ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2014, № 2, с. 111–121

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

УДК 004.4'242

ГЕНЕРАЦИЯ УПРАВЛЯЮЩИХ АВТОМАТОВ ПО ОБУЧАЮЩИМ ПРИМЕРАМ НА ОСНОВЕ МУРАВЬИНОГО АЛГОРИТМА

© 2014 г. И. П. Бужинский, В. И. Ульянцев, Д. С. Чивилихин, А. А. Шалыто

Санкт-Петербург, Санкт-Петербургский национальный исследовательский ун-т информационных технологий, механики и оптики

Поступила в редакцию 18.06.13 г., после доработки 25.10.13 г.

Предлагается метод генерации управляющих автоматов по обучающим примерам, в котором в качестве алгоритма поисковой оптимизации используется муравьиный алгоритм. Эффективность метода оценивается на примере генерации автоматов управления моделью беспилотного самолета. При этом показано, что предлагаемый метод обеспечивает производительность и качество выше, чем у метода, основанного на генетических алгоритмах.

DOI: 10.7868/S0002338814020048

Введение. Автоматное программирование – парадигма, в рамках которой программы проектируются в виде систем автоматизированных объектов управления [1]. Автоматизированный объект управления состоит из объекта управления и системы управления, которая может содержать один или несколько управляющих конечных автоматов (далее – автоматов). Одна из областей, для которой целесообразно применение автоматного программирования, – управление объектами со сложным поведением, т.е. объектами, которые могут демонстрировать различное поведение при одинаковых входных воздействиях. Примерами применения автоматного программирования при решении задач данного класса могут служить работы [2–4]. В [3, 4] автоматы строятся автоматически, поскольку ручное их построение является затруднительным. Существуют задачи, для которых ручное построение автоматов и вовсе невозможно.

Для автоматизации процесса построения автоматов необходимо ввести критерий качества. Один из способов это сделать – задать *функцию приспособленности* (ФП), сопоставляющую каждому конечному автомату вещественное число. Нахождение автомата с заданным значением ФП можно осуществлять на основе алгоритмов поисковой оптимизации (например, эволюционных алгоритмов [5–9]). Альтернативный подход к заданию критерия качества состоит в описании набора ограничений, которым должен удовлетворять автомат. Этот подход использован в [100, 111], где задача генерации конечного автомата сводится к задаче о выполнимости булевой формулы (SAT) [122].

В [133] для построения автомата с дискретными выходными воздействиями, управляющего моделью беспилотного самолета, был использован такой алгоритм поисковой оптимизации, как генетический алгоритм. При этом ФП автомата вычислялась на основе моделирования в авиасимуляторе, и поэтому полный цикл генерации автомата занимал *несколько недель* на двух двухъядерных персональных компьютерах. В [144] также был применен генетический алгоритм, но ФП автомата вычислялась не на основе моделирования, а на основе обучающих примеров, что позволило сократить время генерации автомата на персональном компьютере до *нескольких часов*. Отличительная особенность публикации [144] – использование наряду с дискретными непрерывных выходных воздействий.

В настоящей работе предлагается метод генерации автоматов, являющийся усовершенствованием метода, предложенного в [144]. При этом в качестве алгоритма поисковой оптимизации используется модификация муравьиного алгоритма [155, 166], предложенная в [177], что позволяет сократить время генерации автоматов на компьютере с четырьмя ядрами до четверти часа.

1. Постановка задачи. Управляющим конечным автоматом Мили называется шестерка $(S, E, A, \delta, \lambda, s_0)$, где S – конечное множество состояний, E – множество входных событий, A – множество выходных воздействий, $\delta: S \times E \to S$ – функция переходов, $\lambda: S \times E \to A$ – функция выходов, s_0 – начальное состояние.

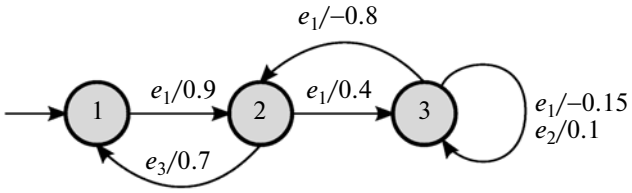

Рис. 1. Пример управляющего автомата

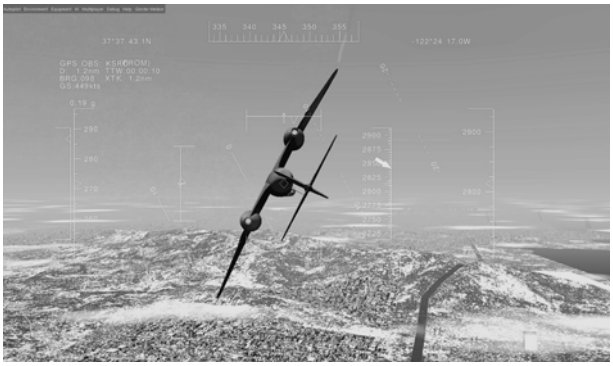

Рис. 2. Авиасимулятор *FlightGear* (снимок экрана)

Пример автомата приведен на рис. 1. Круги на диаграмме соответствуют состояниям автомата, а цифры внутри кругов – их номерам. Начальное состояние 1 отмечено входящей в него слева стрелкой. На каждой из остальных стрелок, обозначающих переходы, слева от косой черты находится входное событие, по которому совершается переход, справа – выходное воздействие (в данном случае – вещественное число), вырабатываемое при совершении перехода. Крайняя правая стрелка соответствует двум переходам сразу.

