Санкт-Петербургский государственный университет
информационных технологий, механики и оптики
Факультет информационных технологий и программирования
Кафедра «Компьютерные Технологии»

А. С. Афанасьева

# Отчет по курсовой работе «Библиотека структур данных и генетических операторов, используемых для генерации автоматов с помощью генетических алгоритмов»

Санкт-Петербург
2011

# Оглавление

# Введение

Целью выполнения курсовой работы является реализация структур данных, которые могут использоваться в генетических алгоритмах для представления особей и вычисления функции приспособленности. Для представления особей используются следующие структуры данных: контейнеры строк, контейнеры строк и значимых предикатов, контейнеры деревьев решений. Эти структуры, а также определенные для них генетические операторы, рассматриваются в разд. 2. Для вычисления функции приспособленности особи конвертируются в автоматы, задаваемые полными таблицами, сокращенными таблицами и деревьями решений соответственно. Автоматам посвящен разд. 1.

Использование готовых структур данных позволяет получить ряд достоинств. Устраняется необходимость заново реализовывать известные структуры данных каждый раз при решении той или иной задачи. Как следствие, уменьшается вероятность ошибок при реализации решения задачи. Также удобством использования библиотеки является то, что она предоставляет наиболее эффективные реализации структур данных и операторов.

Написанная библиотека позволяет абстрагироваться от способа представления особи при решении задачи. Выбранная структура данных может быть заменена на другую в любой момент. Также с использованием библиотеки можно сравнивать работу генетических алгоритмов, использующих разные структуры данных. Пример подобного исследования приведен в разд. 3.

Таким образом, разработанная библиотека структур данных позволяет сконцентрироваться на научной работе при исследовании генетических алгоритмов. Библиотека реализована на языке *Java*. Далее будут рассмотрены структуры данных, разработанные в ходе выполнения курсовой работы.

# 1 Представление автоматов

## 1.1 Автомат

В библиотеке реализован структурный смешанный автомат Мура и Мили [1]. Интерфейс автомата `Automaton` позволяет получить доступ к текущему состоянию и соответствующему списку действий. Действия, выполняемые в зависимости от значения текущего состояния, будем называть действиями Мура. Также имеется возможность выполнить переход в зависимости от передаваемых значений входных предикатов. Операция перехода возвращает список выполняющихся на этом переходе действий. Обозначим их как действия Мили.

Описанный интерфейс реализуется классом `MultiAutomaton`. Функция переходов в реализованном автомате может задаваться разными способами для каждого из состояний. Например, множество значений функции переходов в первом состоянии может быть задано полной таблицей, во втором состоянии – деревом решений и т. д. Будем называть такой автомат мультиавтоматом.

Интерфейс `TransitionRepresenter` описывает функцию переходов, заданную для одного состояния. Он позволяет получить описание перехода, состоящее из состояния, в которое выполняется переход, и списка действий Мили. Экземпляры классов, реализующие этот интерфейс, используются для описания переходов в мультиавтомате. UML-диаграмма классов, участвующих в формировании автомата, представлена на рис. 1.

Мультиавтомат, а также классы, описывающие функции переходов, являются неизменяемыми структурами данных. Это позволяет уменьшить вероятность ошибок при реализации генетических алгоритмов. Далее рассмотрим способы представления функции переходов, заданной для одного состояния. Иными словами, рассмотрим различные реализации интерфейса `TransitionRepresenter`.

