На правах рукописи

Буздалов Максим Викторович

Генерация тестов для определения неэффективных решений олимпиадных задач по программированию с использованием эволюционных алгоритмов

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

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Санкт-Петербург — 2014

Работа выполнена в Санкт-Петербургском национальном исследовательском университете информационных технологий, механики и оптики.

Научный руководитель: доктор технических наук, профессор
Шалыто Анатолий Абрамович

Официальные оппоненты: **Баранов Сергей Николаевич**,
доктор физико-математических наук, профессор, заведующий лабораторией теории и технологии программирования Санкт-Петербургского института информатики и автоматизации Российской академии наук

Крючкова Елена Николаевна,
кандидат физико-математических наук, доцент, профессор кафедры прикладной математики Алтайского государственного технического университета им. И.И. Ползунова

Ведущая организация: Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Санкт-Петербургский государственный университет»

Защита состоится 22 декабря 2014 г. в 15 часов 30 минут на заседании диссертационного совета Д 212.227.06 при Санкт-Петербургском национальном исследовательском университете информационных технологий, механики и оптики по адресу: 197101, г. Санкт-Петербург, Кронверкский просп., д. 49., конференц-зал центра интернет-образования.

С диссертацией можно ознакомиться в библиотеке Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики по адресу: 197101, г. Санкт-Петербург, Кронверкский просп., д. 49. и на сайте `fppo.ifmo.ru`.

Автореферат разослан _____ 2014 года.

Ученый секретарь
диссертационного совета
Д 212.227.06, к.ф.-м.н. Лобанов Игорь Сергеевич

Общая характеристика работы

Актуальность проблемы. В мире проводится большое число олимпиад и соревнований по программированию для школьников, студентов и всех желающих. Олимпиады помогают выявить школьников и студентов, одаренных в области информатики и программирования, а IT-компаниям, в свою очередь, сформировать перечень кандидатов, среди которых они смогут найти способных и талантливых сотрудников.

Существуют работы, посвященные олимпиадам и соревнованиям по программированию, предназначенные главным образом для участников соревнований и для их тренеров. Методические аспекты олимпиад обсуждаются на некоторых конференциях, таких как, например, *ISSEP* (*Informatics in Schools: Situation, Evolution and Perspectives*). Журнал *Olympiads in Informatics*, публикуемый Институтом математики и информатики Вильнюсского университета, всецело посвящен различным аспектам олимпиад по программированию. Однако, в литературе было найдено крайне мало публикаций на тему того, как следует готовить задачи для олимпиад по программированию. Информация на эту тему может быть найдена в виде методических рекомендаций на сайте олимпиад или разослана непосредственно будущим авторам задач, как это делают организаторы системы соревнований *TopCoder*.

Решение олимпиадной задачи по программированию — это, как правило, программа, которая читает входные данные, решает поставленную задачу и выводит ответ. При этом решение должно уложиться в ограничения на время работы и на объем используемой памяти, указанные в условии задачи.

Тест для олимпиадной задачи по программированию — это входные данные для задачи. В большинстве случаев тест хранится в виде текстового файла, удовлетворяющего определенному формату. Для одной задачи может быть один или несколько тестов. Тесты составляются жюри до начала соревнования и, как правило, неизвестны участникам. Правильность ответа на каждом тесте проверяется программой, написанной жюри.

В данной работе предполагается, что дано одно или несколько решений рассматриваемой олимпиадной задачи по программированию, причем каждое из них может реализовывать тот или иной алгоритм. Назовем *корректным* решение, которое на любом тесте способно найти и вывести правильный ответ, а затем завершиться, уложившись в ограничения по времени и памяти. Определим также *решение, корректное по тестам*, которое может быть некорректным, но на всех тестах *из имеющегося набора*, как и корректное решение, находит правильный ответ, укладываясь в указанные ограничения. Решение называется *неэффективным*, если существует хотя бы один тест, на котором оно не укладывается в указанные ограничения (соответственно, любое корректное решение является *эффективным*). Можно также определить *неверное* решение, которое хотя бы для одного теста генерирует неверный ответ или некорректно завершается. В данной работе рассматривается построение тестов только против неэффективных решений.

Будем считать, что тест A *сложнее* теста B для некоторого решения по критерию *времени работы*, если это решение работает на тесте A дольше, чем на тесте B. Аналогично, тест A *сложнее* теста B для некоторого решения по критерию *используемой памяти*, если это решение использует на тесте A больше памяти, чем на тесте B. Назовем набор тестов *слабым*, если существует достаточно много решений, которые выдают правильные ответы, укладываясь в ограничения по времени и памяти, на всех тестах этого набора, но при этом существуют тесты, удовлетворяющие условию задачи, на которых указанные решения не работают.