Сформулируем решаемую задачу. Задан набор обучающих примеров, на основе которых необходимо сгенерировать автомат, имеющий дискретные и непрерывные выходные воздействия, который управляет объектом со сложным поведением. Обучающие примеры записываются человеком и являются примерами желаемого поведения объекта управления.

В настоящей работе, как и в [14], в качестве объекта управления рассматривается модель беспилотного самолета, а желаемое ее поведение – выполнение некоторой фигуры пилотажа. Для моделирования используется авиасимулятор, необходимые требования к которому состоят в возможности записи во время полета параметров самолета (скорости, угла крена и т.п.) и положений органов управления самолетом, характеризуемых числами. В качестве такого симулятора выбран *FlightGear* [18] (рис. 2).

1.1. Управление объектом. Опишем, как происходит взаимодействие автомата и объекта управления. *Кортежем входных воздействий* автомата назовем упорядоченный набор из p вещественных чисел, описывающих состояние объекта управления, подаваемый на вход автомата. Применительно к задаче управления моделью беспилотного самолета кортеж входных воздействий состоит из параметров полета – высоты, скорости, углов курса, тангажа, крена и т.д.

Объект управления обладает *органами управления*, положение которых может изменять автомат. Положения органов управления будем задавать числами. Органы управления, множество положений которых конечно, являются *дискретными*. Примером дискретного органа управления является стартер, который может быть включен или выключен. Обозначим множество значений l-го, $l = \overline{1,d}$ дискретного органа управления как V_l. Положение других органов управления можно задать числами из некоторого отрезка вещественной оси – это *непрерывные* органы управления. Например, положение руля направления, варьирующееся от крайнего левого до крайнего правого, может быть задано числами из отрезка $[-1, 1]$. Обозначим нижнюю и верхнюю границы отрезка возможных значений k-го, $k = \overline{1,c}$ непрерывного органа управления как m_k и M_k соответственно.

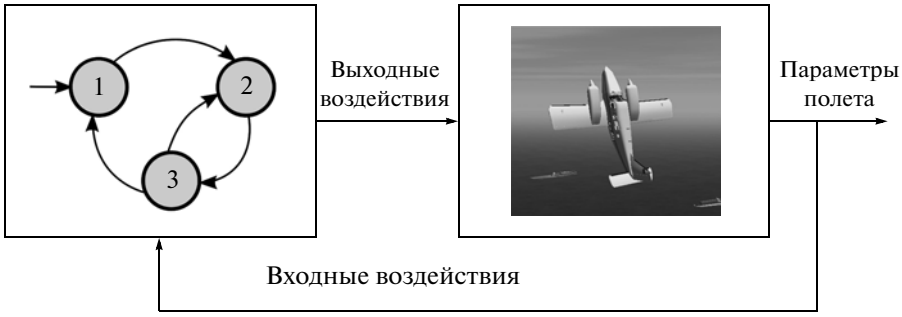

Рис. 3. Процесс взаимодействия автомата и объекта управления

Кортежем выходных воздействий автомата назовем пару из двух упорядоченных наборов чисел: d целых и c вещественных, где d – число дискретных органов управления, c – число непрерывных органов управления. Кортеж выходных воздействий является "снимком" положений всех органов управления в некоторый момент времени. Множество выходных воздействий автомата будем считать образованным всеми возможными кортежами выходных воздействий: $A = V_1 \times \ldots \times V_d \times [m_1, M_1] \times \ldots \times [m_c, M_c]$.

Предикатом будем называть булеву переменную, зависящую от состояния объекта управления или среды, в которой объект находится. Примерами предикатов являются утверждения "вертикальная скорость положительна" или "угол тангажа больше 5°". Будем считать, что всю информацию, необходимую для вычисления предикатов, можно извлечь из кортежей входных воздействий. Фиксируем набор из m предикатов $P_1, \ldots, P_m$. Выбор предикатов для каждого набора обучающих примеров осуществляется вручную.

Реализуем автомат как синхронный – все такты его работы происходят через равные промежутки времени (интервал между тактами равен 0.1 с). В качестве входных событий для автомата будем использовать утверждения "P_i истинно" и "P_i ложно" для каждого предиката P_i. Для каждого события в каждом состоянии автомата будем хранить переход, для которого заданы конечное состояние и кортеж выходных воздействий (более точно – кортеж *изменений* выходных воздействий).

Такт состоит из m переходов: i-й по очередности переход происходит по событию, соответствующему значению предиката P_i. Пусть на некотором такте выполнились переходы, которым сопоставлены кортежи выходных воздействий с непрерывными составляющими, представленными c-мерными векторами $z_1, \ldots, z_m$, тогда непрерывная часть кортежа выходных воздействий всего такта z' формируется по правилу

$$z' = z + \sum_{i=1}^{m} z_i, \tag{1.1}$$

где z – непрерывная часть кортежа выходных воздействий на предыдущем такте. Далее нам будет удобно считать непрерывные воздействия на переходах не ограниченными отрезками $[m_1, M_1], \ldots, [m_c, M_c]$, при этом элементы z' при выходе за границы этих отрезков приравниваются соответствующим границам. Дискретные выходные воздействия такта устанавливаются равными дискретным воздействиям, соответствующим последнему выполнившемуся переходу.

