На правах рукописи

Сердюков Константин Евгеньевич

РАЗРАБОТКА СИСТЕМ ИНТЕЛЛЕКТУАЛЬНОЙ ПОДДЕРЖКИ АНАЛИЗА И ТЕСТИРОВАНИЯ ПРОГРАММ

Специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени
кандидата технических наук

Новосибирск – 2022

Работа выполнена в Федеральном государственном бюджетном образовательном учреждении высшего образования «Новосибирский государственный технический университет»

Научный руководитель:	доктор технических наук, профессор **Авдеенко Татьяна Владимировна**
Официальные оппоненты:	**Денисова Людмила Альбертовна,** доктор технических наук, доцент, Федеральное государственное автономное образовательное учреждение высшего образования «Омский государственный технический университет», кафедра автоматизированных систем обработки информации и управления, профессор; **Попов Фёдор Алексеевич,** доктор технических наук, профессор, Открытое акционерное общество «Федеральный научно-производственный центр «Алтай», отделение вычислительной техники и автоматики, главный научный сотрудник
Ведущая организация:	Федеральное государственное бюджетное образовательное учреждение высшего образования «Томский государственный университет систем управления и радиоэлектроники», г. Томск

Защита диссертации состоится «09» сентября 2022 г. в 15:00 часов на заседании диссертационного совета Д 212.173.06 при Федеральном государственном бюджетном образовательном учреждении высшего образования «Новосибирский государственный технический университет» по адресу: 630073, Новосибирск, пр. К. Маркса, 20, I корпус, конференц-зал.

С диссертацией можно ознакомиться в библиотеке Новосибирского государственного технического университета и на сайте http://www.nstu.ru.

Автореферат разослан «___» июля 2022 г.

Ученый секретарь
диссертационного совета, к.т.н.  Фаддеенков Андрей Владимирович

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность и степень разработанности темы исследования. В настоящее время уровень развития интеллектуальных технологий позволяет говорить о возможности их применения в наиболее сложных, инновационных сферах, к которым, несомненно, относится программная инженерия, занимающаяся созданием, внедрением и сопровождением современного программного обеспечения (ПО). Классический жизненный цикл ПО включает в себя такие этапы, как определение и анализ требований, проектирование, программирование, тестирование и отладка, эксплуатация и сопровождение.

Для обеспечения высокого качества разрабатываемого ПО большое значение имеют методы верификации. Они применяются на всех этапах жизненного цикла и позволяют сделать вывод о качестве разрабатываемого ПО на основе проверки соответствия между программой и заявленными к ней требованиями. Таким образом, верификация играет важную роль в процессе разработки, а тестирование, относящееся к группе методов динамической верификации, выделяется в отдельный этап жизненного цикла ПО.

Этап тестирования, направленный на проверку соответствия между ожидаемыми результатами и реальным поведением программы на специально подобранном наборе тестов (тестовых данных), является одним из самых дорогостоящих и трудозатратных, и может занимать до 40–60% от общего времени создания ПО. Поэтому разработка моделей и алгоритмов интеллектуальной поддержки этапа тестирования ПО является актуальной задачей.

Тестированием программного обеспечения, в том числе системного, а также разработкой тестов, занимались такие ученые как Бурдонов И.Б., Денисова А.Л, Панков Д.А., Мутилин В.С. Новые методы верификации предикатных и автоматных программ, а также средств их визуализации, предложены в работах Шелехова В.И., Непомнящего В.А., Зюбина В.Е. Вопросами тестирования и верификации программных средств при разработке автоматизированных систем управления в области спецхимии занимались Попов Ф.А., Кащеева Е.В.; в области добычи угля – Окольнишников В.В; в аэрокосмической области – Тюгашев А.А.

Генерация тестовых данных – сложный и трудоемкий процесс. Его автоматизация, хотя бы частичная, является актуальной исследовательской задачей, решение которой могло бы повысить эффективность тестирования ПО. Анализ существующих исследований, методов и подходов в области применения методов автоматической генерации тестовых данных показал, что в настоящее время преимущественно используемым в производстве программного обеспечения подходом является применение слепой стратегии случайной генерации тестовых данных. В то же время, анализ проводимых научных исследований показывает, что существуют подходы, разработка и применение которых может значительно улучшить качество генерируемых тестов, выражаемое в степени покрытия ими

тестируемого кода (термин «покрытие кода» означает прохождение вычислений тестируемой программы, инициированное множеством тестовых наборов, по максимально возможному числу путей[1]).

Среди таких продвинутых подходов к генерации тестовых данных исторически первыми появились статические методы символьного анализа кода программы. Генерация тестовых данных в результате такого анализа сводилась к автоматическому формированию и разрешению в символьном виде системы уравнений и неравенств, получающихся логическим объединением и пересечением всех условий в тестируемой программе. Несомненным достоинством статического подхода является получение результатов в символьном виде, что дает возможность аналитически определять подобласти значений тестовых наборов, которые гарантируют проход вычислений по заданным частям кода.

Однако существенным ограничением возможности применения статического подхода является проблема вычислительной сложности символьных вычислений даже для задач относительно небольшой размерности. Поэтому в настоящее время более реалистичным и эффективным для практического использования в компаниях по производству ПО является динамический подход, основанный на фактическом выполнении тестируемой программы при сгенерированных специальным образом значениях входных переменных (наборе тестов) и последующем анализе потоков данных. Развитием этого направления занимались Давыдов А.А., Demillo R., Gelrich R., Nikravan E., Parsa S. и т.д.

Наиболее перспективными методами реализации динамического подхода к генерации тестовых данных являются эволюционные методы оптимизации. Эволюционная парадигма, которая лежит в основе генетического алгоритма (ГА), использует множество случайных тестовых данных, сгенерированных на начальном этапе, после чего проводится последовательная «эволюция» данных с целью улучшения качества покрытия тестируемого кода. В связи с вышеизложенным возникает предположение о возможности адаптации генетического алгоритма для реализации идеи эволюционного улучшения тестовых данных с точки зрения максимизации покрытия ими тестируемого кода.

Вопросы применения генетического алгоритма к генерации тестовых данных рассматриваются в работах Anusha M., Berndt D., Girdis M. Maragathavalli P., Praveen R., Watkins A. и др. Однако существующие исследования сосредоточены на решении локальных проблем, например, на нахождении конкретного набора данных, покрывающего отдельные заданные операторы. В то же время для решения практической задачи всестороннего тестирования ПО актуальна разработка методов генерации множества наборов, обеспечивающих наиболее

[1] Spillner, A., Linz, T., Schaefer, H. Software Testing Foundations. A Study Guide for the Certified Tester Exam // Rocky Nook. – 2014. – 305 p.

полное покрытие всего тестируемого кода, с учетом его многосвязной сложнологической структуры, включая рекурсию.