Иметь слабый набор тестов для олимпиадной задачи по программированию крайне нежелательно. Действительно, участники олимпиад не знают, какое решение жюри считает корректным и какие тесты присутствуют в тестовом наборе. Например, некорректно указывать в условии, что некоторое число N не превосходит 10000, а генерировать только такие тесты, в которых $N \leq 1000$. Это даст меньше шансов опытным участникам, так как они потратят лишнее время на реализацию эффективного решения, и даст преимущество неопытным участникам, которые будут пытаться наудачу посылать решения, возможно, неэффективные.

Одной из важнейших проблем при разработке олимпиадных задач по программированию является генерация тестов, на которых неэффективные решения превышали бы ограничения на время и память. Проблемы, подобные указанной, успешно решаются для определенных классов программ с использованием методов поисковой инженерии программного обеспечения, в частности, различных метаэвристик, включая эволюционные алгоритмы. Однако, существующие методы, основанные на таких алгоритмах, имеют следующие недостатки:

- большинство методов направлены на максимизацию покрытия кода тестами, а не на максимизацию времени работы кода;
- методы генерации тестов, направленные на максимизацию времени работы кода, разработаны для встраиваемых систем;
- методы, основанные на переборе возможных путей выполнения, применимы только для кода без циклов и рекурсии, что не выполняется практически ни для одного решения олимпиадной задачи.

Поэтому проблема генерации тестов для определения неэффективных решений олимпиадных задач по программированию с использованием эволюционных алгоритмов является **актуальной**.

Целью работы является разработка методов и технологии генерации тестов для определения неэффективных решений олимпиадных задач по программированию с использованием эволюционных алгоритмов.

Основные задачи диссертационной работы состоят в следующем:

1. Показать целесообразность применения эволюционных алгоритмов для генерации тестов, определяющих неэффективные решения олимпиадных задач по программированию, на примере задач дискретной матема-

тики, алгоритмы решения которых могут работать существенно быстрее, чем ожидается согласно верхней оценке на время работы.
2. Разработать алгоритм выбора из нескольких функций приспособленности функции, оптимальной по времени решения задачи оптимизации.
3. Разработать и внедрить технологию генерации тестов на основе эволюционных алгоритмов для определения неэффективных решений олимпиадных задач по программированию на примере задач дискретной математики указанного класса.

Научная новизна. В работе получены следующие новые научные результаты, которые выносятся на защиту:

1. Метод генерации тестов для алгоритмов решения NP-трудных задач на основе генетического алгоритма (на примере задачи о рюкзаке). Показано, что разработанный метод с большим уровнем статистической значимости генерирует более сложные тесты, чем наилучшие методы случайной генерации тестов из известных на сегодняшний день.
2. Метод генерации тестов для алгоритмов решения графовых задач на основе генетического алгоритма (на примере задачи о поиске максимального потока). Показано, что разработанный метод генерирует более сложные тесты, чем метод случайной генерации тестов, а также большинство известных методов генерации тестов для указанной задачи. Для некоторых алгоритмов, например для алгоритма Диница, тесты, построенные генетическим алгоритмом, превосходят тесты, построенные другими методами, в несколько раз по времени работы.
3. Алгоритм выбора из нескольких функций приспособленности функции, оптимальной по времени решения задачи оптимизации. Показано, что данный алгоритм выбирает функцию приспособленности на каждой итерации эволюционного алгоритма, а также организует его перезапуск, таким образом, что математическое ожидание числа итераций эволюционного алгоритма, необходимых для решения задачи оптимизации, меньше, чем $4K \cdot \min_G T_G$, где K — число функций приспособленности, а T_G — математическое ожидание числа итераций эволюционного алгоритма при использовании функции приспособленности G.

Методы исследований. В работе используются методы дискретной математики, эволюционных вычислений, теории вероятностей и математической статистики.

Достоверность научных положений, выводов и практических рекомендаций, полученных в диссертации, подтверждается корректным обоснованием постановок задач, точной формулировкой критериев, результатами экспериментов по использованию предложенных в диссертации методов и их статистическим анализом.

Практическое значение работы состоит в том, что предложена технология, позволяющая генерировать тесты для определения неэффективных решений олимпиадных задач по программированию рассмотренных классов в

процессе разработки или доработки этих задач. Технология уменьшает влияние человеческого фактора на качество набора тестов в силу того, что чисто эвристический подход к построению тестов заменен использованием стандартной схемы, характерной для решения задач с помощью эволюционных алгоритмов.

