Министерство образования и науки Российской Федерации

УДК 004.4’242
ГРНТИ 20.01.01, 28.23.29
Инв. № 310275

УТВЕРЖДЕНО:
Исполнитель: **Государственное образовательное учреждение высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики»**
Ректор ГОУВПО «СПбГУ ИТМО» **_____/ В.Н. Васильев/** **М.П.**

НАУЧНО-ТЕХНИЧЕСКИЙ ОТЧЕТ

о выполнении 1 этапа Государственного контракта
№ 16.740.11.0455 от 13 мая 2011 г.

Исполнитель: Государственное образовательное учреждение высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики»
Программа (мероприятие): Федеральная целевая программа «Научные и научно-педагогические кадры инновационной России» на 2009-2013 гг., в рамках реализации мероприятия № 1.2.1 Проведение научных исследований научными группами под руководством докторов наук.
Проект: Разработка метода машинного обучения на основе алгоритмов решения задачи о выполнимости булевой формулы для построения управляющих конечных автоматов
Руководитель проекта: Шалыто Анатолий Абрамович

Санкт-Петербург
2011 г.

СПИСОК ОСНОВНЫХ ИСПОЛНИТЕЛЕЙ

по Государственному контракту 16.740.11.0455 от 13 мая 2011 на выполнение поисковых научно-исследовательских работ для государственных нужд

Организация-Исполнитель: Государственное образовательное учреждение высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики».

Руководитель темы:

доктор технических наук, профессор	А.А. Шалыто

Исполнители темы:

без ученой степени, без ученого звания	Ф.Н. Царев
без ученой степени, без ученого звания	М.В. Буздалов
без ученой степени, без ученого звания	В.И. Ульянцев

РЕФЕРАТ

Отчет 52 с., 4 гл., 13 рис., 4 табл., 45 источников.

Ключевые слова: задача о выполнимости булевой формулы, машинное обучение, автоматное программирование.

В настоящем отчете излагаются результаты выполнения *поисковых научно-исследовательских работ по теме «Разработка метода машинного обучения на основе алгоритмов решения задачи о выполнимости булевой формулы для построения управляющих конечных автоматов»*, выполняемых в рамках государственного контракта, заключенного между Министерством образования и науки Российской Федерации и государственным образовательным учреждением высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики» в соответствии с решением Конкурсной комиссии Министерства образования и науки Российской Федерации № 1 (протокол от 25.03.2011 № 3/0173100003711000031) по лоту шифр «2011-1.2.1-113-002» «Проведение научных исследований научными группами под руководством докторов наук в области информатики» в рамках федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы, утвержденной постановлением Правительства Российской Федерации от 28 июля 2008 года № 568 «О федеральной целевой программе «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы».

Целями настоящего этапа являются:

1. Выполнение аналитического обзора.
2. Проведение патентных исследований.
3. Выбор и обоснование оптимального варианта проведения исследований и выбор архитектуры метода машинного обучения.
4. Подготовка плана проведения теоретических и экспериментальных исследований.

Излагаются результаты выполнения аналитического обзора по следующим направлениям: «Методы машинного обучения для построения конечных автоматов» и «Программные средства для решения задачи о выполнимости булевой формулы».

Приводится обоснование выбора оптимального варианта направления исследований, а также план проведения теоретических и экспериментальных исследований.

Описывается архитектура предлагаемого метода машинного обучения.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящем отчете излагаются результаты выполнения *поисковых научно-исследовательских работ по теме «Разработка метода машинного обучения на основе алгоритмов решения задачи о выполнимости булевой формулы для построения управляющих конечных автоматов»*, выполняемых в рамках государственного контракта, заключенного между Министерством образования и науки Российской Федерации и государственным образовательным учреждением высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики» в соответствии с решением Конкурсной комиссии Министерства образования и науки Российской Федерации № 1 (протокол от 25.03.2011 № 3/01731000037110000031) по лоту шифр «2011-1.2.1-113-002» «Проведение научных исследований научными группами под руководством докторов наук в области информатики» в рамках федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы, утвержденной постановлением Правительства Российской Федерации от 28 июля 2008 года № 568 «О федеральной целевой программе «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы».

Целями настоящего этапа являются:

1. Выполнение аналитического обзора.
2. Проведение патентных исследований.
3. Выбор и обоснование оптимального варианта проведения исследований и выбор архитектуры метода машинного обучения.
4. Подготовка плана проведения теоретических и экспериментальных исследований.

Отчет имеет следующую структуру. В первой главе приводятся результаты выполнения аналитического обзора по двум направлениям:

- методы машинного обучения для построения конечных автоматов;
- программные средства для решения задачи о выполнимости булевой формулы.

Во второй главе обосновывается выбор оптимального направления исследований.

В третьей главе приводится план проведения теоретических и экспериментальных исследований.

В четвертой главе описывается архитектура разрабатываемого метода машинного обучения.

Каждая из глав снабжена выводами, кратко резюмирующими содержание главы. В заключении дается общая оценка работ по этапу.

Машинное обучение – современный раздел информатики, посвященный созданию алгоритмов, которые обучают параметры некоторой модели на заранее подготовленных тестовых примерах, либо в процессе моделирования ее работы в некоторой внешней среде (обучение «на собственных ошибках»).

Задача о выполнимости булевой формулы – одна из важнейших задач информатики. Она состоит в том, чтобы по заданной булевой формуле найти такой набор значений входящих в нее переменных, что результат вычисления формулы – «истина». Эта задача относится к классу NP-полных задач, но для ее решения существует большое число весьма эффективных на практике программных средств. Наиболее современные из этих программных средств такие, как *zChaff* (http://www.princeton.edu/~chaff/zchaff.html), *MiniSat* (http://minisat.se/), *CryptoMiniSat* (http://www.msoos.org/cryptominisat2), *Plingeling* (http://fmv.jku.at/lingeling/) способны обрабатывать формулы длиной в несколько миллионов символов, содержащие десятки и сотни тысяч переменных.

С использованием этих программных средств в настоящее время решаются такие задачи, как проверка эквивалентности формальных моделей программ, верификации моделей программ, фор-

мальной верификации микропроцессоров, генерации тестовых шаблонов, разводка печатных плат и т. д.

Автоматное программирование – парадигма программирования, при использовании которой программу предлагается строить в виде совокупности **автоматизированных объектов управления**, каждый из которых содержит систему управления (взаимодействующие конечные автоматы) и объект управления.

Объект управления характеризуется множеством вычислительных состояний, а также двумя наборами функций: множеством предикатов, отображающих вычислительное состояние в логическое значение (истина или ложь), и множеством действий, позволяющих изменять вычислительное состояние. **Управляющий автомат** определяется конечным множеством управляющих состояний, функцией переходов и функцией действий. Взаимодействие между автоматами может осуществляться различными способами: за счет вложенности автоматов, с помощью обмена сообщениями, с помощью обмена номерами состояний и т. д.

Исследования, проводимые в ряде университетов мира (например, в Стенфордском университете), показывают, что модели, основанные на состояниях, целесообразно применять для построения систем со сложным поведением. Такие системы могут в ответ на одно и то же входное воздействие вырабатывать различные выходные воздействия в зависимости от предыстории работы.

Модели, основанные на состояниях, широко применяются в машинном обучении. Примером таких моделей являются Марковские цепи и скрытые Марковские модели. Они могут применяться в таких задачах, как распознавание речи, и для них разработан ряд алгоритмов обучения. Область их применения ограничивается тем, что эти модели имеют вероятностный характер. Наиболее близкими к ним моделями, имеющими детерминированный характер, являются конечные автоматы.

Важным достоинством автоматных программ является возможность их **автоматической** верификации, что является очень существенным свойством для систем с повышенными требованиями к надежности. Для этих систем важную роль играет влияние человеческого фактора на процесс разработки. Поэтому актуальной задачей является разработка методов машинного обучения для построения конечных автоматов.

Как правило, в машинном обучении конечные автоматы рассматриваются как распознаватели регулярных языков или как преобразователи слов одного регулярного языка в слова другого. В случае построения управляющих автоматов обычно рассматриваются системы с малым числом входных переменных (одной или двумя), что существенно ограничивает область применения таких алгоритмов.

Кроме этого, большая часть существующих алгоритмов машинного обучения для конечных автоматов основаны на принципе обучения «на собственных ошибках» и не являются достаточно эффективными для ряда задач, так как не позволяют эффективно учитывать знания человека. Те алгоритмы, которые основаны на принципе обучения на примерах, обладают недостаточно высокой производительностью для использования на практике – они позволяют работать с автоматами, содержащими пять-десять состояний, а суммарная длина обучающих примеров может составлять не более двухсот-трехсот событий. Построение автомата с помощью существующих алгоритмов может занимать от нескольких минут до нескольких дней.

Для устранения недостатков существующих методов будет разработан **метод машинного обучения, основанный на решении задачи о выполнимости булевой формулы**. В качестве входных данных этот метод будет использовать сценарии работы – один из типов обучающих примеров (тестов). Предварительные исследования показывают, что новый метод будет обеспечивать существенно более высокую производительность (в 10-100 раз более высокую, чем у существующих методов) и сможет обрабатывать автоматы с десятками состояний и тысячами событий.

Изложенное позволяет утверждать, что результаты выполнения научно-исследовательской работы будут превышать мировой уровень разработок в рассматриваемой области.

1. ПРОВЕДЕНИЕ АНАЛИТИЧЕСКОГО ОБЗОРА, ПОДГОТОВКА ПЛАНА ВЫПОЛНЕНИЯ ПНИР И ПРОВЕДЕНИЕ ПАТЕНТНЫХ ИССЛЕДОВАНИЙ

В настоящем разделе приводятся результаты аналитического обзора. Аналитический обзор проводился по следующим направлениям:

- методы машинного обучения для построения конечных автоматов;
- программные средства для решения задачи о выполнимости булевой формулы.

1.1. Методы машинного обучения для построения конечных автоматов

В настоящем разделе приводится обзор существующих методов машинного обучения для построения конечных автоматов.

1.1.1. Машинное обучение и автоматные модели

В этом разделе приводятся основные понятия и определения областей машинного обучения и теории автоматов.

1.1.1.1. Машинное обучение

В соответствии с [7] алгоритм является алгоритмом машинного обучения, если он улучшает свое поведение по мере накопления опыта. Это означает, что алгоритм обучает параметры модели либо на заранее подготовленных тестовых примерах, либо на собственных ошибках, и со временем решает поставленную задачу все лучше и лучше. Некоторые алгоритмы машинного обучения способны подмечать ранее неизвестные закономерности в данных, выделять знания, которых раньше не было.

Модели, основанные на состояниях, широко применяются в машинном обучении. Примером таких моделей являются скрытые Марковские модели. Они могут применяться в таких задачах, как распознавание речи, и для них разработан ряд алгоритмов обучения. Область их применения ограничивается тем, что эти модели имеют вероятностный характер. Наиболее близкими к ним моделями, имеющими детерминированный характер, являются конечные автоматы.

Как правило, в машинном обучении конечные автоматы рассматриваются как распознаватели регулярных языков или как преобразователи слов одного регулярного языка в слова другого. В случае построения управляющих автоматов обычно рассматриваются системы с малым числом входных переменных (одной или двумя), что существенно ограничивает область применения таких алгоритмов. Кроме этого, существующие алгоритмы машинного обучения для конечных автоматов основаны на принципе обучения «на собственных ошибках» и не являются достаточно эффективными для ряда задач, так как не позволяют эффективно учитывать знания человека и не позволяют строить автоматы с гарантированным поведением.

1.1.1.2. Типы автоматных моделей

В настоящем разделе описываются типы автоматных моделей. Абстрактные конечные автоматы принято описывать в следующих терминах. Задано конечное множество символов *X*, которое называется (входным) алфавитом. Множество всех возможных цепочек (последовательностей, строк, слов), составленных из символов алфавита *X* обозначается *X**. Пустая последовательность символов обозначается ε, $\varepsilon \in X^*$. Подмножество *L* множества всех цепочек над алфавитом *X*, $L \subset X^*$, называется языком. Рассматривается следующая проблема: задан язык $L \subset X^*$ и цепочка $\xi \in X^*$. Необходимо определить, принадлежит ли указанная цепочка языку ($\xi \in L$).

Если абстрактный вычислитель способен решить эту проблему для определенного языка $L \subset X^*$ и произвольной строки $\xi \in X^*$, то говорят, что вычислитель распознает язык *L*. Таким

образом, абстрактные автоматы описываются в терминах тех языков, которые они распознают. Различные автоматные модели могут распознавать разные классы языков или, другими словами, обладают разной вычислительной мощностью (вычислительная мощность модели абстрактных автоматов тем больше, чем шире класс распознаваемых ими языков).

1.1.1.3. Конечные автоматы-распознаватели

Детерминированный конечный автомат (ДКА) – это пятерка $\langle X, Y, \delta, y_0, F \rangle$, где X – конечный алфавит входных символов, Y – конечное множество состояний, $\delta : X \times Y \to Y$ – функция переходов, $y_0 \in Y$ – начальное (стартовое) состояние, $F \subset Y$ – множество допускающих состояний.

Расширенная функция переходов $\hat{\delta} : X^* \times Y \to Y$, сопоставляющая новое состояние текущему состоянию и цепочке символов, определяется индуктивно следующим образом [16]:

$$\forall y \in Y \, (\hat{\delta}(\varepsilon, y) = y);$$

$$\forall y \in Y \ \forall \xi \in X^* \ \forall x \in X \left(\hat{\delta}(\xi x, y) = \delta(x, \hat{\delta}(\xi, y))\right).$$

В таком случае, если $\hat{\delta}(\xi, y_0) \in F$ (стартуя в начальном состоянии и обработав цепочку ξ, автомат оказывается в одном из допускающих состояний), то говорят, что он допускает эту цепочку. Множество допускаемых цепочек образует язык L, распознаваемый ДКА: $L = \left\{ \xi \in X^* \mid \hat{\delta}(\xi, y_0) \in F \right\}$.

Класс языков, распознаваемых ДКА, называют регулярными языками. Известно, что он совпадает с классом языков, описываемых регулярными выражениями и автоматными грамматиками [16].

1.1.1.4. Конечные автоматы-преобразователи

Для того чтобы наделить модель абстрактного конечного автомата способностью не только давать ответ типа «да/нет», но и выполнять какие-то преобразования, в модель добавляют конечный алфавит выходных символов Z и функцию выхода φ. Если функция выхода имеет вид $\varphi : X \times Y \to Z$, то вычислитель называется автоматом Мили, а если $\varphi : Y \to Z$ – автоматом Мура. Таким образом, рассматривается шестерка $\langle X, Y, Z, \delta, \varphi, y_0 \rangle$. Она определяет автоматное отображение $f : X^* \to Z^*$ (преобразование, выполняемое автоматом) следующим образом:

$$f(\varepsilon) = \varepsilon$$

$$\forall \xi \in X^* \ \forall x \in X \left(f(\xi x) = f(\xi)\varphi(x, \hat{\delta}(\xi, y_0))\right)$$

Автоматные отображения – это отображения «без предсказания»: перерабатывая цепочку слева направо, они «не заглядывают вперед». Например, отображение, которое сопоставляет цепочке ее саму, записанную в обратном порядке, не является автоматным.

1.1.1.5. Управляющие конечные автоматы

Понятие управляющего (структурного) автомата Мура аналогично понятию абстрактного автомата Мура, введенному выше. В таком автомате выходное воздействие зависит только от состояния и не зависит от входного воздействия. На рис. 1 приведен пример графа переходов автомата Мура.