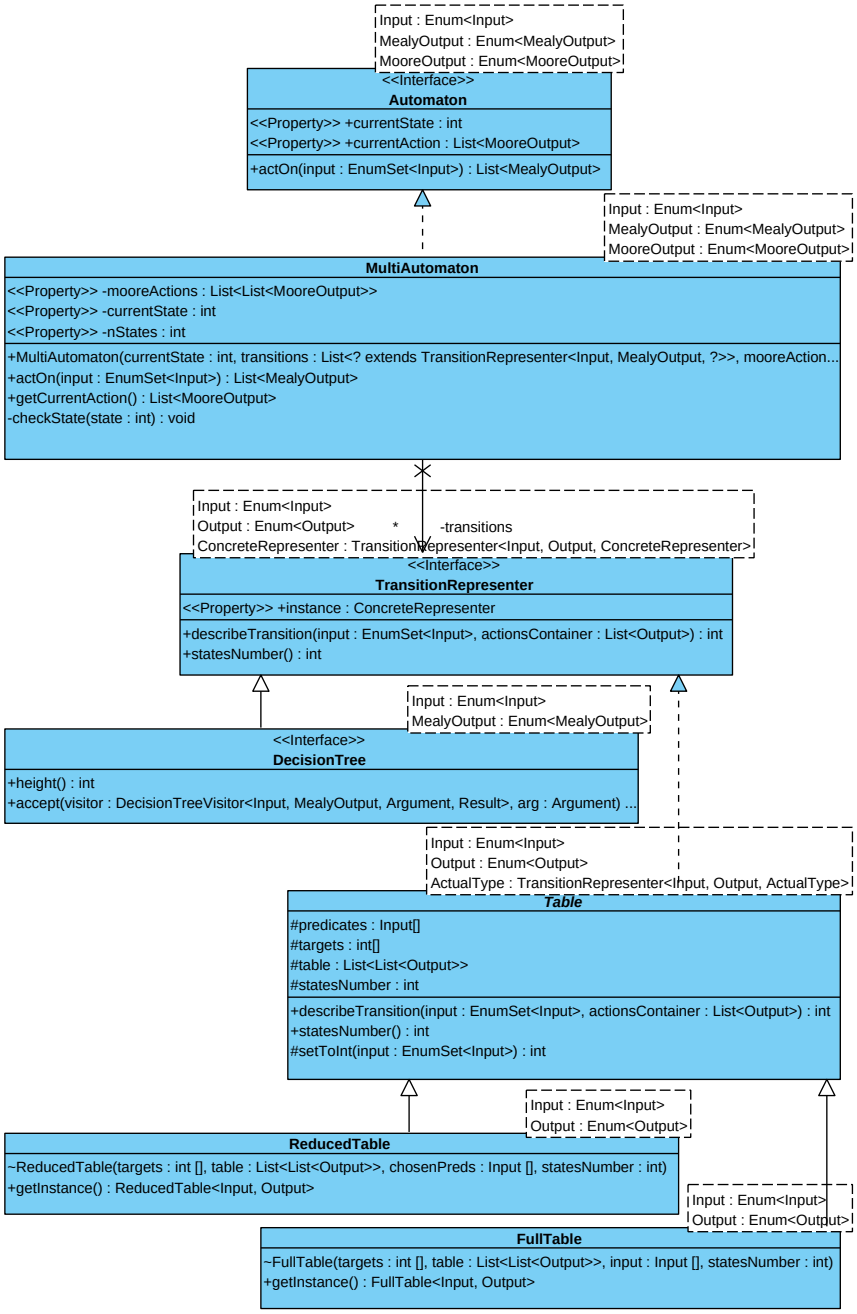


Рис. 1. Диаграмма классов, поясняющая структуру представления автомата

## 1.2 Полные таблицы

Полная таблица представлена классом `FullTable`.FullTable хранит $2^n$ списков выходных воздействий Мили, где $n$ – число входных предикатов.

`FullTable` является неизменяемым классом, как и все другие реализации интерфейса `TransitionRepresenter`. Для создания экземпляров класса `FullTable` следует использовать `FullTableBuilder`.

## 1.3 Сокращенные таблицы

Сокращенные таблицы учитывают лишь ограниченное число входных предикатов, что позволяет сократить расходы памяти и сделать автомат более понятным для человека [2].

Сокращенную таблицу представляет класс `ReducedTable`. По сути, сокращенная таблица является полной таблицей для множества значимых предикатов. Само множество значимых предикатов неизменно для данного экземпляра таблицы, определяется при его создании и хранится вместе с ним.

Экземпляры класса `ReducedTable` создаются с помощью класса `ReducedTableBuilder`, аналогичного классу `FullTableBuiler`.