Внедрение результатов работы. Дополнительные тесты, полученные с помощью методов, предложенных в диссертации, были внедрены в архив задач с проверяющей системой *Timus Online Judge* (функционирующий на базе Уральского федерального университета имени первого президента России Б. Н. Ельцина, г. Екатеринбург), используемый для подготовки к олимпиадам по программированию (задачи «Ships. Version 2» и «Work for Robots»).

Результаты диссертации были использованы в учебном процессе кафедры «Компьютерные технологии» Университета ИТМО при руководстве шестью бакалаврскими работами и одной магистерской диссертацией.

Апробация результатов работы. Основные результаты работы докладывались на следующих конференциях:

- Третья Всероссийская конференция «Нечеткие системы и мягкие вычисления» (2009, Волгоград);
- Всероссийская научная конференция по проблемам информатики СПИСОК (2011, 2012, 2013, 2014, Матмех СПбГУ);
- Genetic and Evolutionary Computation Conference (2011, Дублин, 2013, Амстердам);
- International Conference on Machine Learning and Applications (2012, Бока-Ратон, США, 2013, Майами, 2014, Детройт);
- IEEE Congress on Evolutionary Computation (2013, Канкун, Мексика);
- International Symposium on Search-Based Software Engineering (2013, Санкт-Петербург);
- International Conference on Soft Computing MENDEL (2014, Брно, Чехия);
- International Conference on Bio-Inspired Computing: Theories and Applications (2014, Вухан, Китай).

Личный вклад автора. Решение задач диссертации, разработанные методы, алгоритмы, технология и их реализация принадлежат лично автору.

Публикации. Основные результаты по теме диссертации изложены в 17 публикациях, две из которых изданы в журналах, рекомендованных ВАК, девять — в изданиях, индексируемых в международных базах цитирования *Web of Science* и *Scopus*. В работах, выполненных в соавторстве, авторство принадлежит соавторам в равных долях.

Свидетельства о регистрации программы для ЭВМ. Автором по теме диссертации получено три свидетельства о регистрации программы для ЭВМ.

Участие в научно-исследовательских работах. Результаты диссертации были использованы при выполнении **под руководством автора** научно-исследовательской работы «Разработка методов автоматической генерации тестов на основе эволюционных алгоритмов» по Государственному контракту

№ 14.740.11.1430, 2011–2012 гг. Автор работы являлся победителем конкурса грантов Санкт-Петербурга для студентов, аспирантов, молодых ученых, молодых кандидатов наук 2011 г., тема проекта — «Генерация тестов для олимпиадных задач по теории графов с использованием эволюционных алгоритмов».

Объем и структура работы. Диссертация состоит из введения, семи глав, заключения и одного приложения. Объем диссертации составляет 204 страницы с девятью рисунками, 13 таблицами и пятью листингами. Список литературы содержит 168 наименований.

Содержание работы

В **первой главе** приводится обзор работ, посвященных олимпиадам по программированию, эволюционным алгоритмам, поисковой инженерии программного обеспечения. В частности, в обзоре описаны известные подходы поисковой инженерии для генерации тестов, максимизирующих время работы программы (worst-case execution time test generation), а также современные программные средства, основанные на динамическом символьном выполнении кода.

При этом показано, что первые из них плохо справляются с задачей генерации тестов для олимпиадных задач по программированию — функцией приспособленности в них является непосредственно время работы программы, что хорошо подходит для встраиваемых систем, но плохо применимо для программ, работающих под управлением многозадачных операционных систем. Динамическое символьное выполнение при максимизации времени работы неизбежно столкнется с необходимостью решать задачи удовлетворения ограничений с размером, линейно растущим со временем работы, при этом время, требуемое для решения таких задач, будет расти экспоненциально. Таким образом, данный подход также неприменим для максимизации времени работы решений олимпиадных задач по программированию.

На основе результатов обзора формулируются задачи, решаемые в диссертации.

Во **второй главе** для NP-трудных задач на примере задачи о рюкзаке показывается целесообразность применения эволюционных алгоритмов для генерации тестов, определяющих неэффективные решения олимпиадных задач по программированию. Задача о рюкзаке выбрана, так как она известна как «самая простая NP-трудная задача» — многие решения этой задачи быстро работают на случайных тестах.