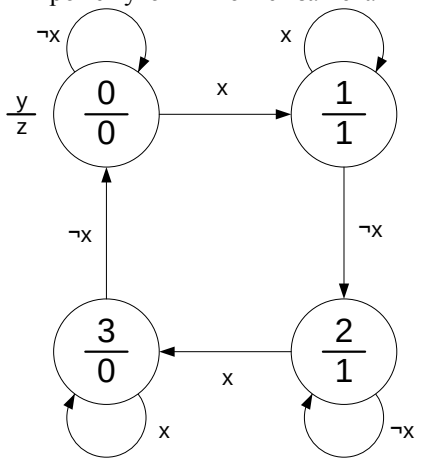

Рис. 1. Пример графа переходов автомата Мура

По аналогии с абстрактными автоматами, структурный автомат, выходные воздействия которого зависят не только от состояния, но и от входных воздействий, называется автоматом Мили. Известно [15], что для любого автомата Мили можно построить эквивалентный ему автомат Мура. Число состояний в таком автомате будет не меньше, чем в исходном.

В качестве примера рассмотрим последовательный двоичный одноразрядный сумматор, который выполняет сложение одноименных бит двух двоичных чисел с учетом переноса. Результатом работы сумматора является одноименный бит суммы и перенос в следующий разряд. На рис. 2 приведен автомат Мили, реализующий это «устройство».

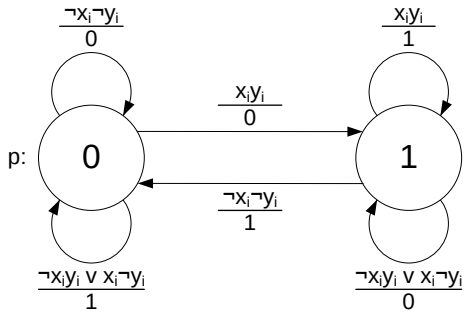

Рис. 2. Автомат Мили последовательного двоичного одноразрядного сумматора

На этом рисунке x_i и y_i – складываемые биты, состояния соответствуют значениям переноса, а значение бита суммы формируется как выходное воздействие на переходах.

Если часть выходных переменных автомата зависит только от состояний, а остальные – также и от входных воздействий, удобно разделить функцию выходов φ на две составляющие: $\varphi_1(y)$ и $\varphi_2(x, y)$. В структурную схему автомата в этом случае вводится не один, как в автоматах Мура и Мили, а два выходных преобразователя. Такие автоматы называются смешанными, автоматами Мура-Мили или С-автоматами.

1.1.2. Генетические алгоритмы и эволюционные вычисления

В этом разделе приведено краткое описание *традиционного генетического алгоритма* (*conventional genetic algorithm*) [38] с одноточечным скрещиванием и мутацией. В этом алгоритме каждая особь представляет собой битовую строку длины L, а размер популяции равен n.

Шаг 1. Генерации начальной популяции $\{x_i\}_{i=1}^{n}$. Для этого, например, можно сгенерировать n битовых строк, в которых каждый из L битов равен 0 или 1 с одинаковой вероятностью.

Шаг 2. Вычислить функцию приспособленности $f(x)$ для каждой особи из текущей популяции. Этот шаг обычно является наиболее трудоемким по времени во всем алгоритме. Отметим, что при вычислении функции приспособленности, как правило, необходимо осуществить декодирование некоторого объекта из двоичной строки.

Шаг 3. Переход к следующему поколению популяции. При создании нового поколения могут использоваться различные стратегии. В качестве примера приведем так называемый *метод рулетки* (*roulette wheel selector*) [2].

Создание нового поколения в этом случае разбивается на n итераций, на каждой из которых в популяцию добавляется одна особь. Для этого случайно с вероятностями, пропорциональными их функция приспособленности, из текущего поколения популяции выбираются две родительские особи x и y.

После этого с некоторой наперед заданной вероятностью происходит скрещивание (рекомбинация, кроссовер, кроссинговер) родительских особей. В результате образуются их «потомки» – z_1 и z_2. Происходит это следующим образом: случайно (с равномерным распределением на множестве $\{1, 2, \ldots, L\}$) выбирается число k. Хромосомы «потомков» z_1 и z_2 теперь строятся следующим образом: для z_1 – первые k символов совпадают с первыми k символами x, последние $(L-k)$ – с последними $(L-k)$ символами y, для z_2 – наоборот. Например, если $L=7$, $x=1001010$, $y=0001001$, $k=3$, то $z_1=1001001$, $z_2=0001010$. После этого равновероятно выбирается одна из особей z_1 либо z_2 – выбранную особь обозначим как z.

С некоторой (как правило, порядка 0.01) вероятностью производится мутация особи z – в ней случайно выбирается один бит и изменяется.

Получившаяся в результате особь z добавляется в следующее поколение.

Шаг 4. Повторять шаги 2 и 3 до тех пор, пока не будет выполнено условие останова (максимальная приспособленность в популяции достигла целевого значения, прекратился рост максимальной приспособленности, число поколений достигло некоторого предела и т. д.).

Отметим, что кроме описанных выше методов одноточечного кроссовера, алгоритма рулетки и мутации изменением случайного бита, существуют и другие – обзор приведен в работе [8].

Генетическое программирование (*genetic programming*), предложенное J.R. Koza в 1992 году [32], предполагает применение генетических алгоритмов для автоматизированного построения программ.

Основное отличие генетического программирования от традиционных генетических алгоритмов состоит в способе кодирования особей. Если в генетическом алгоритме особи кодируются с помощью битовых строк, то в генетическом программировании используется более *высокоуровневое* представление: деревья разбора, тексты программ на языках программирования с несложной структурой, диаграммы переходов конечных автоматов и т. д.

Такой подход позволяет определить генетические операции скрещивания и мутации, которые лучше подходят для решаемой задачи. Например, если каждая особь представляет собой программу, то возможны так называемые *операции, изменяющие архитектуру*, (*architecture-altering operations*) – например, добавление подпрограммы [10].

Генетические алгоритмы и генетическое программирование основаны на моделировании процесса эволюции и являются частью более общей группы методов – эволюционных вычислений [25].

1.1.3. Построение конечных автоматов-распознавателей по обучающим примерам

Из теории формальных языков известно, что конечные детерминированные автоматы способны распознавать регулярные языки [16]. В связи с этим актуальна задача построения автомата, распознающего по множеству примеров некий язык. Задача может быть усилена до построения автомата с минимальным числом состояний. Однако, в работе [27] показано, что эта задача является NP-трудной.

В настоящем разделе приводится обзор алгоритмов машинного обучения для задачи построения автоматов-распознавателей по обучающим примерам.

1.1.3.1. Эволюционные алгоритмы и их сравнение с алгоритмами, основанными на объединении состояний

В работе [34] производится сравнение алгоритмов машинного обучения для построения конечных автоматов-распознавателей. При этом рассматриваются два типа алгоритмов: эволюционные и основанные на построении префиксного дерева (бора) и дальнейшем объединении состояний на основе свидетельств (*Evidence-Driven State Merging*, далее *EDSM*).

Входными данными для алгоритма построения конечного автомата-распознавателя является набор слов, для каждого из которых известно, должно ли оно допускаться автоматом или не должно. Этот набор слов в дальнейшем будет называться множеством обучающих примеров. Выходными данными для рассматриваемого алгоритма является описание построенного конечного автомата, который удовлетворяет входным данным.

Для представления конечного автомата-распознавателя в рассматриваемой работе применяется табличная форма задания его функции переходов. При этом начальным состоянием всегда считается состояние, имеющее номер «0». В рассматриваемой работе рассматриваются только автоматы, обрабатывающие слова над двухсимвольным алфавитом, поэтому число возможных таблиц переходов для n состояний равно n^{2n}. Если учитывать то, что каждое состояние автомата может быть допускающим или недопускающим, то общий размер пространства поиска равен $2^n \cdot n^{2n}$.

Для сокращения пространства поиска в работе [34] применяется специальный метод определения того, какие состояния автомата являются допускающими, а какие – нет. Для этого предлагается использовать так называемый «алгоритм расстановки пометок« на состояниях. Он состоит в следующем. Для каждой строки t из множества обучающих примеров на единицу увеличивается значение $h[f(t)][c]$, где $f(t)$ – номер состояния, в которое приходит рассматриваемый конечный автомат после обработки строки t, а число c равно единице, если строка s должна допускаться автоматом, и равно нулю, если не должна допускаться автоматом. После этого те состояния s, для которых $h[s][1] > h[s][0]$, помечаются как допускающие, а все остальные – как недопускающие. За счет применения описанного алгоритма размер пространства поиска существенно сокращается – его размер становится равным n^{2n}.

В качестве эволюционного алгоритма в рассматриваемой работе применяется так называемая (1+1)-эволюционная стратегия (hill climbing). Работа этого алгоритма осуществляется следующим образом: хранится текущее решение и на каждой итерации к текущему решению применяется операция мутации. Если эта мутация приводит к увеличению значения функции приспособленности, то эта мутация принимается и текущее решение заменяется новым. В противном случае текущее решение не изменяется. Если за заданное заранее число итераций не происходит увеличение значения функции приспособленности, то текущее решение записывается, а процесс поиска решения перезапускается. Отметим, что рассматриваемый алгоритм не использует операцию скрещивания.

Для вычисления функции приспособленности определяется доля строк из обучающего набора, которые правильно классифицируются (допускаются или не допускаются) автоматом.

Сравнение эволюционного алгоритма проводилась на двух наборах тестовых примеров. Первый набор тестовых данных предложен в работах [37, 44]. На этом наборе сравнение проводилось не только с алгоритмом *EDSM*, но и с алгоритмом генетического программирования, предложенным в [37]. Это сравнение показывает, что разработанный алгоритм эволюционного программирования превосходит и алгоритм *EDSM*, и алгоритм генетического программирования, как по точности построенных автоматов, так и по времени работы. Например, среднее время, которое требуется предложенному в рассматриваемой работе алгоритму для построения конечного автомата, составляет 1.6 миллисекунды, а алгоритму *EDSM* требуется 37 миллисекунд в среднем. Вычислительные эксперименты проводились на компьютере с процессором *Pentium IV* с тактовой частотой 2.4 ГГц. Алгоритмы реализовывались на языке программирования *Java*.

Второй набор тестовых данных строился следующим образом:

1. Выбиралось число n.
2. Генерировался случайный ориентированный граф из $5n/4$ вершин, в котором каждая вершина имеет степень два.
3. Случайным образом выбиралась вершина – начальное состояние.
4. Рассматривались все достижимые из начального состояния.
5. Каждое из состояний с равной вероятностью становилось либо допускающим, либо недопускающим.
6. После этого генерировалось множество строк над двухсимвольным алфавитом, для каждой из которых с помощью построенного на шагах 1–5 конечного автомата определялось, допускается она или нет.

На этом тестовом наборе разработанный эволюционный алгоритм при n = 4, 8, 16 превосходил алгоритм *EDSM*, но при n = 32 ему уступал.

На основании проведенных экспериментов был сделан вывод, что разработанный эволюционный алгоритм превосходит алгоритм *EDSM* в том случае, когда необходимо построить автомат с относительно небольшим числом состояний, а обучающее множество примеров содержит небольшую долю строк заданной длины.

В работе [35] также рассматривается задача построения конечных автоматов-распознавателей с помощью эволюционных алгоритмов. Предлагаемый в рассматриваемой работе эволюционный алгоритм является улучшенной версией алгоритма, предложенного в работе [34].

Конечный распознаватель в рассматриваемой работе представляется так же, как и в работе [34] – в хромосому входит таблица переходов, а пометки на состояниях расставляются с помощью специального алгоритма. В качестве эволюционного алгоритма в работе [35] также используется (1+1)-эволюционная стратегия, но операцию мутации предлагается выполнять не так, как в работе [34].

В работе [34] при выполнении мутации случайным образом выбиралась одна из ячеек таблицы переходов, после чего значение в ней заменялось на случайно сгенерированное. Предлагаемый в рассматриваемой работе метод выполнения операции мутации учитывает поведение автомата при обработке обучающего набора.

Для каждого перехода t в автомате вычисляются две величины. Число строк, которые были правильно классифицированы автоматом и при обработке которых использовался переход t, обозначается как $c(t)$. Число строк, неправильно классифицированных автоматом и при обработке которых использовался переход t, обозначается как $e(t)$. После этого вероятность того, что переход t будет выбран при выполнении операции мутации, вычисляется по следующей формуле: $p(t) = \frac{e(t)}{c(t) + e(t)}$. При этом считается, что $p(t)$=0, если $e(t)$=0. Такой подход к выполнению операции мутации требует выполнения двух проходов для вычисления значения функции

приспособленности и указанных вероятностей. На первом из этих проходов неизвестно, какие состояния являются допускающими, а какие – нет, так как он необходим для алгоритма пометки состояний. На втором известно, какие состояния являются допускающими, поэтому указанные вероятности могут быть вычислены.

Кроме описанного способа выполнения операции мутации в рассматриваемой работе также предлагается следующий способ – для каждого состояния запоминаются строки, которые оно должно принимать, и строки, которое оно не должно принимать. В соответствии с алгоритмом расстановки пометок, состояние будет допускающим, если строк, которое оно должно допускать больше, чем строк, которое оно не должно допускать, и недопускающим – в противном случае. При выполнении операции мутации случайным образом выбирается строка, которую автомат классифицирует неправильно. После этого случайным образом выбирается один из переходов, которые используется при обработке выбранной строки. Изменению подвергается та ячейка таблицы переходов, которая ему соответствует.

Для сравнения алгоритмов машинного обучения в рассматриваемой работе применялось два набора тестовых данных. Первый строился так же, как второй набор тестовых данных в работе [34] (описание приведено выше). Второй набор тестовых данных был зашумленным – для некоторых строк из обучающего набора было неправильно указано, должны они допускаться автоматом или нет.

На первом наборе тестовых данных проводилось сравнение четырех эволюционных методов машинного обучения:

- метода, не использующего алгоритм расстановки пометок (*plain*);
- метода из работы [34] (*smart labeling*);
- метода, использующего первый из предлагаемых в настоящей работе подходов к выполнению операции мутации (*sampled*);
- метода, использующего второй из предлагаемых в настоящей работе подходов к выполнению операции мутации (*quick-sampled*).

Результаты вычислительных экспериментов показали, что при восьми состояниях автомата, на основе которого генерировались обучающие наборы, из 50 случаев алгоритм *plain* построил автомат в 26 случаях, алгоритм *smart labeling* – в 42, алгоритм *sampled* – в 47, а алгоритм *quick-sampled* – в 43. При использовании автомата из 16 состояний результаты были такими: *plain* – 5 из 50, *smart* – 21 из 50, *sampled* – 26 из 50, *quick-sampled* – 4 из 50. Если же в исходном автомате было 32 состояния, то только два алгоритма справились с задачей: *smart* правильно строил автомат в 6 случаях из 50, а *sampled* – в 12 случаях из 50.

На втором наборе тестовых данных, который содержал зашумленные обучающие наборы, сравнение проводилось с алгоритмом *EDSM* и с другими алгоритмами, участвовавшими в соревновании *GECCO 2004* [33]. Результаты вычислительных экспериментов показали, что предлагаемый в рассматриваемой работе эволюционный алгоритм (*sampled*) превосходит все существующие методы машинного обучения конечных автоматов на этом обучающем наборе по точности построения автоматов. При этом отметим, что предложенный алгоритм эффективен только для построения автоматов из относительно небольшого числа состояний (порядка нескольких десятков), в то время как с помощью алгоритма *EDSM* могут быть успешно построены автоматы из сотен состояний.

