УДК 004.4'242

ПОСТРОЕНИЕ АВТОМАТНЫХ ПРОГРАММ ПО СПЕЦИФИКАЦИИ С ПОМОЩЬЮ МУРАВЬИНОГО АЛГОРИТМА НА ОСНОВЕ ГРАФА МУТАЦИЙ

Д.С. Чивилихин[a], В.И. Ульянцев[a], В.В. Вяткин[a,b,c], А.А. Шалыто[a]

[a]Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, chivdan@rain.ifmo.ru
[b] Университет Аалто, Хельсинки, FI-00076, Финляндия
[c] Технологический университет Лулео, Лулео, SE-971 87, Швеция

Аннотация. Традиционный в области разработки программ процесс тестирования не может гарантировать их корректности, поэтому при повышенных требованиях к надежности программ прибегают к верификации. Верификация позволяет проверять некоторые свойства программы во всех возможных ее состояниях, однако сам процесс верификации сложен. В методе проверки моделей (model checking) строится модель программы (обычно вручную), а требования к ней записываются на языке темпоральной логики. Выполнение или невыполнение этих требований к модели может быть проверено автоматически. Основной проблемой такого подхода является разрыв между программой и ее моделью. Парадигма автоматного программирования позволяет устранить указанный разрыв. В автоматном программировании логика работы программ описывается управляющими конечными автоматами, модели которых могут быть построены автоматически. В работе рассматривается применение муравьиного алгоритма на основе графа мутаций для решения задач построения автоматных программ по их спецификации, заданной сценариями работы и темпоральными свойствами. Апробация предложенного подхода проведена на примере задачи построения автомата управления дверьми лифта, а также на случайных данных. Полученные результаты показывают, что муравьиный алгоритм в два–три раза эффективнее ранее применявшегося генетического. Предложенный подход может быть рекомендован для автоматизированного построения управляющих программ для ответственных систем.

Ключевые слова: конечный автомат, муравьиный алгоритм, верификация.

Благодарности. Работа выполнена при государственной финансовой поддержке ведущих университетов Российской Федерации (субсидия 074-U01), при поддержке РФФИ в рамках научного проекта № 14-07-31337 мол_а.

AUTOMATA PROGRAMS CONSTRUCTION FROM SPECIFICATION WITH AN ANT COLONY OPTIMIZATION ALGORITHM BASED ON MUTATION GRAPH

D.S. Chivilikhin[a], V.I. Ulyantsev[a], V.V. Vyatkin[a,b,c], A.A. Shalyto[a]

[a]ITMO University, Saint Petersburg, 197101, Russian Federation, chivdan@rain.ifmo.ru
[a]Aalto University, Helsinki, FI-00076, Finland
[c] Luleå University of Technology, Luleå, SE-971 87, Sweden, valeriy.vyatkin@aalto.fi

Abstract. The procedure of testing traditionally used in software engineering cannot guarantee program correctness; therefore verification is used at the excess requirements to programs reliability. Verification makes it possible to check certain properties of programs in all possible computational states; however, this process is very complex. In the model checking method a model of the program is built (often, manually) and requirements in terms of temporal logic are formulated. Such temporal properties of the model can be checked automatically. The main issue in this framework is the gap between the program and its model. Automata-based programming paradigm gives the possibility to overcome this limitation. In this paradigm, program logic is represented using finite-state machines. The advantage of finite-state machines is that their models can be constructed automatically. The paper deals with the application of mutation-based ant colony optimization algorithm to the problem of finite-state machine construction from their specification, defined by test scenarios and temporal properties. The presented approach has been tested on the elevator doors control problem as well as on randomly generated data. Obtained results show the ant colony algorithm is two-three times faster than the previously used genetic algorithm. The proposed approach can be recommended for inferring control programs for critical systems.

Keywords: finite-state machine, ant colony algorithm, verification.

Acknowledgements. The work is partially financially supported by the Government of the Russian Federation (grant 074-U01), and also partially supported by RFBR (scientific project № 14-07-31337 мол_а).

Введение

Проектирование программного обеспечения является сложной задачей, особенно когда область применения требует высокого уровня надежности программ. В таких областях, как космическая и военная промышленность, авиация, энергетика и медицина, цена ошибок очень высока, так как их возникновение может привести к большим потерям ресурсов или нанести вред здоровью человека. Программы в таких случаях не могут подвергаться одной лишь процедуре тестирования, поскольку она может только указать на наличие ошибок, но не может гарантировать их отсутствия.