Схема взаимодействия автомата и самолета приведена на рис. 3. Подаваемые на вход автомата кортежи входных воздействий зависят от текущего состояния самолета (параметров полета).

1.2. О б у ч а ю щ и е п р и м е р ы. Теперь формализуем понятие обучающего примера. Обозначим число моментов времени, записанное в i-м обучающем примере, как L_i ($i = \overline{1, N}$, где N – число обучающих примеров). Далее это число будем называть *длиной* i-го обучающего примера. Каждый обучающий пример состоит из двух частей. Первая из них – последовательность кортежей входных воздействий I_i, состоящая из чисел $I_{i,t,j}$, где $t = \overline{1, L_i}$ – момент времени, $j = \overline{1, p}$ – номер входного воздействия в кортеже. Вторая часть – последовательность кортежей выходных воздействий O_i, состоящая из чисел $D_{i,t,l}$ и $C_{i,t,k}$, где i и t определяют обучающий пример и момент времени, $l = \overline{1, d}$ – номер дискретного органа управления, а $k = \overline{1, c}$ – номер непрерывного органа

Таблица 1. Образец обучающего примера ($p = 4$, $d = 1$, $c = 3$, $L_i = 235$)

Значения	Описание	$t = 1$	...	$t = 10$	...	$t = 20$	...	$t = 235$
$I_{i,t,1}$	Угол тангажа, град	3.078	...	3.544	...	4.112	...	2.412
$I_{i,t,2}$	Угол крена, град	−0.076	...	0.351	...	3.413	...	1.759
$I_{i,t,3}$	Угол курса, град	198.03	...	198.11	...	198.41	...	205.64
$I_{i,t,4}$	Скорость, узлы	251.42	...	252.29	...	253.20	...	289.40
$D_{i,t,1}$	Стартер	0	...	0	...	0	...	0
$C_{i,t,1}$	Положение элеронов (число от −1 до 1)	0.000	...	0.032	...	0.073	...	−0.003
$C_{i,t,2}$	Положение руля направления (число от −1 до 1)	0.000	...	0.016	...	0.037	...	−0.001
$C_{i,t,3}$	Положение руля высоты (число от −1 до 1)	−0.035	...	−0.039	...	−0.037	...	−0.011

управления. Разности между соседними моментами времени, записанными в обучающем примере, равны интервалу между тактами автомата. Образец обучающего примера приведен в табл. 1.

2. Метод генерации автоматов. В предлагаемом в настоящей статье методе генерация конечных автоматов производится на основе модификации муравьиного алгоритма [17]. Однако в этой работе рассматривались автоматы только с дискретными выходными воздействиями.

2.1. Функция приспособленности. Используемая в настоящей работе ФП основана на сходстве поведения, которое демонстрирует автомат, принимая на вход кортежи входных воздействий из обучающих примеров, и "эталонного" поведения, записанного в тех же обучающих примерах. Фиксируем некоторый управляющий автомат. Обозначим последовательность кортежей выходных воздействий, выработанных автоматом, которому последовательно были переданы кортежи входных воздействий из i-го обучающего примера, как $\tilde{O}_i$, а ее дискретные и непрерывные элементы – как $\tilde{D}_{i,t,l}$ и $\tilde{C}_{i,t,k}$ соответственно ($t = \overline{1, L_i}, l = \overline{1, d}, k = \overline{1, c}$). При этом $\tilde{D}_{i,1,l} = D_{i,1,l}$, $\tilde{C}_{i,1,k} = C_{i,1,k}$ (начальные выходные воздействия автомата равны начальным выходным воздействиям, записанным в обучающем примере), а по окончании такта с номером t автомат вырабатывает воздействия $\tilde{D}_{i,t+1,l}$ и $\tilde{C}_{i,t+1,k}$.

Будем рассматривать расстояние между последовательностями O_i и $\tilde{O}_i$ в качестве штрафа для автомата:

$$\rho\left(O_i, \tilde{O}_i\right) = \sqrt{\frac{1}{L_i}\sum_{t=2}^{L_i}\frac{1}{d+c}\left(\sum_{l=1}^{d}\left[\tilde{D}_{i,t,l} \neq D_{i,t,l}\right] + \sum_{k=1}^{c}\left(\frac{\tilde{C}_{i,t,k} - C_{i,t,k}}{M_k - m_k}\right)^2\right)}.$$

Квадратные скобки (скобки Айверсона) в формуле выше преобразуют истинные утверждения в единицу, в ложные – в ноль.