В работе [20] автоматы представлялись с помощью таблицы переходов. Разработанный в этой работе метод позволяет поддерживать в популяции автоматы с разным числом состояний. Функция приспособленности учитывает три компонента – число верно распознанных тестовых примеров, число состояний и переходов автомата, степень общности языка, соответствующего построенному автомату. Это позволяет сузить область поиска, и находить языки, соответствующие определенным

критериям. В обучающий набор могут входить примеры слов, которые как принадлежат, так и не принадлежат языку.

Приведем описание генетической операции репродукции. Число состояний потомка выбирается случайно из диапазона $[N - 2, N + 2]$, где N – число состояний первого родителя. После этого каждый переход копируется из родителей, причем вероятность копирования перехода из заданной особи прямо пропорциональна значению ее функции приспособленности. Переходы, представленные только в одном из родителей, копируются из того родителя, в котором присутствуют. Переходы, отсутствующие в обоих родителях, генерируются случайно.

Генетическая операция мутации осуществляется изменением случайного перехода. При этом переход разрешается удалять. Это приводит к распаду графа переходов на несколько компонент. Результаты экспериментов показали, что удаление недостижимых состояний замедляет вывод – поэтому оно не производится.

Предложенный подход был проверен на практических задачах. Авторам удалось построить автомат из семи состояний, который распознает по множеству из ста примеров русские двухсложные слова.

В работе [43] также рассматривается задача построения конечных автоматов-распознавателей. В качестве метода представления конечных автоматов выбраны двоичные строки, в качестве операций мутации и скрещивания используется однородное скрещивание и мутация.

В качестве исходных данных для генетического алгоритма выступают пары, состоящие из входных слов и последовательностей пометок состояний, которые используются при обработке соответствующих слов.

Рассматривается три варианта функции приспособленности. Первый из них основан на строгом сравнении строк, второй – на вычислении редакционного расстояния, третий – на вычислении длины общего префикса строк. В качестве метода генерации следующего поколения используется метод рулетки.

Алгоритмы, предлагаемые в рассматриваемой работе, реализованы в инструментальном средстве *GeSM*. В этом инструментальном средстве кроме самого генетического алгоритма также реализован алгоритм удаления недостижимых состояний из автоматов.

Экспериментальное исследование разработанных генетических алгоритмов проводилось на задачах построения автоматов для проверки четности, элемента задержки на два такта, распознавателя паттернов и счетчика.

1.1.3.2. Алгоритм, основанный на методах решения задачи о выполнимости булевой формулы

В работе [30] представлен алгоритм построения автоматов-распознавателей по тестовым наборам, принципиально отличающийся от предлагаемых ранее алгоритмов для решения поставленной задачи. Отличие состоит в том, что данный алгоритм основан на сведении поставленной задачи к задаче о выполнимости булевой формулы (*Boolean satisfiability – SAT*).

Напомним, что входными данными для алгоритма построения конечного автомата-распознавателя A является набор слов, для каждого из которых известно, должно ли оно допускаться автоматом (множество слов S^+) или не должно (множество S^-). Подробней опишем этапы предлагаемого алгоритма.

Как и в алгоритме *EDSM*, первым этапом работы алгоритма является построение префиксного дерева (*augmented prefix tree acceptor – APTA*). На рис. 3 приведен пример префиксного дерева, построенного по входным данным $S^+ = \{a, abaa, bb\}$, $S^- = \{abb, b\}$.

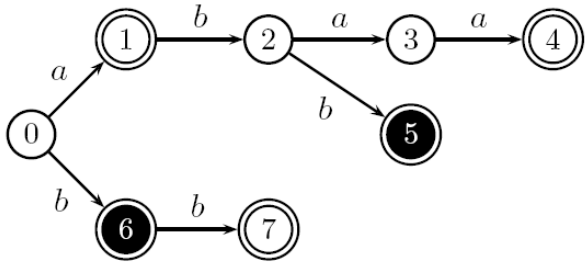

Рис. 3. Пример префиксного дерева

Вторым этапом работы алгоритма является построение графа совместимости по полученному префиксному дереву. Множество вершин этого графа совпадает с множеством вершин префиксного дерева, и две вершины соединены ребром, если они не могут соответствовать одному состоянию искомого автомата-распознавателя. На рис. 4 приведен пример графа совместимости, построенного по префиксному дереву, приведенному на рис. 3.

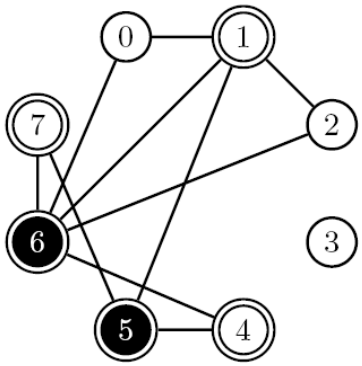

Рис. 4. Пример графа совместимости

Третий, самый сложный этап, заключается в построении и упрощении КНФ-формулы, кодирующей требования, накладываемые графом совместимости на автомат-распознаватель. В работе сравниваются два способа построения КНФ-формулы. Первый способ, носящий название *direct encoding*, строит формулу, состоящую из $O(k^2|S|^2)$ скобок. Здесь, как k обозначен размер искомого автомата, а $|S|$ – суммарная длина обучающих примеров. Второй способ, предложенный авторами, строит формулу, состоящую из $O(k^2|S|)$ скобок. Таким образом, авторами предложен способ построения КНФ-формулы, существенно превосходящий по качеству существующие методы. Также авторы предложили набор способов упрощения полученной формулы, используя методы нахождения полных подграфов в графе совместимости.

Полученная КНФ-формула подавалась на вход программе *picosat*, находящей выполняющую постановку для формулы. Затем, по найденной выполняющей подстановке строился искомый автомат-распознаватель.

Результаты экспериментальных исследований показали, что производительность предлагаемого метода выше всех ранее рассматриваемых методов построения автоматов-распознавателей.

На основании полученных в рассматриваемой работе результатов сделан вывод, что разработка методов машинного обучения на основе методов решения задачи о выполнимости булевой формулы является перспективной областью.

1.1.4. Построение конечных преобразователей по обучающим примерам

По построению конечных преобразователей на основе обучающих примеров существует намного меньше работ, чем по построению конечных распознавателей. Одной из работ по конечным автоматам-распознавателям является работа [36]. Такие автоматы при обработке символов входного слова при выполнении переходов записывают в выходной поток символы выходного алфавита, тем самым осуществляя преобразование слов одного формального языка в слова другого.

Так же, как и в рассмотренных выше работах [34, 35], посвященных построению конечных автоматов-распознавателей, в работе [36] применяется (1+1)-эволюционная стратегия. Конечный преобразователь представляется в виде набора двух таблиц – таблицы значений функции переходов и таблицы значений функции выходов. При этом предполагалось, что на каждом из переходов конечный преобразователь может вывести не более одного символа.

Входными данными для построения конечного автомата-распознавателя является набор обучающих примеров – пар слов, одно из которых является входным, а второе – соответствующим ему выходным.

Опишем алгоритм выполнения операции мутации в алгоритме, предложенном в рассматриваемой работе. Выполняется мутация одного из трех типов:

- добавление состояния – выполняется с вероятностью 0.1, если автомат содержит меньше заданного максимального числа состояний. При этом переходы из нового состояния генерируются случайным образом;
- удаление состояния;
- случайное изменение одного перехода.

Последние две мутации из перечисленных выбираются равновероятно.

В рассматриваемой работе рассматриваются три варианта функции приспособленности:

- на основе строгого сравнения (*strict*);
- на основе вычисления расстояния Хэмминга [29] (*hamming*);
- на основе вычисления редакционного расстояния (расстояния Левенштейна) [6] (*edit*).

Опишем указанные функции приспособленности подробнее. Пусть конечный автомат-преобразователь, задаваемый рассматриваемой особью эволюционного алгоритма, на обучающем примере, который содержит входную строку *s* и эталонную выходную строку *t*, выдал строку *q*. При использовании строго сравнения значение функции приспособленности f будет вычисляться по следующей формуле: $f = \begin{cases} 0, t = q \\ 1, t \neq q \end{cases}$.

При использовании расстояния Хэмминга функция приспособленности вычисляется по формуле: $f = \frac{\sum_{i=1}^{\min(|t|,|q|)} \Delta(t_i, q_i)}{\max(|t|,|q|)}$, где как $|t|$ и $|q|$ обозначены длины строк t и q соответственно. Как Δ в формуле обозначена функция, значение которой равно нулю, если значения ее аргументов совпадает, и единице – в противном случае.

При использовании редакционного расстояния функция приспособленности вычисляется по формуле: $f = \frac{\text{edits}(t, q)}{\max(|t|,|q|)}$. Здесь как edits($t$, q) обозначено редакционное расстояние между строками t и q, которое равно минимальному числу операций вставки, замены или удаления символа, которое необходимо выполнить, для того чтобы преобразовать строку q в строку t. Отметим, что значения всех трех указанных функций приспособленности находится в отрезке от 0 до 1, а меньшее значение соответствует большему соответствию выходной строки эталону.

Экспериментальное исследование методов машинного обучения проводилось на задаче построения конечного преобразователя. Преобразователь предназначен для преобразования представления двумерного изображения из цепного кода для четырех направлений в цепной код для восьми направлений. Использовались два набора тестовых данных – в «простом» было достаточно много коротких строк, а в «сложном» их доля была намного меньше.

Результаты экспериментов показали, что наилучшие результаты на «сложном» тестовом наборе достигаются при использовании функции приспособленности, основанной на редакционном расстоянии. Второй по эффективности является функция, основанная на расстоянии Хэмминга, а третьей – основанная на строгом сравнении. На «простом» наборе ситуация примерно такая же, только функции *strict* и *hamming* меняются местами.

1.1.5. Построение управляющих конечных автоматов на основе принципа «обучения на собственных ошибках»

В работе [42] рассматривается задача построения управляющего конечного автомата на основе принципа «обучения на собственных ошибках». В указанной работе рассматривается задача построения автоматов для игры «Соревнование за ресурсы» (*Competition for Resources*). Игровое поле в этой игре имеет форму тора размером 10 на 10 клеток (тор можно представить в виде квадрата, у которого верхняя сторона склеена с нижней, а правая – с левой). Оно моделирует компьютерную сеть, в которой могут распространяться программные агенты.

В игре участвуют два агента. Находясь в некоторой клетке поля, агент получает информацию о четырех соседних клетках (соседи справа, слева, сверху и снизу). Каждая клетка может находиться в одном из трех состояний – никому не принадлежать, принадлежать первому агенту или принадлежать второму агенту. На основании полученной информации агент может переместиться в одну из соседних клеток, при этом перемещение разрешается выполнять либо в клетку, которая никому не принадлежит, либо в клетку, принадлежащую рассматриваемому агенту. Агент обязан совершить перемещение.

Изначально первый агент находится в левом верхнем углу, а второй – в правом нижнем. Агенты делают ходы по очереди. Игра заканчивается, когда каждая клетка принадлежит либо первому агенту, либо второму. Победителем признается тот агент, которому на момент окончания игры принадлежит большее число клеток. В случае одинакового числа клеток, принадлежащих агентам, победителем признается агент, ходящий вторым.

В рассматриваемой работе стратегия поведения второго агента была задана, а цель состояла в построении конечного автомата для управления первым агентом, который позволял бы достаточно часто выигрывать игру. Стратегия второго агента в рассматриваемой была такой: на каждом ходе при наличии хотя бы одной никому не принадлежащей соседней клетки, перемещение осуществлялось в случайно выбранную из них, а при отсутствии – случайно выбиралась одна из соседних клеток, принадлежащих этому агенту. Таким образом, стратегия второго агента имеет стохастический характер.

Для решения описанной задачи в работе [42] применяется генетический алгоритм. Конечный автомат представляется в виде таблицы значений функции переходов и таблицы значений функции действий. В работе рассматривалось построение автоматов, содержащих от одного до десяти состояний. При построении автомата из n состояний размер таблиц значений указанных функции составляет $n \times 80$, так как существует 80 комбинаций входных переменных. Это объясняется тем, что имеется четыре переменные (соответствующие состояниям соседних клеток), каждая из которых может принимать три возможных значения, при этом ровно одна комбинация (все четыре соседние клетки принадлежат другому агенту) не может встретиться в игре. Это объясняется тем, что клетка, из которой агент пришел в текущую, принадлежит ему.

Мутации с некоторой заданной вероятностью подвергалась каждая из ячеек таблицы значений функции переходов и функции действий. При этом с вероятностью p изменялось состояние, в которое ведет переход, а с вероятностью $(1-p)$ – действие, выполняемое на этом переходе. Кроме того, использовалась операция однородного скрещивания. Для каждой позиции (i, j) в таблицах значений функций переходов и действий производился обмен ячеек между «родителями» с некоторой заданной вероятностью.

Так как поведение второго агента носит вероятностный характер, то оптимальным вариантом функции приспособленности была бы вероятность выигрыша для первого агента, поведение которого задается конечным автоматом, описываемым особью. Однако вычисление этой вероятности сопряжено с перебором всех вариантов поведения первого агента и требует больш*и*х вычислительных ресурсов. Поэтому указанная вероятность вычисляется приближенно – вычисляется не вероятность выигрыша, а доля игр, выигранных агентом.

Вычисление функции приспособленности выполняется в два этапа – на первом из них для всех автоматов, описываемых особями текущего поколения, проводится 500 игр против заданной стратегии поведения второго агента. Если в текущем поколении находится автомат, который по результатам этого вычисления показывает результат, превосходящий наилучший автомат, рассмотренный в процессе работы алгоритма, то для автомата из текущего поколения проводится более точное вычисление функции приспособленности – по итогам 10000 игр.

В качестве метода отбора, применяемого при генерации нового поколения, использовалась комбинация пропорционального отбора и элитизма. Это означает, что особь, имеющая наибольшее значение функции приспособленности в текущем поколении напрямую переходила в следующее, а для остальных вероятность перехода пропорциональна значению функции приспособленности.

Вычислительные эксперименты проводились при значениях $n = 1..10$. Их результаты показывают, что при решении этой задачи существенное преимущество имеют автоматы, имеющие более двух состояний. Автоматы с числом состояний от трех до десяти показывали примерно одинаковые результаты – около 90% побед. Кроме этого, были проведены эксперименты, в которых в процессе мутации разрешалось добавлять и удалять состояния. При этом было замечено, что в построенных автоматах используется меньше половины состояний. На основании результатов экспериментов был сделан вывод о том, что операции добавления и удаления состояний являются слишком «разрушительными» для их применения в эволюционных алгоритмах построения конечных автоматов.

Анализ поведения агентов, управляемых построенными автоматами, показал, что поведение агента достаточно быстро зацикливается, что приводит к ухудшению его результатов. Для того чтобы устранить этот недостаток, в рассматриваемой работе предлагается метод, основанный на наблюдении за поведением агента во время игры. Этот метод состоит в том, что к автомату добавляется память размером в 20 ячеек, которая позволяет отслеживать циклы в поведении агента. В этой памяти записывались ячейки, посещенные агентом. Если далее обнаруживалось, что действие, генерируемое автоматом, приводит к зацикливанию, то действие выбиралось случайным образом. Применение такого метода позволило построить систему управления агентом, которая добивалась побед в 96% игр.

В рамках исследований по теме «Технология генетического программирования для генерации автоматов управления системами со сложным поведением» на кафедре «Технологии программирования» СПбГУ ИТМО был разработан ряд методов генерации конечных автоматов с помощью генетических алгоритмов [11–14].