## 1.4 Деревья решений

Деревья решений также позволяют сэкономить память, необходимую для представления автомата [3]. В библиотеке деревья решений представлены классом `DecisionTree`. Этот класс предоставляет интерфейс для совместного использования с шаблоном проектирования *Visitor* [4], что, в частности, позволяет независимо добавлять новые действия над деревом, не меняя уже написанной реализации.

В качестве примера реализации шаблона *Visitor* можно рассмотреть класс `DecisionTreeSimplifier`, позволяющий устранять недостижимые ветви дерева. Недостижимые ветви могут появляться при генерации случайного дерева, а также в результате некоторых генетических операций. `DecisionTreeSimplifier` возвращает новое дерево, в котором узел с уже встречавшимся предикатом заменен на его дочернюю вершину. Генетические операции над деревьями решений рассмотрены в разд. 2.3.

# 2 Представление особей

Особи, участвующие в генетических алгоритмах, представляются с помощью специальных структур данных, для которых реализованы соответствующие генетические операторы. К этим структурам данных относятся контейнер строк `StringsContainer`, контейнер строк и значимых предикатов `ReducedStrContainer`, а также контейнер деревьев решений `TreesContainer`.

Каждая особь хранит представление некоторого автомата. При возникновении необходимости вычислить функцию приспособленности особи, представленной одним из упомянутых контейнеров, она конвертируются в мультиавтоматы, функции переходов которых представлены полными таблицами, сокращенными таблицами или деревьями решений соответственно.

Для каждого типа особей разработан набор генетических операторов. Генетические операторы реализованы независимо друг от друга и отдельно от самих особей. Это позволяет проводить конфигурацию генетического алгоритма, добавляя в него те или иные операторы. Таким образом, можно легко сравнивать эффективность различных операторов. Пример подобного исследования приведен в разд. 3.

Все структуры данных, описывающие особи, являются неизменяемыми. Это позволяет уменьшить вероятность ошибок при реализации генетических алгоритмов. Далее каждая из структур данных, предназначенных для описания особей, будет рассмотрена более подробно.

## 2.1 Контейнер строк

### 2.1.1 Структура контейнера строк

Контейнер строк `StringsContainer` состоит из трех строк: строки состояний, в которые совершаются переходы (будем называть такие состояния целевыми), строки действий Мили и строки действий Мура. Этот контейнер хранит в соответствующих строках для каждого состояния по $2^n$ целевых состояний и выходных воздействий Мили, где $n$ – число входных предикатов. Также для каждого состояния в строке действий Мура хранится по одному действию Мура. Это представление особи сходно с представлением автомата с помощью строк, описанного в работе [5],

за исключением того, что оно наряду с действиями Мили задает действия Мура.

Пример контейнера строк представлен на рис. 2. В этом примере имеется один входной предикат, который может принимать значения, равные нулю или единице. Число состояний равно трем. Действия Мили могут принимать значения Mealy1, Mealy2 или Mealy3. Предусмотрено два варианта действий Мура: Moore1 и Moore2.

Контейнер строк может быть конвертирован в мультиавтомат, функции переходов в котором представлены полными таблицами. Строка действий Мура или строка действий Мили могут быть пустыми. В этом случае контейнер будет кодировать автомат Мили или автомат Мура соответственно.

