randint(0, self.population_count - 1)
            self.parent_pop.append(deepcopy(self.population[i])
                                   if self.population[i]['fit_func'] < self.population[j]['fit_func']
                                   else deepcopy(self.population[j]))
    elif self.sel_type == 'Roulete':
        s = 0
        for i in self.population:
            s += i['fit_func']
        wheel = [(self.population[0]['fit_func'] / s) * 100]
        for i in range(1, self.population_count):
```

```
  wheel.append(wheel[i-1] + (self.population[i]['fit_func'] / s) * 100)
        for i in range(self.population_count):
            p = randint(1, 100)
            for j in range(self.population_count):
                if p <= wheel[j]:
                    self.parent_pop.append(deepcopy(self.population[j]))
                    break
        print(wheel)
    def crossing(self):
        for _ in range(self.population_count//2):
            f_pos = randint(0, self.population_count - 1)
            while self.parent_pop[f_pos]['crossed']:
                f_pos = randint(0, self.population_count - 1)
            self.parent_pop[f_pos]['crossed'] = True
            kid_1 = deepcopy(self.parent_pop[f_pos])
            f_pos = self.get_hemming_max(f_pos)
            self.parent_pop[f_pos]['crossed'] = True
            kid_2 = deepcopy(self.parent_pop[f_pos])
            if self.cross_type == 'One point':
                cross_point = randint(1, self.gen_count - 1)
                for i in range(cross_point, self.gen_count):
                    kid_1['gen'][i], kid_2['gen'][i] = kid_2['gen'][i], kid_1['gen'][i]
                    kid_1['gen_parent'][i], kid_2['gen_parent'][i], = kid_2['gen_parent'][i],
kid_1['gen_parent'][i]
            elif self.cross_type == 'Two points':
                a = randint(1, self.gen_count - 1)
                b = randint(1, self.gen_count - 1)
                for i in range(min(a, b), max(a, b)):
                    kid_1['gen'][i], kid_2['gen'][i] = kid_2['gen'][i], kid_1['gen'][i]
                    kid_1['gen_parent'][i], kid_2['gen_parent'][i], = kid_2['gen_parent'][i],
kid_1['gen_parent'][i]
            elif self.cross_type == 'Mask':
                for i in range(self.gen_count):
                    if randint(1, 100) % 2 == 0:
                        kid_1['gen'][i], kid_2['gen'][i] = kid_2['gen'][i], kid_1['gen'][i]
```

```
                kid_1['gen_parent'][i], kid_2['gen_parent'][i], = kid_2['gen_parent'][i],
kid_1['gen_parent'][i]
        kid_1['crossed'] = False
        kid_2['crossed'] = False
        self.kid_pop.append(deepcopy(kid_1)), self.kid_pop.append(deepcopy(kid_2))
    def mutation(self):
        x = randint(1, 100)
        index = randint(0, self.population_count - 1)
        if x <= self.mut_value:
            for j, key in enumerate(self.db.data):
                if randint(0, 100) < 35:
                    self.kid_pop[index]['gen'][j] = randint(0, len(self.db.data[key]) - 1)
                    self.kid_pop[index]['gen_parent'][j] = -2
        for i in self.kid_pop:
            while self.check_compatibility(i):
                self.get_compatibility(i, self.check_compatibility(i))
            self.get_price(i)
    def rerun(self):
        fit_func = min([i['fit_func'] for i in self.population])
        if sum([1 for i in self.population if i['fit_func'] == fit_func]) >= self.population_count - 1:
            for i in range(self.population_count// 2, self.population_count):
                for j, key in enumerate(self.db.data):
                    self.population[i]['gen'][j] = randint(0, len(self.db.data[key]) - 1)
                    self.population[i]['gen_parent'][j] = i
                self.population[i]['index'] = i
                while self.check_compatibility(self.population[i]):
                    self.get_compatibility(self.population[i],
self.check_compatibility(self.population[i]))
                self.get_price(self.population[i])
    def change_population(self):
        self.population = deepcopy(self.kid_pop)
        for j, i in enumerate(self.population):
            i['index'] = j
        self.parent_pop.clear()
        self.kid_pop.clear()
```

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О.В. Непомнящий
подпись инициалы, фамилия
«28» 06 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Программная система оптимизации комплектации технических систем на основе генетического алгоритма
тема

Руководитель	20.06.19	доцент, канд.тех.наук	Н.Ю. Сиротинина
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	26.06.19		Э.О. Оюн
	подпись, дата		инициалы, фамилия
Нормоконтролер	27.06.19	доцент, канд.тех.наук	В.И. Иванов
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2019