Один из таких методов – метод сокращенных таблиц переходов был предложен Н. Поликарповой и В. Точилиным в [9]. Этот метод позволяет решить проблему экспоненциального роста размера описания автомата, которая возникает при использовании полных таблиц переходов –

число строк в таблице есть 2^n, где n – число предикатов. Опыт показывает, что в реальных задачах управляющие автоматы, построенные вручную, имеют гораздо меньше переходов, чем $|S| \cdot 2^n$ (здесь за $|S|$ обозначен размер множества состояний автомата). Причина этого состоит в том, что в большинстве задач предикаты имеют «локальную природу» по отношению к управляющим состояниям. В каждом состоянии *значимым* является лишь определенный, небольшой поднабор предикатов, остальные же не влияют на значение управляющей функции. Именно это свойство позволяет существенно сократить размер описания состояний. Кроме того, использование этого свойства в процессе оптимизации позволяет получить результат, более похожий на автомат, построенный вручную, а, следовательно, и более понятный человеку.

Свойство локальности предикатов можно применять для сокращения описания управляющего состояния разными способами. При разработке метода сокращенных таблиц был выбран один из подходов, при котором число значимых в состоянии предикатов ограничивается некоторой константой *r*.

К таблице, задающей сужение управляющей функции на данное состояние, в этом случае добавляется битовый вектор, описывающий множество значимых предикатов (рис. 5).

x_0	x_1	x_2	x_3	x_4	x_5
0	1	0	1	0	0

x_1	x_3	S	z_0	z_1	z_2
0	0	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	1	2	1	1	1

Рис. 5. Хромосома состояния: сокращенная таблица (n = 6, m = 3, r = 2, |S| = 3)

Число строк таблицы в этом случае 2^r, однако, константа *r* обычно невелика. Ее выбор зависит от сложности задачи. Как показывает опыт, для большинства автоматизированных объектов среднее по всем состояниям значение r не больше пяти.

Второй метод – метод представления автоматов деревьями решений предложен В. Даниловым в [4, 5]. Дерево решений является удобным способом задания дискретной функции, зависящей от конечного числа логических переменных. Оно представляет собой помеченное дерево, метки в котором расставлены по следующему правилу:

- внутренние узлы помечены символами переменных;
- ребра – значениями переменных;
- листья – значениями искомой функции.

Для определения значения функции по значениям переменных необходимо спуститься от корня до листа, и сформировать значение, которым помечен полученный лист. При этом из вершины, помеченной переменной *x*, переход производится по тому ребру, которое помечено тем же значением, что и значение переменной *x*.

Метод представления автомата с помощью деревьев решений заключается в следующем: функции переходов и выходов автомата выражаются с помощью деревьев решений. Более формально: зададим для каждого состояния $q \in Q$ функцию $\sigma_q : X \to Q \times Y$, такую что $\sigma_q(x) = (\delta(q, x), \lambda(q, x))$ для $\forall x \in X$. Здесь Q – множество состояний автомата, X – множество входных воздействий, Y – множество выходных воздействий, $\delta : Q \times X \to Q$ – функция переходов, $\lambda : Q \times X \to Y$ – функция выхода. Каждая из этих функций может быть определена

собственным деревом решений. Таким образом, автомат в целом может быть представлен упорядоченным набором деревьев решений и стартовым состоянием.

Каждое состояние автомата представляется с помощью соответствующего дерева решений. Деревья решений состоят из узлов, среди которых выделяется корень. Каждый узел представляется следующим образом:

- номер переменной, соответствующей метке узла;
- указатель на дочерний узел, соответствующий нулевому значению переменной (для внутренних узлов);
- указатель на дочерний узел, соответствующий единичному значению переменной (для внутренних узлов);
- ассоциированное выходное действие (для листьев);
- номер состояния, в которое ведет переход из данной вершины (для листьев).

Следовательно, при использовании описываемого метода автомат представляется объектом следующего вида на языке *Java*:

```
class TreeAutomata {
    Tree[] trees;
    int startState;
}

class Tree {
    Node root;

    private class Node {
        Node left;
        Node right;
        int transState;
        int action;
    }
}
```

Третий метод – метод совместного применения конечных автоматов и нейронных сетей был предложен Ф. Царевым в работах [17, 18]. При использовании этого метода для управления системой со сложным поведением предлагается совместно применять нейронную сеть и конечный автомат. Данные объекты совместно строятся генетическим алгоритмом.

При этом, как отмечалось выше, нейронная сеть используется для классификации значений вещественных входных переменных и выработки входных логических переменных для автомата, а автомат – для выработки выходных воздействий на систему со сложным поведением (рис. 6).

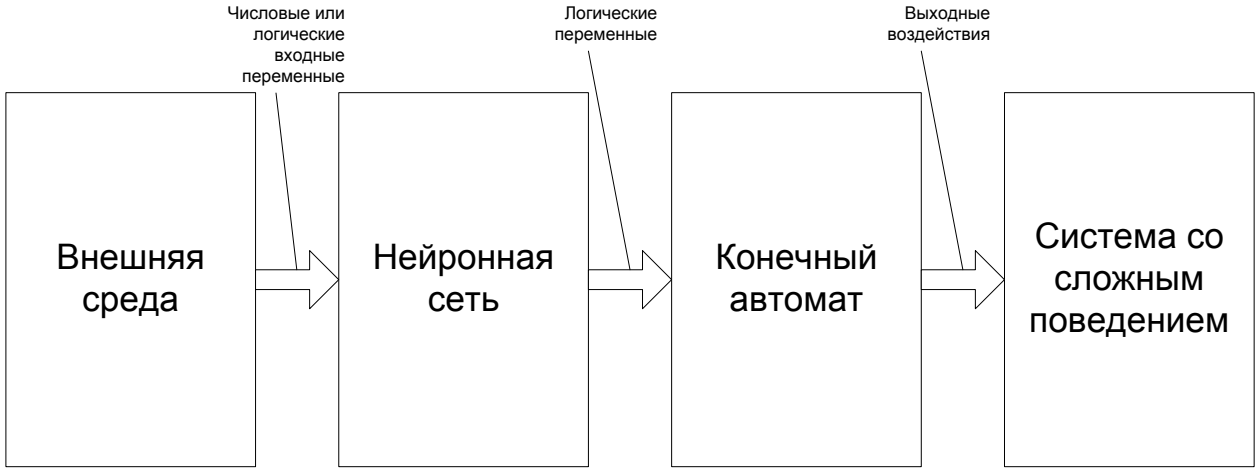

Рис. 6. Структурная схема системы управления при использовании метода совместного применения конечных автоматов и нейронных сетей

Одна из возможных структур нейронной сети и способ ее взаимодействия с конечным автоматом показаны на рис. 7. Отметим, что для решения других задач структура нейронной сети может быть изменена.

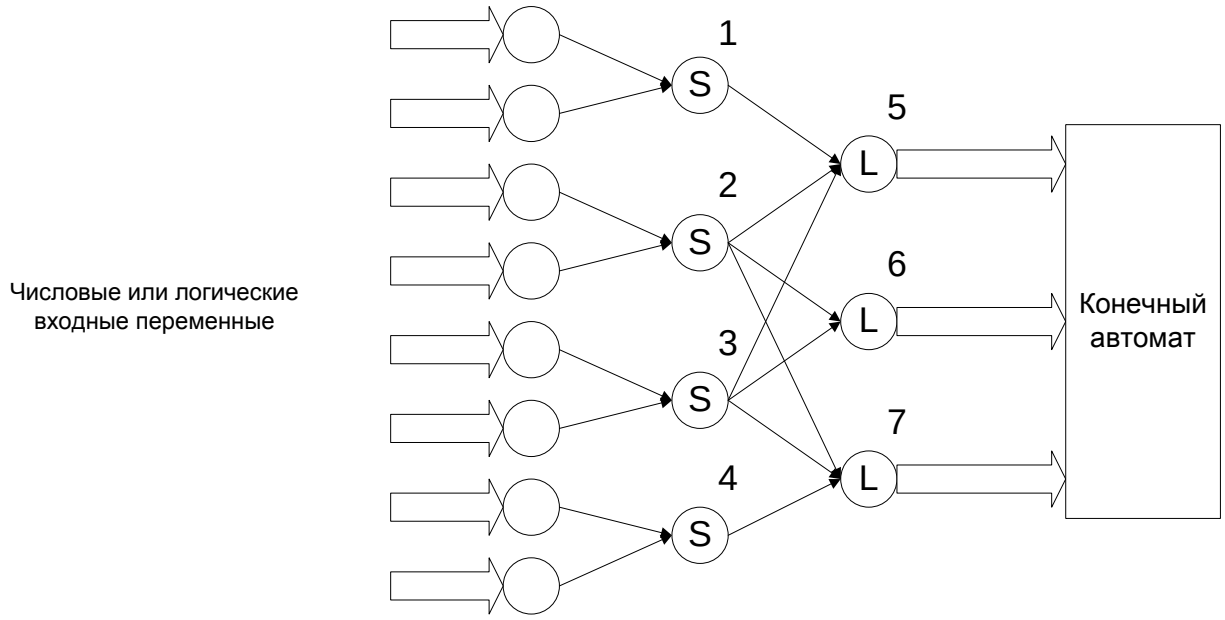

Рис. 7. Возможная структура нейронной сети и ее взаимодействие с конечным автоматом

Символами *S* на рис. 7 обозначены нейроны с сигмоидальной функцией активации, символами *L* – нейроны с пороговой функцией активации. Рядом с нейронами указаны их номера (они используются при описании операции скрещивания нейронных сетей). На каждый из трех выходов нейронной сети поступает число равное нулю или единице. Таким образом, существует восемь вариантов комбинаций выходных сигналов нейронной сети (000, 001, 010, 011, 100, 101, 110, 111), подаваемых на вход конечного автомата.

Экспериментальное исследование указанных трех методов проводилось на построении автомата для задачи «Умный муравей-3».

Приведем сначала описание классической постановки задачи «Умный муравей» [19, 31]. Используется двумерный тор размером 32 на 32 клетки (рис. 8). На некоторых клетках поля расположены яблоки – черные клетки на рис. 8. Яблоки расположены вдоль некоторой ломаной линии, но не на всех ее клетках. Клеткам ломаной, на которых яблок нет, соответствует серый цвет. Белые клетки не принадлежат ломаной и не содержат яблок. Всего на поле 89 яблок.

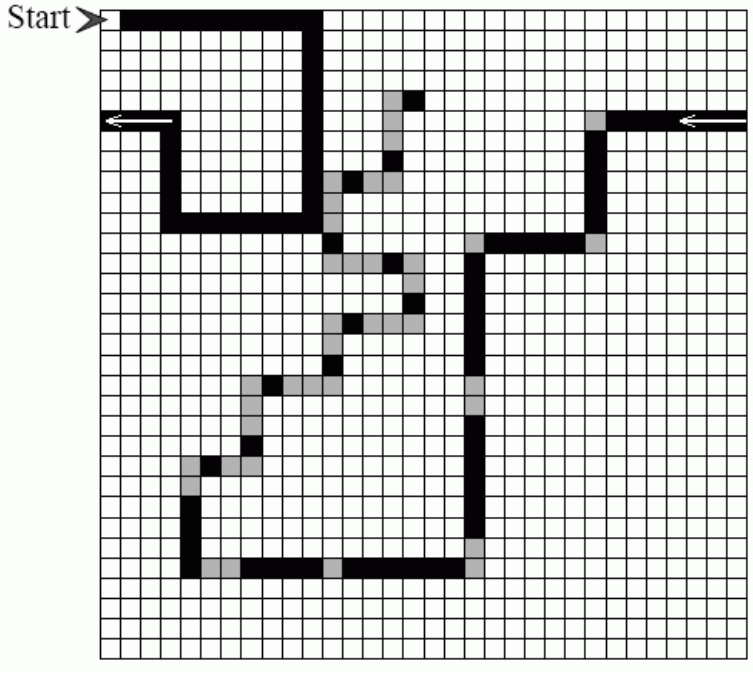

Рис. 8. Поле с яблоками

В клетке с пометкой «Start» находится муравей. Он занимает клетку поля и смотрит в одном из четырех направлений (соседи справа, слева, сверху и снизу). В начале игры муравей смотрит на восток. Он умеет определять находится ли яблоко непосредственно перед ним. За ход муравей совершает одно из четырех действий:

- идет вперед на одну клетку, съедая яблоко, если оно находится перед ним;
- поворачивается вправо;
- поворачивается влево;
- стоит на месте.

Съеденные муравьем яблоки не восполняются. Муравей жив на всем протяжении игры – еда не является необходимым ресурсом для его существования. Никаких других персонажей, кроме муравья, на поле нет. Ломаная *строго задана*. Муравей может ходить по любым клеткам поля. Игра длится не более 200 ходов, на каждом из которых муравей совершает одно из четырех описанных выше действий. В конце игры подсчитывается число яблок, съеденных муравьем. Это значение является результатом игры.

Цель игры – создать муравья, который не более чем за 200 ходов съест как можно больше яблок. Муравьи, съевшие одинаковое количество яблок, заканчивают игру с одинаковым результатом вне зависимости от числа ходов, затраченных каждым из них на процесс еды. Однако эта задача может иметь различные модификации, например такую, в которой при одинаковом количестве съеденных яблок, лучшим считается муравей, съевший яблоки за меньшее число ходов. Ниже будет показано, что поведение муравья может быть задано конечным автоматом. При этом может быть поставлена задача о построении автомата с минимальным числом состояний для муравья,

съедающего все яблоки, или автомата для муравья, съедающего максимальное число яблок при заданном числе состояний.

Постановка задачи «Умный муравей-3», предложенной в работе [1], содержит несколько существенных отличий.

Во-первых, расширена область обзора муравья – вместо одной клетки он видит восемь. Таким образом, множество значений входных переменных содержит $2^8 = 256$ элементов. На рис. 9 изображена область обзора муравья (клетка, в которой находится муравей, обозначена серым цветом).

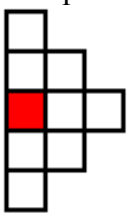

Рис. 9. Область видимости муравья

Во-вторых, расположение еды на поле не фиксировано, а генерируется случайным образом. При этом вероятность того, что яблоко окажется в некоторой клетке, одинакова для всех клеток поля и равна μ.

В этом случае число яблок, съеденных муравьем за 200 ходов, есть случайная величина ξ (определяемая муравьем) на дискретном множестве элементарных исходов Ω – множестве расположений еды – битовых матриц 32×32. Каждому исходу ω_i, содержащему k единиц поставим в соответствие вероятность $p(\omega_i) = \mu^k(1-\mu)^{n-k}$, где $n = 32\times32$.

Для вычисления этой величины в общем случае необходимо перебрать все возможные битовые матрицы размером 32×32. Поэтому для оценки эффективности автомата, задающего поведение муравья, вместо точного вычисления этого математического ожидания, оно будет вычислять приближенно – с помощью моделирования поведения муравья на 10000 случайно сгенерированных полях.

Экспериментальные исследования, проведенные в [14], показали, что при построении автомата для задачи «Умный муравей–3» указанные три метода представления автоматов показали примерно одинаковые результаты – максимальное достигнутое значение функции приспособленности составило примерно 27 для автоматов из 4–16 состояний.

1.1.6. Сравнение рассмотренных алгоритмов машинного обучения

В табл. 1 приведено сравнение рассмотренных в обзоре алгоритмов машинного обучения для построения конечных автоматов. Сравнение алгоритмов проводится по следующим параметрам:

- Тип автоматов, которые строятся (распознаватель, преобразователь, управляющий автомат).
- Задача, на которой проводится экспериментальное исследование.
- Метод представления конечного автомата.
- Тип используемого алгоритма.
- Особенности предлагаемого алгоритма.
- Тип алгоритма машинного обучения.

