ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

Факультет Информационных технологий и программирования

Направление подготовки 010400 «Прикладная математика и информатика»

Специализация 010400.68.04 «Технология проектирования и разработки ПО»

Квалификация (степень) Магистр прикладной математики и информатики

Специальное звание не предусмотрено

Кафедра Компьютерных технологий Группа 6539

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

Метод построения конечных автоматов

на основе муравьиного алгоритма

Автор магистерской диссертации Чивилихин Д. С. (подпись)
(Фамилия, И., О.)

Научный руководитель Царев Ф. Н. (подпись)
(Фамилия, И., О.)

Руководитель магистерской программы Васильев В. Н. (подпись)
(Фамилия, И., О.)

К з а щ и т е д о п у с т и т ь
Зав. кафедрой Васильев В. Н. (подпись)
(Фамилия, И., О.)

“____” 2013 г.

Санкт-Петербург, 2013 г.

Магистерская диссертация выполнена с оценкой _____

Дата защиты “____”_____ 2013 г.

Секретарь ГАК _____

Листов хранения _____

Чертежей хранения _____

ВВЕДЕНИЕ

В последнее время проводится все больше исследований в области *поисковой инженерии программного обеспечения* (*search-based software engineering*) [14]. В этом подходе методы поисковой оптимизации применяются для автоматизированного построения программ для прикладных задач. Одним из классов таких задач является управление объектами со сложным поведением. Для таких задач весьма эффективно автоматное программирование [33].

Основными преимуществами автоматного программирования являются возможность автоматической генерации кода по автомату и возможность верификации автоматных программ с помощью метода *Model Checking* [30]. В некоторых случаях автоматы для программ этого класса удается построить эвристически, однако для многих задач ручное построение автоматов затруднительно или невозможно. Поэтому подход, в котором автоматы для программ рассматриваемого класса строятся методами поисковой оптимизации, является весьма актуальным. Основными методами поисковой оптимизации, применяемыми при построении конечных автоматов, являются эволюционные алгоритмы [31] – генетические алгоритмы [22, 27, 34, 36] и эволюционные стратегии [2, 19, 21].

В последнее время широкое развитие получили так называемые алгоритмы *роевого интеллекта* [3] (*swarm intelligence*), моделирующие поведение коллективов живых существ для решения комбинаторных задач. Подклассом алгоритмов роевого интеллекта являются *муравьиные алгоритмы* [4, 9–12, 25, 26], в основном применяющиеся для решения задач оптимизации на графах.

Целью настоящей работы является разработка методов построения конечных автоматов, основанных на муравьиных алгоритмах. Для этого решаются следующие задачи:

1. Исследование возможности использования классического сведения задачи построения автоматов к оптимизации на графе и применения классических муравьиных алгоритмов для ее решения.
2. Разработка альтернативных способов сведения задачи построения автоматов к оптимизации на графе.
3. Разработка муравьиного алгоритма нового типа для решения задачи построения автоматов с альтернативным сведением.
4. Экспериментальное сравнение характеристик производительности разработанных методов с известными подходами.

Основным результатом работы является новый эффективный метод построения конечных автоматов, основанный на муравьином алгоритме. Важнейшими компонентами предложенного метода являются представление пространства поиска в виде ориентированного графа специального вида и муравьиный алгоритм нового типа, применяющийся для поиска решений в этом графе. Экспериментальные исследования предложенного метода, проведенные для нескольких известных задач, подтверждают его эффективность. Новизна предложенного подхода заключается в том, что ни один из методов роевого интеллекта до сих пор не применялся для построения конечных автоматов.

В главе 1 приводится обзор существующих подходов к построению конечных автоматов. Даются определения рассматриваемых типов автоматов. Приводится общая формулировка задачи построения конечных автоматов по заданной функции приспособленности, рассматриваются известные метаэвристические алгоритмы. Обозреваются работы, посвященные построению конечных автоматов.

Глава 2 содержит подробные описания предлагаемых методов построения конечных автоматов, основанных на муравьиных алгоритмах. Разработанные методы основаны на различных способах сведения задачи построения автоматов к оптимизации на графе.

В главе 3 описываются задачи, на которых проводились экспериментальные исследования разработанных методов: задача об «Умном

муравье» и задача о построении автоматов по тестовым примерам. Приводятся результаты экспериментального сравнения разработанных методов с известными подходами.

ГЛАВА 1. МЕТАЭВРИСТИЧЕСКИЕ МЕТОДЫ ПОСТРОЕНИЯ КОНЕЧНЫХ АВТОМАТОВ

В данной главе приводится обзор метаэвристических методов построения конечных автоматов. В разд. 1.1 даются определения основных типов конечных автоматов. В разд. 1.2 в общем виде формулируется задача построения автоматов по заданной функции приспособленности. В разд. 1.3 приводится обзор основных метаэвристических алгоритмов, а в разд. 1.4 дается обзор статей, посвященных задачам построения конечных автоматов различных типов.

1.1. Конечные автоматы

В настоящем разделе вводятся определения типов автоматов, рассматриваемых в диссертации. Приводятся определения конечного автомата-преобразователя, управляющего конечного автомата и автомата-распознавателя. Также описывается используемый способ представления автоматов.

1.1.1. Конечные автоматы-преобразователи

Конечный автомат-преобразователь – это шестерка $\langle S, \Sigma, \Delta, \delta, \lambda, s_0 \rangle$, где S – множество состояний, Σ – множество входных событий, Δ – множество выходных воздействий, $\delta: S \times \Sigma \to S$ – функция переходов, $\lambda: S \times \Sigma \to \Delta$ – функция выходов, а $s_0 \in S$ – стартовое состояние.

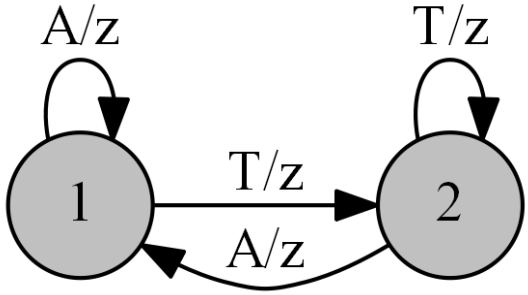

Рис. 1. Пример конечного автомата-преобразователя с двумя состояниями, множеством входных событий Σ = {*T*, *A*}, множеством выходов Δ = {*z*} и стартовым состоянием $s_0 = 1$

Для представления автоматов-преобразователей в данной работе используются полные таблицы переходов и выходов. Пример конечного автомата-преобразователя с двумя состояниями приведен на рис. 1. В табл. 1, 2 приведены примеры таблиц переходов и выходов автомата, представленного на рис. 1.

Таблица 1. Пример таблицы переходов автомата из рис. 1

δ	**Событие**	
Состояние	***A***	***T***
1	**1**	**2**
2	**1**	**2**

Таблица 2. Пример таблицы выходов автомата из рис. 1

λ	**Событие**	
Состояние	***A***	***T***
1	***z***	***z***
2	***z***	***z***

1.1.2. Управляющие конечные автоматы

Управляющий конечный автомат отличается от автомата-преобразователя наличием дополнительных пометок на переходах – булевых формул от логических входных переменных. Он определяется как семерка $\langle S, X, \Sigma, \Delta, \delta, \lambda, s_0 \rangle$, где S – множество состояний, X – множество булевых входных переменных, Σ – множество входных событий, Δ – множество выходных воздействий, δ: $S \times \Sigma \times 2^X \to S$ – функция переходов, λ: $S \times \Sigma \times 2^X \to \Delta^*$ – функция выходов, а $s_0 \in S$ – стартовое состояние.

Пример управляющего конечного автомата приведен на рис. 2. В силу специфики рассматриваемой в диссертации задачи построения управляющих конечных автоматов, используется упрощенный способ их представления, похожий на способ представления автоматов-преобразователей. Этот способ представления управляющих автоматов описан в разд. 3.2.

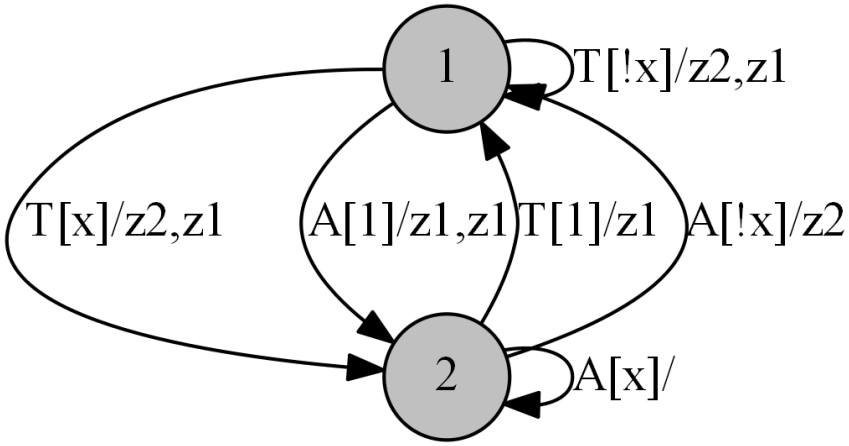

Рис. 2. Пример управляющего конечного автомата

1.1.3. Конечные автоматы-распознаватели

Конечный автомат-распознаватель – это пятерка $\langle S, F, \Sigma, \delta, s_0 \rangle$, где $F \subset S$ – множество *допускающих* состояний. Таким образом, автомат-распознаватель отличается от конечного автомата-преобразователя, определенного выше, тем, что, с одной стороны, не включает в себя функцию действий λ и множество действий Δ, а с другой стороны, включает в себя множество допускающих состояний *F*. Говорят, что автомат-распознаватель *принимает* строку $w \in \Sigma^*$, если после обработки последнего символа строки он переходит в допускающее состояние. В противном случае говорят, что автомат не принимает строку. Пример диаграммы переходов автомата-распознавателя приведен на рис. 3.

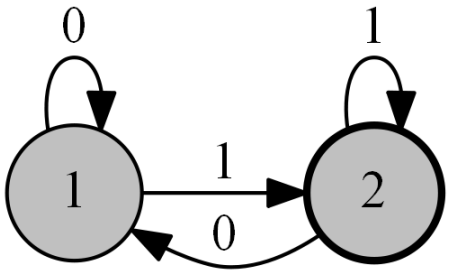

Рис. 3. Пример конечного автомата-распознавателя с Σ = {0, 1}. Стартовое состояние – первое, допускающее состояние выделено жирной рамкой

1.2. Общая формулировка задачи построения автоматов

При использовании метаэвристических алгоритмов для построения конечных автоматов вводится так называемая *функция приспособленности f* автомата (ФП). Эта вещественная функция определена на множестве всех конечных автоматов. Значения ФП пропорциональны близости наблюдаемого поведения автомата к желаемому.

Пусть задано число состояний N, множество входных событий Σ и множество выходных воздействий Δ целевого автомата. Пусть также задана вещественная функция приспособленности автомата f, определенная на множестве всех конечных автоматов с параметрами (N, Σ, Δ). Требуется найти автомат x с достаточно большим значением функции приспособленности $f(x) \geq f_b$.

1.3. Эвристические и метаэвристические алгоритмы

При решении сложных комбинаторных задач встает вопрос быстродействия алгоритмов. Так, для *NP*-трудных задач точные алгоритмы, гарантированно находящие оптимальное решение задачи, будут работать, в худшем случае, за экспоненциальное от размера входных данных время. Поэтому, при решении таких задач часто приходится пользоваться приближенными алгоритмами – алгоритмами, находящими решения, близкие к оптимальным, за сравнительно небольшое время. Такие приближенные алгоритмы часто называют *эвристиками*.

Основной проблемой при применении эвристических алгоритмов является проблема выхода из локальных оптимумов. Для преодоления этой проблемы были изобретены метаэвристические алгоритмы или *метаэвристики* – эвристики высокого уровня, направляющие работу проблемно-ориентированных эвристик в более выгодные области пространства поиска. Двумя основными классами метаэвристик являются эволюционные алгоритмы [31] и методы роевого интеллекта [3].

1.3.1. Эволюционные алгоритмы

Эволюционные алгоритмы [31] это метаэвристики, принципы работы которых заимствованы из законов эволюции живых существ. Основным объектом в эволюционных алгоритмах является популяция особей. Каждая особь представляет собой определенное решение задачи. Размер популяции (число особей) может быть как постоянным, так и изменяться по какому-либо закону. В зависимости от типа эволюционного алгоритма, к популяции и отдельным ее особям применяются те или иные *генетические операторы*: мутация, скрещивание и селекция.

Оператор мутации применяется к отдельным особям и изменяет один или несколько частей представления особи – генов. Оператор скрещивания применяется к двум особям и выдает на выход также две особи. Принцип его работы состоит в том, что родительские особи обмениваются генами таким образом, что каждая дочерняя особь содержит часть генов каждой из родительских особей. Оператор селекции применяется к популяции в целом и определяет, какие особи из текущей популяции получат возможность скрещиваться или напрямую перейти в следующее поколение. Основными типами эволюционных алгоритмов являются эволюционные стратегии и генетические алгоритмы, которые рассматриваются ниже.

1.3.1.1. Эволюционные стратегии

Эволюционные стратегии [2] (ЭС) отличаются от других эволюционных алгоритмов тем, что в них не используется оператор скрещивания. Выделяют два типа эволюционных стратегий: (μ, λ)-ЭС и $(\mu + \lambda)$-ЭС. В обоих типах стратегий популяция состоит из μ особей. В (μ, λ)-ЭС в новую популяцию переходят μ лучших из λ вновь сгенерированных особей. В $(\mu + \lambda)$-ЭС в новую популяцию переходят μ особей из λ вновь сгенерированных и μ особей текущей популяции.