Рассматривается пять известных алгоритмов решения этой задачи, обладающих различной эффективностью. Три из них относятся к *переборным* алгоритмам, при этом один из этих алгоритмов разработан и реализован автором в качестве простейшего алгоритма перебора с отсечениями, в то время как два других алгоритма, предложенные Д. Писингером, могут работать на случайных тестах практически за линейное время. Два других алгоритма, так-

же предложенные Д. Писингером, решают задачу о рюкзаке с использованием *динамического программирования*.

Предложен генетический алгоритм, позволяющий генерировать сложные тесты для таких алгоритмов. Приводятся результаты экспериментов, сравнивающих предложенный генетический алгоритм с известными методами генерации тестов (генерация случайных тестов, генерация сильно коррелированных тестов, генерация тестов вида «сумма подмножеств» — вес каждого предмета равен его стоимости). Показано, что генетический алгоритм генерирует более сложные тесты, в отдельных случаях — до 10 раз. При анализе тестов, сгенерированных генетическим алгоритмом, обнаружен новый класс тестов, сложных для переборных решений — тесты с двумя типами предметов.

В **третьей главе** для графовых задач на примере задачи о поиске максимального потока показывается целесообразность применения эволюционных алгоритмов для генерации тестов, определяющих неэффективные решения олимпиадных задач по программированию. Задача о поиске максимального потока выбрана ввиду того, что она обладает следующим свойством: тесты, сгенерированные случайным образом, редко бывают сложными, а оценки на время работы алгоритмов решений этой задачи могут существенно переоценивать их время работы.

Рассматривается шесть известных алгоритмов решения задачи о поиске максимального потока: алгоритм Форда-Фалкерсона с масштабированием пропускной способности, алгоритм Эдмондса-Карпа, основанный на идее поиска кратчайших дополняющих путей, алгоритм Эдмондса-Карпа с масштабированием пропускной способности, алгоритм Диница, неоптимальная реализация алгоритма Диница (существует легко допустимая ошибка в реализации алгоритма Диница, при которой алгоритм работает правильно, но медленно), улучшенный алгоритм поиска кратчайших дополняющих путей. Предложен генетический алгоритм, позволяющий генерировать сложные тесты для указанных алгоритмов. Он сравнивается с известными генераторами тестов для алгоритмов решения задачи о максимальном потоке: генератор случайных графов, генератор случайных ациклических графов, генератор транзитных решеток, генератор случайных слоев, генератор Черкасского и Гольдберга, генераторы семейства «washington», тесты Заде.

Параметрами задачи о максимальном потоке является число вершин V, число ребер E и максимальная пропускная способность C_{max}. В качестве особи генетического алгоритма предложено использовать список ребер, в котором для каждого ребра задается номер начальной вершины, номер конечной вершины и пропускная способность ребра. Число ребер в списке фиксировано и равно E. При этом истоком всегда считается вершина с номером 1, а стоком — вершина с номером V. В экспериментальных исследованиях рассматривались как произвольные особи, так и особи, строящие *ациклические графы* — в них для каждого ребра номер начальной вершины всегда меньше номера конечной вершины.

Рассматривались два оператора скрещивания: одноточечное скрещивание и двухточечное скрещивание со сдвигом, а в качестве функции приспособленности — следующие величины: астрономическое время работы алгоритма, процессорное время работы алгоритма, число посещенных ребер, число посещенных вершин, число совершенных обходов в ширину или глубину, а также функции приспособленности, специфичные для некоторых алгоритмов.

По результатам экспериментальных исследований было установлено, что генерация тестов для *различных* алгоритмов поиска максимального потока наиболее эффективна при применении *одной и той же конфигурации* генетического алгоритма (использование особей, строящих ациклические графы, одноточечного скрещивания, а в качестве функции приспособленности — числа посещенных ребер). Для рассмотренных алгоритмов поиска максимального потока наилучшими оказались следующие тесты:

- для **алгоритма Диница** и его **неоптимальной реализации** — тест, полученный генетическим алгоритмом при генерации тестов для неоптимальной реализации алгоритма Диница. При этом значение приспособленности в случае алгоритма Диница составило 594 662, в то время как у лучшего негенетического теста оно равно 180 589.
- для **алгоритма Эдмондса-Карпа** — тест, полученный генетическим алгоритмом при генерации тестов для этого же алгоритма. При этом значение приспособленности составило 4 613 284, в то время как у лучшего из известных тестов — теста Заде — это значение равно 4 609 566.
- для **улучшенного алгоритма поиска кратчайших дополняющих путей** — тест, полученный генетическим алгоритмом при генерации тестов для этого же алгоритма. При этом значение приспособленности составило 820 538, в то время как у лучшего негенетического теста оно равно 441 289.
- для **алгоритма Эдмондса-Карпа с масштабированием** лучшим тестом оказался тест Заде.
- для **алгоритма Форда-Фалкерсона с масштабированием** лучшим тестом оказался случайный ациклический граф.