Метод проверки моделей (model checking [1]) применяется для проверки некоторых свойств программы во всех возможных ее состояниях. При традиционном подходе сначала строится модель рассматриваемой программной системы, а требования к модели записываются на языке темпоральной логики. Модель чаще всего строится вручную. Темпоральные свойства проверяются для модели, а не для исходной программы, что в общем случае указывает на разрыв между программой и ее моделью.

Парадигма автоматного программирования [2], в которой логика работы программ представляется в виде одного или нескольких управляющих конечных автоматов, позволяет преодолеть это ограничение. Программы, разработанные с помощью этой парадигмы, могут быть автоматически преобразованы в используемые в методе проверки моделей структуры [3, 4]. Таким образом, нет никакого разрыва между автоматными программами и их моделями.

Разрабатываемые в настоящей работе методы можно применять для построения систем управления, основанных на автоматах. Например, в стандарте проектирования распределенных систем управления IEC 61499 [1] конечные автоматы используются как ключевые управляющие элементы функциональных блоков. Методы, рассматриваемые в данной работе, могут быть применены для автоматизации различных этапов построения распределенных систем управления [5].

Основным достоинством автоматного программирования является возможность создавать программы автоматически по спецификации, которая может содержать, например, сценарии работы и темпоральные свойства. Для построения программ, в частности, применяются алгоритмы поисковой оптимизации, выполняющие направленный перебор решений-кандидатов. Так, в [3, 4] для этого применялись эволюционные алгоритмы. К сожалению, эти алгоритмы не обладают достаточной эффективностью, хотя и позволяют находить решение во много раз быстрее, чем прямым перебором. В [6] был предложен муравьиный алгоритм построения управляющих конечных автоматов, который для известной модельной задачи об «умном» муравье оказался существенно эффективнее генетического алгоритма.

Целью данной работы является экспериментальная проверка эффективности указанного муравьиного алгоритма на задаче построения автоматов по спецификации, состоящей из сценариев работы и темпоральных формул [4]. Эксперименты показали, что муравьиный алгоритм обладает более высокой производительностью по сравнению с ранее применявшимися генетическими алгоритмами. Настоящая работа основана на [7]. Был применен более современный метод настройки параметров алгоритмов, проведены более обширные экспериментальные исследования, в том числе на случайных данных.

Построение автоматных программ по спецификации

В работе управляющим конечным автоматом будем называть семерку $\langle X, E, Y, Z, y_0, \phi, \delta \rangle$, где X – множество булевых входных переменных; E – множество входных событий; Y – множество состояний; $y_0 \in Y$ – начальное состояние; Z – множество выходных воздействий; $\phi : Y \times E \times 2^X \to Y$ – функция переходов, а $\delta : Y \times E \times 2^X \to Z^*$ – функция выходов.

Проще говоря, управляющий конечный автомат – это такой детерминированный автомат, каждый переход которого помечен событием, булевой формулой от входных переменных и последовательностью выходных воздействий. Пример графа переходов управляющего автомата с тремя состояниями приведен на рис. 1. Начальное состояние выделено жирной рамкой. На каждом переходе перед косой чертой записано входное событие (A и H), в квадратных скобках записана булева формула от входных переменных x_1 и x_2, а после косой черты записана последовательность выходных воздействий.

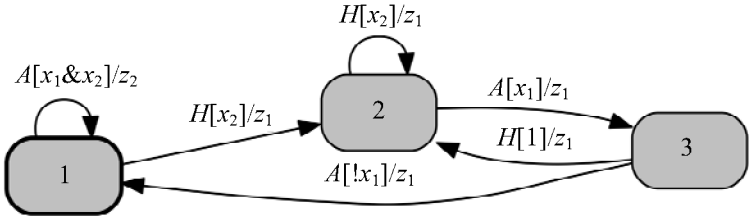

Рис. 1. Пример управляющего конечного автомата. Здесь $X = \{x_1, x_2\}$, $E = \{A, H\}$, $Y = \{1,2,3\}$, $y_0 = 1$, $Z = \{z_1, z_2\}$

В работе рассматриваются спецификации, содержащие сценарии работы и темпоральные формулы. Сценарием работы называется последовательность троек $\langle e, \varphi, O \rangle$, называемых элементами сценария, где $e \in E$ – событие, $\varphi \in 2^X$ – булева формула от входных переменных и $O \in Z^*$ – последовательность выходных воздействий (возможно, пустая). Говорят, что автомат удовлетворяет элементу сценария $\langle e, \varphi, O \rangle$ в состоянии y, если в этом состоянии автомат содержит переход, помеченный событием e, последовательностью выходных воздействий O и охранным условием, равным φ как булева формула. Автомат удовлетворяет сценарию работы, если он удовлетворяет всем последовательно обработанным элементам сценария в соответствующих состояниях.