Отдельно стоит выделить (1 + 1)-эволюционную стратегию как наиболее простой эволюционный алгоритм. В этом алгоритме популяция состоит из

единственной особи. На каждой итерации к текущей особи применяется оператор мутации. Текущая особь заменяется на новую, если значение ФП новой особи не меньше значения ФП текущей особи (в случае задачи максимизации ФП).

1.3.1.2. Генетические алгоритмы

Основным отличием генетических алгоритмов [22] от эволюционных стратегий является применение оператора скрещивания. На каждой итерации с помощью оператора селекции определяются особи, которые получают возможность скрещиваться. Особи, получившие право размножения, скрещиваются, и дочерние особи добавляются в новую популяцию. Когда новая популяция уже сформирована, к каждой особи с некоторой вероятностью применяется оператор мутации.

Существуют также другие, более сложные модели генетических алгоритмов, например, островная модель. В этой модели популяция разделена на несколько частей-островов, каждая из которых развивается независимо от других. Раз в несколько итераций происходит миграция особей между островами.

1.3.2. Методы роевого интеллекта: муравьиные алгоритмы

Методы роевого интеллекта [3] (*swarm intelligence*) применяют принципы самоорганизации коллективов живых существ для решения комбинаторных задач. Коллектив обычно состоит из относительно простых особей-агентов. Процессы самоорганизации приводят к тому, что интеллект коллектива существенно превосходит интеллект отдельных агентов. В данном разделе рассматривается один из классов методов роевого интеллекта – муравьиные алгоритмы [4, 9–12, 25, 26]. Примерами других методов роевого интеллекта являются: метод оптимизации роем частиц [17], пчелиные алгоритмы [23], светлячковые алгоритмы [29].

Муравьиные алгоритмы – семейство алгоритмов оптимизации, основанных на поведении колоний муравьев. Первый муравьиный алгоритм

был разработан Марко Дориго (Marco Dorigo) и применялся для решения задачи о коммивояжере [9]. Муравьиные алгоритмы успешно применяются для решения таких сложных комбинаторных задач, как, например, задача о рюкзаке, задача об упаковке в контейнеры, квадратичная задача о назначениях [12].

Опишем классический способ решения комбинаторных задач с помощью муравьиных алгоритмов [12]. Пусть некая комбинаторная задача представлена в виде тройки $(\hat{S}, \hat{f}, \Omega)$, где $\hat{S}$ – множество решений-кандидатов, $\hat{f}$ – целевая функция и Ω – множество ограничений, определяющее допустимые решения. Задача состоит в нахождении глобально оптимального допустимого решения s^*.

Описанная задача следующим образом сводится к задаче, которую можно решить с помощью муравьиного алгоритма. Пусть $C = \{c_1, \ldots, c_K\}$ – конечное множество компонентов, а $X = \left\{ x = \left\langle c_{i_1}, c_{i_2}, \ldots, c_{i_h}, \ldots \right\rangle, |x| < +\infty \right\}$ – множество состояний задачи. Множество допустимых состояний $\tilde{X} \subset X$ определяется как множество состояний $x \in X$, из которых может быть составлено решение-кандидат, удовлетворяющее ограничениям Ω. Множество допустимых решений есть $\tilde{S} = \tilde{X} \bigcap \hat{S}$, а множество оптимальных допустимых решений S^* является подмножеством множества $\tilde{S}$.

Далее вводится так называемый *граф конструирования* (*construction graph*) $G_C = (C, L)$, вершинами которого являются компоненты задачи C. Граф G_C полон, то есть каждая пара вершин из C соединена ребром из L. Таким образом, исходная задача сводится к поиску пути в графе G_C, обладающего максимальным значением функции $\hat{f}$, удовлетворяющего ограничениям Ω и соответствующего допустимому решению.

Каждому ребру (u, v) графа (u и v – вершины графа) ставится в соответствие число τ_{uv}, называемое *значением феромона*. Также на ребре может быть задано число η_{uv}, называемое *эвристической информацией*. Различие между этими двумя величинами состоит в том, что эвристическая информация

задается изначально, исходя из условий задачи, и не меняется во время выполнения алгоритма, в то время как значения феромона изменяются агентами-муравьями в процессе построения решений. Общая схема любого муравьиного алгоритма состоит из трех последовательных этапов, которые повторяются, пока не будет найдено решение или не выполнится какой-либо критерий останова. Примером такого критерия останова является исчерпание выделенных алгоритму вычислительных ресурсов.

На первом этапе, называемом `ConstructAntSolutions`, колония муравьев строит решения. В начале этого этапа каждый муравей помещается в некоторую вершину графа. Размещение муравьев по начальным вершинам обычно зависит от конкретной задачи. Каждый муравей добавляет компоненты в свое текущее решение, переходя по вершинам графа до тех пор, пока он не построит полное решение. Выбор следующей вершины осуществляется с помощью некоторого вероятностного алгоритма.

На этапе `UpdatePheromones` выполняется обновление значений феромона на ребрах графа. Значение феромона на ребре графа может увеличиться, если ребро вошло в путь некоторого муравья, либо уменьшиться вследствие *испарения феромона* – пропорционального уменьшения значений феромона на всех ребрах графа.

На третьем, необязательном этапе `DaemonActions` выполняются операции, которые не могут быть выполнены отдельными муравьями. Например, могут производиться локальные оптимизации найденных на данной итерации решений.

Конкретные реализации трех описанных этапов могут различаться для различных муравьиных алгоритмов. Примерами реализаций муравьиных алгоритмов являются *Ant Colony System* [10], *MAX MIN Ant System* [25], *Rank-based Ant System* [4] и *Elitist Ant System* [9].

1.4. Методы построения конечных автоматов

В данном разделе приводится обзор известных методов построения конечных автоматов. В основе большинства методов лежат эволюционные алгоритмы. Также иногда применяются обучение с подкреплением и метод имитации отжига. Для некоторых типов задач существуют методы, решающие задачу построения автоматов путем сведения ее к задаче о выполнимости булевой формулы (*SAT*). Как показывает проведенный обзор, ни один из методов роевого интеллекта до сих пор не применялся для построения конечных автоматов.

1.4.1. Построение конечных автоматов с помощью эволюционных стратегий

Известны применения эволюционных стратегий для построения таких типов автоматов, как автоматы-распознаватели и автоматы-преобразователи. Так, в работе [21] (1 + 1)-ЭС использовалась для построения автоматов-распознавателей по набору тестовых примеров. Для представления автоматов использовались полные таблицы переходов. Особенностью этой работы является то, что пометки состояний автомата не хранились и не строились эволюционным алгоритмом. Вместо этого был предложен алгоритм, позволяющий оптимальным образом расставить эти пометки исходя из обучающих примеров. Эволюционная стратегия сравнивалась с лучшим на тот момент алгоритмом слияния состояний [19]. Было показано, что эволюционная стратегия эффективнее слияния состояний, когда число состояний автомата невелико.

Автор [13] использует алгоритм расстановки пометок, предложенный в [21], однако в качестве метода поисковой оптимизации предлагает новый гибридный алгоритм, совмещающий постепенное обучение (*incremental learning*) и эволюционный алгоритм. Предлагается перед началом работы отсортировать строки обучающего набора по возрастанию длины. Обучающий набор разделяется на заданное число блоков. Основная идея состоит в том, что

на начальных итерациях автоматы обучаются на первом блоке, а остальные блоки добавляются в активный обучающий набор по мере достижения идеальной приспособленности на уже включенных блоках. Этот подход позволил получать чуть менее качественные автоматы, чем в работе [21], но в несколько раз быстрее.

В работе [20] эволюционные стратегии использовались для построения автоматов-преобразователей по тестовым примерам. Исследовалось применение функций приспособленности на основе полного совпадений строк, расстояния Хэмминга и расстояния Левенштейна. Было показано, что функция приспособленности на основе расстояний Левенштейна обеспечивает наибольшую эффективность эволюционного алгоритма.

1.4.2. Построение конечных автоматов с помощью генетических алгоритмов

Множество работ, посвященных построению различных типов конечных автоматов, было выполнено студентами и сотрудниками кафедры «Компьютерные технологии» НИУ ИТМО. В работе [32] генетические алгоритмы применялись для построения конечных автоматов, решающих задачу о флибах. Эта задача состоит в построении автомата-преобразователя, предсказывающего состояние некой среды, заданное битовой маской определенной длины. Идеальный автомат, получив на вход любой бит этой маски, должен выдать на выход следующий бит.

Генетические алгоритмы также применялись при построении автоматов-преобразователей для задачи об «Умном муравье» [16] в работе [36]. В указанной работе был разработан специальный оператор скрещивания, учитывающий поведение автомата при выполнении первых сорока ходов. В англоязычной литературе также встречаются примеры использования генетических алгоритмов для построения автоматов в задаче об «Умном муравье» (*Artificial Ant problem*) [6, 18].

Авторы [24] применяют генетический алгоритм для построения автоматов в задаче о завоевании ресурсов (*Competition for Resources*). Функция приспособленности в этой задаче основана на моделировании поведения агента в некоторой игре. Рассматривалась задача построения конечно-автоматной системы управления агентом для игры против псевдослучайной стратегии. Одним из интересных результатов этого исследования является то, что конечно-автоматной модели в чистом виде не хватило для достаточно успешной игры агента. Для повышения эффективности было предложено добавить к модели несколько ячеек памяти, которые использовались для предотвращения нежелательного циклического поведения агента.

В работах [1, 34] генетические алгоритмы применяются для решения значительно более сложной задачи построения управляющих конечных автоматов для управления беспилотным летательным аппаратом. В работе [34] используется функция приспособленности, основанная на моделировании, в связи с чем построение автомата требует значительных вычислительных затрат. В работе [1] предлагается строить автоматы для автопилота по тестовым примерам поведения, которые записываются экспертом. Этот подход в значительной мере ускорил построение автоматов.

Авторы [27] применяют генетические алгоритмы для построения автоматов на основе тестовых примеров и верификации. Предлагается специализированный оператор скрещивания автоматов, учитывающий их поведение на тестовых примерах.

1.4.3. Методы построения конечных автоматов на основе сценариев работы

Помимо работ, описанных в предыдущем разделе, в которых для построения автоматов по тестовым примерам используются генетические алгоритмы, существуют работы, в которых предлагаются специализированные методы построения различных классов автоматов по сценариям работы. Сценарии работы отличаются от тестовых примеров тем, что для каждого

входного события (и охранного условия) известны конкретные выходные воздействия. Эти методы основаны на сведении задачи построения автоматов к задаче выполнимости булевой формулы (*SAT*).

Так, в работе [15] по сценариям работы строятся автоматы-распознаватели, а в работе [28], выполненной на кафедре «Компьютерные технологии», по сценариям работы строятся управляющие конечные автоматы. В основе этих методов лежит построение по сценариям префиксного дерева сценариев и дальнейшая раскраска этого дерева в заданное число цветов. Число цветов, в которое раскрашивается дерево сценариев, соответствует числу состояний автомата. Для определения того, в какие цвета раскрасить вершины дерева, применяется сведение к задаче *SAT*. Для решения экземпляров этой задачи применяются так называемые *SAT*-солверы.

Как показывается в указанных работах, скорость работы этих методов на несколько порядков превышает скорость работы методов, основанных на эвристическом слиянии состояний. Однако нельзя забывать, что эти методы позволяют решать лишь задачу построения автоматов по сценариям и неприменимы в случае функции приспособленности, основанной на моделировании.

Выводы по главе 1

1. Существует множество методов построения конечных автоматов различных типов, основанных на эволюционных стратегиях и генетических алгоритмах.
2. Для задач построения автоматов по сценариям работы разработаны эффективные методы, основанные на сведении задачи построения автомата к задаче *SAT*. Развитие метаэвристических алгоритмов для решения этих типов задач не представляется перспективным направлением исследований.
3. Методы роевого интеллекта, в частности, муравьиные алгоритмы, весьма эффективны для решения многих *NP*-полных задач. Это направление в науке находится в стадии бурного развития, что подтверждается большим числом публикаций по этой тематике.
4. В ходе проведения аналитического обзора не было обнаружено ни одного метода построения конечных автоматов, основанного на методах роевого интеллекта, в том числе на муравьиных алгоритмах.

ГЛАВА 2. МЕТОДЫ ПОСТРОЕНИЯ КОНЕЧНЫХ АВТОМАТОВ С ПОМОЩЬЮ МУРАВЬИНЫХ АЛГОРИТМОВ

В данной главе приводится описание разработанных в настоящей работе методов построения конечных автоматов, основанных на применении муравьиных алгоритмов.

2.1. Метод на основе классического сведения

Применительно к построению конечных автоматов-преобразователей, классическое сведение этой задачи к виду, приемлемому для применения муравьиного алгоритма, выглядит следующим образом. Множество решений-кандидатов $\hat{S}$ есть множество всех конечных автоматов с параметрами (N, Σ, Δ), где N – число состояний, Σ – множество входных событий а Δ – множество выходных воздействий. Здесь целевая функция $\hat{f}$ есть функция приспособленности автомата *f*. Множество компонентов *C* это множество всех возможных переходов автоматов с параметрами (N, Σ, Δ):

$$C = \{t = \langle i, j, e, a \rangle, i \in S, j \in S, e \in \Sigma, a \in \Delta\},$$

где $i \in S$ – стартовое состояние перехода, $j \in S$ – конечное состояние перехода, $e \in \Sigma$ – событие, по которому выполняется переход и $a \in \Delta$ – выходное воздействие, выполняющееся на переходе. Пример графа конструирования приведен на рис. 4. В приведенном примере граф конструирования был построен для автоматов с двумя состояниями, множество событий Σ состоит и двух элементов *x* и !*x*, а множество выходных воздействий Δ состоит из единственного элемента *z*.