Цель работы заключается в разработке и исследовании методов автоматической генерации тестовых данных на основе модификаций генетического алгоритма для наиболее полного покрытия программного кода.

Для достижения заданной цели поставлены и решены следующие **задачи**:

1. Провести анализ существующих исследований, методов и подходов в области применения методов автоматической генерации тестовых данных;
2. Разработать алгоритм генерации набора тестовых данных для покрытия наиболее сложного пути тестируемого кода на основе метрик оценки сложности;
3. Модифицировать разработанный алгоритм для получения множества наборов данных, обеспечивающих наиболее полное покрытие кода;
4. Исследовать различные варианты функции приспособленности ГА для обеспечения наибольшего покрытия за счёт большего разнообразия множества сгенерированных тестовых наборов;
5. Реализовать приложение для генерации множества наборов тестовых данных с использованием предложенного алгоритма и его модификаций.

Область исследования. Диссертация соответствует области исследования п.1 «Модели, методы и алгоритмы проектирования и анализа программ и программных систем, их эквивалентных преобразований, верификации и тестирования» паспорта специальности 05.13.11 – «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей».

Объектом исследования является процесс генерации наборов тестовых данных для исходного кода программного обеспечения как основа для этапа тестирования в программной инженерии.

Предметом исследования являются подходы, методы и алгоритмы структурного тестирования, обеспечивающие генерацию тестовых данных.

Теоретическая значимость проведённого исследования заключается в развитии эволюционного подхода для решения задачи автоматической генерации тестовых данных для наиболее полного покрытия кода за счет формулирования специального вида функции приспособленности, учитывающей не только сложность пути, но и разнообразие множества тестовых наборов.

Практическая значимость. Результаты диссертационного исследования могут использоваться в компаниях, занимающихся разработкой ПО, для автоматической генерации тестовых наборов данных с целью сокращения времени и затрат, повышения качества тестирования в целом.

Научная новизна исследования заключается в следующем:

1. Осуществлена формальная постановка задачи генерации тестовых данных для ее решения с помощью генетического алгоритма, включающая математический вид функции приспособленности на основе метрик оценки сложности кода;

2. Сформулированы две новые модификации функции приспособленности, направленные на увеличение степени покрытия тестируемого кода. Первая модификация заключается во введении дополнительной аддитивной компоненты в функцию приспособленности, отвечающей за разнообразие популяции. Во второй модификации разнообразие достигается за счет динамического изменения весов операторов в зависимости от степени их покрытия в предшествующем поколении;

3. Разработаны эвристические алгоритмы генерации тестовых данных на основе предложенных формальных эволюционных постановок задач и различных вариантов функции приспособленности;

4. Разработано программное приложение, реализующее предложенные методы генерации тестовых наборов с модифицированной функцией приспособленности для обеспечения максимального покрытия тестируемого кода с минимально необходимым количеством наборов.

Методы исследования. Основой методологии представленного исследования является теория программной инженерии, модели, методы и алгоритмы тестирования ПО, методы оптимизации, генетический алгоритм.

Основные положения, выносимые на защиту:

1. Формальная постановка задачи генерации тестовых данных и математический вид функции приспособленности на основе метрик сложности кода;

2. Алгоритмы генерации данных на основе многократного запуска и алгоритма с дополнительным параметром, отвечающим за разнообразие;

3. Модификации функции приспособленности, обеспечивающие большее разнообразие тестовых наборов;

4. Программное приложение, реализующее предложенные методы и подходы.

Степень достоверности результатов работы. Достоверность полученных результатов определяется использованием современного научно-методического аппарата, а также взаимной согласованностью результатов, полученных для различных сочетаний параметров и модификаций разработанных алгоритмов.

Представленные в работе научные результаты получены в рамках выполнения работ по грантам – Грант Министерства образования и науки РФ в рамках проектной части государственного задания, проект № 2.2327.2017/4.6 «Интеграция моделей представления знаний на основе интеллектуального анализа больших данных для поддержки принятия решений в области программной инженерии» (2017-2019 г.); Грант Российского фонда фундаментальных

исследований в рамках выполнения научного проекта № 19-37-90156 (Аспиранты) «Разработка и исследование методов интеллектуального анализа и тестирования программного кода» (2019-2021 г.); Грант Министерства Науки и Высшего Образования Госзадания проект № FSUN-2020-0009 «Моделирование системной организации когнитивных функций с применением интеллектуального анализа массивов психометрических и нейрофизиологических данных» (2020-2022 г.).

Предлагаемые автором методы и алгоритмы генерации тестовых данных на основе эволюционных алгоритмов используются при разработке программного обеспечения в ООО «Дежавю». Результаты работы используются в учебном процессе Новосибирского государственного технического университета в рамках дисциплин «Интеллектуальные информационные системы», «Интеллектуальные системы и технологии», «Программная инженерия», «Методы оптимизации».

Апробация работы. Основные результаты работы были представлены на конференциях: 12th International Conference on Advances in Swarm Intelligence (ICSI'21) (Qingdao, China, 2021); 14th International Symposium "Intelligent Systems – 2020" (INTELS'20) (г. Москва, 2020); 12th International Conference "Data Analytics and Management in Data Intensive Domains" (DAMDID) (г. Воронеж, 2020); междунар. конф. и молодеж. шк. «Информационные технологии и нанотехнологии» (г. Самара, 2017, 2018, 2019, 2020, 2021); Раб. семинар в рамках 12 междунар. Ершовской конф. по информатике (PSI'19) (г. Новосибирск, 2019); всерос. науч.-техн. конф. студентов, аспирантов и молодых ученых с междунар. участием «Измерения, автоматизация и моделирование в промышленности и научных исследованиях» (г. Бийск, 2018, 2019); всерос. науч. конф. молодых ученых «Наука. Технологии. Инновации» (г. Новосибирск, 2016, 2017).

Публикации. По теме диссертационной работы опубликовано 23 работы, в том числе 2 статьи опубликованы в научных журналах из перечня ВАК, 10 публикаций - в изданиях, индексируемых в Web of Science и Scopus, 2 свидетельства о регистрации программы для ЭВМ, 9 публикаций в сборниках трудов международных и российских конференций.

Личный вклад автора. Результаты научных исследований, представленных в диссертационной работе, были получены при непосредственном участии соискателя, которое заключалось в разработке и реализации алгоритмов генерации тестовых данных, постановке вычислительных экспериментов и апробации полученных результатов. Доля личного вклада в публикациях, выполненных в соавторстве, составляет не менее 50%.

Структура и объем работы. В диссертации представлено введение, четыре главы по тематике исследования, заключение, список литературы, в котором содержится 110 наименований, и 4 приложения. Полный объём работы составляет 166 страниц, включая 10 таблиц и 50 рисунков.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обосновывается актуальность темы диссертационной работы, формулируются цели и задачи исследования, описывается научная новизна, формулируются положения, выносимые на защиту, приводятся сведения об апробации, указывается личный вклад соискателя.