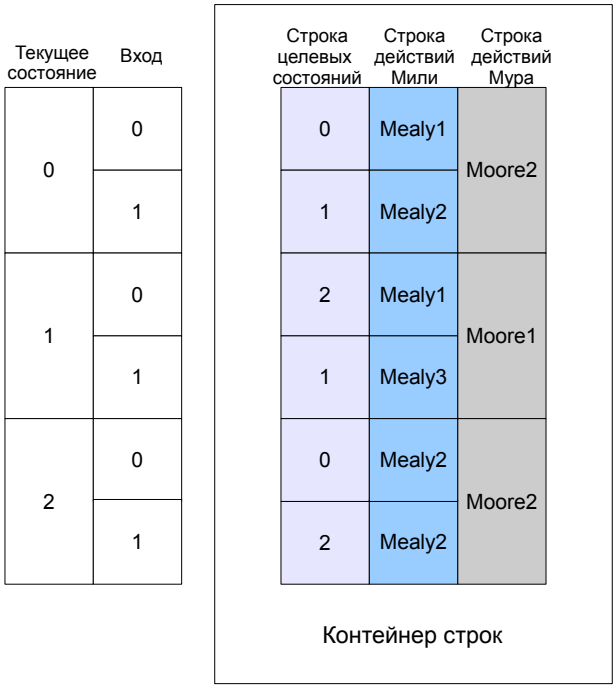


Рис. 2. Пример контейнера строк

Использование контейнеров строк вместо непосредственного использования автоматов, построенных на полных таблицах, позволяет применить универсальный подход к реализации генетических операторов, применяющихся к соответствующим особям. В основе этих операторов

лежат операторы над строками, являющиеся фундаментальной единицей построения всех реализованных в библиотеке генетических операторов.

Реализованы генетические операторы мутации и скрещивания, работающие с контейнерами строк. В их основе лежат мутация и скрещивание строк, механизм которых описан в книге [6].

### 2.1.2 Мутация контейнера строк

При мутации контейнера строк независимо мутируют строки действий Мили и Мура, а также строка целевых состояний. Каждый символ с определенной вероятностью заменяется на один из возможных для данной строки.

### 2.1.3 Многоточечное скрещивание

Скрещивание также выполняется независимо для каждой строки. Случайным образом выбирается несколько точек кроссовера, число которых можно задать в качестве параметра оператора. Затем происходит обмен генетическим материалом.

## 2.2 Контейнер строк и значимых предикатов

### 2.2.1 Структура контейнера строк и значимых предикатов

Контейнер строк и значимых предикатов `ReducedStrContainer` предназначен для представления особи, конвертируемой в мультиавтомат, функции переходов в котором описываются сокращенными таблицами. В этом контейнере хранятся множества значимых предикатов для каждого из состояний. Число значимых предикатов $r$ одинаково для всех состояний.

Также контейнер содержит строку действий Мили, строку целевых состояний и строку действий Мура. Для каждого из $n$ состояний хранится по $2^r$ действий Мили и целевых состояний, а также по одному действию Мура. Таким образом, длины строк действий Мили и строк целевых состояний одинаковы и равны $n \times 2^r$, длина строки действий Мура равна $n$. Так же, как и в `StringsContainer`, строки Мура или Мили могут быть пустыми. В таком случае, контейнер будет кодировать автомат Мили или Мура соответственно.

### 2.2.2 Генетические операторы для множеств значимых предикатов

Для `ReducedStrContainer` реализованы следующие генетические операторы: мутация строк, мутация множеств значимых предикатов, кроссовер строк и множеств значимых предикатов. Мутация и кроссовер строк реализованы так же, как и для `StringsContainer`. Рассмотрим подробнее генетические операторы, работающие с множествами значимых предикатов.

При мутации множества значимых предикатов каждый из предикатов с некоторой вероятностью заменяется другим, который не принадлежит множеству. Кроссовер предикатов выполняется следующим образом: предикаты, значимые для обоих родителей, наследуются обоими потомками, а каждый из тех предикатов, которые были значимы лишь для одного родителя, равновероятно достается одному из детей. Затем вызывается метод, уравнивающий при необходимости размеры множеств предикатов.

## 2.3 Контейнер деревьев решений

### 2.3.1 Структура контейнера деревьев решений

Контейнер деревьев решений `TreesContainer` предназначен для представления особи, конвертируемой в мультиавтомат, функции переходов которого хранятся в виде деревьев решений [3]. Контейнер содержит строку, элементами которой являются деревья решений. Индексы элементов строки соответствуют номерам состояний кодируемого автомата. Таким образом, строка деревьев решений задает функции переходов для каждого из состояний.

Реализованы генетические операторы, работающие с контейнерами деревьев. Представление списка деревьев в виде строки позволяет применить универсальный подход для написания ряда операторов.