Ограничения Ω описывают то, что конечный автомат должен быть детерминированным, т.е. из каждого состояния по каждому входному событию в автомате должен существовать ровно один переход. Для реализации ограничений *k*-й муравей использует свою внутреннюю память для хранения множества посещенных им компонентов L_t^k. Пусть *k*-й муравей располагается в вершине $u \in S$. Обозначим множество вершин, инцидентных *u*, как N_u. Перед

выбором следующей вершины муравей сканирует множества N_u и L_t^k, формируя множество компонентов $\hat{N}_u$. Это множество $\hat{N}_u$ содержит только такие компоненты $t \in \hat{N}_u$, что для любого стартового состояния $y \in S$ и входного события $\varepsilon \in \Sigma$ множество посещенных компонентов L_t^k не содержит переходов с тем же стартовым состоянием и входным событием. Следующая вершина v выбирается k-м муравьем из множества $\hat{N}_u$ с вероятностью, вычисляемой по формуле:

$$p_v = \frac{\tau_{uv}^{\alpha}}{\sum_{w \in \hat{N}_u} \tau_{uw}^{\alpha}},$$

где $\alpha \in (0,+\infty)$. Эта формула не учитывает эвристическую информацию, поскольку не было найдено значимого способа ее определения. Сплошные ребра на рис. 4 обозначают путь муравья, построившего автомат, изображенный на рис. 1.

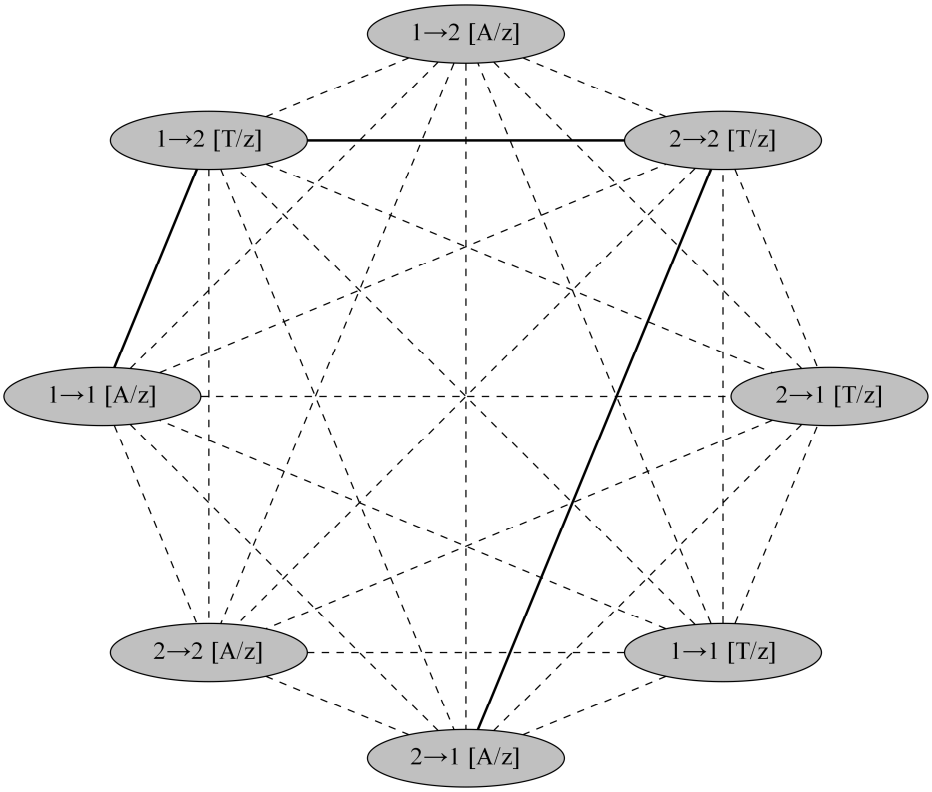

Рис. 4. Пример графа конструирования для построения автоматов с помощью классических муравьиных алгоритмов

В качестве конкретных муравьиных алгоритмов предлагается использовать *Elitist Ant System* (EAS) [9] и $ACO_{bs,\tau_{\min}}$ [26]. В алгоритме EAS лучшее найденное решение s^{best} откладывает феромон наряду с решениями, найденными на текущей итерации. Значения феромона обновляются по формуле:

$$\tau_{uv} = (1-\rho)\tau_{uv} + \sum_{k=1}^{N_{\text{ants}}} \Delta\tau_{uv}^{k} + e \cdot \Delta\tau_{uv}^{\text{best}},$$

где параметр $e \in [0,1]$ определяет влияние элитарного решения, а

$$\Delta\tau_{uv}^{k} = \begin{cases} f(s_k), \text{если } (u,v) \in L_t^k \\ 0, \text{иначе} \end{cases} \quad \text{и} \quad \Delta\tau_{uv}^{\text{best}} = \begin{cases} f(s^{\text{best}}), \text{если } (u,v) \in L_t^{\text{best}} \\ 0, \text{иначе} \end{cases}.$$

Здесь, s_l – решение, найденное l-м муравьем на текущей итерации, а L_t^{best} – множество ребер, посещенных лучшим муравьем. Процедура обновления феромона в $ACO_{bs,\tau_{\min}}$ аналогична процедуре обновления феромона в EAS, с тем исключением, что только лучший муравей откладывает феромон и e = 1. В обоих алгоритмах значения феромона поддерживаются не ниже минимального значения $\tau_{\min}$.

Описанные алгоритмы отличаются тем, что для них доказана *сходимость в значениях* [12]. Сходимость в значениях означает, что при наличии достаточного количества ресурсов алгоритм гарантированно найдет оптимальное решение, если оно существует.

2.2. Метод на основе полного недетерминированного конечного автомата

Рассмотрим представление конечного автомата в виде графа переходов. При этом подходе состояния автомата соответствуют вершинам графа, а переходы – его ребрам. Любой автомат из множества всех автоматов с параметрами (N, Σ, Δ) как граф является подграфом *полного недетерминированного конечного автомата* (НКА) с теми же параметрами. Под полным НКА понимается автомат, в котором из каждого из N состояний по

каждому входному событию из Σ существует переход в каждое состояние с каждым выходным воздействием из Δ. Пример такого автомата с двумя состояниями при Σ = {*T*, A} и Δ = {z_1, z_2} приведен на рис. 5.

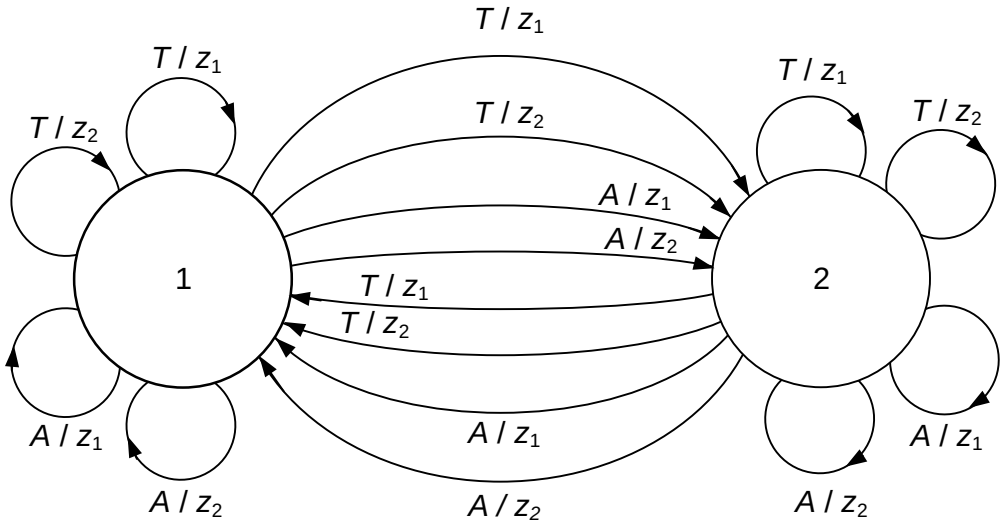

Рис. 5. Пример полного недетерминированного конечного автомата с двумя состояниями, Σ = {*T*, *A*}, Δ = {z_1, z_2}

Задача построения автоматов может быть переформулирована следующим образом. Задано множество входных событий Σ, множество выходных воздействий Δ и число состояний *N*. На полном НКА с параметрами (*N*, Σ, Δ) требуется найти такой путь, что детерминированный конечный автомат, составленный из ребер и вершин найденного пути, имеет достаточно большое значение функции приспособленности.

В начале работы алгоритма всем ребрам полного НКА сопоставляется некоторое одинаковое положительное значение феромона τ_0. В каждую вершину полного НКА помещается по муравью. На этапе построения решений муравьями каждый из них выбирает переходы из своего состояния по каждому входному символу, исходя из значений феромона на переходах полного НКА. Так, пусть *k*-й муравей находится в состоянии *k* полного недетерминированного автомата. Муравей перебирает все входные события $e \in \Sigma$. Для каждого входного события *e* переход выбирается из множества $N_{k,e}$ всех возможных

переходов из состояния k по событию e. Переход $t \in N_{k,e}$ выбирается с вероятностью:

$$p_t = \frac{\tau_t^\alpha}{\sum_{w \in N_{k,e}} \tau_w^\alpha},$$

где $\alpha \in (0,+\infty)$. Заметим, что в отличие от классического муравьиного алгоритма, описанного в разд. 1.3.2, каждый муравей строит переходы только из своего состояния, не перемещаясь в другие состояния (вершины). По выбранным муравьями состояниям и переходам строится детерминированный конечный автомат. Удовлетворение ограничениям детерминированности автомата вытекает из алгоритма работы муравьев.

Далее вычисляется значение функции приспособленности построенного автомата. При этом запоминаются переходы, которые он совершал. К весам посещенных при вычислении функции приспособленности переходов НКА добавляются дополнительные веса, пропорциональные вычисленному значению. Отметим, что веса тех переходов, которые не были посещены при вычислении функции приспособленности, не изменяются, а сами переходы удаляются из ДКА. Также из ДКА удаляются непосещенные состояния.

На следующем этапе со всех ребер НКА испаряется феромон – веса всех ребер уменьшаются в одинаковое число раз. Наконец, проверяются условия сходимости (достижения требуемого значения функции приспособленности) и стагнации. Под стагнацией понимается ситуация, при которой значение функции приспособленности лучшего автомата не увеличивается в течение некоторого фиксированного промежутка времени (числа шагов алгоритма). В таком случае алгоритм перезапускается. При достижении требуемого значения функции приспособленности алгоритм прекращает свою работу, выдавая лучший из построенных автоматов.

2.3. Метод на основе графа мутаций автоматов

В данном разделе приводится описание третьего из разработанных в рамках диссертации методов построения конечных автоматов. Метод основан на альтернативном сведении задачи построении автомата к оптимизации на графе. Центральным понятием здесь является понятие мутации конечного автомата, описываемое в разд. 2.3.1. Природа так называемого *графа мутаций*, в котором происходит поиск решений, такова, что для решения задачи оптимизации на этом графе невозможно применить классические муравьиные алгоритмы. В связи с этим был разработан муравьиный алгоритм нового типа, описание которого дается в разд. 2.3.3 – 2.3.7.

Отметим, что разработанный муравьиный алгоритм нового типа может быть применен и для решения других комбинаторных задач помимо построения автоматов. Для применения разработанного метода к решению некоторой комбинаторной задачи требуется лишь задать способ представления решения задачи и определить набор операторов мутации. Примеры операторов мутации для некоторых комбинаторных задач приведены в разд. 2.3.8.

2.3.1. Операторы мутации конечных автоматов

Под *мутацией* конечного автомата понимается небольшое изменение его структуры – таблиц, задающих функции переходов и выходов. В данной работе используются следующие два типа мутаций конечных автоматов. Эти способы выполнения мутации гарантируют, что одна мутация вызывает ровно одно изменение в автомате.

1. Изменение состояния, в которое ведет переход. Для случайно выбранного перехода в автомате состояние s, в которое он ведет, заменяется другим состоянием, выбранным случайным образом из множества $S \setminus \{s\}$.
2. Изменение действия на переходе. Действие a на случайно выбранном переходе заменяется на другое действие, выбранное случайным образом из множества $\Delta \setminus \{a\}$.

Выбранный набор операторов мутации может быть при необходимости дополнен другими операторами. Также существует возможность применения нескольких мутаций на одном ребре графа мутаций.

2.3.2. Представление пространства поиска в виде графа специального вида

Основой предлагаемого метода построения конечных автоматов является представление пространства поиска – множества всех автоматов с заданными параметрами – в виде ориентированного графа G. Этот граф, который мы будем называть *графом мутаций*, имеет следующие свойства.

1. Каждая вершина графа G ассоциирована с конечным автоматом.
2. Пусть u – вершина, ассоциированная с автоматом A_1, а v – вершина, ассоциированная с автоматом A_2. Тогда, если автомат A_1 может быть получен из автомата A_2 применением одной операции мутации, то граф G содержит ребра $u \to v$ и $v \to u$. В противном случае, вершины u и v не связаны ребром.
3. Таким образом, для любой пары автоматов A_1 и A_2 и соответствующей пары вершин u и v в графе G существует путь из u в v и из v в u.

Небольшой пример графа мутаций приведен на рис. 6. Каждое ребро графа помечено мутацией, записанной в нотации, описанной ниже. Значение сплошных и штрихованных ребер будет объяснено позднее в разд. 2.3.5.

- Изменение состояния, в которое ведет переход. Перед стрелкой записаны состояние и событие, определяющие переход, а после стрелки – новое состояние, в которое ведет этот переход после мутации:

  Tr: (состояние, событие) → «новое состояние».

- Изменение действия на переходе. Перед стрелкой записаны состояние и событие, определяющие переход, а после стрелки – новое выходное воздействие, выполняющееся на этом переходе после мутации.