[1] Function blocks: International Standard IEC 61499 Part 1: Architecture. International Electrotechnical Commission, 2011-12.

Для представления темпоральных свойств автомата в работе используется логика линейного времени (Linear Temporal Logic, LTL). Язык LTL состоит из пропозициональных переменных **Prop**, логических операторов **and**, **or**, **not** и набора темпоральных операторов, таких как, например, G (глобально в будущем), X (в следующий момент времени) и F (когда-либо в будущем). Для проверки того, удовлетворяет ли автомат LTL-формуле, применяется верификатор автоматных программ, разработанный в [3, 8]. Указанный верификатор принимает на вход автомат и LTL-формулу и либо выдает на выход контрпример (путь в модели Крипке [9], нарушающий формулу), либо сообщение о том, что автомат удовлетворяет формуле.

Задачу построения управляющего автомата по спецификации можно сформулировать так: требуется построить автомат с не более чем N_{states} состояниями, удовлетворяющий набору сценариев работы и множеству LTL-формул. В [10] показано, что уже задача построения автомата только по сценариям является, как минимум, NP-трудной. Этим обусловливается необходимость применения метаэвристических алгоритмов [11] при решении поставленной задачи.

Рассматриваемый в настоящей работе муравьиный алгоритм [6], как и генетический, представляет собой алгоритм направленного перебора. Степень соответствия каждого решения-кандидата поставленной задаче в таких алгоритмах оценивается с помощью так называемой функции приспособленности (ФП), которую необходимо максимизировать.

Функция приспособленности

Для оценки соответствия управляющих автоматов заданной спецификации используется ФП, предложенная в [8]. Первый компонент ФП f_{tests} оценивает, насколько хорошо автомат соответствует заданному набору сценариев работы, второй компонент f_{LTL} оценивает соответствие автомата темпоральным формулам. Выражение для ФП учитывает также число переходов автомата и имеет вид

$$f = f_{tests} + f_{LTL} + \frac{M - n_{transitions}}{100 \cdot M},$$

где $n_{transitions}$ – число всех переходов автомата, а M – число, гарантированно превосходящее $n_{transitions}$. Величины f_{tests} и f_{LTL} принимают значения от нуля до единицы, а последний член в указанной формуле для f не превышает 0,01. Таким образом, автоматы со значением ФП, большим двух, удовлетворяют всем сценариям работы и темпоральным формулам.

Муравьиный алгоритм построения автоматов

Основой метода построения конечных автоматов из [6] является представление пространства поиска (множества всех автоматов с заданными параметрами) в виде ориентированного графа G. Каждая вершина этого графа, который будем называть графом мутаций, ассоциирована с конечным автоматом, а ребра соответствуют мутациям автоматов.

Под мутацией конечного автомата понимается небольшое изменение его структуры – функций переходов φ и выходов δ. В данной работе используются следующие операторы мутации конечных автоматов.

1. **Изменение состояния, в которое ведет переход**. Для случайно выбранного перехода в автомате состояние y, в которое он ведет, заменяется другим состоянием, выбранным случайным образом из множества $Y \setminus \{y\}$.
2. **Добавление или удаление переходов**. Наличие некоторых переходов в состоянии автомата может сделать его противоречащим LTL-формуле. В связи с этим необходимо иметь возможность периодически удалять и соответственно добавлять переходы. Оператор мутации, предложенный в [4], сканирует состояния автомата и с определенной вероятностью изменяет набор переходов в выбранном состоянии. Случайным образом решается, добавить или удалить переход. Переход добавляется лишь в том случае, когда в этом состоянии нет перехода, помеченного какой-либо встречающейся в сценариях комбинацией входного события и булевой формулы. Тогда в состояние добавляется новый переход, который ведет в случайно выбранное состояние. В случае удаления перехода из текущего состояния удаляется случайно выбранный переход.