### 2.3.2 Мутация контейнеров деревьев решений

Мутация контейнеров деревьев решений основана на мутации строк. Каждый элемент строки с некоторой вероятностью заменяется на новый, как и при мутации контейнера строк. Отличие состоит в том, что новый элемент получается путем генерации случайного дерева. В случае

с контейнером строк новый элемент выбирается из заранее определенного конечного множества действий или целевых состояний. Реализован и другой вариант мутации, при котором дерево заменяется на него же с заново сгенерированным поддеревом.

Обобщение кода оператора мутации достигается путем применения генератора элементов строки. В случае контейнера строк оператору передается генератор элементов конечного множества, в случае контейнера деревьев решений соответствующие операторы работают с генераторами случайных деревьев и с генераторами деревьев со случайным поддеревом.

### 2.3.3 Скрещивание контейнеров деревьев решений

Многоточечное скрещивание контейнеров деревьев представляет из себя многоточечное скрещивание соответствующих строк, рассмотренное в разд. 2.1. Специфическим для контейнеров деревьев решений генетическим оператором является второй вариант скрещивания, при котором происходит обмен поддеревьями между двумя случайно выбранными деревьями из родительских контейнеров.

# 3 Демонстрация использования библиотеки

## 3.1 Модифицированная постановка задачи о флибах

В качестве модельной задачи для демонстрации возможностей написанной библиотеки была использована модифицированная задача о флибах. За основу была взята задача о флибах, описанная в работе [7]. Задача состоит в том, чтобы построить флиб – автомат Мили, который предсказывает битовое значение некоторого параметра окружающей среды, обладающего периодичностью. Модификация задачи состоит в том, что на вход флиба подается не только битовое значение окружающей среды, но и некая дополнительная константа. Введение дополнительного параметра позволяет продемонстрировать использование сокращенных таблиц и деревьев решений.

В ходе исследования были реализованы решения как исходной, так и модифицированной задач о флибах. Совместно с разработанной библиотекой применялся генетический алгоритм, реализация которого была взята из системы эволюционных вычислений *Watchmaker* (http://watchmaker.uncommons.org/). Благодаря использованию структур данных и генетических операторов, разработанных в ходе выполнения курсовой работы, решение задач свелось к реализации вычисления функции приспособленности.

Для обеих задач была применена одна и та же реализация функции приспособленности, учитывающая число использованных состояний автомата. Идея минимизации числа состояний в задаче о флибах изложена в работе [8].

## 3.2 Условия эксперимента

Проводился эксперимент, аналогичный первому эксперименту, описанному в работе [8]. В качестве битовой маски, соответствующей одному периоду воздействий среды на флиб, была выбрана маска, имеющая вид 11110100101111101001. Число воздействий среды на флиб – 100, число поколений – 400, число особей в одном поколении - 100. Особи представлялись в виде строк.

Результаты решения исходной задачи сравнимы с результатами, полученными в упомянутой работе. Для успешного решения модифицированной задачи потребовалось большее число поколений.

В следующих разделах исследуется влияние различных параметров на характеристики решения модифицированной задачи о флибах. Таким образом демонстрируются возможности библиотеки по проведению подобных исследований.

## 3.3 Результаты работы алгоритма при разном числе особей в поколении

На рис. 3 представлены графики зависимости значения функции приспособленности (ФП) лучшего индивидуума в поколении от числа вычислений функции приспособленности. Графики построены для различного числа особей в поколении. Исходя из полученных результатов, можно сделать вывод, что наиболее эффективным является использование 10 - 20 особей.