  Out: (состояние, событие) → «новое выходное воздействие».

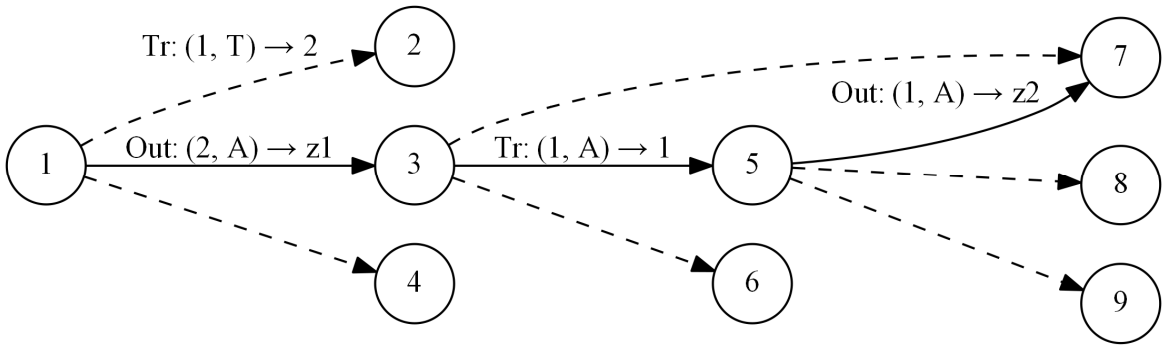

Рис. 6. Пример графа мутаций. Здесь *A* и *T* – входные события, z1 и z2 – выходные воздействия

2.3.3. Эвристическая информация на ребрах графа

На каждом ребре (u, v) графа предлагается задать значение эвристической информации:

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

где $\eta_{\min}$ – небольшая положительная константа, например, 10^{-3}. Максимум разности значений ФП и положительной константы берется для того, чтобы значения эвристической информации всегда были положительными, что необходимо для применения вероятностной формулы выбора пути, о которой пойдет речь в разд. 2.3.5.

2.3.4. Построение начального решения

В начале работы алгоритма строится некоторое начальное решение, которое добавляется в граф и становится его первой вершиной. При построении этого решения сначала строится случайный конечный автомат с заданным числом состояний *N*, таблицы переходов и выходов которого заполняются случайным образом. Далее, случайно построенное решение улучшается с помощью применения простейшей (1+1)-эволюционной стратегии [21] в течение небольшого фиксированного числа итераций.

Эволюционная стратегия работает следующим образом: к текущему решению применяется оператор мутации, случайно выбранный из набора операторов, перечисленных в разд. 2.3.1, и вычисляется значение ФП измененного решения. Если значение ФП нового решения не меньше значения

ФП текущего решения, то текущее решение заменяется новым. Процедура повторяется до тех пор, пока не будет достигнуто максимальное число вычислений ФП, отведенное эволюционной стратегии.

2.3.5. Построение решений муравьями

Процедуру построения решений муравьями можно разделить на два этапа. На первом этапе выбираются вершины графа, из которых муравьи начнут построение путей. Экспериментально было установлено, что одним из наиболее эффективных правил выбора стартовых вершин является выбор вершины, ассоциированной с лучшим на момент выбора решением, в качестве единственной стартовой вершины для всех муравьев.

На втором этапе совершается одна итерация колонии, в ходе которой каждый муравей обходит граф, 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 следующих действий.

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 N_{mut} 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, создать ее и добавить в граф;
   - добавить в граф ребро (u, t).

   Описанная процедура выбора новой вершины путем построения новых решений проиллюстрирована на рис. 7. Отметим, что переход осуществляется даже в том случае, когда значение функции приспособленности у лучшего из построенных с помощью мутаций автоматов меньше значения функции приспособленности автомата A. Это

свойство алгоритма позволяет ему эффективно выходить из локальных оптимумов.

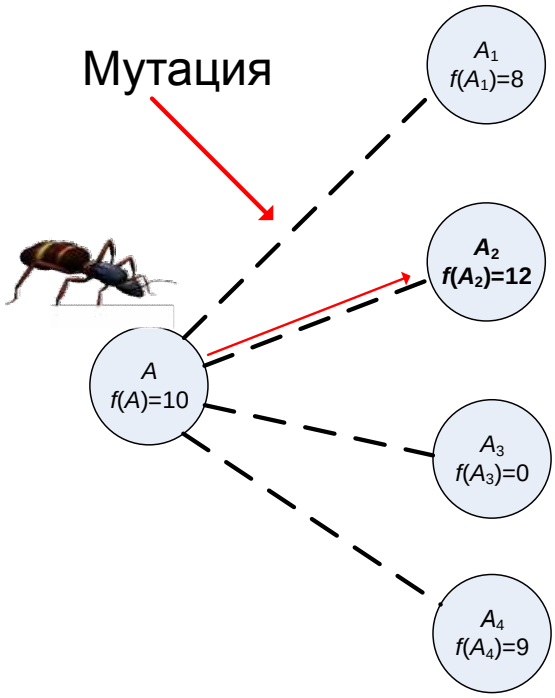

Рис. 7. Выбор новой вершины путем построения новых решений. Муравей производит несколько мутаций автомата A, получая автоматы A_1–A_4. В качестве следующей вершины выбирается вершина A_2 как соответствующая автомату с наибольшим значением функции приспособленности $f(A_4) = 12$

2. **Вероятностный выбор**. С вероятностью $1 - p_{\text{new}}$ муравей выбирает следующую вершину из множества N_u ребер, инцидентных вершине u. Вершина v выбирается с вероятностью, определяемой классической в муравьиных алгоритмах формулой:

$$p_{uv} = \frac{\tau_{uv}^{\alpha} \cdot \eta_{uv}^{\beta}}{\sum_{w \in N_u} \tau_{uw}^{\alpha} \cdot \eta_{uw}^{\beta}}, \alpha, \beta \in (0; +\infty).$$

Если у вершины u нет инцидентных ей ребер, то муравей всегда выполняет построение новых решений. Процедура выбора муравьем следующей вершины из числа существующих вершин проиллюстрирована на рис. 8.

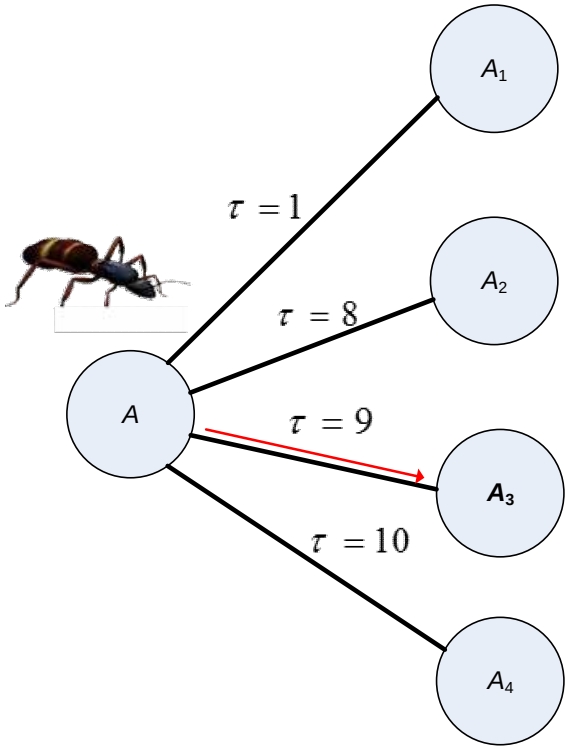

Рис. 8. Выбор новой вершины из числа существующих инцидентных вершин. В силу того, что формула выбора вершины является вероятностной, муравей может перейти не в вершину A_4 по ребру наибольшим значением феромона τ, а в вершину A_3

Все муравьи в колонии запускаются «параллельно» – каждый из них 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. После этого алгоритм перезапускается. Сплошные ребра графа мутаций на рис. 6 обозначают путь муравья, а штрихованные ребра – все остальные мутации, которые были выполнены муравьем.

2.3.6. Обновление значений феромона

Правило обновления значений феромона, применяемое в данной работе, основано на правиле, использующемся в алгоритмах *Elitist Ant System* [10] и *MAX MIN Ant System* [24]. 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 на этом ребре. Последовательно рассматриваются все пути муравьев на текущей итерации алгоритма. Для каждого пути муравья выделяется отрезок от начала пути до вершины, содержащей автомат с наибольшим на пути значением ФП. Для всех ребер из этого отрезка обновляются значения τ_{uv}^{best}, а затем значения феромона на всех ребрах графа *G* обновляются по формуле:

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

где $\tau_{\min} = 0{,}001$ – фиксированная нижняя граница значений феромона, а $\rho \in [0,1]$ – скорость испарения феромона. Введение нижней границы $\tau_{\min}$ исключает чрезмерное испарение феромона с ребер графа.

2.3.7. Основные отличия предложенного муравьиного алгоритма нового типа от классических муравьиных алгоритмов

Предложенный муравьиный алгоритм весьма существенно отличается от других муравьиных алгоритмов оптимизации, хотя и заимствует их основы. Во-первых, используемые в алгоритме графы могут быть чрезвычайно большими – для некоторых задач они могут содержать миллионы вершин. Такие графы невозможно хранить в оперативной памяти персональных компьютеров, поэтому в алгоритме используются специальные механизмы для ограничения размеров рассматриваемых подграфов этих графов.

Во-вторых, в большинстве случаев в муравьиных алгоритмах вершины графа являются компонентами решений, которые являются путями в этом графе. В предложенном алгоритме, наоборот, каждая вершина графа полностью представляет собой решение (конечный автомат), в то время как муравьи переходят от одного решения к другому в поисках оптимального. Эта особенность ведет к необходимости введения специального критерия остановки муравьев, отличающегося от простого критерия, используемого в классических муравьиных алгоритмах.

2.3.8. Применение муравьиного алгоритма на основе графа мутаций для решения других комбинаторных задач

Как уже отмечалось ранее во введении к разд. 2.3, разработанный метод на основе графа мутаций может быть применен не только для построения конечных автоматов, но и для решения других комбинаторных задач. Необходимыми требованиями являются задание способа представления решения комбинаторной задачи и определение набора операторов мутации.

В качестве первого простого примера рассмотрим задачу *OneMax*. В этой задаче решение представляется вектором из нулей и единиц, а функция приспособленности решения есть сумма элементов вектора. Таким образом, целью в этой задаче является нахождение вектора из всех единиц. Для применения предложенного метода на основе графа мутаций достаточно определить один оператор мутации, который будет выбирать случайный элемент вектора и инвертировать его – ноль заменять на единицу, и наоборот.

Более сложный пример – задача о коммивояжере. В этой задаче задан полный взвешенный граф, требуется найти гамильтонов путь минимального веса в этом графе. Решение задачи о коммивояжере может быть представлено в виде перестановки индексов вершин графа. Простейший оператор мутации случайным образом выбирает два элемента перестановки и меняет их местами.

Приведенные примеры демонстрируют теоретическую возможность применения разработанного метода к различным комбинаторным задачам. Экспериментальные исследования эффективности метода применительно к различным комбинаторным задачам выходят за рамки темы магистерской диссертации.

Выводы по главе 2

1. Предложен метод построения конечных автоматов, использующий классический для муравьиных алгоритмов способ сведения задачи к поиску оптимального пути в некотором полном графе.
2. Предложен метод построения конечных автоматов, в котором в качестве пространства поиска используется обобщение детерминированного конечного автомата – полный недетерминированный конечный автомат.
3. Предложен метод построения конечных автоматов, основанный на сведении этой задачи к поиску оптимальной вершины в графе специального вида и муравьином алгоритме нового типа. Разработанный метод является достаточно общим и может быть применен как к построению автоматов различных типов, так и к другим комбинаторным задачам, таким как, например, задача *OneMax* и задача о коммивояжере.

ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ ПРЕДЛОЖЕННЫХ МЕТОДОВ ПОСТРОЕНИЯ КОНЕЧНЫХ АВТОМАТОВ

В данной главе приводятся результаты экспериментальных исследований эффективности разработанных методов построения конечных автоматов. Сначала описываются задачи, для которых проводились экспериментальные сравнения. В последующих разделах приводятся результаты экспериментального сравнения разработанных методов с известными методами построения автоматов.

3.1. Задача об «Умном муравье»

Одной из стандартных задач, используемых для сравнения эффективности метаэвристических алгоритмов, является задача об «Умном муравье» [15]. В этой задаче задано тороидальное поле размером 32×32 клетки. На поле вдоль некоторой ломаной распределены «яблоки». В данной работе рассматриваются две конфигурации задачи – поле Санта-Фе и поле Джона Мура (рис. 9). Оба поля содержат по 89 клеток с яблоками. Черные клетки содержат яблоки, серые клетки обозначают пустые клетки оптимального пути, а белые клетки пусты.