В **первой главе** рассматриваются теоретические основы программной инженерии, описываются основные модели и этапы жизненного цикла разработки ПО. Приводится обзор методов верификации и тестирования для оценки качества разрабатываемых программ. Более подробно анализируется процесс тестирования как важный этап разработки ПО, подчеркивается важность генерации тестовых данных для обеспечения качественного тестирования.

Основной целью генерации тестовых данных является создание такого множества тестовых наборов, которое обеспечило бы достаточный уровень качества конечного программного продукта путем проверки большей части различных путей программы, т.е. обеспечило бы максимальное покрытие кода в соответствии с выбранными критериями качества.

В работе используется динамический подход к генерации данных, который основан на фактическом выполнении кода и динамическом анализе потока данных. Для тестируемой программы можно определить граф потоков управления (Control-flow graph, CFG) как направленный граф $CFG = (V, R, v0, vE)$, где V – набор узлов графа, R – подмножество декартова произведения $V \times V$, определяющее бинарное отношение на V (множество дуг графа), $v0$ и vE – входной и выходной узлы, соответственно, $v0 \in V,\ vE \in V$.

Узел в V соответствует наименьшей исполняемой части оператора, т.е. соответствует оператору присваивания, оператору ввода или вывода, или части <выражения> от *if-then-else* или *while* операторов. Ребро (ветвь) графа $(v_i,\ v_j)$ соответствует возможной передаче управления от узла v_i к узлу v_j. Каждая ветвь в может быть помечена предикатом, определяющим условия, при которых эта ветвь будет пройдена при очередном запуске программы.

Использование графа потоков управления позволяет определить путь, по которому прошли вычисления при выборе соответствующего тестового набора. В этом случае говорят, что тестовый набор обеспечил покрытие упорядоченного подмножества узлов графа, расположенных на этом пути. В диссертации описываются различные критерии покрытия, наибольший интерес из которых представляют критерии покрытия путей и операторов. Критерий покрытия путей определяет долю покрытых тестовым набором путей графа относительно всех возможных путей, а критерий покрытия операторов – долю покрытых узлов графа.

В качестве иллюстративного примера рассмотрим простую программу, граф потоков управления которой представлен на рисунке 1. Как видно из рисунка, было использовано восемь тестовых наборов, которые проходят по четырем различным

путям кода. Проход по пути A обеспечивает покрытие подмножества узлов $\{a, b, d, f, h, i, j, k, l, m, n, o, p, q\}$, проход по пути B – покрытие того же подмножества $\{a, b, d, f, g, i, j, k, l, m, n, o, p, q\}$, проход по пути C – покрытие подмножества $\{a, b, d, e, j, k, l, m, n, p, q\}$, последний путь D обеспечивает покрытие подмножества $\{a, b, c, k, l, p, q\}$.

Таким образом, в данном случае достаточно четырех тестовых наборов, обеспечивающих полное покрытие узлов при проходе по четырем путям, хотя общее число путей графа – 12. В общем случае получаем экспоненциальную зависимость числа путей от количества и глубины ветвлений в программе. И хотя критерий покрытия путей может дать возможность выявления потенциально большего числа ошибок, вычислительные затраты на их выявление чрезмерно высоки. Поэтому в работе используется критерий покрытия операторов как наиболее реалистичный и достаточно результативный (с точки зрения выявления ошибок) критерий качества тестовой выборки, позволяющий сгенерировать наименьшее количество тестовых наборов, обеспечивающих полное покрытие тестируемой программы.

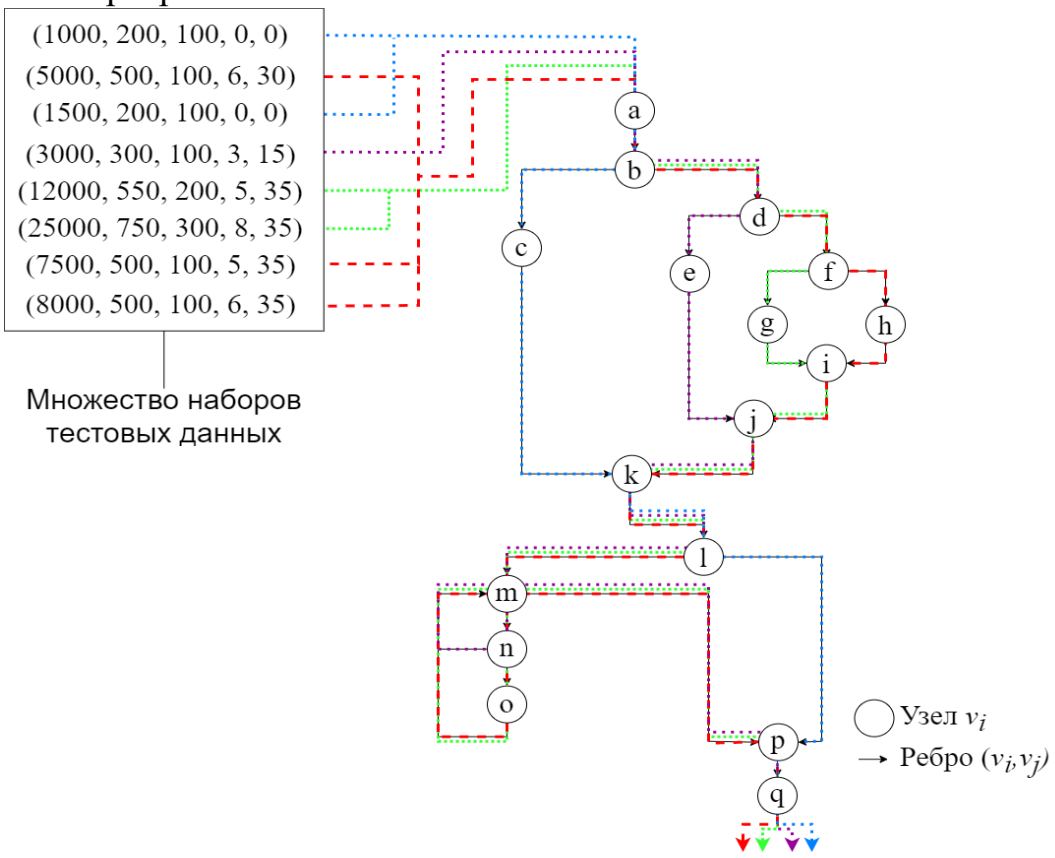

Рисунок 1 – Покрытие тестовыми наборами графа потоков управления

Наиболее часто применяемой на практике стратегией генерации тестовых данных является использование слепой стратегии псевдослучайной генерации. Данные, сгенерированные таким способом, редко достигают приемлемого качества покрытия для всего программного кода. В литературе предлагается использование

эволюционной стратегии, конкретнее, генетического алгоритма, для целенаправленного улучшения начальной выборки, сгенерированной слепым методом. Однако в большинстве работ исследователи ограничиваются нахождением тестовых наборов для покрытия ограниченных частей тестируемого кода, например, отдельных, наиболее трудоемких, путей программы. Цель покрытия всего программного кода заявляется лишь в отдельных исследованиях.