В **четвертой главе** описывается разработанный автором алгоритм выбора оптимальной функции приспособленности, который назван SARA (от англ. Switch-and-Restart Algorithm). Данный алгоритм направлен на решение проблемы выбора такой функции приспособленности из нескольких функций, которая приводит к нахождению хорошего теста за минимальное ожидаемое число итераций алгоритма оптимизации.

Пусть дана задача оптимизации, эволюционный алгоритм и K функций приспособленности, такие что математическое ожидание числа итераций эволюционного алгоритма, необходимых для решения указанной задачи оптимизации при использовании функции приспособленности G, составляет T_G. Разработанный алгоритм выбирает функцию приспособленности на каждой итерации эволюционного алгоритма, а также организует перезапуск этого ал-

горитма таким образом, что общее число итераций эволюционного алгоритма, необходимых для решения исходной задачи оптимизации, будет меньше $4K \cdot \min_G T_G$. При этом требуемый объем памяти равен соответствующей величине для используемого эволюционного алгоритма с добавлением лишь небольшой константы, что существенно меньше, чем при использовании параллельного запуска K эволюционных алгоритмов.

В диссертации разработанный алгоритм приведен в более общей формулировке, в которой он работает с несколькими итеративными алгоритмами оптимизации. В указанной формулировке теорема о свойствах этого алгоритма звучит следующим образом:

Теорема 1. *Пусть поставлена задача оптимизации и имеется K итеративных алгоритмов оптимизации, причем матожидание числа итераций, требуемых для решения задачи оптимизации алгоритмом G, равно T_G. Тогда матожидание общего числа итераций не больше $4K \cdot \min_G T_G$.*

Приведем схему доказательства этой теоремы.

1. Пусть K — число алгоритмов, $P_G(x)$ — вероятность того, что алгоритм G решает задачу оптимизации на итерации с номером x. Пусть A_i — размер i-ой мета-итерации, T — матожидание общего числа итераций для решения задачи оптимизации алгоритмом SARA. Тогда
$$T \le K \sum_{y=1}^{\infty} \left(A_y \prod_{i=1}^{y-1} \left(1 - \sum_{j=1}^{A_i} P_G(j) \right) \right).$$
2. Пусть K — число алгоритмов, T_G — матожидание числа итераций, необходимых алгоритму G для решения задачи оптимизации. Пусть $A_i = \lfloor \alpha^{i-1} \rfloor$ — размер i-ой мета-итерации, T — матожидание общего числа итераций для решения задачи оптимизации алгоритмом SARA. Тогда
$$T \le K \frac{\alpha^2}{\alpha - 1} T_G.$$
3. Для каждого значения $\alpha > 1$ и произвольного $\varepsilon > 0$ существует такое K, такая задача оптимизации и такой набор из K алгоритмов оптимизации, что
$$\frac{T}{\min_G T_G} > K \frac{\alpha^2}{\alpha - 1}(1 - \varepsilon).$$
4. Минимальное время работы достигается при $\alpha = 2$.

В **пятой главе** описывается технология генерации тестов для определения неэффективных решений олимпиадных задач по программированию на основе эволюционных алгоритмов. Предложенная технология (см. рисунок 1) состоит из следующих этапов:

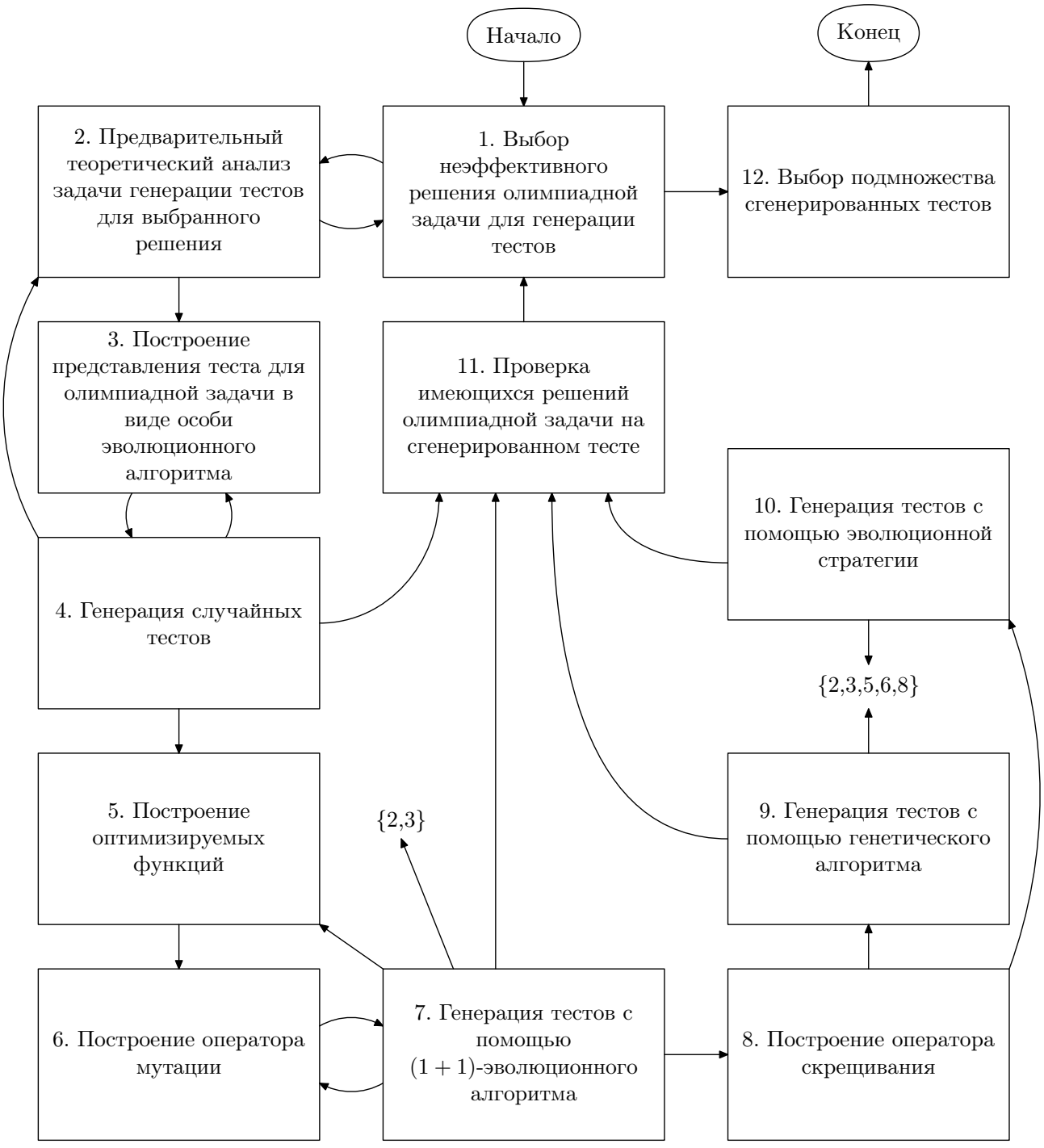

Рисунок 1 – Технология генерации тестов для определения неэффективных решений олимпиадных задач по программированию

1. **Выбор неэффективного решения олимпиадной задачи для генерации тестов**. На данном этапе выбирается решение олимпиадной задачи, для которого на следующих этапах будут генерироваться тесты. Если такое решение не найдено, производится переход к этапу 12.
2. **Предварительный теоретический анализ задачи генерации тестов для выбранного решения**. На данном этапе проводится предварительный анализ того, целесообразно ли генерировать тесты для рассматриваемого решения с использованием эволюционных алгоритмов. Если верх-

няя оценка на время работы решения или на объем используемой памяти показывают, что решение может превышать соответствующие ограничения, но при этом тест, демонстрирующий превышение ограничений, не был найден ранее, то генерация тестов целесообразна.