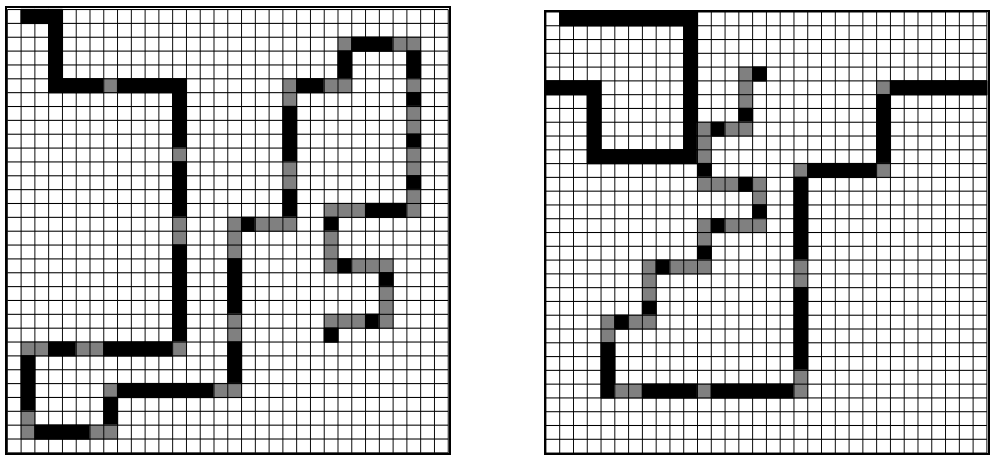

Рис. 9. Поле Санта-Фе (слева) и поле Джона Мура (справа)

Агент, называемый «умным муравьем», изначально располагается в левой верхней клетке и «смотрит» на восток. Находясь в некоторой клетке, он может определить, есть ли в следующей клетке яблоко. На каждом временном шаге

муравей может повернуть налево, повернуть направо или перейти вперед на одну клетку. Если в клетке, в которую переходит муравей, находится яблоко, муравей его «съедает». Число временных шагов s_{max}, доступных муравью, зависит от постановки эксперимента.

В описанной задаче целью является построение системы управления муравьем, которая позволит ему съесть все яблоки за отведенное число ходов. Используемая функция приспособленности имеет вид:

$$f = n_{food} + \frac{s_{max} - s_{last} - 1}{s_{max}},$$

где n_{food} – число яблок, съеденных муравьем, s_{max} – максимальное число ходов, предоставленных муравью и s_{last} – номер хода, на котором было съедено последнее яблоко.

Система управления муравьем строится в виде конечного автомата. У такого автомата есть два входных события – F (следующая клетка содержит яблоко), $!F$ (следующая клетка пуста) и три выходных воздействия: L (повернуть налево), R (повернуть направо) и M (сделать шаг вперед).

3.2. Задача построения конечных автоматов по обучающим примерам

Эта задача состоит в построении автомата с фиксированным поведением, заданным набором обучающих тестовых примеров (тестов) T. В рассматриваемой постановке задачи каждый тестовый пример задается двумя последовательностями: последовательностью входов $In[i]$ и соответствующей последовательностью выходов $Ans[i]$. Последовательность $In[i]$ составлена из элементов множества входных событий Σ, а последовательность выходов $Ans[i]$ – из элементов множества выходов Δ.

Говорят, что конечный автомат удовлетворяет тесту T_i={$In[i]$, $Ans[i]$}, если последовательность $Ans[i]$ является результатом передачи ему на вход последовательности $In[i]$. Задача состоит в нахождении автомата с заданным числом состояний, удовлетворяющего всем тестовым примерам.

Отметим, что в этой задаче выходные воздействия не хранятся в представлении автомата и не получаются с помощью алгоритма построения автоматов. Вместо этого используется так называемый алгоритм расстановки выходных воздействий на переходах [25]. Основная идея алгоритма состоит в использовании *каркаса* автомата. Каркасом автомата называется автомат, в котором вместо самих выходных воздействий на переходах записано число выходных воздействий. Пример каркаса автомата приведен на рис. 10.

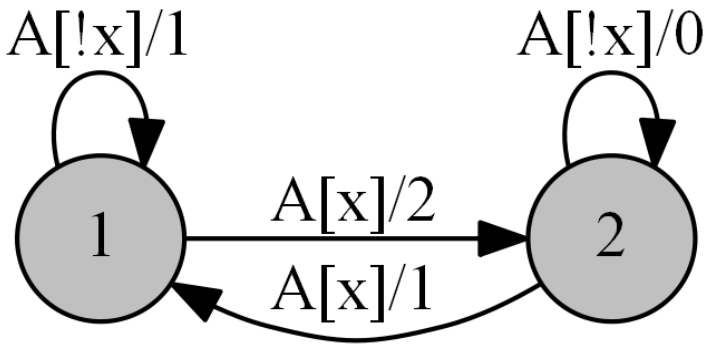

Рис. 10. Пример каркаса автомата

Алгоритм расстановки выходных воздействий принимает на вход каркас автомата и множество тестовых примеров. Результатом работы алгоритма является автомат, в котором выходные воздействия, расставлены оптимальным образом на основе тестов.

Используемая функция приспособленности основана на редакционном расстоянии между строками. Опишем способ вычисления значения функции приспособленности. Как уже упоминалось, сначала каркас автомата и тесты подаются на вход алгоритму расстановки выходных воздействий, который выдает конечный автомат. Далее, каждая входная последовательность $In[i]$ подается на вход построенному автомату, при этом запоминается выходная последовательность $Out[i]$. Полученная выходная последовательность сравнивается с эталонной последовательностью $Ans[i]$. Для оценки схожести полученных последовательностей вычисляется значение выражения:

$$f_1 = \frac{1}{|\mathrm{T}|}\sum_{i=1}^{|T|}\left(1-\frac{\mathrm{ED}(Out[i], Ans[i])}{\max(\mathrm{len}(Out[i]), \mathrm{len}(Ans[i]))}\right),$$

где $|T|$ – число тестовых примеров, len(s) – длина последовательности s, а ED(s_1, s_2) – редакционное расстояние между последовательностями s_1 и s_2. Окончательное выражение для функции приспособленности учитывает число переходов *transitionsCount* автомата:

$$f = \begin{cases} 10 \cdot f_1 + \dfrac{1}{M}\left(M - transitionsCount\right), \text{ если } f_1 < 1 \\ 20 + \dfrac{1}{M}\left(M - transitionsCount\right), \text{ если } f_1 = 1 \end{cases},$$

где M – число, гарантированно превосходящее максимально возможное число переходов автомата. Значения этой функции больше для автоматов, полностью удовлетворяющих всем тестам и имеющим меньшее число переходов и меньше для автоматов с большим числом переходов и не удовлетворяющих всем тестам.

3.3. Исследование эффективности метода на основе полного недетерминированного конечного автомата

Данное исследование было проведено в рамках предварительных исследований по тематике диссертации. Метод на основе полного недетерминированного конечного автомата сравнивался с методом, основанным на генетическом алгоритме [36]. Для сравнения методов рассматривалась задача об «Умном муравье» с полем Джона Мура, описанная в разд. 3.1. Как и в работе [36], было установлено ограничение на максимальное число ходов «умного» муравья $s_{\max} = 200$, а решения искались среди автоматов, содержащих семь состояний.

При использовании функции приспособленности, описанной в разд. 3.1 метод на основе полного автомата не нашел оптимального решения. Тогда была введена функция приспособленности, учитывающая число использованных при ее вычислении состояний автомата:

$$f = n_{\text{food}} + \frac{s_{\max} - s_{\text{last}} - 1}{s_{\max}} + 0{,}1 \cdot \left(N^{+}_{\text{states}} - N_{\text{states}}\right)$$

Здесь, N^{+}_{states} – число состояний, которые были посещены при вычислении значений функции приспособленности. Значения этой функции тем больше, чем меньше число посещенных состояний. При построении автоматов с этой функцией приспособленности решения искались среди автоматов с 12 состояниями. Алгоритм работал до тех пор, пока не был найден автомат, позволяющий муравью съесть все яблоки за отведенное число шагов и использующий при этом не более семи состояний.

Эксперимент был повторен 100 раз. Метод, основанный на полном автомате, позволил найти оптимальное решение в среднем за $24{,}49 \times 10^6$ вычислений ФП. Генетический алгоритм из работы [36] позволяет найти решение лишь за 220×10^6 вычислений ФП, то есть в 8,3 раза медленнее, чем муравьиный алгоритм.

Данный результат, однако, нельзя использовать для адекватной оценки эффективности муравьиного алгоритма на основе полного автомата. Он предлагает лишь новый, самый эффективный из существовавших на момент исследований, способ решения задачи об «Умном муравье», которая сама по себе не представляет практического интереса. Описанное предварительное исследование указало на необходимость разработки новых, более эффективных методов, основанных на муравьиных алгоритмах.

3.4. Сравнение метода на основе классического сведения и метода на основе графа мутаций

Для сравнения эффективности разработанных методов на основе классического сведения и на основе графа мутаций рассматривается задача об «Умном муравье» с полем Санта Фе, описанная в разд. 3.1. Настройка параметров алгоритмов проводилась с помощью полного факторного эксперимента для автоматов из пяти состояний и наибольшего числа шагов муравья $s_{\max} = 600$.

Экспериментальное сравнение алгоритмов проводилось для автоматов из $N_{\text{states}} = [5..10]$ состояний и $s_{\max} = 400$. Эксперимент для каждого числа

состояний был повторен 100 раз. Каждый эксперимент продолжался либо до достижения оптимального решения со значением ФП не менее 89, либо до исчерпания алгоритмом выделенных 30000 вычислений ФП. Для каждого числа состояний автомата подсчитывалось среднее по всем запускам время работы и доля запусков, в которых было найдено оптимальное решение. Результаты, представленные в табл. 3 позволяют утверждать, что метод, основанный на графе мутаций, существенно превосходит по производительности метод на основе классического сведения. Так, метод на основе графа мутаций находит оптимальное решение в 3-9 раз чаще метода на основе классического сведения, причем делает это в 30-400 раз быстрее.

Таблица 3. Результаты запусков методов на основе муравьиного алгоритма с классическим сведением и на основе графа мутаций

	Метод на основе классического сведения			**Метод на основе графа мутаций**		
N_{states}	**Среднее время, сек.**	**Среднее значение ФП**	**Доля успешных запусков, %**	**Среднее время, сек.**	**Среднее значение ФП**	**Доля успешных запусков, %**
5	18,09	64,5	18	0,65	85,98	87
6	28,97	74,9	30	0,54	86,35	87
7	51,06	73,15	21	0,52	86,64	87
8	82,57	75,38	19	0,61	86,48	81
9	142,49	75,51	19	0,45	87,95	92
10	218,49	72,13	10	0,50	87,89	91

Полученные экспериментальные результаты подтверждают необходимость разработки специализированного муравьиного алгоритма для решения задачи построения автоматов.

3.5. Исследование эффективности метода на основе графа мутаций

3.5.1. Построение конечных автоматов для задачи об «Умном муравье»

Во всех экспериментах использовались следующие значения параметров алгоритма: число муравьев $N_{\text{ants}} = 10$, скорость испарения феромона $\rho = 0.7$, параметр стагнации муравья $n_{\text{stag}} = 50$, параметр стагнации колонии $N_{\text{stag}} = 200$, число мутаций $N_{\text{mut}} = 60$, вероятность мутации $p_{\text{new}} = 0.6$, $\alpha = \beta = 1.0$.

3.5.1.1. Поле Санта Фе

Результаты работы предложенного метода для поля Санта Фе сравнивались с результатами, полученными в [8], которые на данный момент являются лучшими из всех опубликованных результатов для этой задачи при $s_{\max} = 600$. Стоит отметить, что в указанной работе строятся не конечные автоматы, а так называемые *S*-выражения из языка программирования *LISP*. Тем не менее, результаты сравнения можно считать корректными, так как существует алгоритм трансформации *S*-выражения в конечный автомат [18].

При проведении экспериментов варьировалось как число состояний в целевом автомате N, так и наибольшее число доступных муравью шагов $s_{\max}$. Эксперимент для каждой комбинации параметров N и $s_{\max}$ был повторен 10000 раз. В каждом из экспериментов предложенный метод нашел автомат, позволяющий муравью съесть всю еду. Например, для автоматов из семи состояний при $s_{\max} = 600$ предложенный метод строит автомат, позволяющий муравью съесть всю еду, в среднем за 7203 вычисления ФП, в то время как в работе [8] для построения аналогичного *S*-выражения требуется 20696 вычислений ФП. На рис. 11 приведены графики зависимости среднего числа вычислений ФП от числа состояний искомого автомата. На рис. 12 приведена диаграмма переходов одного из построенных автоматов, который позволяет муравью съесть всю еду на поле Санта Фе за 394 хода.

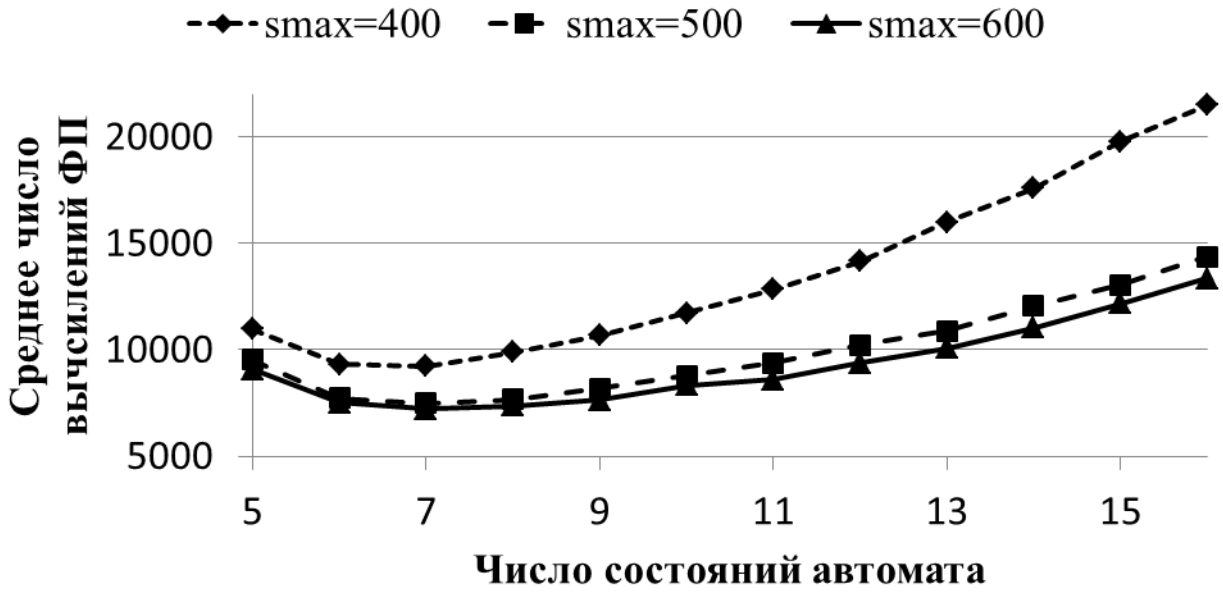

Рис. 11. Поле Санта-Фе: среднее число вычислений ФП при числе состояний автомата N = 5..16 и s_{max} = {400, 500, 600}

Сравнение полученных результатов с приведенными в [8] позволяет утверждать, что предложенный метод генерирует автоматы для поля Санта-Фе быстрее, чем любой другой из ранее опубликованных алгоритмов.