Пример графа мутаций приведен на рис. 2. Вершины соответствуют автоматам, а ребра – мутациям автоматов. Запись на ребре (2, $H[x_2]$) → 1 означает, что мутация изменила состояние, в которое ведет переход из состояния 2 по событию H и формуле x_2, на состояние 1. На каждом ребре (u, v) графа, как и в классических муравьиных алгоритмах [12], задаются две величины: так называемое значение эвристической информации η_{uv} и значение феромона $\tau_{\min}$. Эвристическая информация вычисляется по формуле

$$\eta_{uv} = \max\left(\eta_{\min}, f(v) - f(u)\right),$$

где $\eta_{\min}$ – небольшая положительная константа, например, 10^{-3}. В начале работы алгоритма всем ребрам графа сопоставляется начальное значение феромона $\tau_{\min} = 10^{-3}$.

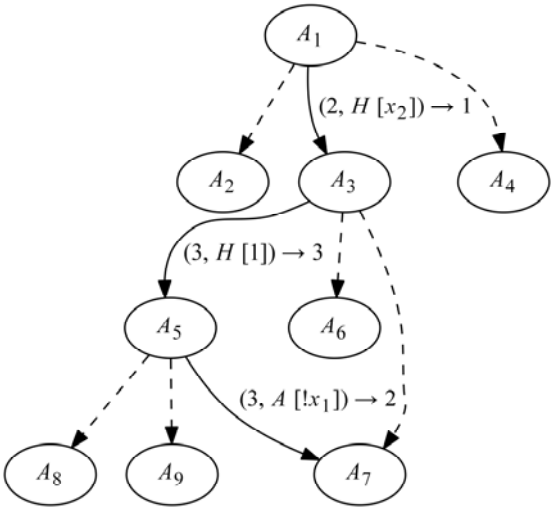

Рис. 2. Пример графа мутаций. Вершины соответствуют автоматам, а ребра – мутациям автоматов. Запись на ребре (2, H [x_2]) → 1 означает, что мутация изменила состояние, в которое ведет переход автомата из состояния 2 по событию H и формуле x_2, на состояние 1

В процессе работы алгоритма значения феромона изменяются муравьями, а значения эвристической информации остаются неизменными. На каждой итерации алгоритма выполняется две операции: построение решений муравьями и обновление значений феромона. Также происходит проверка заданных условий останова.

Построение решений муравьями

Процедуру построения решений муравьями можно разделить на два этапа. На первом этапе выбираются вершины графа, из которых муравьи начнут построение путей. В качестве единственной стартовой вершины для всех муравьев выбирается вершина, ассоциированная с лучшим найденным решением.

На втором этапе совершается одна итерация муравьиной колонии, в ходе которой каждый муравей перемещается по графу, начиная с соответствующей стартовой вершины. Пусть муравей находится в вершине u, ассоциированной с автоматом A. Если у вершины u существуют инцидентные ей ребра, то муравей выполняет одно из двух действий – построение новых решений либо вероятностный выбор. Если у вершины u нет инцидентных ей ребер, то муравей всегда выполняет построение новых решений.

1. **Построение новых решений**. С вероятностью p_{new} муравей пытается создать новые ребра и вершины графа G путем выполнения фиксированного числа N_{mut} мутаций автомата A. После выполнения муравьем всех мутаций он выбирает лучшую из построенных вершин и переходит в нее. Процесс обработки одной мутации автомата A таков:
   - выполнить мутацию автомата A, получить автомат A_{mut};
   - найти в графе G вершину t, ассоциированную с автоматом A_{mut}. Если такой вершины не существует, то создать ее и добавить в граф;
   - добавить в граф ребро (u, t).

   Отметим, что переход осуществляется даже в том случае, когда значение ФП у лучшего из построенных с помощью мутаций автоматов меньше значения ФП автомата A. Это свойство алгоритма позволяет ему выходить из локальных оптимумов.
2. **Вероятностный выбор**. С вероятностью $1 - p_{new}$ муравей выбирает следующую вершину из множества N_u ребер, инцидентных вершине u. Вершина v выбирается с вероятностью, определяемой классической в муравьиных алгоритмах формулой [12]:

$$p_{uv} = \frac{\tau_{uv}^{\alpha} \cdot \eta_{uv}^{\beta}}{\sum\limits_{w \in N_u} \tau_{uw}^{\alpha} \cdot \eta_{uw}^{\beta}},$$

   где $\alpha, \beta \in [0, 1]$ – параметры, определяющие значимость значений феромона и эвристической информации при выборе пути.

Каждый муравей в колонии делает по одному шагу до тех пор, пока все муравьи не остановятся. Каждый муравей может выполнить максимум n_{stag} шагов, на каждом из которых происходит построение новых решений либо вероятностный выбор без увеличения своего значения ФП. После этого муравей будет остановлен. Аналогично, колония муравьев может выполнить максимум N_{stag} итераций без увеличения максимального значения ФП. Затем алгоритм перезапускается.