3. **Построение представления теста для олимпиадной задачи в виде особи эволюционного алгоритма**. На данном этапе строится такое представление теста в виде особи эволюционного алгоритма, которое позволяет с большой вероятностью получать корректные тесты, при этом желательно, чтобы эти тесты было как можно более сложными.
4. **Генерация случайных тестов**. На данном этапе с целью проверить эффективность представления теста генерируется достаточно большое число случайных тестов с использованием разработанного на предыдущем этапе представления теста в виде особи. Если тест, на котором рассматриваемое решение признается неэффективным, найден на этом этапе, тогда производится переход к этапу 11.
5. **Построение оптимизируемых функций**. На данном этапе строятся функции, отражающие сложность теста для рассматриваемого решения, которые будут оптимизироваться эволюционными алгоритмами на дальнейших этапах. Если удалось построить одну функцию, она и будет оптимизироваться, в противном случае для выбора из всех построенных функций рекомендуется использовать алгоритм SARA.
6. **Построение оператора мутации**. На данном этапе строится оператор мутации, который будет применяться на дальнейших этапах в эволюционных алгоритмах.
7. **Генерация тестов с помощью $(1+1)$-эволюционного алгоритма**. На данном этапе делается попытка сгенерировать хороший тест с использованием простейшего эволюционного алгоритма. Если такой тест найден, производится переход к этапу 11.
8. **Построение оператора скрещивания**. На данном этапе рассматривается возможность построения оператора скрещивания. Если такой оператор построить возможно, производится переход к этапу 9, иначе — к этапу 10.
9. **Генерация тестов с помощью генетического алгоритма**. На данном этапе производится генерация тестов с использованием генетического алгоритма и построенных ранее операторов. Если найден хороший тест, производится переход к этапу 11.
10. **Генерация тестов с помощью эволюционной стратегии**. На данном этапе производится генерация тестов с использованием эволюционной стратегии и построенных ранее операторов. Если найден хороший тест, производится переход к этапу 11.
11. **Проверка имеющихся решений олимпиадной задачи на сгенерированном тесте**. На данном этапе проверяется, какие еще решения призна-

ются неэффективными с помощью сгенерированного теста. После этого производится переход к первому этапу.

12. **Выбор подмножества сгенерированных тестов**. На данном (заключительном) этапе полученный набор тестов минимизируется таким образом, чтобы все известные неэффективные решения признавались неэффективными, но при этом общее число тестов было минимально возможным.

Данная технология уменьшает влияние человеческого фактора на качество набора тестов в силу того, что чисто эвристический подход к построению тестов заменен использованием стандартной схемы, характерной для решения задач с помощью эволюционных алгоритмов.

В **шестой главе** описывается внедрение предложенной технологии при генерации тестов для задачи «Ships. Version 2», расположенной в архиве задач с проверяющей системой *Timus Online Judge*, предназначенном для подготовки к олимпиадам по программированию. Данная задача является частным случаем задачи о мультирюкзаке, в котором сумма весов всех предметов равна сумме вместимостей всех рюкзаков. Указанный частный случай является NP-трудной задачей, что в совокупности с ограничениями, указанными в условии задачи, делают ее крайне трудной для решения. Однако в 2009 году по данной задаче на основании имевшихся на тот момент тестов 260 решений были признаны корректными по тестам. Применение технологии позволило признать **все** указанные решения неэффективными.

В **седьмой главе** описывается внедрение предложенной технологии при генерации тестов для задачи «Work for Robots», расположенной в том же архиве задач, что и предыдущая задача. Данная задача состоит в подсчете числа клик в графе и потому является #P-полной, но в отличие от предыдущей задачи, у нее имеется корректное решение, основанное на динамическом программировании по подмножествам и идее «разделяй-и-властвуй».

По состоянию на 2009 год администрацией сайта 86 решений было признано корректными по тестам. Однако кроме указанного корректного решения среди них было множество решений, основанных, главным образом, на идее перебора с возвратом. В результате применения технологии 45 из 86 решений были признаны неэффективными на новых тестах, которые были добавлены в архив.

Заключение

В диссертационном исследовании получены следующие результаты:

1. Предложен метод генерации тестов для алгоритмов решения NP-трудных задач на основе генетического алгоритма (на примере задачи о рюкзаке).
2. Предложен метод генерации тестов для алгоритмов решения графовых задач на основе генетического алгоритма (на примере задачи о поиске максимального потока).

3. Разработан алгоритм выбора из нескольких функций приспособленности функции, оптимальной по времени решения задачи оптимизации.
4. Предложена и внедрена технология генерации тестов для определения неэффективных решений олимпиадных задач по программированию на основе эволюционных алгоритмов.

Статьи в журналах из перечня ВАК

1. *Буздалов М. В.* Генерация тестов для олимпиадных задач по программированию с использованием генетических алгоритмов // Научно-технический вестник СПбГУ ИТМО. — 2011. — 2(72). — С. 72—77. — 0,375 п. л.
2. *Буздалов М. В.* Генерация тестов для олимпиадных задач по теории графов с использованием эволюционных стратегий // Научно-технический вестник СПбГУ ИТМО. — 2011. — 6(76). — С. 123—127. — 0,3125 п. л.