Определим ФП, объединив штрафы $\rho\left(O_i, \tilde{O}_i\right)$, вычисленные на всех обучающих примерах:

$$f = 1 - \sqrt{\frac{1}{N}\sum_{i=1}^{N}\rho^2\left(O_i, \tilde{O}_i\right)}.$$

Таким образом, f учитывает различия поведения автомата и "эталонного" его поведения для всех обучающих примеров. Для автоматов, поведение которых хорошо соответствует поведению, записанному в обучающих примерах, f близка к единице. Похожая ФП, отличающаяся от используемой типом нормировки, применена в [14].

2.2. Особь алгоритма поисковой оптимизации. *Каркасом* автомата будем называть автомат, для которого не задана функция выходов (на переходах автомата не расставлены кортежи выходных воздействий). Использование каркаса в качестве особи для алгоритма поисковой оптимизации позволяет сделать пространство поиска дискретным.

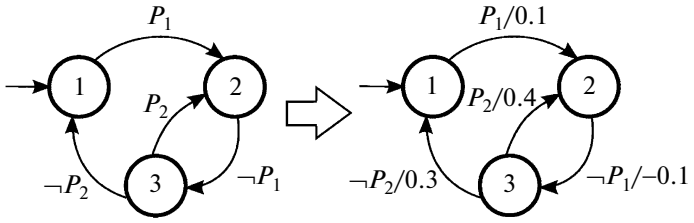

Рис. 4. Процесс расстановки выходных воздействий на ребрах каркаса автомата

Выходные воздействия автомата будем расставлять на каркасе автомата так, чтобы ФП достигла максимума для заданного каркаса автомата, с помощью алгоритма, предложенного в [14]. Такая возможность обусловлена видом функции приспособленности, благодаря которому задача максимизации сводится к задаче решения нескольких систем линейных уравнений. Схематично процесс расстановки выходных воздействий показан на рис. 4. На нем рассмотрен случай наличия у объекта управления двух предикатов, одного непрерывного органа управления и отсутствия дискретных органов управления. Алгоритм расстановки выходных воздействий запускается перед каждым вычислением ФП.

Опишем метод расстановки выходных воздействий. Для максимизации ФП на заданном каркасе автомата необходимо решить следующую оптимизационную задачу, которая может рассматриваться независимо для каждого органа управления:

$$1-\sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{1}{L_i}\sum_{t=2}^{L_i}\frac{1}{d+c}\left(\sum_{l=1}^{d}\left[\tilde{D}_{i,t,l}\neq D_{i,t,l}\right]+\sum_{k=1}^{c}\left(\frac{\tilde{C}_{i,t,k}-C_{i,t,k}}{M_k-m_k}\right)^2\right)}\to\max. \tag{2.1}$$

Искомыми переменными в этой задаче являются дискретные и непрерывные воздействия на каждом переходе автомата, которые пока не присутствуют в (2.1) в явном виде, но от которых зависят величины $\tilde{D}_{i,t,l}$ и $\tilde{C}_{i,t,k}$. Очевидно, что задача упрощается до следующей:

$$\sum_{i=1}^{N}\frac{1}{L_i}\sum_{t=2}^{L_i}\left(\sum_{l=1}^{d}\left[\tilde{D}_{i,t,l}\neq D_{i,t,l}\right]+\sum_{k=1}^{c}\left(\frac{\tilde{C}_{i,t,k}-C_{i,t,k}}{M_k-m_k}\right)^2\right)\to\min. \tag{2.2}$$

Начнем с дискретных органов управления. Зафиксируем, что l – номер дискретного органа управления. Пусть $v_j\in V_l$, $j=\overline{1,n}$, – дискретное выходное воздействие, которое должно быть сопоставлено переходу с номером j, а n – число переходов автомата. За счет избавления от постоянных слагаемых задача (2.2) сводится к следующей:

$$\sum_{i=1}^{N}\frac{1}{L_i}\sum_{t=2}^{L_i}\left[\tilde{D}_{i,t,l}\neq D_{i,t,l}\right]\to\min_{v_1,\ldots,v_n}.$$

Пусть также величина $w_{j,i,t}$, $t=\overline{2,L_i}$, равна единице, если переход j выполнился в конце такта $t-1$ при работе автомата на i-м обучающем примере, и нулю в противном случае (напомним, воздействия $\tilde{D}_{i,t,l}$ при $t>1$ определяются последним переходом такта $t-1$). Получаем:

$$\left[\tilde{D}_{i,t,l}\neq D_{i,t,l}\right]=\sum_{j=1}^{n}\left[v_j\neq D_{i,t,l}\right]w_{j,i,t};$$

$$\sum_{j=1}^{n}\sum_{i=1}^{N}\frac{1}{L_i}\sum_{t=2}^{L_i}\left[v_j\neq D_{i,t,l}\right]w_{j,i,t}\to\min_{v_1,\ldots,v_n}.$$

Слагаемые внешней суммы не зависят друг от друга, поэтому числа v_j могут определяться для каждого перехода независимо. Решение задачи сводится к нахождению значения $v_j\in V_l$, минимизирующего соответствующее ему слагаемое. Нахождение таких значений может быть выполнено за время

$$O\left(\sum_{i=1}^{N}L_i+n|V_l|\right).$$

Перейдем к непрерывным органам управления. Зафиксируем, что k – номер непрерывного органа управления. Пусть $u_j \in \mathbb{R}$, $j = \overline{1,n}$, – непрерывное выходное воздействие, которое должно быть сопоставлено переходу с номером j. Задача оптимизации k-го непрерывного выходного воздействия может быть записана следующим образом:

$$g_k = \sum_{i=1}^{N} \frac{1}{L_i} \sum_{t=2}^{L_i} \left(\tilde{C}_{i,t,k} - C_{i,t,k} \right)^2 \to \min_{u_1,\ldots,u_n}. \tag{2.3}$$