Обновление значений феромона

Значение феромона, которое муравей откладывает на ребрах своего пути, равно максимальному значению ФП всех автоматов, посещенных этим муравьем. Для каждого ребра (u, v) графа G хранится число τ_{uv}^{best} – наибольшее из значений феромона, когда-либо отложенных муравьями на этом ребре. Последовательно рассматриваются все пути муравьев на текущей итерации алгоритма. Для каждого пути выделяется отрезок от начала пути до вершины, содержащей автомат с наибольшим на пути значением ФП $f_{\max}$. Для всех ребер из этого отрезка обновляются значения τ_{uv}^{best}: $\tau_{uv}^{best} \leftarrow \max\left(f_{\max}, \tau_{uv}^{best}\right)$. Затем значения феромона на всех ребрах графа G обновляются по формуле

$$\tau_{uv} \leftarrow \max\left(\tau_{\min}, (1-\rho)\tau_{uv} + \tau_{uv}^{best}\right),$$

где $\rho \in [0, 1]$ – скорость испарения феромона; $\tau_{\min} = 10^{-3}$ – минимальное разрешенное значение феромона. Введение нижней границы $\tau_{\min}$ исключает чрезмерное испарение феромона с ребер графа.

Настройка значений параметров алгоритмов

Для обеспечения адекватного сравнения эффективности муравьиного и генетического алгоритмов перед запуском экспериментов была проведена автоматическая настройка значений параметров алгоритмов с помощью программного средства *irace* [13]. На вход средству подается информация об интервалах возможных значений параметров настраиваемого алгоритма и множество экземпляров задачи, в данном случае – множество наборов сценариев и темпоральных формул. Это множество экземпляров задачи является обучающим для процесса настройки значений параметров. Средство *irace* реализует итеративную процедуру, в которой эффективность того или иного набора параметров настраиваемого алгоритма оценивается путем запуска этого алгоритма на экземплярах задачи из обучающего набора. По ходу работы *irace* с помощью статистического теста определяются неэффективные наборы параметров, которые затем отбрасываются.

Обучающее множество экземпляров задачи состояло из 200 наборов сценариев и темпоральных формул. При создании каждого из экземпляров задачи сначала генерировался случайный конечный автомат, число состояний N_{states} которого выбиралось случайно из отрезка [5, 10]. Остальные параметры автоматов имели следующие значения: $|E| = 2, |X| = 1, |Z| = 2$. По каждому автомату был сгенерирован набор из $5 \cdot |Y|$ сценариев общей длиной $100 \cdot |Y|$, а также две LTL-формулы.

На настройку параметров каждого алгоритма было отведено 12 ч на компьютере с процессором AMD Phenom(tm) II x4 955 3.2 GHz. В результате были получены значения параметров муравьиного и генетического алгоритмов, указанные в табл. 1, которые далее использовались во всех экспериментах.

Генетический алгоритм	
Параметр	**Значение**
Размер популяции	201
Доля элитных особей	0,25
Вероятность мутации	0,06
Число итераций до малой мутации поколения	483
Число итераций до большой мутации поколения	100

Муравьиный алгоритм	
Параметр	**Значение**
Число муравьев N_{ants}	4
Число мутаций N_{mut}	44
Максимальное число итераций колонии без увеличения значения ФП N_{stag}	28
Максимальное число шагов муравья без увеличения значения ФП n_{stag}	45
Скорость испарения феромона ρ	0,52

Таблица 1. Значения параметров муравьиного и генетического алгоритмов, полученные с помощью программного средства *irace*

Экспериментальное исследование предлагаемого подхода

Построение автомата управления дверьми лифта. В первой части экспериментального исследования рассматривалась задача управления конкретной системой (дверьми лифта) [8]. Система характеризуется пятью входными событиями: e_{11} (нажата кнопка «Открыть двери»), e_{12} (нажата кнопка «Закрыть двери»), e_2 (двери успешно открыты или закрыты), e_3 (препятствие помешало дверям закрыться), e_4 (двери заклинило). Существует три возможных выходных воздействия: z_1 (начать открывать двери), z_2 (начать закрывать двери), z_3 (послать аварийный сигнал). Спецификация, являющаяся входными данными для генетического алгоритма [3, 4] и применяемого в работе муравьиного алгоритма, состоит из

9 сценариев работы и 11 LTL-формул. Поиск осуществлялся среди автоматов с не более чем $N_{states} = 6$ состояниями. Автомат, решающий задачу, изображен на рис. 3, а.