Таким образом, в главе обосновывается и формулируется цель диссертационной работы, заключающаяся в разработке и исследовании методов автоматической генерации тестовых данных на основе модификаций генетического алгоритма для наиболее полного покрытия тестируемого кода.

Во второй главе приводится формализация задачи генерации тестовых данных для ее решения с помощью генетического алгоритма, описываются основные особенности эволюционных операций, предлагается алгоритм генерации тестовых данных для одного сложного пути, и исследуются метрики оценки сложности кода для их использования в качестве весов функции приспособленности.

Определим $(var_1, var_2, ..., var_N)$ – вектор входных переменных тестируемого кода; область определения входных переменных $D = D_1 \times D_2 \times ... \times D_N$, где D_i – область определения входной переменной var_i. Также определим путь в графе как набор узлов $P = \langle v0, v_{i_1}, \ldots, v_{i_k}, \ldots, vE \rangle$, таких что $\left(v_{i_k}, v_{i_{(k+1)}} \right) \in R$ (множество дуг графа). Путь P достижим, если существует входной тестовый набор, приводящий к прохождению потока управления по этому пути, в противном случае путь P недостижим.

Цель генерации данных – найти множество тестовых наборов $\{x_1, x_2, \ldots, x_m\}$, $x_i \in D$, инициирующих прохождение по заданному множеству достижимых путей, которое может содержать один путь, несколько путей, или все множество достижимых путей (полное покрытие кода). Для случая, если требуется покрыть только один путь, в качестве критерия качества тестового набора можно использовать функцию, которая задаёт ненулевые веса тем узлам графа, по которым проходит рассматриваемый путь P:

$$\mathrm{F}(x) = \sum_{j=1}^{n(x)} w_j(x), \qquad (1)$$

где $w_j(x)$ – ненулевые веса, соответствующие пути P и входному вектору $x \in D$; $n(x)$ – количество операторов на рассматриваемом пути.

Рассмотрим формальную постановку задачи генерации тестовых данных для ее решения с помощью генетического алгоритма. В соответствии с терминологией ГА определим популяцию особей, состоящую из m хромосом $\{x_1, x_2, \ldots, x_m\}$, где каждая хромосома $x_i = [var_1^i, var_2^i, \ldots, var_N^i]$, соответствующая одному набору тестовых данных, состоит из N генов (значений N входных переменных). Основной цикл ГА для генерации тестовых данных включает следующие этапы, которые

выполняются итерационно до достижения максимально возможного покрытия или заданного числа поколений:

1. *Инициализация.* Исходная популяция формируется случайным образом с учетом ограничений на значения входных переменных. Объем популяции m выбирается на основе размера тестируемой программы.

2. *Оценка популяции.* Каждая хромосома популяции оценивается функцией приспособленности (например, функцией (1) в случае необходимости покрытия заданного пути P).

3. *Селекция (Отбор).* Лучшие 20% хромосом отбираются в неизменном виде для следующего поколения; остальные 80% хромосом следующего поколения будут получены в результате скрещивания. Данная пропорция получена эмпирически и позволяет обеспечить достаточное разнообразие популяции с высокой скоростью сходимости.

4. *Скрещивание.* Половина особей следующего поколения формируется путем случайного скрещивания 20% лучших хромосом предыдущего поколения друг с другом. Остальные хромосомы будут получены путем случайного скрещивания всех хромосом предыдущего поколения друг с другом. Скрещивание происходит путем применения операции смешивания генов, где l-й ген потомка является линейной комбинацией соответствующих генов родительских хромосом:

$$var_l^{offspring} = \beta_l * var_l^{mother} + (1 - \beta_l) * var_l^{father}, l = \overline{1, N},$$

а константа $\beta_l \in [0,1]$ выбирается случайным образом для каждого $l = \overline{1, N}$.

5. *Мутация.* С заданной вероятностью мутации (0.05) каждый ген может изменить свое значение на случайное в рамках заданных ограничений на входные переменные. Основная цель мутации – достижение большего разнообразия.

6. *Формирование тестовых наборов данных в виде пула элитных хромосом.* В каждом поколении происходит отбор особей популяции в пул элитных хромосом, обеспечивающих дополнительное покрытие кода по сравнению с предшествующим покрытием.

Вычисление весов в функции приспособленности (1) может быть проведено с использованием различных метрик сложности кода. Для того, чтобы применить метрики к вычислению функции приспособленности, сформулируем выражение, в котором в явном виде указывается ее связь с операторами тестируемого кода. Обозначим $g(x_i)$ – вектор, являющийся индикатором покрытия операторов, инициированным определенным тестовым набором x_i:

$$g(x_i) = \big(g_1(x_i), g_2(x_i), \dots, g_n(x_i)\big),$$

где n – количество операторов тестируемой программы,

$$g_j(x_i) = \begin{cases} 1, \text{если путь, инициированный набором } x_i, \text{проходит через оператор } j \\ 0, \text{ в противном случае} \end{cases}$$

Введя обозначение $(w_1, w_2, \dots, w_n)$ для вектора весов, присвоенных всем операторам программы, функция приспособленности (1) для отдельной хромосомы x_i может быть представлена в следующем виде:

$$F_1(x_i) = \sum_{j=1}^{n} w_j g_j(x_i). \quad (2)$$

Присваивая различные веса различным операторам, можно учесть тот факт, что разные пути выполнения тестируемой программы имеют отличающуюся сложность. Больший вес назначается критическим операторам, которые расположены на путях, наиболее подверженных ошибкам, или выполняемых наиболее часто.

Значения весов w_j функции приспособленности в выражении (2) могут быть определены в соответствии с метриками сложности кода. В диссертации были проанализированы следующие метрики: метрика SLOC (Source Lines Of Code), метрика ABC; метрика Джилба и метрика NOD (Nested Operation Division). Метрика NOD использует более сложную структуру распределения весов для приоритизации операторов высокого уровня. В результате анализа выбраны две метрики – NOD и SLOC, которые были использованы в дальнейших исследованиях при вычислении функции приспособленности.

В **третьей главе** сформулированы модификации функции приспособленности, направленные на увеличение степени покрытия тестируемого кода. Разработаны эвристические алгоритмы генерации тестовых данных на основе формальных эволюционных постановок задач и различных вариантов функции приспособленности.