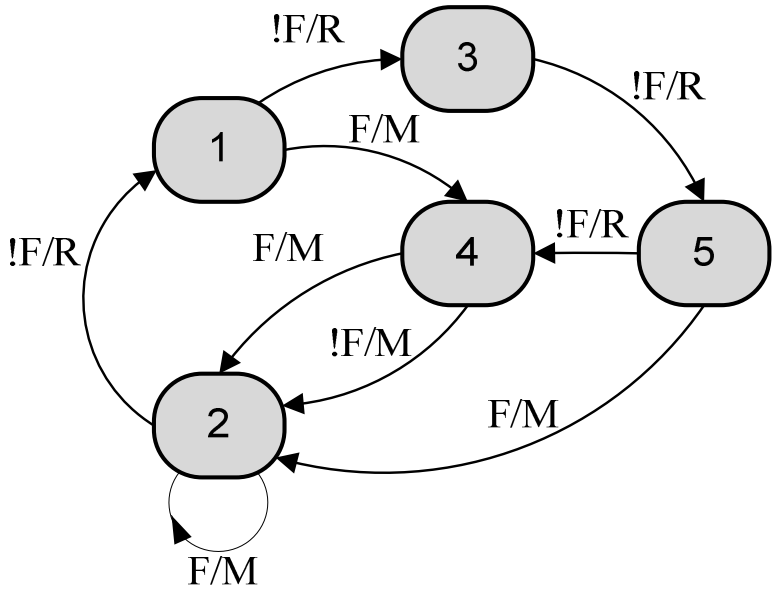

Рис. 12. Диаграмма переходов конечного автомата из пяти состояний, позволяющего агенту в задаче «Умный муравей» съесть всю еду на поле Санта Фе за 394 хода

3.5.1.2. Поле Джона Мура

В первом эксперименте для поля Джона Мура результаты работы предложенного метода сравнивались с результатами работы генетического алгоритма [36]. В экспериментах было установлено ограничение $s_{max} = 200$, а число состояний автомата варьировалось от семи до шестнадцати. Для каждого числа состояний целевого автомата предложенный метод и генетический алгоритм запускались по 100 раз. Для автоматов из семи состояний предложенный метод находит решение в среднем за 31×10^6 вычислений ФП, что приблизительно в 60 раз быстрее, чем генетический алгоритм. Результаты для остальных размеров автоматов приведены на рис. 13. Диаграмма переходов одного из построенных автоматов, позволяющего муравью съесть всю еду за 189 шагов, изображена на рис. 14.

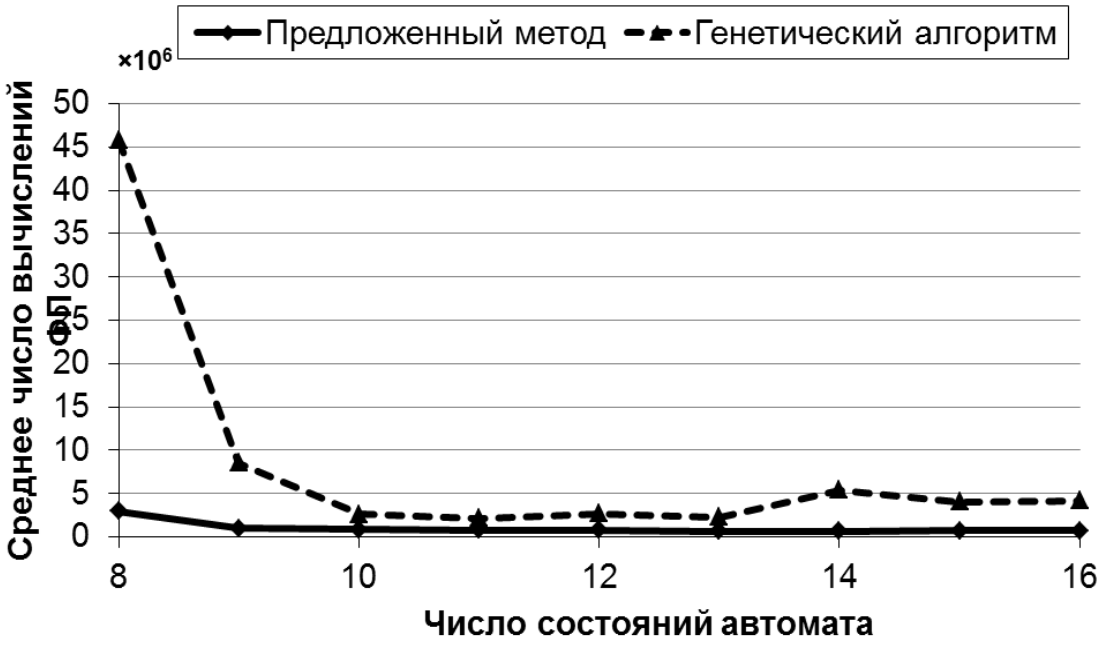

Рис. 13. Поле Джона Мура: среднее число вычислений ФП при $s_{max} = 200$

Во втором эксперименте результаты работы предложенного метода сравнивались с результатами, полученными в [6] при $s_{max} = 600$. В этой работе для построения автоматов применялся эволюционный алгоритм без кроссовера. Наряду с конечными автоматами в [6] строились также системы вложенных конечных автоматов. Кроме того, в эволюционном алгоритме использовались

операторы добавления и удаления состояний, поэтому число состояний автоматов изменялось в течение работы алгоритма. Конечные автоматы при достижении идеального поведения содержали в среднем по 12 состояний. В случае систем вложенных автоматов, автоматы верхнего уровня содержали в среднем семь состояний, в то время как вложенные автоматы содержали до 12 состояний.

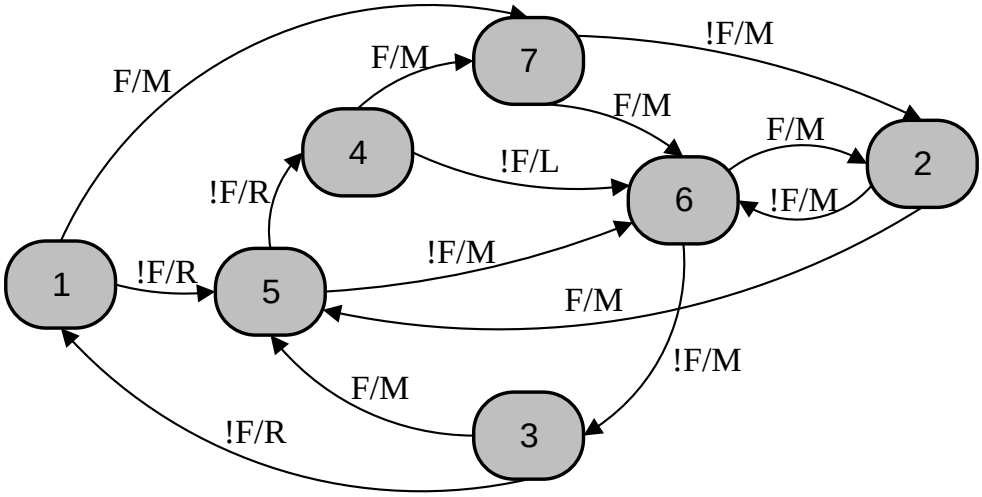

Рис. 14. Диаграмма переходов конечного автомата из семи состояний, позволяющего агенту в задаче «Умный муравей» с полем Санта Фе за 189 ходов

Предложенный метод не использует мутации изменения числа состояний, поэтому в целях проведения сравнения было проведено несколько экспериментов для различного числа состояний целевого автомата. Результаты экспериментов, а также результаты, полученные в [6], представлены на рис. 15. Эксперименты показали, что предложенный метод позволяет решить задачу в среднем в несколько раз быстрее, чем эволюционный алгоритм из [6].

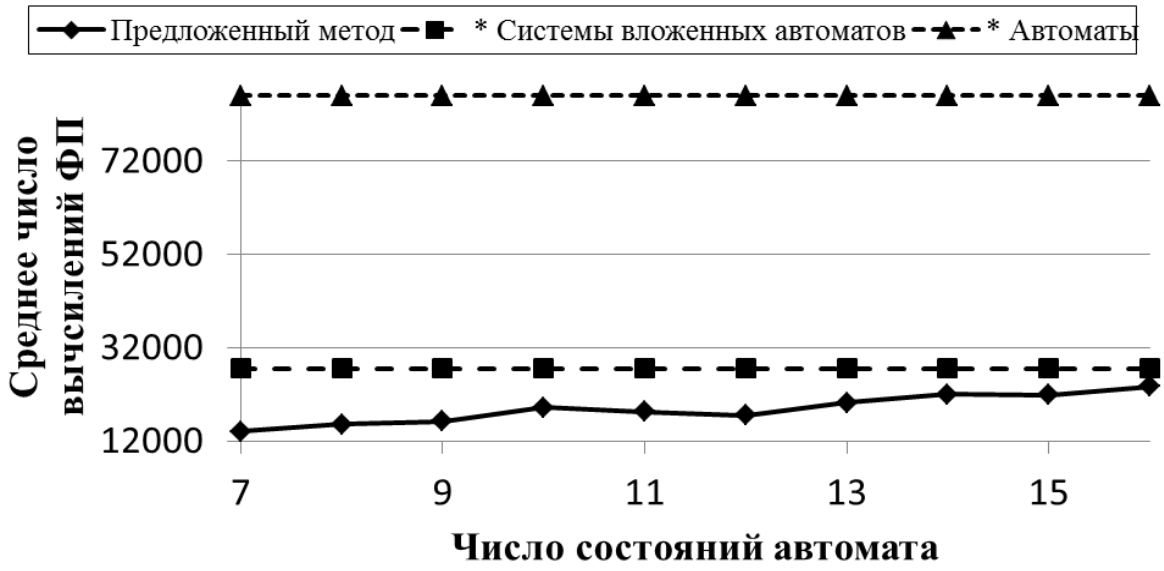

Рис. 15. Поле Джона Мура: среднее число вычислений ФП при s_{max} = 600. (*)Chellapilla, Czarnecki (1999) [5]

Построение управляющих автоматов по обучающим примерам: автомат управления часами с будильником

В качестве первого примера применения разработанного метода для задачи построения управляющих автоматов по обучающим примерам рассматривается задача построения системы управления часами с будильником. У часов с будильником есть три кнопки, которые используются для установки текущего времени, установки времени срабатывания будильника, а также его включения и выключения. Обозначим эти кнопки как “M”, “A” и “H” соответственно.

Имеется два режима работы: установка текущего времени и установка времени срабатывания сигнала. Когда будильник выключен, кнопки “H” и “M” используются для увеличения значений часов и минут текущего времени соответственно. При нажатии на кнопку “A” часы переходят в режим установки времени срабатывания будильника. В этом режиме те же кнопки “H” и “M” устанавливают время срабатывания сигнала. Нажатие кнопки “A” в этом режиме приводит к включению будильника. Во время срабатывания будильника его можно выключить, нажав на кнопку “A”, однако будильник выключится сам по прошествии одной минуты. Часы также содержат таймер,

каждую минуту увеличивающий текущее время. Описанная система содержит семь входных событий и семь выходных воздействий.

В экспериментах использовался набор из 38 тестовых примеров, взятый из [7]. Суммарная длина входных последовательностей была равна 242, длина выходных последовательностей – 195. При построении автоматов пометки на переходах каркасов выбирались из промежутка [0, 7]. Решения искались среди автоматов с четырьмя состояниями. Целью было построение автомата с тремя состояниями, содержащего 14 состояний и имеющего значение функции приспособленности, равное 20,86. Граф переходов этого автомата, который в [33] был построен вручную, приведен на рис. 16.

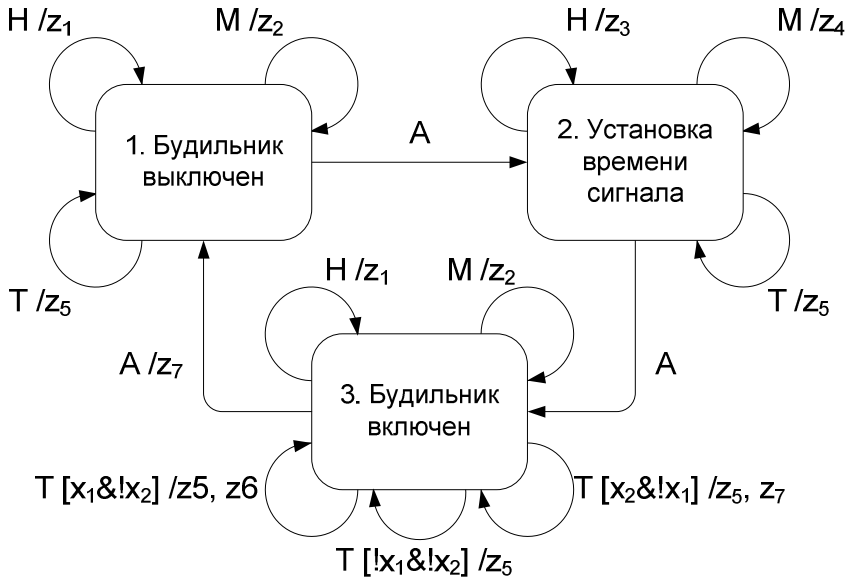

Рис. 16. Автомат управления часами с будильником

Сравнивались следующие алгоритмы:

- Генетический алгоритм с классическим кроссовером (ГА-1).
- Два варианта генетического алгоритма (ГА-2, ГА-2+ЭС), разработанные Царевым и Егоровым [27]. Эти генетические алгоритмы используют специальный оператор кроссовера, учитывающий поведение автомата на обучающих примерах.
- Метод спуска со случайными мутациями (RMHC).
- (1+1)-эволюционная стратегия [2].
- Предлагаемый метод, основанный на графе мутаций.