В каждом эксперименте было проведено по 1000 запусков каждого алгоритма, каждый запуск продолжался до достижения лучшим решением значения ФП, равного 2,0075, что соответствует автомату, полностью удовлетворяющему заданной спецификации.

Для сравнения указанных алгоритмов измерялось медианное время работы и число вычислений ФП. Медианное время работы генетического алгоритма составило 7,3 с, а муравьиного – 3,5 с. Медианное число вычислений ФП генетического алгоритма равно 41902, а муравьиного – 10119. На рис. 3, б, приведена ящичная диаграмма, изображающая распределения запусков муравьиного и генетического алгоритмов по времени работы. Нижний и верхний края «ящика» изображают первый и третий квартили распределения, черта внутри ящика соответствует медиане, а горизонтальные линии снизу и сверху от ящика обозначают края статистически значимой выборки. Точки сверху от этих линий соответствуют выбросам.

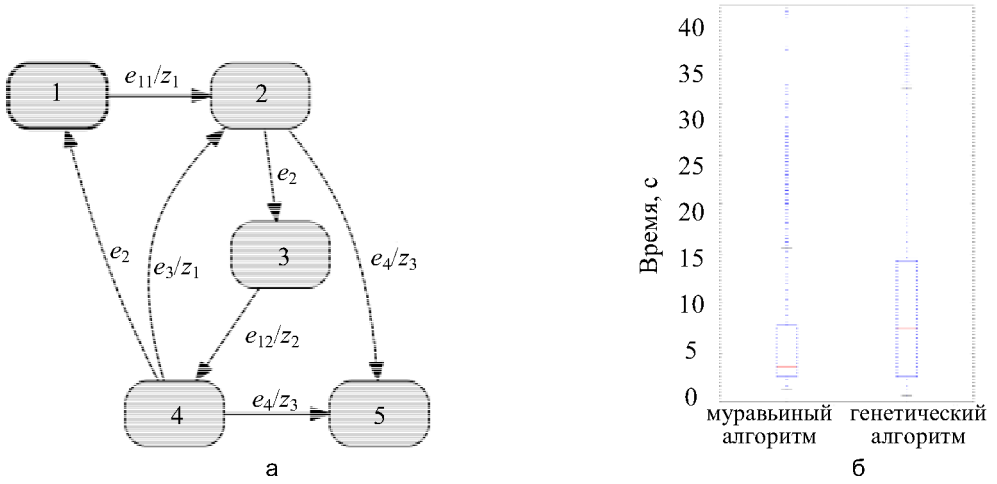

Рис. 3. Автомат управления дверьми лифта (а) и ящичная диаграмма распределения запусков муравьиного и генетического алгоритмов по времени работы (б)

Полученные в эксперименте результаты позволяют утверждать, что муравьиный алгоритм решает рассматриваемую задачу в два–три раза быстрее генетического алгоритма. Более того, как видно из рис. 3, б, муравьиный алгоритм, по сравнению с генетическим алгоритмом, обеспечивает меньший разброс времени работы. Статистическая значимость полученных результатов была подтверждена с помощью теста Уилкоксона [14]. Значение p-value, равное $3{,}93 \cdot 10^{-13}$, указывает на то, что вероятность совпадения среднего времени работы алгоритмов крайне мала.

Построение автоматов по случайным данным. Во второй части экспериментального исследования рассматривались случайно сгенерированные автоматы, содержащие от 4 до 10 состояний.