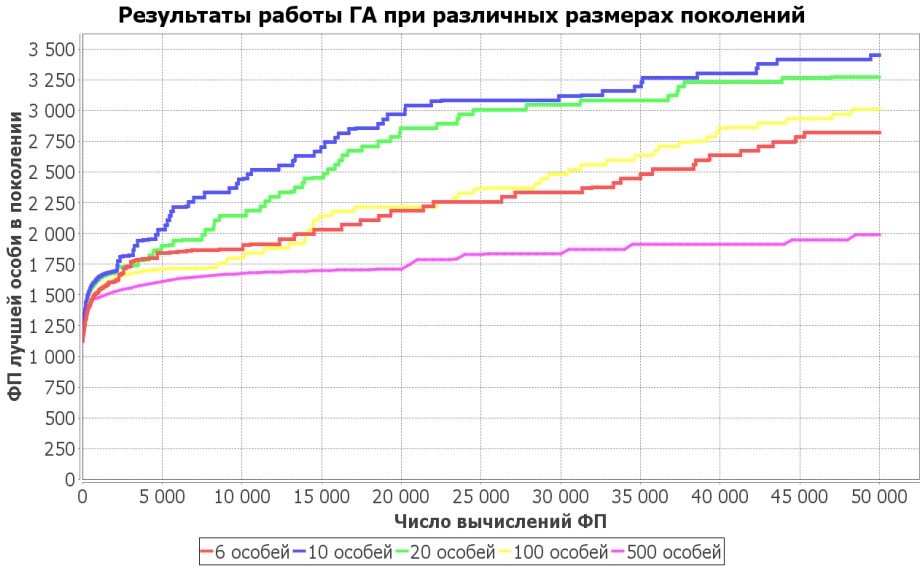


Рис. 3. Графики работы генетического алгоритма на полных таблицах, запущенного с различным числом особей в поколении

## 3.4 Сравнение различных генетических операторов

Для проведения следующих экспериментов была использована модифицированная задача о флибах, число поколений было выбрано равным 3000. Алгоритм запускался 30 раз, затем проводилось усреднение.

Применение библиотеки структур данных позволило сравнить воздействия выбора различных комбинаций генетических операторов на производительность алгоритма. На рисунках 4, 5, 6 представлены сравнительные графики производительности генетического алгоритма, запущенного с использованием различных операторов для разных способов представления особей. На графиках показана зависимость среднего значения функции приспособленности от числа поколений. Можно видеть, что для модифицированной задачи о флибах в случае использования сокращенных таблиц и деревьев решений операторы кроссовера не эффективны.

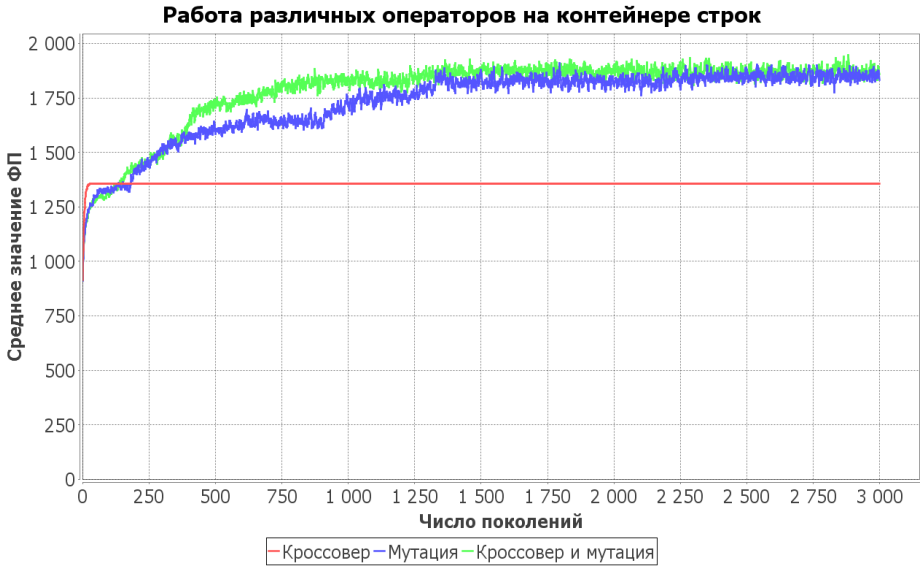


Рис. 4. Графики работы генетического алгоритма, запущенного с использованием различных операторов на контейнерах строк