Эксперимент для каждого метода был повторен 1000 раз. Каждый эксперимент продолжался до получения автомата, удовлетворяющего всем обучающим примерам и содержащего 14 переходов. Результаты экспериментальных запусков приведены в табл. 4.

Таблица 4. Минимальное, максимальное и среднее по 1000 экспериментов число вычислений функции приспособленности, необходимое для генерации искомого автомата

Метод поисковой оптимизации	**Минимальное**	**Максимальное**	**Среднее**
ГА-1	855390	38882588	5805943
RMHC	1150	9592213	1423983
ЭС	1506	9161811	3447390
ГА-2	32830	599022	117977
ГА-2+ЭС	26740	188509	53706
Предложенный метод на основе графа мутаций	2987	211378	**34201**

Проведенные эксперименты показали, что предложенный в данной работе метод позволяет построить целевой автомат в среднем быстрее, чем другие рассмотренные методы. Стоит отдельно отметить, что предложенный метод превосходит по производительности даже генетические алгоритмы, использующие специальный оператор кроссовера.

3.5.2. Построение управляющих автоматов по обучающим примерам: случайно сгенерированные управляющие автоматы

Для полноты исследования рассматривалась задача построения случайно сгенерированных управляющих автоматов. В каждом эксперименте сначала генерировался случайный конечный автомат и по нему генерировались

обучающие примеры. Далее с помощью алгоритмов поисковой оптимизации по обучающим примерам строился удовлетворяющий им управляющий автомат.

Процесс проведения одного эксперимента похож на процесс, использованный в работе [28]. Единственным исключением является то, что в данной работе по автомату строились не сценарии работы, а тестовые (обучающие) примеры. Кратко этот процесс можно описать следующим образом.

- Построение случайного управляющего автомата с заданным числом состояний N_{states}.
- Построение обучающих примеров по автомату. Каждый обучающий пример соответствует случайному пути в автомате длиной от N_{states} до $3{\times}N_{\text{states}}$.
- Применение метода поисковой оптимизации для построения управляющего автомата, удовлетворяющего полученным обучающим примерам.

Генерировались автоматы со следующими параметрами: число входных событий равно двум, число выходных воздействий равно двум, длина последовательности выходных воздействий от одного до двух, число входных переменных равно двум, число переходов автомата равно $4{\times}N_{\text{states}}$.

Сравнение методов поисковой оптимизации проводилось для задач построения автоматов из $N_{\text{states}} = 5$ состояний и наборов обучающих примеров с суммарной длиной l от 200 до 1000. Для каждого значения длины обучающих примеров l проводилось по 100 экспериментов с каждым из рассмотренных алгоритмов. При этом в каждом эксперименте генерировался новый случайный автомат и новые обучающие примеры. Сравнивались показатели эффективности следующих методов:

- эволюционная стратегия;
- предложенный муравьиный алгоритм на основе графа мутаций.

Для обеспечения адекватного сравнения перед проведением экспериментов проводилась настройка параметров алгоритмов. Для настройки

применялся полный факторный эксперимент – перебор всех комбинаций параметров алгоритма в поисках лучшей. Настройка проводилась для задачи построения автомата из пяти состояний по набору обучающих примеров суммарной длины 200. Для каждой комбинации параметров эксперимент повторялся 20 раз. Каждый эксперимент прекращался после 30000 вычислений функции приспособленности. Для каждого алгоритма выбирался набор параметров, обеспечивающий наибольшую долю запусков, в которых был найден автомат, удовлетворяющий всем обучающим примерам.

В ходе основного эксперимента запуск алгоритмов проводился до 50000 вычислений функции приспособленности, вычислялась средняя доля запусков алгоритмов, в которых был найден автомат, удовлетворяющий всем тестам. В каждом эксперименте генерировался новый случайный целевой управляющий автомат и новый набор обучающих примеров. Результаты экспериментальных запусков приведены на рис. 17.

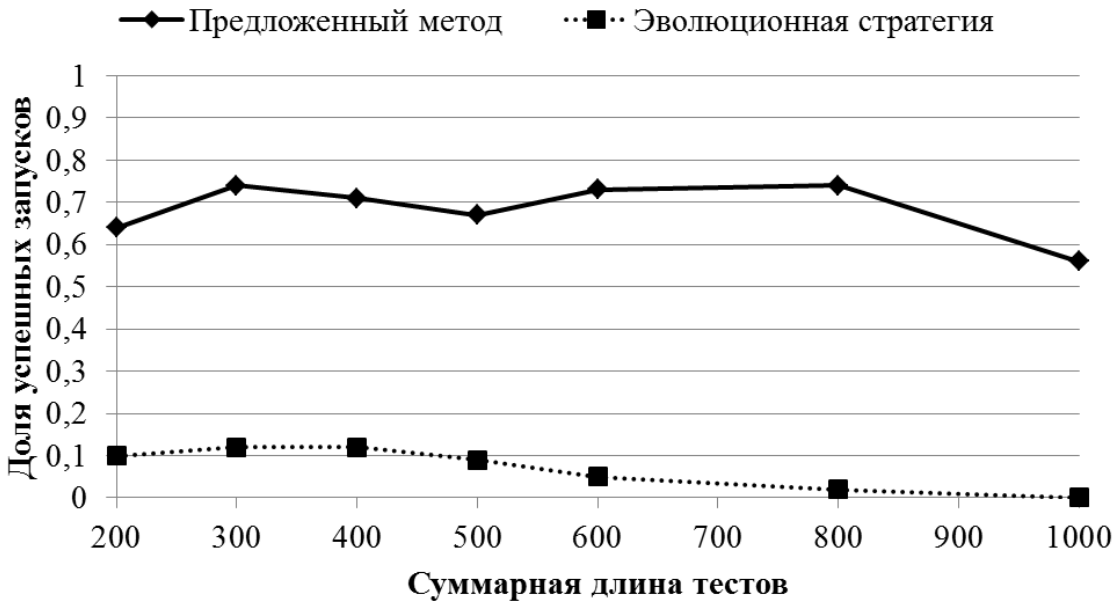

Рис. 17. Зависимость доли успешных запусков алгоритмов от суммарной длины тестов: эволюционная стратегия и предложенный метод на основе графа мутаций

Графики показывают, что предложенный метод в среднем в 6-37 раз эффективнее эволюционной стратегии. Для наибольших тестовых наборов длиной 1000 ни один из запусков эволюционной стратегии не привел к

получению идеального решения, в то время как предложенный метод позволил получить целевое решение примерно в половине запусков.

Также было проведено сравнение с результатами запусков генетического алгоритма, изложенными в диссертации Ф. Царева [35]. Сравнение велось с предложенным там алгоритмом ГА-2+ЭС, который комбинирует генетический алгоритм и эволюционную стратегию. Особенностью генетического алгоритма является то, что в нем применяется специальный оператор кроссовера, учитывающий поведение автомата на тестах. Генетический алгоритм работал до достижения 99% заданного целевого значения функция приспособленности, далее запускалась эволюционная стратегия. Рассматривались автоматы, содержащие от четырех до десяти состояний, два входных события, два выходных воздействия и одну входную переменную. Наибольшая длина выходной последовательности также равнялась двум. По автомату генерировались тесты суммарной длиной $150 \times N_{\text{states}}$.

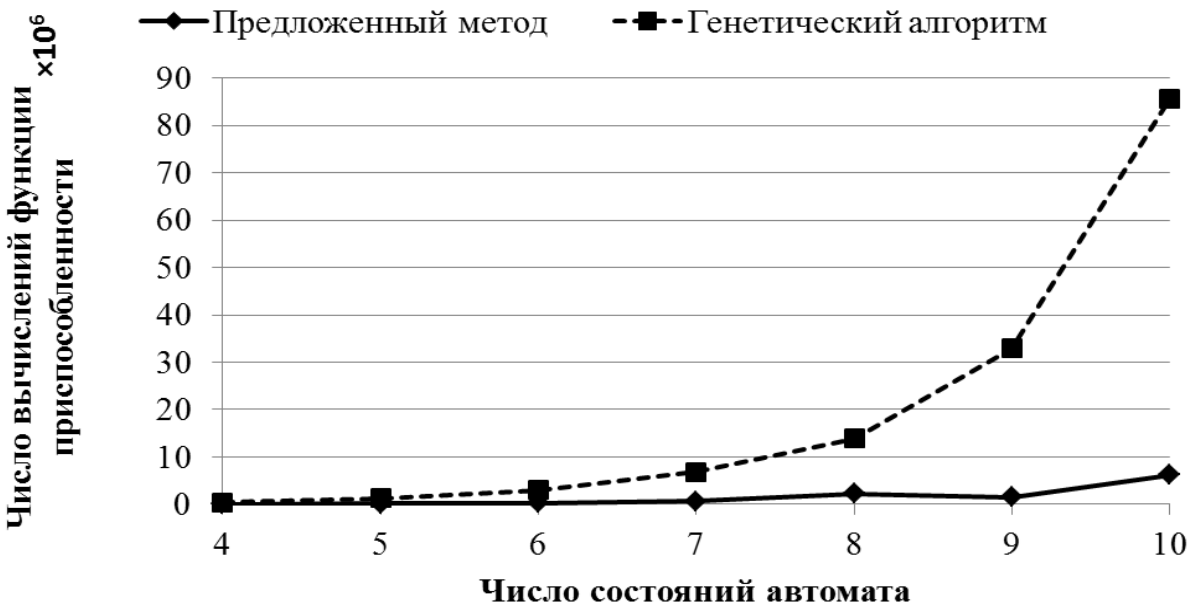

Рис. 18. Среднее число вычислений функции приспособленности для предложенного метода и генетического алгоритма при различных размерах целевых автоматов

Для каждого сгенерированного автомата с 4-10 состояниями проводилось по 100 запусков предложенного метода. В каждом запуске метод работал

вплоть до достижения целевого значения функции приспособленности. Графики средних значений числа вычислений функции приспособленности предложенного метода и генетического алгоритма приведены на рис. 18.

Как видно из графиков, предложенный метод во всех случаях оказался в среднем в 3-20 раз эффективнее генетического алгоритма. В том числе, для автоматов из четырех состояний предложенный метод позволяет найти решение в 8 раз быстрее генетического алгоритма, а для автоматов из девяти состояний в 20 раз быстрее.

3.6. Ленивый метод вычисления функции приспособленности

При разработке методов и проведении экспериментальных исследований был предложен подход, не имеющий прямого отношения к муравьиным алгоритмам, однако позволяющий увеличить их производительность. Предлагается при вычислении значения ФП автомата помечать переходы, которые при этом совершались. Если переход автомата не использовался при вычислении ФП, то поведение автомата при изменении такого перехода не может измениться, следовательно, значение ФП можно не вычислять заново.

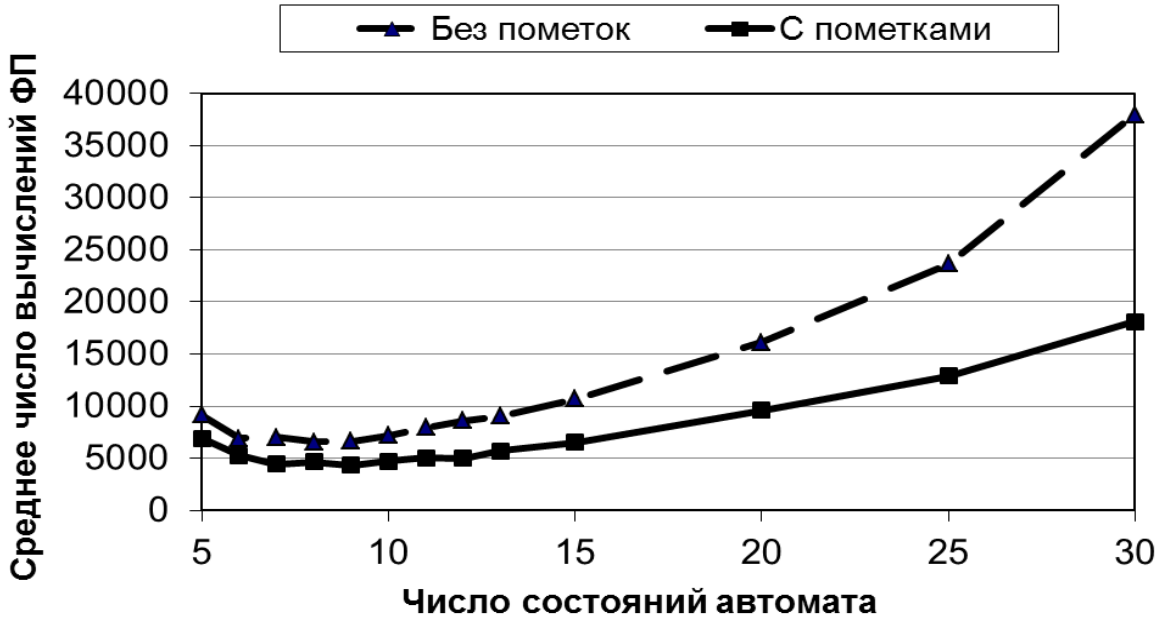

Рис. 19. Влияние пометок об использовании переходов на производительность предложенного алгоритма

Предложенный подход тестировался в применении к методу, основанному на графе мутаций и задаче об «Умном муравье». На рис. 19 приведены графики зависимости среднего числа вычислений ФП от числа состояний автомата с использованием и без использования пометок на переходах для поля Санта Фе при s_{max} = 600. Как видно из графиков, выигрыш от использования пометок увеличивается с ростом числа состояний целевого автомата. Это объясняется тем, что вероятность совершить мутацию, не меняющую поведение автомата, также увеличивается с ростом числа его состояний.