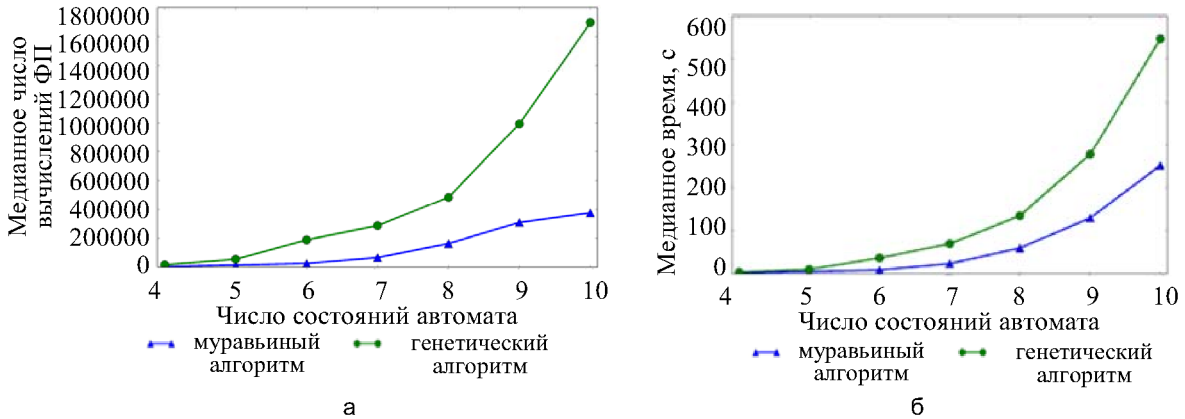

Рис. 4. Графики зависимости медианных значений числа вычислений ФП (а) и времени работы (б) от числа состояний автомата для генетического и муравьиного алгоритмов

Для каждого значения числа состояний N_{states} было сгенерировано по 100 автоматов со следующими параметрами: два входных события, два выходных воздействия, длина последовательности выходных воздействий от нуля до двух, булевы формулы зависят от единственной входной переменной. По каждому из построенных автоматов был построен набор из $5 \cdot N_{states}$ сценариев работы общей длиной $100 \cdot N_{states}$ и

две LTL-формулы. Формулы генерировались таким образом, чтобы они выполнялись для автомата, но не выполнялись для большого числа других автоматов.

N_{states}	p-value
4	$1 \cdot 10^{-7}$
5	$4 \cdot 10^{-6}$
6	$1 \cdot 10^{-10}$
7	$2 \cdot 10^{-6}$
8	$2 \cdot 10^{-4}$
9	$2 \cdot 10^{-6}$
10	$5 \cdot 10^{-4}$

Таблица 2. Результаты статистического теста Уилкоксона для экспериментов со случайно сгенерированными автоматами

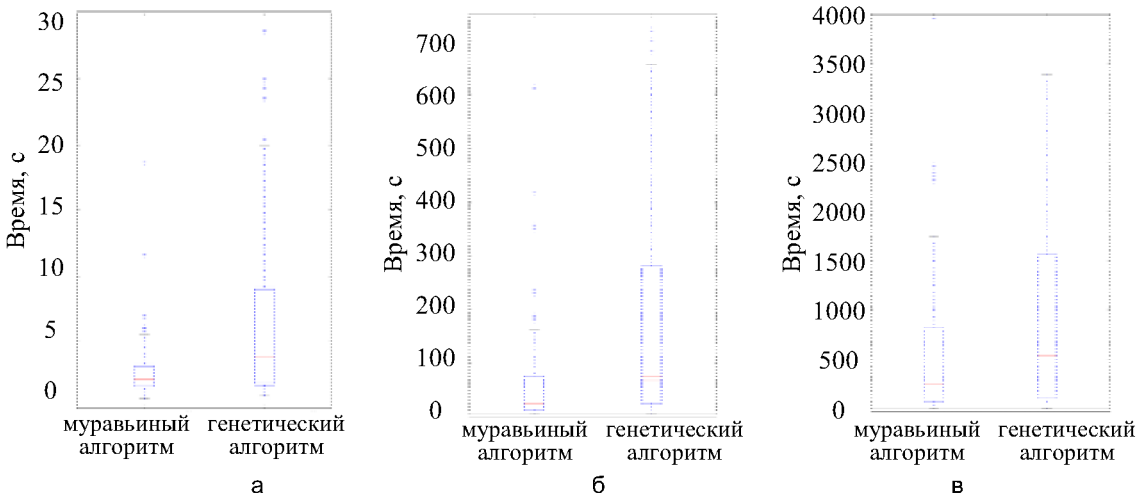

Рис. 5. Ящичные диаграммы распределения запусков муравьиного и генетического алгоритмов по времени работы для экспериментов со случайными автоматами с N_{states}: 4 (а); 7 (б); 10 (в)

Каждый эксперимент продолжался до нахождения алгоритмом решения, удовлетворяющего всем сценариям и темпоральным формулам. Замерялось число совершенных вычислений ФП, а также время работы алгоритмов. Графики медианных значений этих величин в зависимости от числа состояний автомата приведены на рис. 4, ящичные диаграммы распределений запусков алгоритмов по времени работы изображены на рис. 5. Результаты проведения статистического теста приведены в табл. 2. Полученные экспериментальные результаты позволяют сделать вывод о том, что на случайных данных муравьиный алгоритм существенно эффективнее генетического.