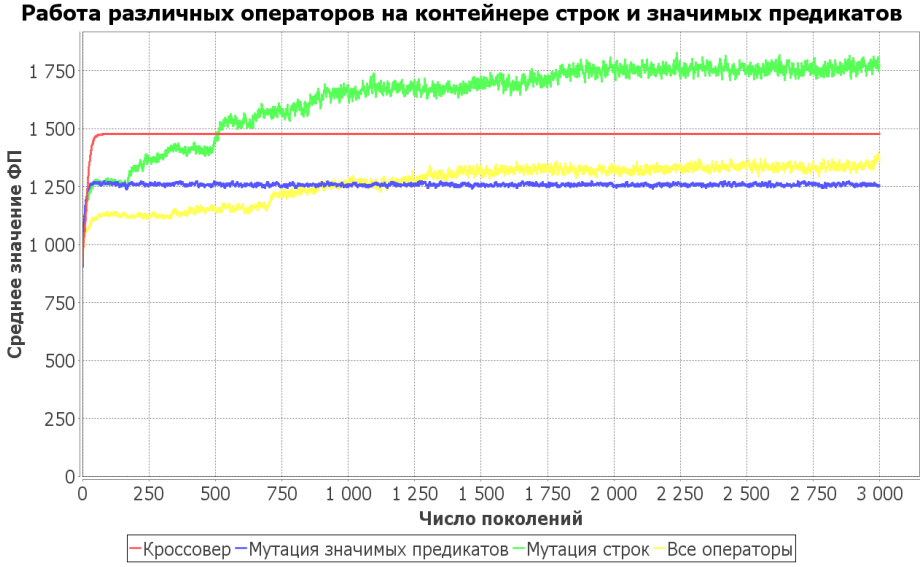


Рис. 5. Графики работы генетического алгоритма, запущенного с использованием различных операторов на контейнерах строк и значимых предикатов

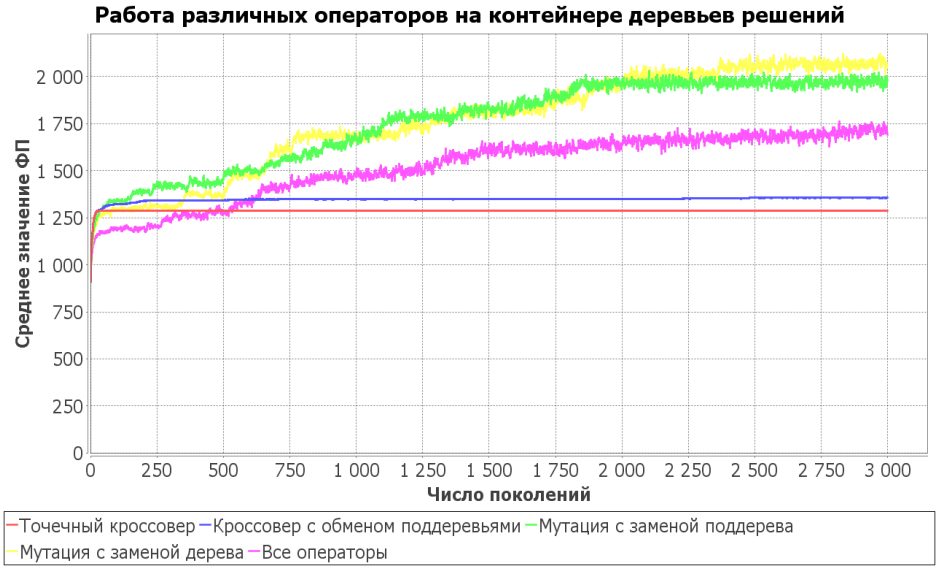


Рис. 6. Графики работы генетического алгоритма, запущенного с использованием различных операторов на контейнерах деревьев решений

## 3.5 Сравнение различных структур данных

Также был проведен анализ пригодности различных способов представления особей для решения модифицированной задачи о флибах. На рис. 7 представлены графики зависимости максимального значения функции приспособленности от числа поколений для контейнеров строк, контейнеров строк и значимых предикатов, а также для контейнеров деревьев решений.

Из графиков следует, что наибольшую производительность показывает решение, использующее контейнеры строк для представления особей и, соответственно, полные таблицы для вычисления функции приспособленности. Таким образом, в данной задаче негативное влияние лишнего входного предиката – дополнительной константы – перевешивает достоинства использования меньшего количества памяти, которые можно было ожидать от использования сокращенных таблиц и деревьев решений.