Теперь вернемся к правилу (1.1). Согласно ему, на каждом такте каждое непрерывное выходное воздействие изменяется на сумму воздействий, сопоставленных выполненным на этом такте переходам. Пусть $\beta_{i,t,j}$ – число выполнений перехода j на такте $t-1$ при работе автомата на i-м обучающем примере, а $\alpha_{i,t,j}$ – число его выполнений от начала работы автомата до начала такта t, т.е.

$$\alpha_{i,t,j} = \sum_{t'=2}^{t} \beta_{i,t',j}.$$

С применением введенных обозначений (1.1) можно переписать, используя сумму по всем переходам автомата, а не только по переходам, совершенным на текущем такте:

$$\tilde{C}_{i,t,k} = \tilde{C}_{i,t-1,k} + \sum_{j=1}^{n} \beta_{i,t,j} u_j.$$

Поскольку $\tilde{C}_{i,1,k} = C_{i,1,k}$, формулу можно замкнуть:

$$\tilde{C}_{i,t,k} = C_{i,1,k} + \sum_{j=1}^{n} \alpha_{i,t,j} u_j.$$

Подставим это выражение в (2.3):

$$g_k = g_k(u_1,\ldots,u_n) = \sum_{i=1}^{N} \frac{1}{L_i} \sum_{t=2}^{L_i} \left(C_{i,1,k} + \sum_{j=1}^{n} \alpha_{i,t,j} u_j - C_{i,t,k} \right)^2 \to \min_{u_1,\ldots,u_n}.$$

Для нахождения минимума функции g_k приравняем нулю ее производные по $u_1,\ldots,u_n$:

$$\frac{\partial g_k}{\partial u_{j_0}} = 2 \sum_{i=1}^{N} \frac{1}{L_i} \sum_{t=2}^{L_i} \alpha_{i,t,j_0} \left(C_{i,1,k} + \sum_{j=1}^{n} \alpha_{i,t,j} u_j - C_{i,t,k} \right) = 0, \quad j_0 = \overline{1,n}.$$

Получившиеся условия, записанные для различных j_0, образуют систему линейных уравнений:

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{N} \frac{1}{L_i} \sum_{t=1}^{L_i} \alpha_{i,t,j_0} \alpha_{i,t,j} \right) u_j = \sum_{i=1}^{N} \frac{1}{L_i} \sum_{t=2}^{L_i} \alpha_{i,t,j_0} \left(C_{i,t,k} - C_{i,1,k} \right), \quad j_0 = \overline{1,n}. \tag{2.4}$$

Мы показали, что оптимальные непрерывные выходные воздействия для заданного непрерывного органа управления k могут быть найдены путем решения системы (2.4). Нахождение ее коэффициентов и ее решение методом Гаусса может быть выполнено за время

$$O\left(n^2 \sum_{i=1}^{N} L_i + n^3 \right).$$

Для различных органов управления k левая часть системы оказывается одинаковой, что позволяет производить расстановку выходных воздействий сразу для всех органов управления без ухудшения асимптотической оценки. Более детальные оценки времени работы процедур расстановки как дискретных, так и непрерывных органов управления приведены в [14].

2.3. Муравьиный алгоритм. В настоящей работе, как отмечено выше, автоматы генерируются при помощи модификации муравьиного алгоритма, предложенной в [17], особенностью которой является представление решений задачи в виде вершин графа поиска, в то время как в классическом муравьином алгоритме решениям соответствуют пути в графе. В [17] алгоритм был применен для построения по обучающим примерам автоматов с дискретными воздей-

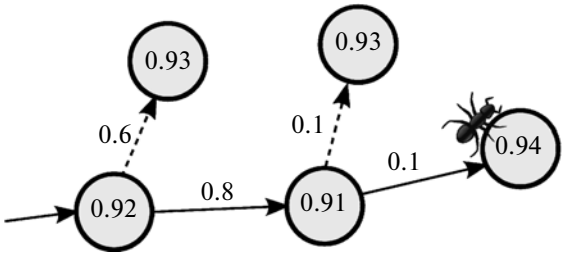

Рис. 5. Часть графа G

ствиями, использование же непрерывных воздействий в настоящей работе не меняет суть алгоритма. Рассмотрим ориентированный граф G, вершинами которого являются автоматы (особи алгоритма, элементы пространства поиска), а ребрами – *мутации* между автоматами. Для данной задачи мутацией будем считать изменение одного перехода каркаса автомата. Граф будем строить постепенно, начиная с единственной вершины – сгенерированного случайным образом каркаса автомата.