Публикации в рецензируемых изданиях, индексируемых Web of Science или Scopus

3. *Buzdalov M.* Generation of Tests for Programming Challenge Tasks Using Evolution Algorithms // Proceedings of Genetic and Evolutionary Computation Conference Companion. — ACM, 2011. — Pp. 763–766. — 0,25 п. л.
4. *Buzdalov M.* Generation of Tests for Programming Challenge Tasks on Graph Theory using Evolution Strategy // Proceedings of the International Conference on Machine Learning and Applications. Vol. 2. — IEEE Computer Society, 2012. — Pp. 62–65. — 0,25 п. л.
5. *Buzdalov M.*, *Buzdalova A.* Adaptive Selection of Helper-Objectives for Test Case Generation // 2013 IEEE Congress on Evolutionary Computation. Vol. 1. — 2013. — Pp. 2245–2250. — 0,375 п. л. / 0,19 п. л.
6. *Buzdalova A.*, *Buzdalov M.*, *Parfenov V.* Generation of Tests for Programming Challenge Tasks Using Helper-Objectives // 5th International Symposium on Search-Based Software Engineering. — Springer, 2013. — Pp. 300–305. — (Lecture Notes in Computer Science ; 8084). — 0,1875 п. л. / 0,07 п. л.
7. *Buzdalov M.*, *Buzdalova A.*, *Petrova I.* Generation of Tests for Programming Challenge Tasks Using Multi-Objective Optimization // Proceedings of Genetic and Evolutionary Computation Conference Companion. — ACM, 2013. — Pp. 1655–1658. — 0,25 п. л. / 0,1 п. л.
8. *Arkhipov V.*, *Buzdalov M.*, *Shalyto A.* Worst-Case Execution Time Test Generation for Augmenting Path Maximum Flow Algorithms using Genetic Algorithms // Proceedings of the International Conference on Machine Learning and Applications. Vol. 2. — IEEE Computer Society, 2013. — Pp. 108–111. — 0,25 п. л. / 0,15 п. л.

9. *Buzdalov M.*, *Shalyto A.* Worst-Case Execution Time Test Generation for Solutions of the Knapsack Problem using a Genetic Algorithm // Proceedings of 9th International Conference on Bio-inspired Computing: Theories and Applications. — Springer, 2014. — Pp. 1–10. — (Communications in Computer and Information Science ; 472). — 0,3125 п. л. / 0,25 п. л.
10. Worst-Case Execution Time Test Generation using Genetic Algorithms with Automated Construction and Online Selection of Objectives / N. Kravtsov, M. Buzdalov, A. Buzdalova, A. Shalyto // Proceedings of 20th International Conference on Soft Computing MENDEL 2014. — Czech Republic, 2014. — Pp. 111–116. — 0,375 п. л. / 0,15 п. л.
11. *Buzdalov M.* A Switch-and-Restart Algorithm with Exponential Restart Strategy for Objective Selection and its Runtime Analysis // Proceedings of the International Conference on Machine Learning and Applications. Vol. 1. — IEEE Computer Society, 2014. — Pp. 112–117. — 0,375 п. л.

Другие публикации

12. *Буздалов М. В.* Применение генетических алгоритмов для определения неэффективных решений олимпиадных задач по программированию (на примере задачи о рюкзаке) // Труды Третьей Всероссийской конференции «Нечеткие системы и мягкие вычисления». Т. 2. — 2009. — С. 16—24. — 0,25 п. л.
13. *Буздалов М. В.* Генерация тестов для олимпиадных задач по программированию с использованием эволюционных стратегий // Материалы Второй межвузовской научной конференции по проблемам информатики СПИСОК. — 2011. — С. 336—338. — 0,09 п. л.
14. *Буздалов М. В.* Применение эволюционных алгоритмов для покрытия кода тестами // Материалы Всероссийской научной конференции по проблемам информатики СПИСОК. — 2012. — С. 404—408. — 0,15 п. л.
15. *Буздалова А. С.*, *Буздалов М. В.* Использование вспомогательных функций приспособленности для тестирования решений олимпиадных задач по программированию // Материалы Всероссийской научной конференции по проблемам информатики СПИСОК. — 2013. — С. 548—555. — 0,25 п. л. / 0,1 п. л.
16. *Якорев В. О.*, *Буздалов М. В.* Генерация тестов для олимпиадных задач по программированию с помощью многокритериальных эволюционных алгоритмов // Материалы Всероссийской научной конференции по проблемам информатики СПИСОК. — 2013. — С. 571—573. — 0,1 п. л. / 0,05 п. л.
17. *Буздалов М. В.*, *Буздалова А. С.* Асимптотически оптимальные алгоритмы для выбора вспомогательных критериев оптимизации // Материалы Всероссийской научной конференции по проблемам информатики СПИСОК. — 2014. — С. 324—329. — 0,2 п. л. / 0,15 п. л.