Выводы по главе 3

1. Методы построения автоматов на основе классического сведения к поиску оптимального пути в полном графе и на основе полного недетерминированного конечного автомата, не являются эффективными для решения поставленной задачи.
2. Предложенный метод построения автоматов на основе графа мутаций и муравьиного алгоритма нового типа для рассмотренных задач превосходит по эффективности эволюционные стратегии и генетические алгоритмы.
3. Эффективность предложенного метода на основе графа мутаций может быть увеличена путем применения разработанного ленивого метода вычисления функции приспособленности.

ЗАКЛЮЧЕНИЕ

Был предложен новый метод построения конечных автоматов, основанный на муравьином алгоритме нового типа и представлении пространства поиска в виде графа мутаций. Полученные экспериментальные результаты свидетельствуют о том, что для рассмотренных задач предложенный метод эффективнее, чем известные методы на основе генетических алгоритмов и эволюционных стратегий.

По теме диссертации сделаны доклады на следующих научных, научно-технических и научно-практических конференциях: I Всероссийский конгресс молодых ученых (10-13 апреля 2012 года, НИУ ИТМО), 3-я Всероссийская конференция по проблемам информатики (СПИСОК'2012, 25-27 апреля 2012 года), XIX Всероссийская научно-методическая конференция (Телематика'2012, 26-28 апреля 2012 года, НИУ ИТМО), Genetic and Evolutionary Computation Conference (GECCO'2012, 7-12 июля 2012 года, Philadelphia, USA), Eighth International Conference on Swarm Intelligence (ANTS'2012, 12-14 сентября 2012 года, Brussels, Belgium), XLII научная и учебно-методическая конференция НИУ ИТМО. (29 января – 1 февраля 2013 года, НИУ ИТМО), II Всероссийский конгресс молодых ученых (9-12 апреля 2013 года, НИУ ИТМО), 4-я Всероссийская конференция по проблемам информатики (СПИСОК'2013, 23-26 апреля 2013 года, СПбГУ), VII-я Международная научно-практическая конференция "Интегрированные модели и мягкие вычисления в искусственном интеллекте". (20-22 мая 2013 года, г. Коломна).

Также приняты доклады на конференции: Genetic and Evolutionary Computation Conference (GECCO'2013, 6–10 июля 2013 года, Amsterdam, Netherlands) и IFAC Conference on Manufacturing Modelling, Management and Control (MIM'2013, 19-21 июня 2013 года, St. Petersburg, Russia).

По теме диссертации было опубликовано 8 научных работ, в том числе две работы в трудах крупных зарубежных конференций, одна работа в журнале из перечня ВАК и пять в трудах всероссийских и международных

конференций. Полнотекстовые материалы докладов по теме диссертации были приняты к публикации в трудах конференций GECCO'2013 и MIM'2013, упомянутых выше.

Таким образом, поставленные в диссертации задачи решены, а цель достигнута. Все результаты, полученные в диссертации, были опубликованы и прошли апробацию на российских и зарубежных конференциях, соответствующих ее тематике.

ПУБЛИКАЦИИ

1. *Chivilikhin D., Ulyantsev V.* MuACOsm - A New Mutation-Based Ant Colony Optimization Algorithm for Learning Finite-State Machines / To appear in Proceedings of the 2013 Genetic and Evolutionary Computation Conference
2. *Chivilikhin D., Ulyantsev V., Shalyto A.* Solving Five Instances of the Artificial Ant Problem with Ant Colony Optimization / To appear in Proceedings of the 2013 IFAC Conference on Manufacturing Modelling, Management and Control
3. *Chivilikhin D., Ulyantsev V., Tsarev F.* Test-Based Extended Finite-State Machines Induction with Evolutionary Algorithms and Ant Colony Optimization / Proceedings of the 2012 GECCO Conference Companion on Genetic and Evolutionary Computation. NY.: ACM. 2012, pp. 603–606
4. *Chivilikhin D., Ulyantsev V.* Learning Finite-State Machines with Ant Colony Optimization // Lecture Notes in Computer Science, 2012, Volume 7461/2012, pp. 268-275
5. *Чивилихин Д. С., Ульянцев В. И.* Метод построения управляющих автоматов на основе муравьиных алгоритмов // Научно-технический вестник информационных технологий, механики и оптики. 2012. №6(82), с. 72-76
6. *Чивилихин Д. С., Ульянцев В. И.* Метод построения конечных автоматов на основе муравьиного алгоритма / Сборник тезисов докладов конгресса молодых ученых, Выпуск 1. СПб: НИУ ИТМО. 2013. с. 235-236
7. *Чивилихин Д. С., Ульянцев В. И.* Применение муравьиных алгоритмов для построения конечных автоматов / Сборник тезисов докладов конгресса молодых ученых, Выпуск 1. Труды молодых ученых. СПб: НИУ ИТМО. 2012. с. 227-228
8. *Чивилихин Д. С., Ульянцев В. И.* Применение муравьиных алгоритмов для построения конечных автоматов / Всероссийская научная конференция по проблемам информатики (СПИСОК-2012). СПб.: ВВМ. СПбГУ. 2012, с. 409-410
9. *Чивилихин Д. С., Ульянцев В. И.* Применение муравьиных алгоритмов для построения конечных автоматов / Труды XIX Всероссийской научно-методической конференции Телематика'2012. Том 1. СПб: Университетские телекоммуникации, 2012. с. 145.

10. *Чивилихин Д. С., Ульянцев В. И., Шалыто А. А.* Метод построения конечных автоматов на основе муравьиного алгоритма / Интегрированные модели и мягкие вычисления в искусственном интеллекте. Сборник тезисов докладов VII-й Международной научно-технической конференции (Коломна, 20-22 мая 2013 г.). В 3-х томах. Т.2. - М.: Физматлит, 2013. с. 931-942

ИСТОЧНИКИ

1. *Alexandrov A., Sergushichev A., Kazakov A., Tsarev F.* Genetic algorithm for induction of finite automata with continuous and discrete output actions / In Proceedings of the 13th annual conference companion on Genetic and evolutionary computation (GECCO '11), Natalio Krasnogor (Ed.). ACM, New York, NY, USA. 2011. pp. 775 – 778.
2. *Beyer H. G.* The Theory of Evolution Strategies. Natural Computing Series. Berlin. NY: Springer-Verlag, 2001.
3. *Bonabeau E., Dorigo M., Theraulaz G.* Swarm Intelligence: from natural to artificial systems. – Oxford University Press. 1999.
4. *Bullnheimer B., Hartl R. F., Strauss C.* A new rank-based version of the Ant System: A computational study // Central European Journal for Operations Research and Economics. 1999. Vol. 7, № 1, pp. 25 – 38.
5. *Clarke J., Dolado J., Harman M., Hierons R., Jones B., Lumkin M., Mitchell B., Mancoridis S., Rees K., Roper M., Shepperd M.* Reformulating software engineering as a search problem / IEEE Proceedings – Software. 2003. Vol. 150(3), pp. 161 – 175.
6. *Chellapilla K., Czarnecki D.* A preliminary investigation into evolving modular finite state machines / Proceedings of the 1999 Congress on Evolutionary Computation. 1999. Vol. 2, pp. 1349 – 1356.
7. *Chivilikhin D., Ulyantsev V., Tsarev F.* Test-Based Extended Finite-State Machines Induction with Evolutionary Algorithms and Ant Colony Optimization / Proceedings of the 2012 GECCO Conference Companion on Genetic and Evolutionary Computation. NY.: ACM. 2012, pp. 603 – 606.
8. *Christensen K., Oppacher F.* Solving the artificial ant on the Santa Fe trail problem in 20,696 fitness evaluations / Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07). 2007. pp. 1574 – 1579.
9. *Dorigo M.* Optimization, Learning and Natural Algorithms. PhD thesis. Polytechnico di Milano, Italy. 1992.
10. *Dorigo M., Gambardella L. M.* Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem // IEEE Transactions on Evolutionary Computation. 1997. Vol. 1, № 1, pp. 55 – 66.
11. *Dorigo M., Maniezzo V., Colorni A.* Ant System: Optimization by a colony of cooperating agents // IEEE Transactions on Systems, Man, and Cybernetics. 1996. Vol. 26, № 1, Part B, pp. 29 – 41.
12. *Dorigo M., Stützle T.* Ant Colony Optimization. – MA: MIT Press, 2004.

13. *Gomez J.* An incremental-evolutionary approach for learning finite automata / In proceedings of the 2006 IEEE Congress on Evolutionary Computation. 2006. pp. 362 – 369.
14. *Harman M.* Software engineering meets evolutionary computation // Computer. 2011. Vol. 44, № 11, pp. 31 – 39.
15. *Heule M., Verwer S.* Exact DFA Identification Using SAT Solvers / Grammatical Inference: Theoretical Results and Applications 10th International Colloquium, (ICGI 2010). 2010. Lecture Notes in Computer Science. Vol. 6339, Springer. pp. 66 – 79.
16. *Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C., Wang A.* Evolution as a theme in artificial life: The Genesys/Tracker system. Technical report. 1990.
17. *Kennedy J., Eberhart R.* Particle swarm optimization / In Proceedings of the 1995 IEEE International Conference on Neural Networks. 1995. Vol. 4, pp. 1942 – 1948.
18. *Kim D.* Memory analysis and significance test for agent behaviours / Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. 2006. pp. 151 – 158.
19. *Lang K. J., Pearlmutter B. A., Price R. A.* Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm // In ICGI. Lecture Notes in Computer Science, Springer. 1998. Vol. 1433, pp. 1 – 12.
20. *Lucas S., Reynolds J.* Learning Finite State Transducers: Evolution versus Heuristic State Merging // IEEE Transactions on Evolutionary Computation. 2007. Vol. 11, № 3, pp. 308 – 325.
21. *Lucas S., Reynolds J.* Learning Deterministic Finite Automata with a Smart State Labeling Evolutionary Algorithm // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2005. Vol. 27, № 7, pp. 1 – 12.
22. *Mitchell M.* An Introduction to Genetic Algorithms. MA: The MIT Press. 1996.
23. *Pham D. T., Ghanbarzadeh A., Koc E., Otri S., Rahim S., Zaidi M.* The Bees Algorithm. Technical note. Manufacturing Engineering Centre, Cardiff University, UK. 2005.
24. *Spears W. M., Gordon D. E.* Evolving finite-state machine strategies for protecting resources / In Proceedings of the International Symposium on Methodologies for Intelligent Systems. 2000. pp. 166 – 175.
25. *Stützle T., Hoos H. H.* MAX MIN Ant System // Future Generation Computer Systems. 2000. Vol. 16, № 8, pp. 889 – 914.
26. *Stützle T., Dorigo M.* A short convergence proof for a class of ACO algorithms // IEEE Transactions on Evolutionary computation. 2002. Vol. 6(4), pp. 358 – 365
27. *Tsarev F., Egorov K.* Finite-state machine induction using genetic algorithm based on testing and model checking / Proceedings of the 2011 GECCO Conference Companion on Genetic and Evolutionary Computation (GECCO'11). 2011. pp. 759 – 762.
28. *Ulyantsev V., Tsarev F.* Extended Finite-State Machine Induction using SAT-Solver / In Proceedings of the 14th IFAC Symposium “Information Control Problems in Manufacturing (INCOM'12)". IFAC. 2012. pp. 512 – 517.

29. *Yang X. S. Nature-inspired metaheuristic algorithms.* – Luniver Press. 2008.
30. *Вельдер С. Э., Лукин М. А., Шалыто А. А., Яминов Б. Р.* Верификация автоматных программ. – СПб: Наука. 2011.
31. *Емельянов В. В., Курейчик В. М., Курейчик В. В.* Теория и практика эволюционного моделирования. М.: Физматлит, 2003.
32. *Лобанов П. Г., Шалыто А. А.* Использование генетических алгоритмов для автоматического построения конечных автоматов в задаче о «Флибах» / Сборник докладов 4-й Всероссийской научной конференции «Управление и информационные технологии» (УИТ-2006). СПбГЭТУ «ЛЭТИ». 2006. с.144 – 149.
33. *Поликарпова Н. И., Шалыто А. А.* Автоматное программирование. – СПб: Питер. 2011.
34. *Поликарпова Н. И., Точилин В. Н., Шалыто А. А.* Метод сокращенных таблиц для генерации автоматов с большим числом входных переменных на основе генетического программирования // Известия РАН. Теория и системы управления. 2010. № 2, с. 100 – 117.
35. *Царев Ф. Н.* Методы построения конечных автоматов на основе эволюционных алгоритмов. Диссертация на соискание ученой степени кандидата технических наук. НИУ ИТМО. 2012. http://is.ifmo.ru/disser/tsarev_disser.pdf
36. *Царев Ф. Н., Шалыто А. А.* О построении автоматов с минимальным числом состояний для задачи об «Умном муравье» / Сборник докладов X международной конференции по мягким вычислениям и измерениям. СПбГЭТУ «ЛЭТИ». 2007. Т. 2, С. 88 – 91.