Работа алгоритма состоит из последовательности итераций, в начале каждой из которых в некоторые вершины графа G помещаются N_{ants} муравьев, которые далее перемещаются по графу в поисках лучшей особи. Каждый муравей запоминает лучшую особь среди тех, что он посетил. Размещение муравьев происходит случайным образом на вершинах пути, который проделал муравей, нашедший лучшую особь из рассмотренных муравьями за все итерации алгоритма.

Путь муравья формируется следующим образом (предполагаем, что муравей находится в некоторой вершине v).

С вероятностью p_{new} в граф G добавляются N_{mut} новых ребер из вершины v в вершины, отличающиеся от v на одну мутацию. При этом если конечной вершины ребра до этого не было в графе, то она в него добавляется вместе с соответствующим ей каркасом автомата. Среди всех вершин, отличающихся от v на одну мутацию и еще не связанных с v ребром, выбор N_{mut} вершин для добавления в граф случаен.

В противном случае, либо в случае невозможности добавления N_{mut} новых ребер из вершины v, муравей выбирает одну из уже присутствующих в графе вершин-соседей и переходит в нее. Вероятность выбора следующей вершины пропорциональна значению *феромона* на ребре, ведущем в нее из v (используется “метод рулетки”).

Феромон – это величина, связанная с ребрами графа, которую муравьи изменяют в процессе работы алгоритма. Феромон каждого созданного ребра графа G инициализируется некоторым значением τ_0 и изменяется в процессе работы алгоритма. Обновление феромона на ребрах происходит в конце каждой итерации алгоритма, когда все N_{ants} муравьев прекратили свое перемещение. Муравей прекращает перемещение, если посещение предыдущих N_{stag} вершин не приводит к улучшению ФП лучшей особи, найденной муравьем.

Обновление феромона происходит следующим образом. Для каждого ребра uv определим две величины: текущее значение феромона τ_{uv} и максимальное значение τ_{uv}^{best}, которое когда-либо откладывалось на ребре uv. Обновление величины τ_{uv}^{best} происходит не на всем пути, пройденном муравьем, а только на его префиксе, заканчивающемся лучшей на пути муравья особью. Значение, которым обновляется τ_{uv}^{best}, монотонно возрастает с ростом ФП лучшей на пути особи. После обновления величин τ_{uv}^{best} текущее значение феромона каждого ребра графа G пересчитывается по классической для муравьиных алгоритмов формуле:

$$\tau'_{uv} = \rho\tau_{uv} + \tau_{uv}^{\text{best}},$$

где ρ – коэффициент испарения феромона ($0 < \rho < 1$). Если в результате обновления феромона его значение становится меньше τ_0, то оно увеличивается до τ_0. В настоящей работе используются следующие значения параметров муравьиного алгоритма: $N_{\text{ants}} = 4$; $N_{\text{stag}} = 40$; $N_{\text{mut}} = 35$; $p_{\text{new}} = 0.25$; $\rho = 0.35$; $\tau_0 = 0.005$.

Небольшой фрагмент графа G и пример конечного участка пути муравья в нем приведены на рис. 5. Внутри вершин показаны ФП автоматов, на ребрах – значения феромона. Путь муравья обозначен сплошными стрелками, остальные ребра графа – пунктирными.

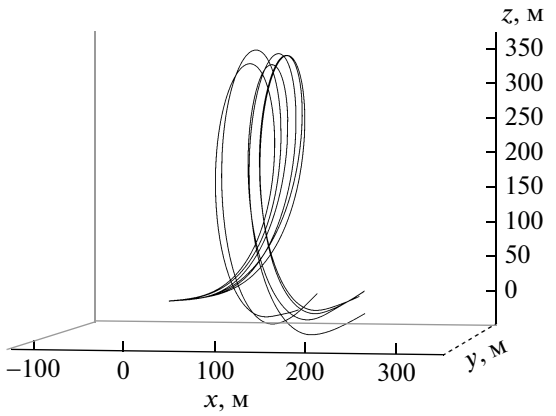

Рис. 6. Примеры траекторий из набора обучающих примеров для “мертвой петли”

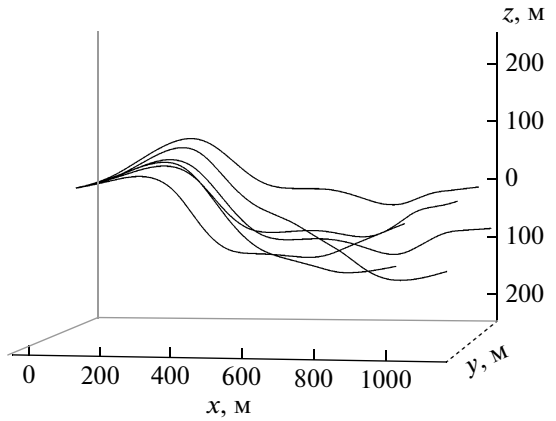

Рис. 7. Примеры траекторий из набора обучающих примеров для “бочки”

3. Экспериментальное исследование. В настоящем разделе представлены результаты применения описанного метода генерации автоматов на выбранных авторами фигурах пилотажа, а также приведено сравнение производительности муравьиного алгоритма с производительностью генетического алгоритма. При реализации генетического алгоритма использовалась работа [14].