Генетический алгоритм на основе функции приспособленности вида (2) хорошо решает локальную задачу поиска набора данных, покрывающего наиболее сложный путь (или заданный путь, если выбирать веса операторов определенным образом). Это означает, что популяция хромосом в последнем поколении будет вырождаться, то есть иметь в своем составе много особей, соответствующих очень небольшому множеству самых нагруженных путей, но не обеспечивающих полного покрытия кода. Однако представляется целесообразным получение полного тестового набора в рамках одной популяции особей, являющейся результатом однократного запуска ГА. Данная задача может быть решена введением в функцию приспособленности компоненты, отвечающей за разнообразие популяции (т.е. особь является более приспособленной не только в случае прохода по более сложному пути, но и в случае наибольшего отличия пути, инициированного данной особью, от путей остальных особей в популяции).

Для вычисления j-го коэффициента схожести $sim_j(x_{i_1}, x_{i_2})$ двух хромосом x_{i_1} и x_{i_2}, сравниваем, покрывается или нет j-й узел графа потоков управления обоими путями, инициированными этими тестовыми наборами

$$sim_j(x_{i_1}, x_{i_2}) = \overline{g_j(x_{i_1}) \oplus g_j(x_{i_2})}, j = \overline{1, n},$$

где $g_j(x_i)$ – индикатор покрытия, а n – количество операторов.

Чем больше совпадающих битов между двумя путями, тем больше значение сходства между хромосомами x_{i_1} и x_{i_2}. Следующая формула учитывает веса соответствующих операторов (узлов графа):

$$sim(x_{i_1}, x_{i_2}) = \sum_{j=1}^{n} w_j \cdot sim_j(x_{i_1}, x_{i_2}),$$

где w_j – вес оператора j.

Значение сходства между хромосомой x_i и остальными хромосомами в популяции можно выразить

$$f_{sim}(x_i) = \frac{1}{(m-1)} \sum_{\substack{s=1 \\ s \neq i}}^{m} sim(x_s, x_i), \quad (3)$$

где m – размер популяции. Теперь можно определить среднее значение сходства особей во всей популяции

$$\overline{f_{sim}} = \frac{1}{m} \sum_{i=1}^{m} f_{sim}(x_i).$$

и далее сформулировать аддитивную компоненту функции приспособленности, ответственную за разнообразие путей в популяции:

$$F_2(x_i) = \left| \overline{f_{sim}} - f_{sim}(x_i) \right|. \quad (4)$$

В итоге, результирующая функция приспособленности для хромосомы x_i с учётом схожести вычисляется по формуле:

$$F(x_i) = F_1(x_i) + k \cdot F_2(x_i), \quad (5)$$

где компоненты $F_1(x_i)$ и $F_2(x_i)$ определяются формулами (2) и (4). Соответственно, первая компонента $F_1(x_i)$ определяет сложность пути, инициированного хромосомой x_i, а вторая компонента $F_2(x_i)$ – удалённость этого пути от всех остальных путей в популяции. Параметр k определяет соотношение между компонентами и определяется эмпирически для каждой тестируемой программы.

Экспериментальное исследование функции (5) показывает, что она не обеспечивает достаточного разнообразия хромосом в популяции из-за наличия циклически повторяющихся хромосом. Для исключения данного «эффекта раскачивания» предлагается использовать значение $ind(x_1, \dots, x_i)$, которое определяется числом хромосом из множества $\{x_1, \dots, x_{i-1}\}$, неразличимых с хромосомой x_i (неразличимыми называются хромосомы, инициирующие прохождение по одному и тому же пути графа потоков управления):

$$\tilde{F}(x_i) = F_1(x_i) + \frac{k}{1 + ind(x_1, \dots, x_i)} \cdot F_2(x_i). \quad (6)$$

Действительно, начальное значение $ind(x_1) = 0$, так как множество, в котором выявляются неразличимые хромосомы, на первом шаге пусто. На каждом следующем шаге значение $ind(x_1, \dots, x_i)$ может либо увеличиваться на 1, если следующая хромосома неразличима с одной из предшествующих, либо сохранять прежнее значение, если следующая хромосома уникальна. Это позволит хромосомам, проходящим по разным путям, более равномерно распределиться по популяции в целом.

Важным фактором, влияющим на качество генерации тестовых данных, является значение константы k. На рисунке 2 показана зависимость покрытия кода SUT2, представленного в диссертации, от параметра k для различного размера популяции. Для тестируемой программы наибольшее среднее значение покрытия достигается при $k = 10$.

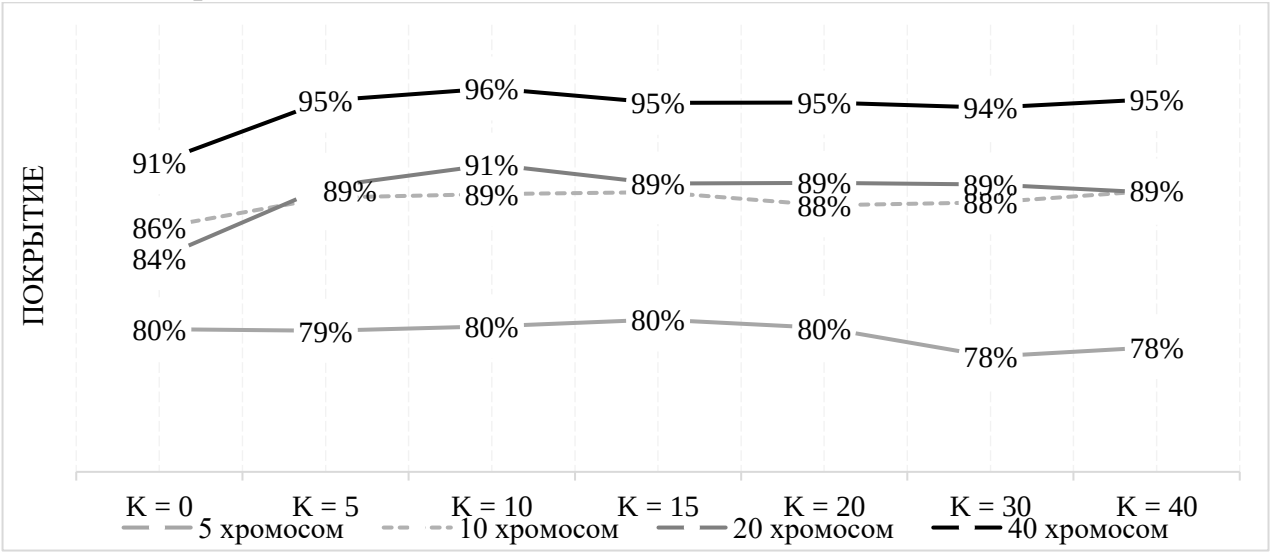

Рисунок 2 – Достигнутое покрытие кода SUT2 в зависимости от константы k

Проведенный анализ позволил выявить существенное влияние значения параметра k на результат покрытия тестируемого кода. Поэтому в диссертации была предложена вторая модификация функции приспособленности, состоящая только из одной компоненты F_1, но, позволяющая внести разнообразие в итоговую популяцию, обеспечивая достаточно высокий процент покрытия кода даже без использования аддитивной компоненты F_2.