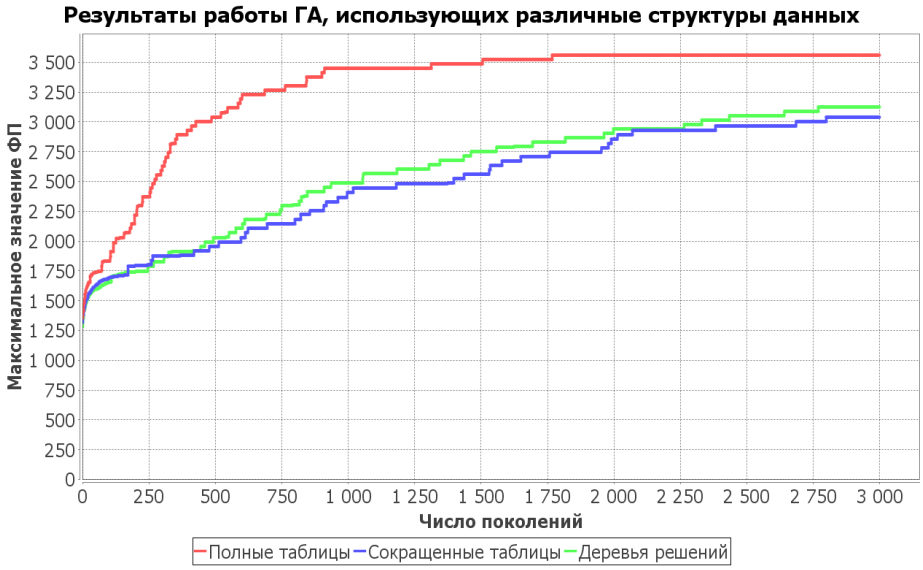


Рис. 7. Графики работы генетического алгоритма, запущенного с использованием различных структур данных

# Заключение

В ходе выполнения курсовой работы была создана библиотека структур данных, используемых в генетических алгоритмах. Библиотека расширяема, предоставляет необходимую функциональность и удобна для использования. Применение библиотеки позволяет свести решение задач генетического программирования к реализации расчета функции приспособленности.

Гибкость использования библиотеки заключается в возможности выбора способа представления особей, а также выбора генетических операторов, наиболее подходящих для решения поставленной задачи. Такая возможность обеспечивается простотой замены представления особей. В ходе исследования применимости библиотеки к решению модельной задачи, которая представляла из себя модифицированную задачу о флибах, был сделан вывод о нецелесообразности использования сокращенных таблиц и деревьев решений для этой задачи.

Разработанная библиотека структур данных позволяет сконцентрироваться на научной работе при исследовании генетических алгоритмов.

# Источники


1. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. – СПб.: Питер, 2009. – 176 с.: ил.

2. Точилин В. Н. Метод сокращенных таблиц для генерации автоматов с большим числом входных воздействий на основе генетического программирования

   http://is.ifmo.ru/papers/tochilin/doc.pdf

3. Данилов В. Р., Шалыто А. А. Метод генетического программирования для генерации автоматов, представленных деревьями решений

   http://is.ifmo.ru/download/2008-03-07-danilov.pdf

4. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2010. – 366 с.: ил.

5. Данилов В. Р. Методы представления функции переходов при генерации автоматов управления на основе генетического программирования

   http://is.ifmo.ru/papers/_2010_02_25_danilov.pdf

6. Mitchell M. An Introduction to Genetic Algorithms. MIT Press. Cambridge. MA, 1996

7. Лобанов П. Г., Шалыто А. А. Использование генетических алгоритмов для автоматического построения конечных автоматов в задаче о «флибах»

   http://is.ifmo.ru/works/flib/

8. Мандриков Е. А., Кулев В. А., Шалыто А. А. Применение генетических алгоритмов для создания управляющих автоматов в задаче о «флибах»

   http://is.ifmo.ru/download/2008-02-23_flibs.pdf