3.1. Фигуры пилотажа. С помощью симулятора *FlightGear* были записаны два набора обучающих примеров: набор из 23 примеров, описывающий выполнение “мертвой петли”, и набор из 28 примеров, описывающих выполнение вращения самолета на 360° вокруг своей главной оси, именуемого “бочкой”. Для выполнения “мертвой петли” применялась модель гражданского самолета *Piper PA*34-200*T*, а для выполнения “бочки” – модель реактивного истребителя *Gloster Meteor* (гражданский самолет не смог выполнить эту фигуру).

На рис. 6, 7 приведены примеры траекторий самолета для обучающих примеров из каждого набора. Начальные точки и углы курса всех траекторий совмещены для удобства восприятия. Как можно заметить, для каждой из фигур пилотажа траектории самолета похожи друг на друга, но не идентичны. Причины этого – множество случайных факторов, влияющих на полет самолета, а также непрофессионализм человека, записавшего обучающие примеры. Тем не менее небольшие ошибки в траекториях не являются проблемой для предложенного метода построения автоматов.

3.2. Результаты экспериментального исследования. Для сравнения производительности муравьиного и генетического алгоритмов было произведено по 25 запусков обоих алгоритмов на наборах обучающих примеров, описывающих выполнение “мертвой петли” и “бочки”. Каждый запуск останавливался, если в течение 10000 вычислений ФП ее максимальное достигнутое значение не увеличивалось. Число состояний автомата было фиксировано и равно четырем. Как показали предварительные исследования, этого числа состояний достаточно для решения указанных задач.

Таблица 2. Среднее число вычислений ФП в запусках алгоритмов поисковой оптимизации

Набор обучающих примеров	Муравьиный алгоритм	Генетический алгоритм
“Мертвая петля”	27000	44000
“Бочка”	22000	41000

Таблица 3. Число запусков, в которых достигалось заданное значение ФП на обучающих примерах с “мертвой петлей”

Значение ФП	Муравьиный алгоритм	Генетический алгоритм
0.9890	3	0
0.9887	8	3
0.9884	15	15
0.9881	16	18
0.9878	17	19

Таблица 4. Число запусков, в которых достигалось заданное значение ФП на обучающих примерах с “бочкой”

Значение ФП	Муравьиный алгоритм	Генетический алгоритм
0.9882	11	4
0.9880	22	14
0.9878	23	19
0.9876	24	23
0.9874	25	24

В табл. 2 приведено усредненное по всем запускам число вычислений ФП, потребовавшееся алгоритмам для работы на различных наборах обучающих примеров, а в табл. 3, 4 – результаты всех произведенных запусков. В левых колонках табл. 3, 4 находятся контрольные значения ФП, а в правых – числа запусков, в которых такие значения ФП были достигнуты сравниваемыми алгоритмами. Напомним, что чем ближе значение ФП к единице, тем лучше поведение автомата соответствует обучающим примерам. Как видно из таблиц, высокие значения ФП достигались муравьиным алгоритмом чаще, чем генетическим. Этот факт, а также данные табл. 2 позволяют сделать вывод о том, что муравьиный алгоритм превосходит генетический по производительности на рассматриваемых фигурах пилотажа. Кроме того, для “мертвой петли” существует значение ФП (0.9890), которое сумел достигнуть только муравьиный алгоритм. Это свидетельствует об улучшении качества построения автоматов с помощью этого алгоритма.

Вычислительные эксперименты проводились на персональном компьютере с четырехъядерным процессором *Intel Core* 2 *Quad Q*9400 с тактовой частотой 2.66 ГГц. В обоих алгоритмах вычисления ФП осуществлялись сразу для групп из нескольких десятков или сотен особей, что позволило рассчитывать значения ФП для различных особей параллельно.

Среднее время запуска муравьиного алгоритма на обоих наборах обучающих примеров составляет около 5 мин. При этом для получения автомата с более высоким значением ФП имеет смысл провести два-три запуска. Следовательно, время построения автоматов при применении муравьиного алгоритма составляет около 15 мин. Отметим, что при использовании генетического алгоритма время генерации автоматов составляет около 30 мин, что тоже превосходит результаты, описанные в [14], даже при учете различий в вычислительных конфигурациях, используемых в настоящей работе и в [14] (одно ядро *Intel Core 2 Duo T7250*, 2 ГГц).

3.3. Анализ полученных результатов. Автоматы управления самолетом, построенные муравьиным алгоритмом, запускались в авиасимуляторе *FlightGear* в условиях, аналогичных условиям, при которых были записаны обучающие примеры. Каждому из рассмотренных автоматов было необходимо выполнить требуемую фигуру пилотажа пять раз. Более 90% автоматов справились с заданием. Наиболее типичные некритические ошибки, которые совершали автоматы, были связаны с выполнением заключительной части фигуры пилотажа: по завершении