Заключение

В работе рассмотрено применение метода построения управляющих конечных автоматов на основе муравьиного алгоритма и графа мутаций к задаче построения автоматов по спецификации, включающей сценарии работы и темпоральные формулы. Проведенные экспериментальные исследования позволяют сделать вывод о том, что муравьиный алгоритм в два–три раза превосходит по производительности ранее применявшийся метод, основанный на генетическом алгоритме.

Литература

1. Clarke E.M., Grumberg O., Peled D.A. Model Checking. MIT press, 1999. 330 p.
2. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. СПб: Питер, 2009. 176 с.
3. Егоров К.В., Царев Ф.Н., Шалыто А.А. Применение генетического программирования для построения автоматов управления системами со сложным поведением на основе обучающих примеров и спецификации // Научно-технический вестник СПбГУ ИТМО. 2010. № 5 (69). С. 81–86.
4. Tsarev F., Egorov K. Finite state machine induction using genetic algorithm based on testing and model checking // Proc. 13th Annual Genetic and Evolutionary Computation Conference, GECCO'11. Dublin, Ireland, 2011. P. 759–762.

5. Yang C.-H., Vyatkin V., Pang C. Model-driven development of control software for distributed automation: a survey and an approach // IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2014. V. 44. N 3. P. 292–305.
6. Chivilikhin D., Ulyantsev V. MuACOsm – a new mutation-based ant colony optimization algorithm for learning finite-state machines // Proc. 15th Genetic and Evolutionary Computation Conference, GECCO 2013. Amsterdam, Netherlands, 2013. P. 511–518.
7. Чивилихин Д.С., Ульянцев В.И., Шалыто А.А. Муравьиный алгоритм для построения автоматных программ по спецификации // Труды XII Всероссийского совещания по проблемам управления ВСПУ-2014. Москва, 2014. С. 4531–4542.
8. Егоров К.В. Генерация управляющих конечных автоматов на основе генетического программирования и верификации: дис. … канд. техн. наук. СПб.: НИУ ИТМО, 2013. 150 с.
9. Вельдер С.Э., Лукин М.А., Шалыто А.А., Яминов Б.Р. Верификация автоматных программ. СПб: Наука, 2011. 244 с.
10. Ульянцев В.И. Построение управляющих конечных автоматов по сценариям работы на основе решения задачи удовлетворения ограничений: магистерская диссертация [Электронный ресурс]. Режим доступа: http://is.ifmo.ru/diploma-theses/2013/master/ulyantsev/ulyantsev.pdf, свободный. Яз. рус. (дата обращения 30.09.2014).
11. Скобцов Ю.А., Федоров Е.Е. Метаэвристики. Донецк: НОУЛИДЖ, 2013. 426 с.
12. Dorigo M., Stützle T. Ant Colony Optimization. MIT Press, 2004. 319 p.
13. López-Ibáñez M., Dubois-Lacoste J., Stützle T., Birattari M. The irace package: iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004. IRIDIA, Belgium, 2011. 20 p.
14. Wilcoxon F. Probability tables for individual comparisons by ranking methods // Biometrics. 1947. V. 3. P. 119–122.

Чивилихин Даниил Сергеевич – аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, chivdan@rain.ifmo.ru

Ульянцев Владимир Игоревич – аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, ulyantsev@rain.ifmo.ru

Вяткин Валерий Владимирович – доктор технических наук, профессор, Университет Аалто, Хельсинки, FI-00076, Финляндия; заведующий кафедрой, Технологический университет Лулео, Лулео, SE-971 87, Швеция, valeriy.vyatkin@aalto.fi

Шалыто Анатолий Абрамович – доктор технических наук, профессор, профессор, главный научный сотрудник, заведующий кафедрой, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, shalyto@mail.ifmo.ru

Daniil S. Chivilikhin – postgraduate, ITMO University, Saint Petersburg, 197101, Russian Federation, chivdan@rain.ifmo.ru

Vladimir I. Ulyantsev – postgraduate, ITMO University, Saint Petersburg, 197101, Russian Federation, ulyantsev@rain.ifmo.ru

Valeriy V. Vyatkin – D.Sc., Professor, Aalto University, Helsinki, FI-00076, Finland; Chaired Professor of Dependable Communications and Computations, Luleå University of Technology, Luleå, SE-971 87, Sweden, valeriy.vyatkin@aalto.fi

Anatoly A. Shalyto – D.Sc., Professor, Department head, ITMO University, Saint Petersburg, 197101, Russian Federation, shalyto@mail.ifmo.ru

Принято к печати 22.08.14
Accepted 22.08.14