В основе модификации алгоритма лежит идея «феромонов», заимствованная из муравьиного алгоритма. Применительно к задаче увеличения покрытия путей, инициируемых наборами входных тестовых данных, идея феромонов приводит к целесообразности динамического (от поколения к поколению) увеличения или уменьшения весов операторов $w_j, j = \overline{1,n}$, в зависимости от количества особей, прошедших ранее (в предшествующих поколениях) по соответствующим узлам графа. Динамическое изменение весов операторов можно представить в виде

$$\widetilde{w}_j^{(q)} = Ph_j^{(q)} w_j,\ j = \overline{1,n};\ q = \overline{1,Q}, \qquad (7)$$

где $\widetilde{w}_j^{(q)}$ – вес, присваиваемый оператору j в поколении q, $Ph_j^{(q)}$ – мультипликатор веса оператора j в поколении q $(0 \le Ph_j^{(q)} \le 1)$, Q – число поколений (итераций ГА). С учетом зависимости (7) динамический вариант компоненты F_1 функции приспособленности будет иметь вид:

$$F_1^{(q)}(x_i) = \sum_{j=1}^{n} \widetilde{w}_j^{(q)} g_j(x_i) = \sum_{j=1}^{n} Ph_j^{(q)} w_j g_j(x_i);\ q = \overline{1,Q}. \quad (8)$$

В диссертации предлагается и исследуется несколько подходов к реализации динамического изменения мультипликатора $Ph_j^{(q)}$, но во всех случаях его величина изменяется от 0 до 1 и обратно, в зависимости от покрытия или отсутствия покрытия оператора j в предшествующем поколении. В одних подходах величина изменения зависит от пропорции покрытия оператора j особями популяции, в других учитывается факт покрытия и количество поколений Q, при достижении которых предполагается получить максимальный процент покрытия. Один из наиболее успешных подходов (метод *Count*-) использует следующую формулу:

$$Ph_j^{(q)} = \begin{cases} 1, \text{если } q = 1, \\ Ph_j^{(q-1)} \left(1 - \widetilde{m}_j^{(q-1)}/m\right), \text{ если } \widetilde{m}_j^{(q-1)} \neq 0, \\ 1, \text{ если } \widetilde{m}_j^{(q-1)} = 0, \end{cases} \quad (9)$$

где $\widetilde{m}_j^{(q-1)}$ – число хромосом в популяции, состоящей из m особей, покрывших оператор j в поколении $(q-1)$.

На рисунке 3 проведены результаты сравнительного анализа покрытия тестируемой программы методом *Count*- с модификацией в виде соотношения (8) – (9) и без модификации при разных значениях параметра k в формуле (6). Используются следующие параметры генетического алгоритма – количество поколений $Q = 50$, размер популяции $m = 25$.

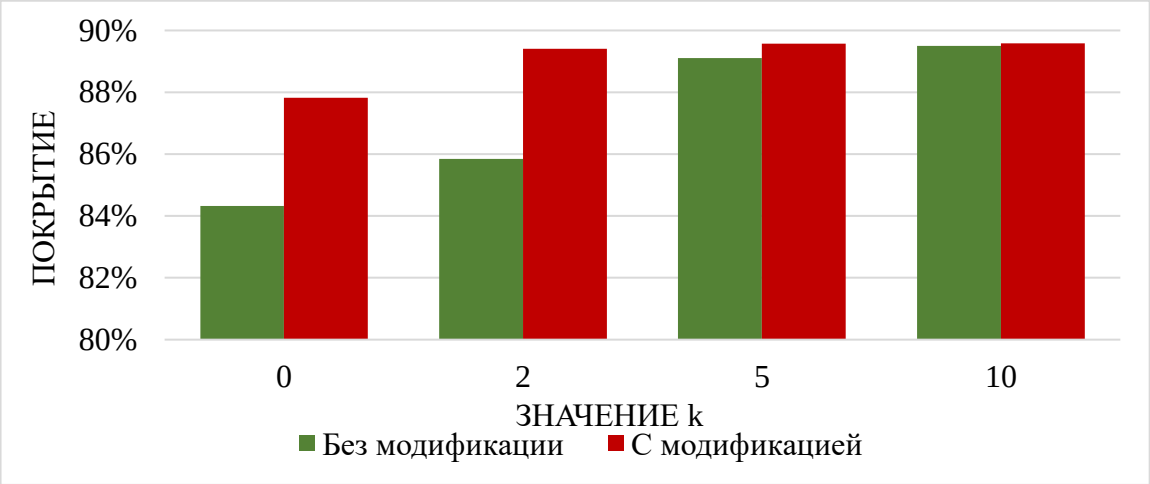

Рисунок 3 – Сравнительный анализ покрытия для модификации с динамическим изменением весов функции приспособленности и без нее

Из рисунка видно, что использование модификации позволяет не только увеличить покрытие кода, но и устранить необходимость определять значение k для каждой отдельной тестируемой программы.

В **четвёртой главе** рассматривается программная реализация предложенного алгоритма. Приложение позволяет генерировать наборы тестовых данных с использованием предложенных в диссертации модификаций функции приспособленности.

Структура программы определяется следующим образом (Рисунок 4):

– *Тестируемый код*. На вход разработанного приложения подается тестируемый код, для которого необходимо сгенерировать тестовые наборы x. Код в виде текста вводится в соответствующее поле интерфейса, что позволяет взаимодействовать с ним в процессе работы приложения.

– *Интерфейс*. Обеспечивает взаимодействие между приложением и пользователем. В модуле интерфейса производится передача тестируемого кода в реализацию, и обеспечивается вывод выходной информации.

– *Реализация*. В модуле реализации производится обработка и анализ тестируемого кода и генерация наборов тестовых данных.

– *Сгенерированные данные*. Выходной информацией для приложения являются сгенерированные тестовые наборы x. В качестве дополнительной информации выводится значение вектора индикаторов покрытия $g(x)$. При необходимости можно также обеспечить вывод всех операторов с указанием соответствующего этому оператору значения индикатора.

Вывод сгенерированных тестовых наборов представляется в зависимости от выбранной цели генерации данных. Если целью является покрытие одного пути, то итоговым решением является первая хромосома из всей популяции, если поставлена цель полного покрытия кода, то решением будет пул элитных хромосом (Unique chromosomes) (Рисунок 5).