Таблица 1. Сравнение рассмотренных в обзоре алгоритмов машинного обучения для построения конечных автоматов

Название работы	Рассматриваемая модель	Метод представления используемой модели	Тип используемого алгоритма	Особенности алгоритма	Тип алгоритма машинного обучения
Learning DFA: Evolution versus Evidence Drive State Merging (2003 год)	Конечные автоматы-распознаватели	Таблица значений функции переходов	**Эволюционная стратегия (1+1)**	Для определения допускающих состояний используется алгоритм расстановки пометок	Обучение на основе примеров
Learning Deterministic Finite Automata with a Smart State Labeling Algorithm (2005 год)	Конечные автоматы-распознаватели	Таблица значений функции переходов	**Эволюционная стратегия (1+1)**	Для определения допускающих состояний используется алгоритм расстановки пометок, предложен метод выполнения операции мутации, учитывающий поведение автомата на обучающих примерах	Обучение на основе примеров
A Genetic Algorithm for Finite State Automata Induction with Application to Phonotactics (1998 год)	Конечные автоматы-распознаватели	Таблица значений функции переходов	**Генетический алгоритм**	Функция приспособленности учитывает не только долю правильно классифицированных обучающих примеров, но и размер конечного автомата	Обучение на основе примеров
Genetic Inference of Finite State Machines (2007 год)	Конечные автоматы-распознаватели	В виде битовых строк	**Генетический алгоритм**	Анализируется не тип состояния, в которое пришел автомат после обработки строки, а последовательность типов состояний, которые были посещены в процессе обработки строки. Используется два типа функции приспособленности.	Обучение на основе примеров

Exact DFA Identification Using SAT Solvers (2010 год)	Конечные автоматы-распознаватели	Таблица значений функции переходов	**Сведение к задаче о выполнимости булевой формулы**	Производится построение графа совместимости, его преобразование в КНФ-формулу, выполняющий набор для которой находится методами решения задачи о выполнимости булевой формулы.	Обучение на основе примеров
Evolving Finite-State Transducers: Some Initial Explorations (2003 год)	Конечные автоматы-преобразователи	Таблица значений функции переходов и таблица значений функции выходов	**Эволюционная стратегия (1+1)**	В работе рассматриваются три варианта функции приспособленности	Обучение на основе примеров
Evolving Finite-State Machine Strategies for Protecting Resources (2000 год)	Управляющие конечные автоматы	Таблица значений функции переходов и таблица значений функции выходов	**Генетический алгоритм**	Для предотвращения зацикливания поведения автомата применялась динамическая верификация.	**Обучение на собственных ошибках**
Применение генетического программирования для генерации автоматов с большим числом входных переменных (2008 год)	Управляющие конечные автоматы	Функции переходов и действий в виде сокращенных таблицы	**Генетический алгоритм**	Сокращенные таблицы позволяют учитывать в каждом состоянии только так называемые «значимые предикаты»	**Обучение на собственных ошибках**
Метод представления автоматов деревьями решений для использования в генетическом программировании (2008 год)	Управляющие конечные автоматы	Функции переходов и действий в виде деревьев решений	**Генетический алгоритм**	Деревья решений позволяют задавать в каждом состоянии свой приоритет переменных	**Обучение на собственных ошибках**

Совместное применение генетического программирования, конечных автоматов и искусственных нейронных сетей для построения системы управления беспилотным летательным аппаратом (2008 год)	Управляющие конечные автоматы	Таблица значений функции переходов и таблица значений функции выходов, нейронная сеть для обработки входных переменных	**Генетический алгоритм**	Нейронные сети выполняют преобразование входных вещественных переменных в логические.	**Обучение на собственных ошибках**

1.2. Программные средства для решения задачи о выполнимости булевой формулы

В настоящем разделе приводится аналитический обзор программных средств для решения задачи о выполнимости булевой формулы (в дальнейшем такое средство будем также называть *SAT-solver*) Будут рассмотрены наиболее известные программные средства, выпущенные за последнее десятилетие.

Все программные средства объединяет то, что булева формула должна быть представлена в конъюнктивной нормальной форме и записана в файл в формате *DIMACS* [41].

1.2.1. Программное средство *zChaff*

Программное средство *zChaff* [45] является реализацией алгоритма *Chaff* [39]. На соревновании *SAT 2002 Competition* это средство был признано лучшим в нескольких категориях. Известны случаи успешного решения им промышленных задач, включающих миллион переменных и десятки миллионов ограничений на переменные.

Программное средство *zChaff* можно скомпилировать в подключаемую библиотеку, что позволяет использовать *SAT-solver* без создания дополнительных файлов. Средство успешно интегрировано в такие продукты, как верификатор моделей *NuSMV* [40] и доказатель теорем *GrAnDe* [28].

Как и большинство алгоритмов, используемых для решения задачи о выполнимости булевой формулы, алгоритм *Chaff* является частным случаем алгоритма *DPLL* [24] (*Davis-Putnam-Logemann-Loveland*). Опишем схему работы данного алгоритма.

Функции *DPLL* подается на вход формула *f*. Если формула является тривиальной, то функция возвращает, разрешима (TRUE) или не разрешима (FALSE) данная формула *f* (имеется ли выполняющая подстановка). Если формула *f* не является тривиальной, то некоторым образом выбирается переменная *v* и рекурсивно запускается функция *DPLL* с допущением *v* = TRUE. Если при данном допущении формула *f* имеет выполняющую подстановку, то возвращаем TRUE – формула разрешима. В противном случае повторяем процесс с допущением *v* = FALSE. Если ни одно из допущений не привело к нахождению выполняющей подстановки, то возвращаем FALSE – формула неразрешима. Иллюстрация работы данного алгоритма приведена на рис. 10.

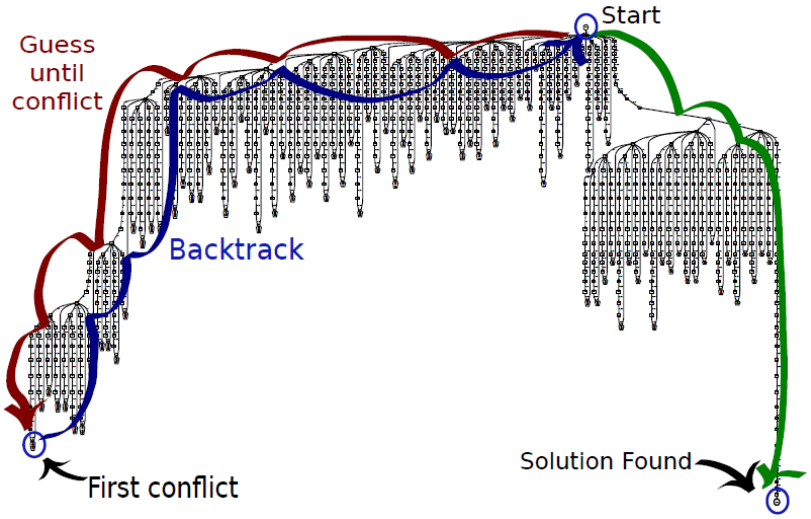

Рис. 10. Иллюстрация работы алгоритма *DPLL*

1.2.2. Программное средство *MiniSat*

MiniSat [26] – одно из самых популярных программных средств для решения задачи о выполнимости булевой формулы. Этот *SAT-solver* популярен за счет того, что авторы выполнили поставленную перед собой задачу – создать средство с малым числом строк исходного кода, который будет понятен начинающим исследователям и разработчикам.

На соревновании *SAT 2005 competition* программное средство *MiniSat* стало победителем в четырех категориях, что позволяет говорить о высокой производительности данного программного средства. На соревнованиях последующих лет *MiniSat* почти всегда попадал в пятерку лучших средств для решения задачи о выполнимости булевой формулы.

Особенностью данного программного средства является использование препроцессора *SatELite* [22], упрощающего исходную задачу за счет удаления некоторых переменных и ограничений на них. На рис. 11 приведены графики, демонстрирующие на двух тестовых наборах улучшение производительности программного средства *MiniSat* за счет использования препроцессора.

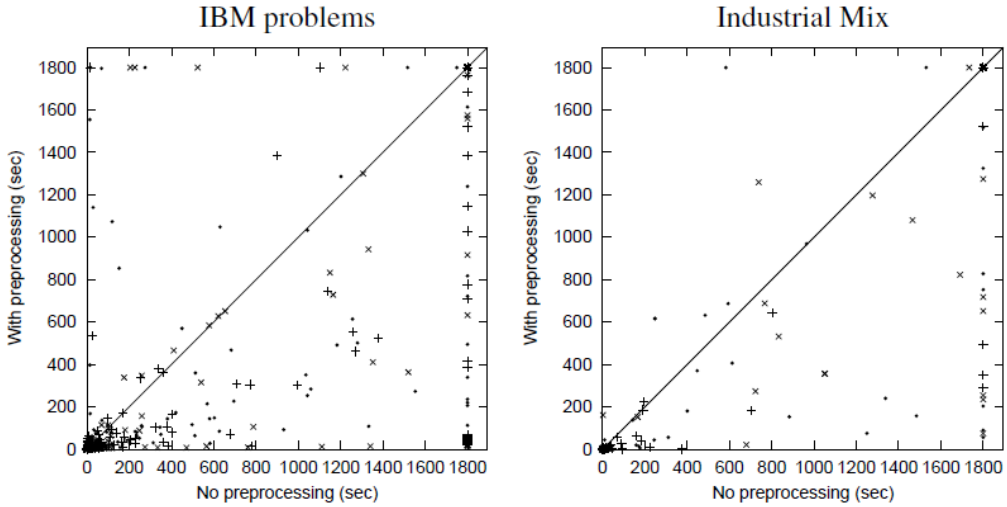

Рис. 11. Сравнение производительности средства *MiniSat* с использованием и без использования препроцессора *SatELite* на двух тестовых наборах

1.2.3. Программное средство *CryptoMiniSat*

Программным средством с самой высокой производительностью на данный момент является *CryptoMiniSat* [23]. Данное средство победило в главной категории (*main track*) соревнования *SAT-Race 2010*. Основной целью его автора являлась разработка программного средства, подбирающего код для некоторого шифрования. Цель была достигнута – за два дня работы *CryptoMiniSat* был подобран ключ для программатора *HiTag2*, используемого в современных системах безопасности автомобилей.

В качестве ядра программы было использовано ранее описанное средство *MiniSat*, содержащее примерно 1500 строк исходного кода, в то время как число строк исходного кода *CryptoMiniSat* достигает 13000. Код является открытым – распространяется по лицензии *GPL*.

Отличительной особенностью *CryptoMiniSat* является то, что он может обрабатывать не только ограничения, записанные через бинарную операцию «или», но и через бинарную операцию «исключающее или» (сложение по модулю два, *XOR*).

В дальнейшем предполагается использование именно этого программного средства.

1.2.4. Программное средство *Plingeling*

Plingeling [21] – распределенная реализация программного средства *Lingeling*. Это средство заняло второе место в главной категории соревнования *SAT-Race 2010.*, Это средство заняло первое место в категории распределенных средств. В данной категории программным средствам разрешалось использовать восемь ядер двух процессоров одного персонального компьютера.

1.2.5. Сравнение рассмотренных программных средств для решения задачи о выполнимости булевой формулы

В табл. 2 приведено сравнение рассмотренных в обзоре программных средств для решения задачи о выполнимости булевой формулы. Сравнение средств проводится по следующим параметрам:

- Язык, на котором написано программное средство.
- Особенности программного средства.
- Награды, полученные данным средством на соревнованиях.

Таблица 2. Сравнение рассмотренных программных средств для решения задачи о выполнимости булевой формулы

Название программного средства, ссылка	Авторы	Язык	Особенности	Награды
zChaff (http://www.princeton.edu/~chaff/zchaff.html)	Lintao Zhang, Yogesh Mahajan (Принстонский университет, США)	*C++*, код открыт	Использование алгоритма Chaff	*SAT 2002 Competition*, Best Complete Solver (категории industrial, handmade)
MiniSat (http://minisat.se/)	Niklas Eén, Niklas Sörensson (CSE, Гётеборгский университет, Швеция)	*C++*, код открыт	Использование препроцессора SatELite	*SAT 2005 competition*, победитель в четырех категориях, множество других наград
CryptoMiniSat (http://www.msoos.org/cryptominisat2)	Mate Soos (INRIA, Франция)	*C++*, код открыт	Использование ядра MiniSat	Победитель основной категории *SAT-Race 2010*
Plingeling (http://fmv.jku.at/lingeling/)	Команда во главе с профессором Армином Биере (институт формальных моделей и верификации, Австрия)	*C++*, код открыт	Распределенное средство	Победитель категории распределенных средств *SAT-Race 2010*

1.3. Выводы по главе 1

1. Алгоритмы машинного обучения обучают параметры модели либо на заранее подготовленных тестовых примерах, либо на собственных ошибках. При этом достаточно распространенным типом моделей являются модели, содержащие состояния. Такие модели могут быть как вероятностными (скрытые Марковские модели), так и детерминированными (конечные автоматы).
2. Конечные автоматы могут использоваться в задачах распознавания языков, в задачах преобразования последовательностей, а также в задачах управления.
3. В большинстве работ по разработке алгоритмов машинного обучения конечных автоматов рассматриваются конечные автоматы-распознаватели. Эксперименты, проведенные в рамках работ по построению других типов конечных автоматов, показывают, что есть надежда на то, что алгоритмы машинного обучения, основанные на эволюционных вычислениях, будут эффективных и для решения задач построения автоматов других типов.
4. Как правило, в эволюционных алгоритмах число состояний всех автоматов, рассматриваемых в процессе работы алгоритма, одинаково (при этом, однако часть состояний в каждом из автоматов может быть недостижима).
5. Один из существующих алгоритмов машинного обучения основан на применении методов решения задачи о выполнимости булевой формулы. Производительность этого метода превосходит методы машинного обучения, основанные на других принципах.

2. ВЫБОР И ОБОСНОВАНИЕ ОПТИМАЛЬНОГО ВАРИАНТА ПРОВЕДЕНИЯ ИССЛЕДОВАНИЙ И ВЫБОР АРХИТЕКТУРЫ МЕТОДА МАШИННОГО ОБУЧЕНИЯ

В настоящем разделе представлено обоснование оптимального варианта проведения исследований, а также описан выбор архитектуры метода машинного обучения.

2.1. Выбор и обоснование оптимального варианта проведения исследований

Основным недостатком существующих методов машинного обучения является их недостаточно высокая производительностью для использования на практике – они позволяют работать с автоматами, содержащими пять-десять состояний, а суммарная длина обучающих примеров может составлять не более двухсот-трехсот событий. Построение автомата с помощью существующих методов может занимать от нескольких минут до нескольких дней.

Для устранения недостатков существующих методов будет разработан метод машинного обучения, основанный на решении задачи о выполнимости булевой формулы. Предварительные исследования показывают, что новый метод будет показывать существенно более высокую производительность (в 10-100 раз более высокую, чем у существующих методов) и сможет обрабатывать автоматы с десятками состояний, а размер входных данных сможет составлять тысячи событий.

При проведении теоретических исследований в рамках настоящего Государственного контракта планируется разработать четыре алгоритма:

- алгоритм построения дерева сценариев;
- алгоритм построения графа совместимости вершин дерева сценариев;
- алгоритм построения булевой КНФ-формулы, задающей требования к раскраске построенного графа и выражающей непротиворечивость системы переходов результирующего автомата;
- алгоритм построения автомата по найденному выполняющему набору значений переменных.

При проведении экспериментальных исследований разработанный метод будет сравниваться с аналогами на примере различных задач построения управляющих конечных автоматов. При этом будут рассматриваться автоматы с различным числом состояний. На основании результатов экспериментов разработанный метод построения управляющих конечных автоматов может быть доработан, после чего будет реализован прототип инструментального средства.