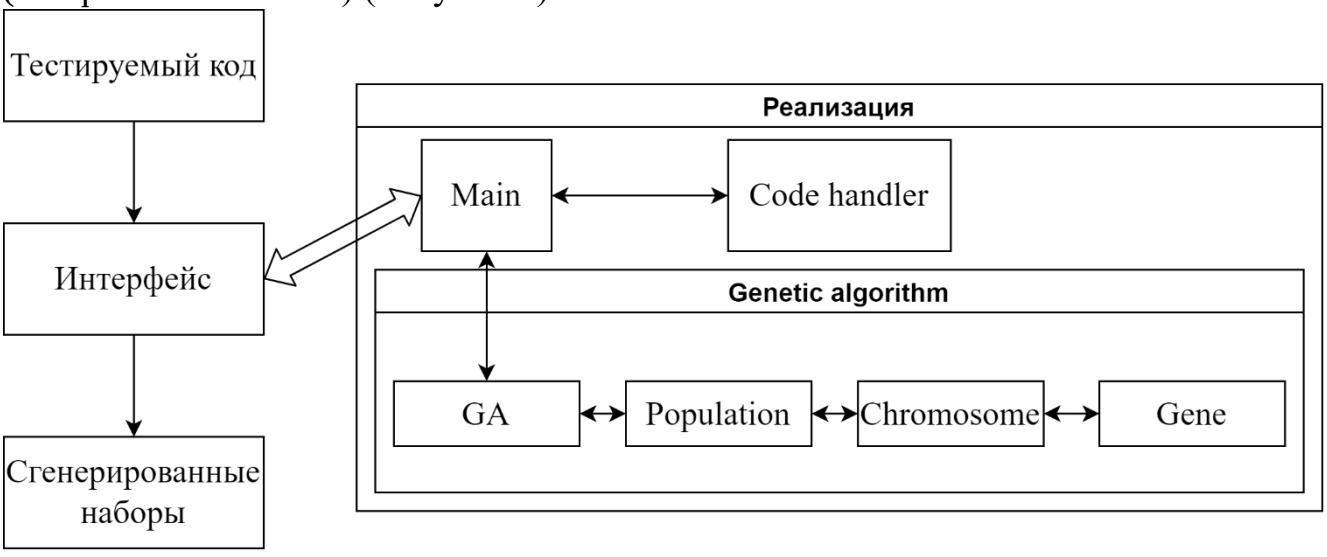

Рисунок 4 – Структурная схема разработанного приложения

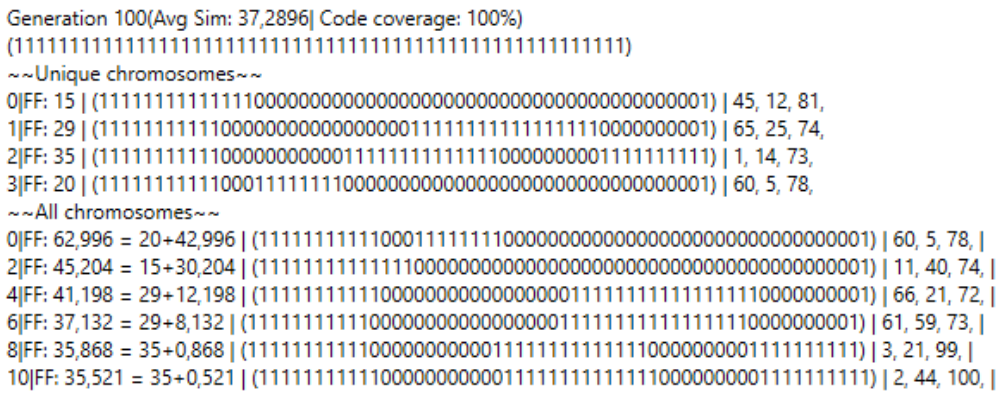

Рисунок 5 – Вывод сгенерированных наборов данных

Таким образом, программа позволяет генерировать множество входных наборов тестовых данных на основе введенного тестового кода, настраивать основные параметры генетического алгоритма для каждого тестового кода и проводить промежуточный анализ достижимых путей кода на основе критерия покрытия операторов.

В **заключении** подведены итоги проведённого исследования.

Приложения содержат листинги анализируемых программ SUT1 и SUT2, листинг разработанного приложения, свидетельства о регистрации программы для ЭВМ, акты внедрения в учебный процесс НГТУ и производственный процесс компании ООО «Дежавю».

ОСНОВНЫЕ РЕЗУЛЬТАТЫ И ВЫВОДЫ

Проведенные в диссертационной работе исследования образуют теоретическую и практическую основу для разработки систем интеллектуальной поддержки анализа и тестирования программ.

Основными результатами проведённого исследования являются:

1. Сформулирована общая постановка задачи генерации тестовых данных для ее решения с помощью генетического алгоритма, включая математический вид функции приспособленности, определяющей качество тестового набора. Для вычисления весов функции приспособленности предложены 4 метрики оценки сложности кода, 2 из которых выбраны для дальнейших исследований.

2. Предложены два подхода к генерации множества тестовых наборов, обеспечивающих максимальное покрытие кода, на основе модификации функции приспособленности ГА. Первая модификация заключается во введении дополнительной аддитивной компоненты, отвечающей за разнообразие популяции. Во второй модификации функции приспособленности разнообразие достигается за счет динамического изменения весов операторов в зависимости от степени их покрытия тестовыми наборами, полученными на предыдущей итерации.

3. Разработано семейство алгоритмов генерации тестовых данных на основе предложенных формальных эволюционных постановок задач и различных

вариантов функции приспособленности ГА, обеспечивающих максимально возможное покрытие исследуемых программ.

4. Разработано программное приложение, реализующее предложенные алгоритмы генерации тестовых данных. На вход приложения подается текст анализируемой программы, на выходе получается множество сгенерированных тестовых данных с дополнительной информацией о покрытии операторов, значениях функции приспособленности ГА, позволяющей провести анализ качества тестируемого ПО.

5. С использованием разработанного приложения проведен анализ эффективности построенных алгоритмов, найдены значения параметров ГА для повышения скорости сходимости, предложены эвристические решения, повышающие качество тестовых наборов за счет увеличения степени покрытия ими тестируемого кода.

ОСНОВНЫЕ ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

Статьи в журналах, рекомендованных ВАК РФ:

1. Сердюков, К.Е. Исследование методов определения сложности кода при формировании наборов входных тестовых данных / К.Е. Сердюков, Т.В. Авдеенко // Южно-Сибирский научный вестник. – 2019. – № 4-2 (28). – С. 65–69.

2. Сердюков, К.Е. Исследование метрик оценки кода при формировании наборов данных с использованием генетического алгоритма / К. Е. Сердюков, Т. В. Авдеенко // Известия Тульского государственного университета. Технические науки. – 2019. – Вып. 10. – С. 430–442.

Публикации в изданиях, индексируемых в базах данных WoS и Scopus:

3. Avdeenko, T., Serdyukov, K. Automated Test Data Generation Based on a Genetic Algorithm with Maximum Code Coverage and Population Diversity [Electronic resource] // Applied Sciences. – 2021. – Vol.11. – Art. 4673. – 17 p.