2.2. Архитектура разрабатываемого метода машинного обучения

В настоящем разделе приведено описание архитектуры разрабатываемого метода машинного обучения.

2.2.1. Основные определения

В рамках работ по настоящему Государственному контракту *управляющим конечным автоматом* будем называть детерминированный конечный автомат, каждый переход которого помечен *событием*, последовательностью *выходных воздействий* и *охранным условием*, представляющим собой логическую формулу от *входных переменных*.

Автомат получает события от так называемых *поставщиков событий* (в их роли могут выступать внешняя среда, интерфейс пользователя и т. д.) и генерирует выходные воздействия для *объекта управления*. При поступлении события автомат выполняет переход в соответствии с охранными условиями и значениями входных переменных. При выполнении перехода генерируются выходные воздействия, которыми он помечен, и автомат переходит в соответствующее состояние.

Отметим, что состояния такого автомата не делятся на допускающие и не допускающие. Пример управляющего автомата приведен на рис. 12.

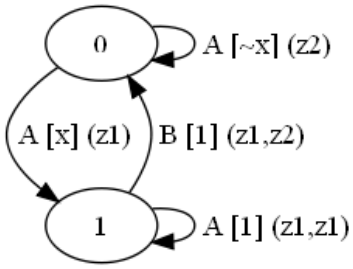

Рис. 12. Пример управляющего конечного автомата

Для данного автомата множество входных событий равно $\{A, B\}$, охранные условия зависят от единственной логической входной переменной x, множество выходных воздействий равно $\{z1, z2\}$. Далее, состояние автомата с номером 0 будем считать начальным.

В рамках работ по настоящему Государственному контракту в качестве исходных данных для построения управляющего конечного автомата используется множество *сценариев работы*. Сценарием работы будем называть последовательность $T_1 \ldots T_n$ троек $T_i = <e_i, f_i, A_i>$, где e_i – входное событие, f_i – булева формула от входных переменных, задающая охранное условие, A_i – последовательность выходных воздействий. В дальнейшем тройки T_i будем называть *элементами сценария*.

Будем говорить, что автомат, находясь в состоянии *state*, *удовлетворяет элементу сценария* T_i, если из *state* исходит переход, помеченный событием e_i, последовательностью выходных воздействий A_i и охранным условием, тождественно равным f_i как булева формула. Автомат *удовлетворяет сценарию работы* $T_1 \ldots T_n$, если он удовлетворяет каждому элементу данного сценария, находясь при этом в состояниях пути, образованного соответствующими переходами.

В рамках работ решается задача построения управляющего конечного автомата с заданным числом состояний C по заданному множеству сценариев работы Sc, которым автомат должен удовлетворять.

2.2.2. Архитектура разрабатываемого метода

Разрабатываемый метод предполагается реализовывать в пять этапов:

1. Построение дерева сценариев.
2. Построение графа совместимости вершин дерева сценариев.
3. Построение булевой КНФ-формулы, задающей требования к раскраске построенного графа и выражающей непротиворечивость системы переходов результирующего автомата.
4. Запуск сторонней программы, решающей задачу о выполнимости булевой КНФ-формулы.
5. Построение автомата по найденному выполняющему набору значений переменных.

Каждому из этапов будет соответствовать модуль программы. Архитектура разрабатываемого метода приведена на рис. 13.

Рис. 13. Архитектура разрабатываемого метода

Построение дерева сценариев предполагается производить путем последовательного добавления в него сценариев из заданного множества. Построение графа совместимости предполагается производить методами динамического программирования. Предполагается, что КНФ-формула будет задавать требования к раскраске построенного графа и выражать непротиворечивость системы переходов результирующего автомата. Далее, планируется найти выполняющую подстановку для данной формулы с помощью программы *CryptoMiniSa*t, которая на данный момент обладает самой высокой производительностью. Построение автомата будет производиться по найденному выполняющему набору значений переменных.

Выводы по главе 2

1. Выбрано и обосновано оптимальное направление проведения исследований.
2. Описана выбранная архитектура разрабатываемого метода машинного обучения.

3. ПОДГОТОВКА ПЛАНА ПРОВЕДЕНИЯ ТЕОРЕТИЧЕСКИХ И ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ

В настоящем разделе представлен план проведения теоретических и экспериментальных исследований.

3.1. План проведения первого этапа теоретических и экспериментальных исследований

На первом этапе (работы проводятся с даты подписания Государственного контракта по 12.07.2011) проведения исследований планируется провести следующие работы:

- патентные исследования;
- разработка архитектуры метода машинного обучения.

Результаты работ первого этапа:

- научно-технический отчет;
- отчет о патентных исследованиях.

3.2. План проведения второго этапа теоретических и экспериментальных исследований

На втором этапе (работы проводятся с 13.07.2011 по 18.11.2011) проведения исследований планируется провести следующие работы:

- теоретическая оценка сложности задачи построения управляющего автомата по сценариям работы программы;
- разработка алгоритма построения дерева сценариев работы;
- программная реализация алгоритма построения дерева сценариев работы;
- разработка алгоритма построения графа совместимости вершин дерева сценариев;
- программная реализация алгоритма построения графа совместимости вершин дерева сценариев.

Результаты работ второго этапа:

- научно-технический отчет.

3.3. План проведения третьего этапа теоретических и экспериментальных исследований

На третьем этапе (работы проводятся с 01.01.2012 по 12.07.2012) проведения исследований планируется провести следующие работы:

- разработка алгоритма построения булевой формулы по графу совместимости;
- программная реализация алгоритма построения булевой формулы по графу совместимости;
- подготовка заяви на регистрацию программы для ЭВМ;
- подготовка статьи для публикации в журнале из перечня ВАК;
- разработка алгоритма построения автомата по найденной выполняющей подстановке;
- программная реализация алгоритма построения автомата по выполняющей подстановке.

Результаты работ третьего этапа:

- научно-технический отчет;
- копия заявки на регистрацию программы для ЭВМ;
- копия статьи для публикации в журнале из перечня ВАК.

3.4. План проведения четвертого этапа теоретических и экспериментальных исследований

На четвертом этапе (работы проводятся с 13.07.2012 по 19.11.2012) проведения исследований планируется провести следующие работы:

- выбор сторонней программы для решения задачи о выполнимости булевой формулы;
- составление плана проведения экспериментальных исследований;
- подготовка заявки на регистрацию программы для ЭВМ;
- проведение вычислительных экспериментов;
- анализ результатов вычислительных экспериментов;
- подготовка статьи для публикации в журнале из перечня ВАК.

Результаты работ четвертого этапа:

- научно-технический отчет;
- копия заявки на регистрацию программы для ЭВМ;
- копия статьи для публикации в журнале из перечня ВАК;
- протоколы экспериментов.

3.5. План проведения пятого этапа теоретических и экспериментальных исследований

На пятом этапе (работы проводятся с 01.01.2013 по 12.07.2013) проведения исследований планируется провести следующие работы:

- корректировка разработанных алгоритмов с учетом результатов проведенных экспериментов;
- программная реализация прототипа инструментального средства, реализующего разработанный метод машинного обучения;
- подача заявки на регистрацию программы для ЭВМ;
- подготовка статьи для публикации в журнале из перечня ВАК.

Результаты работ пятого этапа:

- научно-технический отчет;
- копия заявки на регистрацию программы для ЭВМ;
- копия статьи для публикации в журнале из перечня ВАК.

3.6. План проведения шестого этапа теоретических и экспериментальных исследований

На шестом этапе (работы проводятся с 13.07.2013 по 19.11.2013) проведения исследований планируется провести следующие работы:

- разработка рекомендаций по возможности использования результатов проведенной поисковой научно–исследовательской работы в реальном секторе экономики;
- разработка рекомендаций по использованию результатов поисковой научно–исследовательской работы при разработке научно–образовательных курсов;
- разработка научно–образовательных курсов на электронных носителях по новейшим направлениям науки и технологий;
- разработка научно–популярных материалов для школьников и школьных учителей с участием руководителей поисковых научно–исследовательских работ.

Результаты работ шестом этапа:

- итоговый научно-технический отчет.

Выводы по главе 3

1. Разработан план проведения теоретических и экспериментальных исследований.
2. На первом этапе теоретических исследований будет разработана архитектура метода машинного обучения.

4. ПРОВЕДЕНИЕ ПАТЕНТНЫХ ИССЛЕДОВАНИЙ

В настоящем разделе представлены результаты патентных исследований.

4.1. Перечень сокращений, условных обозначений, символов, единиц, терминов

НИР – научно-исследовательская работа.
FSM – Конечная машина состояний (*Finite State Machine*).
FSA – Конечный автомат (*Finite State Automaton*).
SAT – Задача о выполнимости булевой формулы (*SATisfiability problem*)
SAT-solver – Программа, решающая задачу *SAT*.
ПО – Программное обеспечение.
ЭВМ – Электронно-вычислительная машина.

4.2. Общие данные об объекте исследований

Патентный поиск проводился с целью определения патентоспособности планируемых результатов научно-исследовательской работы **по лоту «Проведение научных исследований научными группами под руководством докторов наук в следующей области информатики» – тема «Разработка метода машинного обучения на основе алгоритмов решения задачи о выполнимости булевой формулы для построения управляющих конечных автоматов» (шифр «2011-1.2.1-113-002-047»)**, выполняемой в рамках Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы **по государственному контракту № 16.740.11.0455**, заключенному между Министерством образования и науки Российской Федерации и Государственным образовательным учреждением высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики» на основании решения Конкурсной комиссии Министерства образования и науки Российской Федерации (протокол от 25.03.2011 г. № 3/0173100003711000031), а также для получения сведений об охранных и иных документах, которые могут препятствовать применению результатов данной НИР в Российской Федерации, и условиях использования таких документов.

В соответствии с задачей исследования, разрабатываемая технология генерации автоматов должна удовлетворять следующим требованиям:

- возможность создания автоматов управления системами со сложным поведением;
- использование алгоритмов решения задачи о выполнимости булевой формулы.

Патентный поиск проводился в соответствии с ГОСТ Р. 15.011-96 «Система разработки и постановки продукции на производство. Патентные исследования» [1]. Проверка патентоспособности проводимой научно-исследовательской работы осуществлялась на основе поиска патентных и других открытых документов, описывающих решения, максимально полно удовлетворяющие задаче исследования. Поиск патентной информации проводился в Бюллетене Федеральной службы по интеллектуальной собственности, патентам и товарным знакам Российской Федерации «Программы для ЭВМ, базы данных, топологии интегральных микросхем» (Роспатент, www.fips.ru), Бюро по патентам и товарным знакам США (USPTO, www.uspto.gov) и Европейского патентного бюро (EPO, ep.espacenet.com).

Патентный поиск проводился с 06 июня 2011 г. по 09 июня 2011 г.

4.3. Исследования патентов

Тема исследования ограничивается применением алгоритмов решения задачи выполнимости булевой формулы для построения управляющих конечных автоматов. Однако, для повышения

полноты результатов поиска, рассматриваемая тематическая область была расширена на следующие направления:

- автоматическое построение программ;
- применение автоматных моделей вычислений;
- построение управляющих систем.

В случаях, когда в рамках некоторого направления была возможность выделить блоки меньшего объема, блоки, заведомо не соответствующие направлению исследования, из рассмотрения исключались. Таким образом, были выбраны элементы патентных классификаций для ограничения области поиска.

В рамках патентного поиска производился анализ патентов, принадлежащих следующим категориям международной патентной классификации (рассматривались патенты, принадлежащие самому низкому указанному уровню):

Категория	Содержание категории (рус.)	Содержание категории (англ.)
G	Физика	Physics
G06	Вычисление; счет	Computing; calculating; counting
G06F	Обработка цифровых данных с помощью электрических устройств	Electrical digital data processing
G06F 17	Устройства или методы цифровых вычислений или обработки данных, специально предназначенные для специфических функций	Digital computing or data processing equipment or methods, specially adapted for specific functions
G06F 17/10	Комплексные математические операции	Complex mathematical operations
G06F 17/50	Автоматизированное проектирование	Computer-aided design

Кроме того, рассматривались следующие категории патентной классификации США (перевод на русский язык исполнителя отчета):

Категория	Содержание категории (рус.)	Содержание категории (англ.)
716	Системы автоматизированного проектирования и анализа схем и полупроводниковых масок	Computer-aided design and analysis of circuits and semiconductor masks
716/100		Integrated circuit design processing
716/101		Logic design processing
706	Обработка данных: искусственный интеллект	Data processing: Artificial intelligence
706/12	Машинное обучение	Machine learning

Также производился поиск без ограничения разделов классификаций по следующим ключевым словам: *FSM, FSA, automaton, automata, inference, induction, SAT, satisfiability*.

Так как патенты, полностью удовлетворяющие поставленным перед исследованием требованиям, найдены не были, в качестве результата поиска рассмотрим наиболее близкие к теме исследования патенты:

1. Патент US 7685547 от 23.03.2010 *Method, system, and computer program product for generating automated assumption for compositional verification*.

Предложен способ построения минимального автомата для использования в задаче верификации электронных схем с использованием выборки и решения задачи выполнимости булевой формулы. Минимальный детерминированный конечный автомат строится методом обобщения неполного недетерминированного конечного автомата, который, в свою очередь, получен по множеству поведений электронной схемы. При данном подходе автомат строится по имеющейся электронной схеме и в дальнейшем используется в целях ее верификации. Кроме того, построенный автомат является автоматом-распознавателем. Таким образом, описанный подход неприменим к задаче построения управляющих конечных автоматов, а следовательно, не решает задачу настоящего исследования.

2. Патент WO 2003/052591 от 26.06.2003 *Processor architecture selectively using finite-state-machine for control code*.

Предлагается архитектура процессора, в которой часть инструкций предлагается реализовать с помощью конечных автоматов. Данный подход является более экономичным по отношению к использованию кеш-памяти и потреблению энергии. Данный патент не рассматривает вопрос генерации такого автомата, поэтому методы, описанные в патенте, не могут быть применимы в данном исследовании.

3. Патент US 5943659 от 24.08.1999 *Deterministic encoding of fuzzy finite state automata in continuous recurrent neural networks.*

На основе представления детерминированного конечного автомата в виде рекуррентной нейронной сети предлагается кодирование в виде нейронной сети нечеткого автомата, распознающего некоторый нечеткий язык с любой заданной точностью. Широко известна взаимная эквивалентность конечного автомата и рекуррентной нейронной сети, а также способы применения генетических алгоритмов для оптимизации таких сетей. В проводимом исследовании не применяется такой подход, не рассматриваются нечеткие автоматы, а распознавание языка не является типичной задачей.

4. Патенты EP 1296281 от 26.03.2003 и US 6839698 от 04.01.2005 *Fuzzy genetic learning automata classifier.*

Описывает построение нечеткого автомата, наилучшим образом разделяющего на классы входные последовательности. Классификация последовательностей или распознавание языка – одна из наиболее известных областей применения конечных автоматов. Однако нечеткие автоматы и задачи классификации не характерны для темы данного патентного поиска. Кроме того, в патентах не рассматривается способ модификации автомата.

5. Патент EP 1202141 от 02.05.2002 *Method of reducing finite controlling automata and corresponding computer-readable medium.*

Часто несколько небольших управляющих автоматов объединяют в один большой. Большой автомат сложно преобразовывать и часто не представляется возможным изобразить графически. Для применения техник верификации большой автомат сокращают. Для этого можно применять описанный в патенте способ. Из описания автомата исключаются избыточные переходы, недостижимые состояния и переходы внутри одного состояния без действий. Таким образом, патент охватывает только упрощение заданного автомата, что может быть применено после исследуемого построения автомата с применением алгоритмов решения задачи выполнимости булевой формулы, но не заменяет его.

Сводная информация по основным различиям рассмотренных патентов и задач проверяемого на патентоспособность исследования приведена в табл. 3.

Из изложенного следует, что поиск не выявил наличия патентов, препятствующих проведению исследований по теме «Разработка метода машинного обучения на основе алгоритмов решения задачи о выполнимости булевой формулы для построения управляющих конечных автоматов».

Таблица 3. Отличия описываемых в патентах задач от темы исследования

Патент	Нечеткий автомат	Не управляющий автомат	Не построение
US 7685547		Да	
WO 2003/052591			Да
US 5943659	Да	Да	
EP 1296281	Да	Да	
US 6839698	Да	Да	
EP 1202141			Да

4.4. Исследования программ для ЭВМ

Согласно Приложению 2 к Государственному контракту, в ходе выполнения работ планируется регистрация программ для ЭВМ. В связи с этим, были рассмотрены данные о регистрации программ для ЭВМ из выпусков Электронного бюллетеня «Программы для ЭВМ, базы данных, топологии интегральных микросхем» за 2008-2011 годы. В табл. 4 приведена сводная информация по рассмотренным данным.

Таблица 4. Результаты рассмотрения данных о регистрации программ для ЭВМ

Рег. номер	Название программы	Краткое описание программы	Комментарий
2008613501	Эволюционный генератор конечных автоматов для задач резолютивного вывода	Программа генерирует автоматы, представляющие собой эвристику поиска пустого дизъюнкта, с помощью генетического алгоритма.	Используется генетический алгоритм, а не *SAT-solver*
2008614247	Генератор модели конечного преобразователя продукционных правил	Программа генерирует автоматы, предназначенные для генерации базового множества ядер продукционных правил, с помощью генетического алгоритма.	Используется генетический алгоритм, а не *SAT-solver*
2008610354	*MicroSWITCH*	Программа предназначена для разработки ПО микроконтроллеров путем описания системы взаимодействующих конечных автоматов.	Вопрос автоматического построения автоматов не рассматривается
2008610423	Пакет прикладной программ *Distributed-SAT 1.0*	Программа предназначена для решения задачи SAT с применением ресурсов распределенной вычислительной сети	Может использоваться в качестве составной части системы, разрабатываемой в рамках Государственного контракта
2008610473	Программная система «Генетический генератор автоматов»	Программа предназначена для генерации автоматных программ с большим числом входных	Используется генетический алгоритм, а не *SAT-*

		воздействий и сложными условиями на переходах.	*solver*, автомат строится не по тестовым примерам, а по оценке работы автомата
2009610226	Визуальный редактор состояний для платформы *Eclipse* (*EclipseStateEditor*)	Программа предназначена для описания модели программы в виде конечного автомата.	Может использоваться для редактирования программ, полученных с помощью разрабатываемой системы.
2010615014	Программное средство для генерации автоматов, представленных линейными бинарными графами, на основе генетического программирования	Программа предназначена для автоматической генерации автоматов управления с помощью генетического программирования.	Используется генетический алгоритм, а не *SAT-solver*
2010614197	Программное средство для построения управляющих конечных автоматов на основе обучающих примеров с использованием генетических алгоритмов	Программа предназначена для автоматической генерации автоматов управления с помощью генетических алгоритмов на основе тестовых примеров.	Используется генетический алгоритм, а не *SAT-solver*

Из изложенного следует, что поиск не выявил наличия программ для ЭВМ, реализующих требуемую функциональность и тем самым препятствующих проведению исследований по теме «Разработка метода машинного обучения на основе алгоритмов решения задачи о выполнимости булевой формулы для построения управляющих конечных автоматов».

4.5. Исследования непатентных источников

Согласно Патентному закону Российской Федерации [2], уровень техники, в сравнении с которым выявляется новизна изобретения, определяется не только зарегистрированными в России патентами, но также имеющейся во всем мире общедоступной информацией. В связи с этим был проведен анализ публикаций, содержащих упоминания построения конечных автоматов или подобных им моделей с применением алгоритмов решения задачи о выполнимости булевой формулы.

4.5.1. Общий обзор публикаций

В настоящее время наиболее распространенными методами машинного обучения для построения управляющих конечных автоматов являются методы, основанные на генетических и эволюционных алгоритмах. Такие методы машинного обучения для построения управляющих конечных автоматов по используемому принципу обучения можно разделить на три типа:

- методы, основанные на принципе обучения «на собственных ошибках»;

- методы, основанные на принципе обучения на примерах (тестах);
- методы, основанные на принципе обучения на спецификации.

В работе [42] рассматривается задача построения управляющего конечного автомата на основе принципа «обучения на собственных ошибках». В этой работе рассматривается задача распространения программных агентов в компьютерной сети. Для решения этой задачи применялся генетический алгоритм, в котором конечный автомат был представлен в виде таблицы значений функции переходов и таблицы значений функции действий.

В рамках исследований по теме «Технология генетического программирования для генерации автоматов управления системами со сложным поведением» в СПбГУ ИТМО был разработан ряд методов генерации конечных автоматов с помощью генетических алгоритмов [11–14].

Один из этих методов – метод сокращенных таблиц переходов, был предложен в [8]. Второй метод – метод представления автоматов деревьями решений, предложен в [4].

В [34] производится сравнение алгоритмов машинного обучения для построения конечных автоматов-распознавателей. При этом рассматриваются два типа алгоритмов: эволюционные и основанные на построении префиксного дерева (бора) и дальнейшем объединении состояний на основе свидетельств (*Evidence-Driven State Merging* – EDSM).

В [35] также рассматривается задача построения конечных автоматов-распознавателей с помощью эволюционных алгоритмов. Предлагаемый в рассматриваемой работе эволюционный алгоритм является улучшенной версией алгоритма, предложенного в [34].

В [20] автоматы представлялись с помощью таблицы переходов. Разработанный в этой работе метод позволяет поддерживать в популяции автоматы с разным числом состояний. Функция приспособленности учитывает три компонента – число верно распознанных тестовых примеров, число состояний и переходов автомата, степень общности языка, соответствующего построенному автомату. Это позволяет сузить область поиска, и находить языки, соответствующие определенным критериям. В обучающий набор могут входить примеры слов, которые как принадлежат, так и не принадлежат языку.

В [43] также рассматривается задача построения конечных автоматов-распознавателей. В качестве метода представления конечных автоматов выбраны двоичные строки, в качестве операций мутации и скрещивания используется однородное скрещивание и мутация.

По построению конечных преобразователей на основе обучающих примеров существует намного меньше работ, чем по построению конечных распознавателей. Одной из работ по конечным автоматам-распознавателям является [36]. Такие автоматы при обработке символов входного слова при выполнении переходов записывают в выходной поток символы выходного алфавита, тем самым осуществляя преобразование слов одного формального языка в слова другого.

В [16] посвящена построению взаимодействующих конечных автоматов с использованием генетического программирования с функцией приспособленности, при вычислении которой используется верификация моделей [17]. Верификация моделей – это набор методов и алгоритмов, которые позволяют определять, соответствует ли программа некоторому множеству требований. При этом требования выражаются на языке темпоральной логики – записываются утверждения о поведении программы во времени. В рассматриваемой работе применяется темпоральная логика *CTL* (Computation Tree Logic – логика деревьев вычислений), а для верификации используется система *SMV* (http://www.cs.cmu.edu/~modelcheck/smv.html).

В [18] предлагается метод генетического программирования, основанный на совместном использовании тестовых примеров и верификации моделей программ. Этот метод позволяет строить автоматы с гарантированным поведением за счет использования верификации.

Основным недостатком существующих методов машинного обучения является их недостаточно высокая производительностью для использования на практике – они позволяют

работать с автоматами, содержащими пять-десять состояний, а суммарная длина обучающих примеров может составлять не более двухсот-трехсот событий. Построение автомата с помощью существующих методов может занимать от нескольких минут до нескольких дней.

Для устранения недостатков существующих методов будет разработан метод машинного обучения, основанный на решении задачи о выполнимости булевой формулы. Предварительные исследования показывают, что новый метод будет показывать существенно более высокую производительность (в 10-100 раз более высокую, чем у существующих методов) и сможет обрабатывать автоматы с десятками состояний, а размер входных данных сможет составлять тысячи событий.

4.5.2. Обзор избранных работ

Далее будут подробно рассмотрены работы, наиболее близко соответствующие направлению проводимого по контракту исследования, и указаны их отличия.

4.5.2.1. Evolving Finite-State Machine Strategies for Protecting Resources

В [42] рассматривается задача построения управляющего конечного автомата на основе принципа «обучения на собственных ошибках». В этой работе рассматривается задача распространения программных агентов в компьютерной сети. Для решения этой задачи применялся генетический алгоритм, в котором конечный автомат был представлен в виде таблицы значений функции переходов и таблицы значений функции действий.

Вычисление функции приспособленности выполняется в два этапа – на первом из них значение вычисляется приближенно, а на втором – оно уточняется, но только для части автоматов, показавших наилучшие результаты на первом этапе.

В отличие от этой работы, в исследованиях, проводимых по контракту, автомат будет строиться на основе тестовых примеров с помощью алгоритма решения задачи о выполнимости булевой формулы.

4.5.2.2. Применение генетического программирования для генерации автоматов с большим числом входных переменных

В [8] был предложен метод сокращенных таблиц переходов. Этот метод позволяет решить проблему экспоненциального роста размера описания автомата, которая возникает при использовании полных таблиц переходов – число строк в таблице есть 2^n, где n – число предикатов. Опыт показывает, что в реальных задачах управляющие автоматы, построенные вручную, имеют гораздо меньше переходов, чем $|S| \cdot 2^n$ (здесь как $|S|$ – число состояний автомата). Причина этого состоит в том, что в большинстве задач предикаты имеют «локальную природу» по отношению к управляющим состояниям. В каждом состоянии значимым является лишь определенный, небольшой поднабор предикатов, остальные же не влияют на значение управляющей функции. Именно это свойство позволяет существенно сократить размер описания состояний. Кроме того, использование этого свойства в процессе оптимизации позволяет получить результат, более похожий на автомат, построенный вручную, а, следовательно, и более понятный человеку.

В отличие от этой работы, в исследованиях, проводимых по контракту, автомат будет строиться с помощью алгоритма решения задачи о выполнимости булевой формулы.

4.5.2.3. Метод представления автоматов деревьями решений для использования в генетическом программировании

В [4] предложен метод представления автоматов деревьями решений. Дерево решений является удобным способом задания дискретной функции, зависящей от конечного числа логических переменных. Оно представляет собой помеченное дерево, метки в котором расставлены по следующему правилу:

- внутренние узлы помечены символами переменных;
- ребра – значениями переменных;
- листья – значениями искомой функции.

Для определения значения функции по значениям переменных необходимо спуститься от корня до листа, и сформировать значение, которым помечен полученный лист. При этом из вершины, помеченной переменной *x*, переход производится по тому ребру, которое помечено тем же значением, что и значение переменной *x*.

Метод представления автомата с помощью деревьев решений заключается в следующем: функции переходов и выходов автомата выражаются с помощью деревьев решений. Более формально: зададим для каждого состояния $q \in Q$ функцию $\sigma_q : X \to Q \times Y$, такую что $\sigma_q(x) = (\delta(q,x), \lambda(q,x))$ для $\forall x \in X$. Здесь Q – множество состояний автомата, X – множество входных воздействий, Y – множество выходных воздействий, $\delta : Q \times X \to Q$ – функция переходов, $\lambda : Q \times X \to Y$ – функция выхода. Каждая из этих функций может быть определена собственным деревом решений.

В отличие от этой работы, в исследованиях, проводимых по контракту, автомат будет строиться с помощью алгоритма решения задачи о выполнимости булевой формулы, а сам автомат будет представлен в виде таблицы переходов.

4.5.2.4. Learning DFA: Evolution versus Evidence Driven State Merging

В [34] производится сравнение алгоритмов машинного обучения для построения конечных автоматов-распознавателей. При этом рассматриваются два типа алгоритмов: эволюционные и основанные на построении префиксного дерева (бора) и дальнейшем объединении состояний на основе свидетельств (*Evidence-Driven State Merging, EDSM*).

Входными данными для алгоритма построения конечного автомата-распознавателя является набор слов, для каждого из которых известно, должно ли оно допускаться автоматом или не должно. Этот набор слов в дальнейшем будет называться множеством обучающих примеров. Выходными данными для рассматриваемого алгоритма является описание построенного конечного автомата, который удовлетворяет входным данным.

В качестве эволюционного алгоритма в рассматриваемой работе применяется так называемая (1+1)-эволюционная стратегия. Такой алгоритм не использует операцию скрещивания. Работа этого алгоритма осуществляется следующим образом: хранится текущее решение, и на каждой итерации к текущему решению применяется операция мутации. Если эта мутация приводит к увеличению значения функции приспособленности, то эта мутация принимается, и текущее решение заменяется новым. В противном случае текущее решение остается без изменений. Если за заданное заранее число итераций не происходит увеличение значения функции приспособленности, то текущее решение записывается, а процесс поиска решения перезапускается.

На основании проведенных экспериментов был сделан вывод, что разработанный эволюционный алгоритм превосходит алгоритм *EDSM* в том случае, когда необходимо построить автомат с относительно небольшим числом состояний, а обучающее множество примеров содержит небольшую долю строк заданной длины.

В исследованиях, проводимых по контракту, автомат будет строиться на основе префиксного дерева, но, в отличие от данной работы, состояния будут объединяться на основе решения задачи о выполнимости булевой формулы.

4.5.2.5. A Genetic Algorithm for Finite State Automata Induction with Application to Phonotactics

В [20] автоматы представляются с помощью таблицы переходов. Разработанный в этой работе метод позволяет поддерживать в популяции автоматы с разным числом состояний. Функция приспособленности учитывает три компонента – число верно распознанных тестовых примеров, число состояний и переходов автомата, степень общности языка, соответствующего построенному автомату. Это позволяет сузить область поиска, и находить языки, соответствующие определенным критериям. В обучающий набор могут входить примеры слов, которые как принадлежат, так и не принадлежат языку.

Приведем описание генетической операции репродукции. Число состояний потомка выбирается случайно из диапазона $[N - 2, N + 2]$, где N – число состояний первого родителя. После этого каждый переход копируется из родителей, причем вероятность копирования перехода из заданной особи прямо пропорциональна значению ее функции приспособленности. Переходы, представленные только в одном из родителей, копируются из того родителя, в котором присутствуют. Переходы, отсутствующие в обоих родителях, генерируются случайно.

Генетическая операция мутации осуществляется изменением случайного перехода. При этом переход разрешается удалять. Это приводит к распаду графа переходов на несколько компонент. Результаты экспериментов показали, что удаление недостижимых состояний замедляет вывод – поэтому оно не производится.

Предложенный подход был проверен на практических задачах. Авторам удалось построить автомат из семи состояний, который распознает по множеству из ста примеров русские двухсложные слова.

В отличие от этой работы, в исследованиях, проводимых по контракту, автомат будет строиться с помощью алгоритма решения задачи о выполнимости булевой формулы. Кроме того, будет строиться не автомат-распознаватель, а управляющий автомат.

4.5.2.6. Exact DFA Identification Using SAT Solvers

В [30] представлен алгоритм построения автоматов-распознавателей по тестовым наборам, основанный на сведении поставленной задачи к задаче *SAT*. Тестовые наборы представляют собой два набора слов V^+ и V^-, которые искомый автомат должен принимать и отвергать, соответственно.