4. Avdeenko, T. V. Genetic algorithm fitness function formulation for test data generation with maximum statement coverage / T. V. Avdeenko, K. E. Serdyukov. // Lecture Notes in Computer Science. – 2021. – Vol. 12689: 12 International Conference on Advances in Swarm Intelligence (ICSI 2021), China, Qingdao. – P. 379-389.

5. Avdeenko, T.V. Formulation and research of new fitness function in the genetic algorithm for maximum code coverage [Electronic resource] / T. V. Avdeenko, K. E. Serdyukov, Z. B. Tsydenov // Procedia Computer Science. – 2021. – Vol. 186. – P. 713-720.

6. Serdyukov, K.E. Development and Research of the Test Data Generation Approach Modifications / K.E. Serdyukov, T.V. Avdeenko. // The 7 international conference on information technology and nanotechnology (ITNT-2021) – Samara: IEEE, 2021. – 6 p.

7. Serdyukov, K. E. Researching of methods for assessing the complexity of program code when generating input test data [Electronic resource] / K. E. Serdyukov, T. V. Avdeenko // CEUR Workshop Proceedings. – 2020. – Vol. 2667: Information Technology and Nanotechnology, ITNT-DS 2020, Samara. – P. 299-304.

8. Serdyukov, K. The Study of the Sequential Inclusion of Paths in the Analysis of Program Code f or the Task of Selecting Input Test Data [Electronic resource] / K. Serdyukov, T. Avdeenko // CEUR Workshop Proceedings. – 2020. – Vol. 2790: Management in Data Intensive Domains (DAMDID/RCDL 2020), Voronezh, 13-16 Oct. 2020. - P. 79-88.

9. Serdyukov, K. E. Using genetic algorithm for generating optimal data sets to automatic testing the program code / K.E. Serdyukov, T.V. Avdeenko // CEUR Workshop Proceedings. – 2019. – Vol. 2416: Information Technology and Nanotechnology: Data Science. – P. 173–182.

10. Serdyukov, K. E. Automatic data generation for software testing based on the genetic algorithm / K. E. Serdyukov, T. V. Avdeenko // Actual problems of electronic instrument engineering (APEIE–2018) – Новосибирск: Изд-во НГТУ, 2018. – Т. 1, ч. 4. – С. 535-540.

11. Serdyukov, K. E. Method of application of the genetic algorithm for automatic generation of test data / K. E. Serdyukov, T. V. Avdeenko // CEUR Workshop Proceedings. – 2018. – Vol. 2212: Data Science. Information Technology and Nanotechnology, Samara. – P. 424-430.

12. Serdyukov, K. E. Investigation of the genetic algorithm possibilities for retrieving relevant cases from big data in the decision support systems / K. E. Serdyukov, T. V. Avdeenko // CEUR Workshop Proceedings. – 2017. – Vol.1903: Data Science. Information Technology and Nanotechnology, DS-ITNT. – P. 36-41.

Публикации в других научных изданиях:

13. Сердюков, К. Е. Разработка и исследование модификаций подхода генерации тестовых данных / К. Е. Сердюков, Т. В. Авдеенко. // Информационные технологии и нанотехнологии (ИТНТ-2021) - Самара, 2021 – № 33343 – 3 с.

14. Сердюков, К. Е. Исследование способов оценки сложности программного кода при генерации входных тестовых данных / К. Е. Сердюков, Т. В. Авдеенко // Информационные технологии и нанотехнологии (ИТНТ-2020). Т. 4: Науки о данных. – Самара: Изд-во Самар. нац. исслед. ун-та, 2020. – С. 662–671.

15. Сердюков, К. Е. Исследование возможностей применения генетического алгоритма для формирования наборов данных и первичной отладки программного кода / К. Е. Сердюков, Т. В. Авдеенко // Информационные технологии и нанотехнологии (ИТНТ-2019). – Самара: Новая техника, 2019. – С. 685–694.

16. Сердюков, К. Е. Исследование методов определения сложности кода при формировании наборов входных данных / К. Е. Сердюков, Т. В. Авдеенко //

Измерения, автоматизация и моделирование в промышленности и научных исследованиях (ИАМП–2019) – Бийск: Изд-во АлтГТУ, 2019. – С. 352–355.

17. Сердюков, К. Е. Применение генетического алгоритма для генерации входных данных при тестировании программного кода / К. Е. Сердюков, Т. В. Авдеенко // Наукоемкое программное обеспечение. – Новосибирск: Изд-во НГУ, 2019. – С. 130–137.

18. Сердюков, К. Е. Исследование метода применения генетического алгоритма для формирования набора данных при тестировании программного обеспечения / К. Е. Сердюков, Т. В. Авдеенко // Измерения, автоматизация и моделирование в промышленности и научных исследованиях. – Барнаул: Изд-во АлтГТУ, 2018. – С. 528–531.

19. Сердюков, К. Е. Исследование возможностей генетического алгоритма для извлечения релевантных прецедентов в системах поддержки принятия решений / К. Е. Сердюков, Т. В. Авдеенко, Е. С. Макарова // Информационные технологии и нанотехнологии. – Самара, 2017. – С. 1872–1878.

20. Сердюков, К. Е. О возможностях генетического алгоритма для разработки вариантов тестов программного обеспечения / К. Е. Сердюков; науч. рук. Т. В. Авдеенко // Наука. Технологии. Инновации. – Новосибирск: Изд-во НГТУ, 2017. – Ч. 2. – С. 186–190.

21. Сердюков, К. Е. Гибридизация прецедентного подхода к представлению знаний и генетических алгоритмов в экономике / К. Е. Сердюков; науч. рук. Т. В. Авдеенко // Наука. Технологии. Инновации. – Новосибирск, 2016.– С. 61–63.

Свидетельства о государственной регистрации программы для ЭВМ:

22. Программа генерации тестовых данных на основе модификации генетического алгоритма с использованием методов оценки сложности кода / К.Е. Сердюков, Т.В. Авдеенко (Россия). – №2020662849; заявл. 28.10.2020; зарег. в реестре программ для ЭВМ 28.10.2020 г.

23. Программа генерации тестовых данных на основе расширенной функции приспособленности, учитывающей меру сложности кода и разнообразие особей в популяции / К.Е. Сердюков, Т.В. Авдеенко (Россия). – №2020665707; заявл. 07.12.2020; зарег. в реестре программ для ЭВМ 07.12.2020 г.

Отпечатано в типографии
Новосибирского государственного технического университета
630073, г. Новосибирск, пр. К. Маркса, 20, тел. (383) 346-08-57.
Формат 60×84/16. Усл. печ. л. 1.5. Тираж 100 экз.
Подписано в печать 07.07.2022. Заказ №1185.