В данной работе предлагается метод построения и упрощения КНФ-формулы, кодирующей требования, накладываемые тестовыми примерами на автомат-распознаватель. Данная КНФ-формула подается на вход программе picosat, находящей выполняющую подстановку для построенной формулы. Затем, по найденной выполняющей подстановке строится искомый автомат-распознаватель. Результаты экспериментальных исследований показали, что производительность предлагаемого метода выше других методов построения автоматов-распознавателей. Таким образом, была показана целесообразность использования методов решения задачи *SAT*.

В отличие от этой работы, в исследованиях, проводимых по контракту, будет строиться не автомат-распознаватель, а управляющий автомат.

4.5.3. Результаты непатентных исследований

Исследование публикаций по построению конечных автоматов с применением алгоритмов решения задачи о выполнимости булевой формулы показало, что работы по этой тематике в мире

проводятся. Ни в одной из найденных публикаций, также как и в патентах, не было найдено описания подхода, полностью удовлетворяющего требованиям, предъявляемым к автоматам в рамках работ по контракту.

4.6. Отличия проводимого исследования

Проверяемое на патентоспособность исследование ориентировано на применение построения автоматов с использованием алгоритмов решения задачи о выполнимости булевой формулы для автоматизации широкого класса программных систем со сложным поведением, в то время как многие из рассмотренных патентов и публикаций применимы только в узкоспециальных областях. Так, патенты US 7685547, US 5943659, EP 1296281, US 6839698 ориентированы на построение только конечных автоматов-распознавателей. Патенты же WO 2003/052591 и EP 1202141 не рассматривают автоматическое построение автоматов.

Большинство зарегистрированных программ для ЭВМ, применяемых для автоматического построения конечных автоматов, делает это с помощью генетического алгоритма (2008613501, 2008614247, 2008610473, 2010615014, 2010614197). Другие программы либо предназначены не для автоматической генерации автоматов, а для их редактирования человеком (2008610354, 2009610226), либо могут использоваться в качестве реализации одной из составных частей исследования (2008610423).

В [3–9, 11–13, 15, 17] рассматривается построение различных автоматов с помощью генетических алгоритмов. В [14, 17] построение автоматов производится, как и в исследованиях, проводимых по контракту, на основе тестовых примеров, но при этом либо используются генетические алгоритмы, либо строятся автоматы-распознаватели. Отдельно стоит выделить [10], в которой рассматривается построение префиксного дерева (бора) по тестовым примерам, на основе которого строится автомат. В исследовании будет использован именно этот подход, однако, в отличие от работы [10], построение автомата по бору будет производиться с помощью решения задачи о выполнимости булевой формулы, что эффективнее.

Таким образом, выполнены анализ патентных и непатентных источников, а также зарегистрированных программ для ЭВМ показал, что рассматриваемая в работе по контракту задача в настоящее время не решена.

4.7. Выводы по главе 4

Данное патентное исследование, проведенное в рамках первого этапа работы по государственному контракту № 16.740.11.0455, показало, что в настоящее время отсутствуют патенты и иные охранные документы, которые могут препятствовать применению в Российской Федерации технологии генерации автоматов управления системами со сложным поведением с использованием алгоритмов решения задачи о выполнимости булевой формулы, разрабатываемой в рамках указанной работы. Таким образом, в соответствии с пунктом 5 Технического задания и пунктом 6.1 Детализированного предложения о качестве поисковых научно-исследовательских работ обеспечена патентная чистота результатов и отсутствие препятствий для их применения.

4.8. Литература и источники

1. ГОСТ Р. 15.011-96 «Система разработки и постановки продукции на производство. Патентные исследования»
2. *Патентный закон РФ.* http://www.legal-support.ru/information/laws/intellect/patent-law.html
3. *Spears W., Gordon D.* Evolving Finite-State Machine Strategies for Protecting Resources. Lecture Notes in Computer Science. Springer Berlin / Heidelberg. Volume 1932/2009, pp. 5–28.

4. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. Промежуточный отчет по этапу I «Выбор направлений исследований и базовых компонентов». http://is.ifmo.ru/genalg/_2007_01_report-genetic.pdf
5. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. Промежуточный отчет по этапу II «Теоретические исследования поставленных перед НИР задач». http://is.ifmo.ru/genalg/_2007_02_report-genetic.pdf
6. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. Промежуточный отчет по этапу III «Экспериментальные исследования поставленных перед НИР задач». http://is.ifmo.ru/genalg/_2007_03_report-genetic.pdf
7. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. Промежуточный отчет по этапу IV «Обобщение и оценка результатов исследований». http://is.ifmo.ru/genalg/_2007_04_report-genetic.pdf
8. *Поликарпова Н. И., Точилин В. Н., Шалыто А. А.* Применение генетического программирования для генерации автоматов с большим числом входных переменных // Известия РАН. Теория и системы управления. 2010. № 2, с.100–117.
9. *Данилов В. Р.* Метод представления автоматов деревьями решений для использования в генетическом программировании //Научно-технический вестник СПбГУ ИТМО. 2008. Выпуск 53. Автоматное программирование, с. 103–108.
10. *Lucas S., Reynolds J.* Learning DFA: Evolution versus Evidence Driven State Merging / The 2003 Congress on Evolutionary Computation (CEC '03). Vol. 1, pp. 351–358.
11. Lucas S., Reynolds J. Learning Deterministic Finite Automata with a Smart State Labeling Algorithm // IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 27. 2005. № 7, pp. 1063–1074.
12. *Belz A., Eskikaya B.* A Genetic Algorithm for Finite State Automata Induction with Application to Phonotactics / Proceedings of the ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing. Saarbruecken. 1998, pp. 9–17.
13. Heule M., Verwer S. Exact DFA Identification Using SAT Solvers // Grammatical Inference: Theoretical Results and Applications 10th International Colloquium (ICGI 2010). Lecture Notes in Computer Science. 2010. V 6339, pp. 66–79.
14. *Spichakova M.* Genetic Inference of Finite State Machines. Masters thesis. Tallinn. 2007. http://s-ma-u-g.googlecode.com/files/thesis.pdf
15. Lucas S. Evolving Finite-State Transducers: Some Initial Explorations // Lecture Notes in Computer Science. Springer Berlin. Heidelberg. 2003. V. 2610, pp. 241–257.
16. Johnson C. Genetic Programming with Fitness based on Model Checking. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007. V. 4445, pp. 114–124.
17. Кларк Э., Грамберг О., Пелед Д. Верификация моделей программ. М.: МЦНМО, 2002.
18. 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 Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2010. №.5 (69), с. 81–86.

ЗАКЛЮЧЕНИЕ

В результате исследований на первом этапе работ по контракту были выполнены следующие работы:

- выполнен аналитический обзор;
- проведены патентные исследования;
- выбран и обоснован оптимальный вариант проведения исследований;
- выбрана архитектура метода машинного обучения;
- подготовлен план проведения теоретических и экспериментальных исследований.

Аналитический обзор был выполнен по следующим направлениям:

- методы машинного обучения для построения конечных автоматов;
- программные средства для решения задачи о выполнимости булевой формулы.

В результате выполнения аналитического обзора разработан план проведения теоретических и экспериментальных исследований. В соответствии с этим планом в рамках теоретических исследований будут разработаны следующие алгоритмы, составляющие машинного обучения на основе алгоритмов решения задачи о выполнимости булевой формулы для построения управляющих конечных автоматов:

- алгоритм построения дерева сценариев;
- алгоритм построения графа совместимости вершин дерева сценариев;
- алгоритм построения булевой КНФ-формулы, задающей требования к раскраске построенного графа и выражающей непротиворечивость системы переходов результирующего автомата;
- алгоритм построения автомата по найденному выполняющему набору значений переменных.

По результатам экспериментальных исследований будет проведена модификация разработанных методов. В заключение работы будет проведено обобщение и оценка результатов исследований.

Результаты выполненных работ позволяют утверждать, что научно-технический уровень исследований превышает уровень исследований в рассматриваемой области, проводимых в лучших исследовательских центрах мира.

СПИСОК ЛИТЕРАТУРЫ

1. *Бедный Ю. Д., Шалыто А. А.* Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». СПбГУ ИТМО, 2007. http://is.ifmo.ru/works/ant/
2. *Гладков Л. А., Курейчик В. В., Курейчик В. М.* Генетические алгоритмы. М.: Физматлит, 2006.
3. *Гуров В. С., Мазин М. А., Нарвский А. С., Шалыто А. А.* UML. SWITCH-технология. Eclipse // Информационно-управляющие системы. 2004. № 6. с. 12–17.
4. *Данилов В. Р.* Метод представления автоматов деревьями решений для использования в генетическом программировании //Научно-технический вестник СПбГУ ИТМО. 2008. Выпуск 53. Автоматное программирование, с. 103–108.
5. *Данилов В. Р.* Технология генетического программирования для генерации автоматов управления системами со сложным поведением. СПбГУ ИТМО. Бакалаврская работа. 2007.
6. *Левенштейн В. И.* Двоичные коды с исправлением выпадений, вставок и замещений символов. Доклады Академии Наук СССР 1963. № 4, с. 845–848.
7. *Николенко С. И., Тулупьев А. Л.* Самообучающиеся системы. М.: МЦНМО, 2009.
8. *Норенков И. П., Арутюнян Н. М.* Метагенетический алгоритм оптимизации и структурного синтеза проектных решений //Информационные технологии. 2007. № 3.
9. *Поликарпова Н. И., Шалыто А. А.* Автоматное программирования. СПб: Питер, 2009.
10. Сайт по генетическому программированию. www.genetic-programming.com
11. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. Промежуточный отчет по этапу I «Выбор направлений исследований и базовых компонентов». http://is.ifmo.ru/genalg/_2007_01_report-genetic.pdf
12. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. Промежуточный отчет по этапу II «Теоретические исследования поставленных перед НИР задач». http://is.ifmo.ru/genalg/_2007_02_report-genetic.pdf
13. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. Промежуточный отчет по этапу III «Экспериментальные исследования поставленных перед НИР задач». http://is.ifmo.ru/genalg/_2007_03_report-genetic.pdf
14. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. Промежуточный отчет по этапу IV «Обобщение и оценка результатов исследований». http://is.ifmo.ru/genalg/_2007_04_report-genetic.pdf
15. *Фридман А., Меннон П.* Теория и проектирование переключательных схем. М.: Мир, 1978.
16. *Хопкрофт Дж., Мотвани Р., Ульман Дж.* Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002.
17. *Царев Ф. Н.* Разработка метода совместного применения генетического программирования и конечных автоматов на примере игры «Летающие тарелки». СПбГУ ИТМО. Бакалаврская работа. 2007.
18. *Царев Ф. Н.* Совместное применение генетического программирования, конечных автоматов и искусственных нейронных сетей для построения системы управления беспилотным летательным аппаратом //Научно-технический вестник СПбГУ ИТМО. 2008. Выпуск 53. Автоматное программирование, с. 42–60.
19. *Angeline P., Pollack J.* Evolutionary Module Acquisition / Proceedings of the Second Annual Conference on Evolutionary Programming. Cambridge: MIT Press. 1993, pp. 154–163. http://www.demo.cs.brandeis.edu/papers/ep93.pdf
20. *Belz A., Eskikaya B.* A Genetic Algorithm for Finite State Automata Induction with Application to Phonotactics / Proceedings of the ESSLLI-98 Workshop on Automated Acquisition of Syntax and

Parsing, Saarbruecken, 1998, pp. 9–17. http://www.itri.brighton.ac.uk/~Anja.Belz/Publications/A_GA_for_FSA_induction_with_an_application_to_phonotactics.ps
21. *Biere A.* Lingeling, Plingeling, PicoSAT and PrecoSAT at SAT Race 2010. Technical Report 10/1. 2010. FMV Reports Series. Institute for Formal Models and Verification. Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria.
22. *Biere A. Een N.* Effective Preprocessing in SAT through Variable and Clause Elimination / Proceedings of the SAT 2005. http://minisat.se/downloads/SatELite.pdf
23. CryptoMiniSat SAT-solver. http://www.msoos.org/cryptominisat2
24. *Davis M., Logemann G., Loveland D.* A machine program for theorem proving // Communications of the ACM. 1962. № 5, зз. 394–397.
25. *De Jong K.* Evolutionary computation: a unified approach. MIT Press, Cambridge MA, 2006.
26. *Een N., Sörensson N.* An extensible SAT-solver. Chalmers University of Technology. Sweden. 2003. http://minisat.se/downloads/MiniSat.pdf
27. *Gold E. M.* Complexity of automaton identification from given data // Information and Control. 1978, № 37, pp. 302−320.
28. GrAnDe (Ground And Decide) – system for CNF first order problems with a finite Herbrand universe. http://www.cs.miami.edu/~tptp/ATPSystems/GrAnDe/
29. *Hamming R.* Error detecting and error correcting codes //Bell System Technical Journal 29 (2), pp. 147–160.
30. *Heule M., Verwer S.* Exact DFA Identification Using SAT Solvers //Grammatical Inference: Theoretical Results and Applications 10th International Colloquium (ICGI 2010). Lecture Notes in Computer Science. V. 6339, pp. 66–79.
31. *Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C., Wang A.* The Genesys System: Evolution as a Theme in Artificial Life /Proceedings of Second Conference on Artificial Life. MA: Addison-Wesley. 1992, pp. 549–578. www.cs.ucla.edu/~dyer/Papers/AlifeTracker/Alife91Jefferson.html
32. *Koza J.* Genetic programming. On the Programming of Computers by Means of Natural Selection. MA: The MIT Press, 1998.
33. Learning DFS from Noisy Samples. A contest from GECCO 2004. http://cswww.essex.ac.uk/staff/sml/gecco/NoisyDFA.html
34. *Lucas S., Reynolds J.* Learning DFA: Evolution versus Evidence Driven State Merging // The 2003 Congress on Evolutionary Computation (CEC '03). Vol. 1, pp. 351–358.
35. *Lucas S., Reynolds J.* Learning Deterministic Finite Automata with a Smart State Labeling Algorithm // IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 27. 2005. №7, pp. 1063–1074.
36. *Lucas S.* Evolving Finite-State Transducers: Some Initial Explorations. Lecture Notes in Computer Science. Springer Berlin / Heidelberg. Volume 2610/2003, pp. 241–257. http://www.springerlink.com/content/41a34vg70fp1hltb/
37. *Mitchell M., Holland J., Forrest S.* When will a genetic algorithm outperform hill climbing? / Advances in Neural Information Processing Systems 6, pp. 51–58. Morgan Kaufman, San Mateo, California, 1994.
38. *Mitchell M.* An Introduction to Genetic Algorithms. MA: The MIT Press, 1996.
39. *Moskewicz M.W., Madigan C.F., Zhao Y., Zhang L., Malik S.* Chaff: engineering an efficient SAT solver. / Proceedings «Design Automation Conference» 2001, pp. 530–535.
40. NuSMV: a new symbolic model checker. http://nusmv.fbk.eu/

41. *Satisfiability* suggested format DIMACS http://www.domagoj-babic.com/uploads/ResearchProjects/Spear/dimacs-cnf.pdf
42. *Spears W., Gordon D.* Evolving Finite-State Machine Strategies for Protecting Resources //Lecture Notes in Computer Science. Springer Berlin / Heidelberg. 2009. V. 1932, pp. 5–28.
43. *Spichakova M.* Genetic Inference of Finite State Machines. Masters thesis. Tallinn. 2007. http://s-ma-u-g.googlecode.com/files/thesis.pdf
44. *Tomita M.* Dynamic construction of finite automata from examples using hill climbing /Proceedings of the 4th Annual Cognitive Science Conference. USA. 1982, pp. 105–108.
45. *zChaff SAT-solver*. http://www.princeton.edu/~chaff/zchaff.html