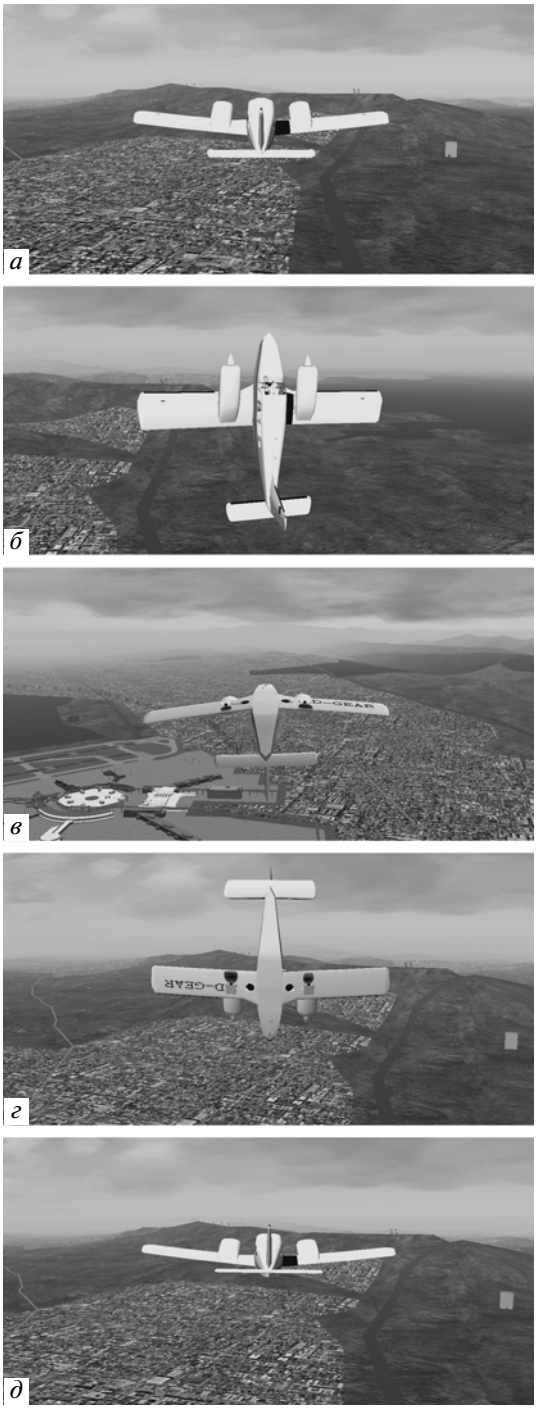

Рис. 8. Снимки экрана, показывающие различные моменты выполнения “мертвой петли” под управлением одного из автоматов в *FlightGear*

“мертвой петли” или “бочки” самолет должен был выровнять свое положение в воздухе и продолжить лететь ровно. Авторы видят следующие возможные решения этой проблемы:

конечные участки обучающих примеров могут быть записаны пилотом более аккуратно;

набор предикатов может быть выбран более удачно;

стабилизацию положения самолета может выполнять отдельный автомат, управление которому будет передаваться автоматически.

На рис. 8, *а*–*д* приведены несколько снимков экрана авиасимулятора *FlightGear*, на которых отражены различные моменты выполнения “мертвой петли” под управлением одного из автома-

тов, сгенерированных муравьиным алгоритмом. Автомат имеет четыре состояния. Визуализация самого автомата оказалась затруднительной из-за большого числа переходов автомата. Их число равно 56.

Заключение. В работе предложен усовершенствованный метод генерации управляющих конечных автоматов. Этот метод, как и прототип, позволяет генерировать по обучающим примерам автоматы, имеющие как дискретные, так и непрерывные выходные воздействия. Повышение производительности и качества метода на рассмотренных примерах достигнуто за счет применения одной из модификаций муравьиного алгоритма в качестве алгоритма поисковой оптимизации, которая используется вместо генетического алгоритма.

СПИСОК ЛИТЕРАТУРЫ

1. *Поликарпова Н.И., Шалыто А.А.* Автоматное программирование. СПб: Питер, 2011. 176 с. http://is.ifmo.ru/books/_book.pdf.
2. *Клебан В.О., Шалыто А.А.* Разработка системы управления малоразмерным вертолетом // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2011. № 2 (72). С. 12–16. http://is.ifmo.ru/works/2011/Vestnik/72-2/02-Kleban-Shalyto.pdf.
3. *Царев Ф.Н., Шалыто А.А.* Применение генетического программирования для генерации автоматов в задаче об "умном муравье" // Тр. IV Междунар. научно-практической конф. "Интегрированные модели и мягкие вычисления в искусственном интеллекте". Т. 2. М.: Физматлит, 2007. С. 590–597. http://is.ifmo.ru/genalg/_ant_ga.pdf.
4. *Царев Ф.Н.* Совместное применение генетического программирования, конечных автоматов и искусственных нейронных сетей для построения системы управления беспилотным летательным аппаратом // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2008. № 53. С. 42–60. http://is.ifmo.ru/works/2008/Vestnik/53/03-genetic-neuro-automata-flying-plates.pdf.
5. *Гладков Л.А., Курейчик В.В., Курейчик В.М.* Генетические алгоритмы. М.: Физматлит, 2006.
6. *Рассел С., Норвиг П.* Искусственный интеллект: современный подход. М.: Вильямс, 2006.
7. *Koza J.R.* Genetic Programming: on the Programming of Computers by Means of Natural Selection. Cambridge: MIT Press, 1992.
8. *Курейчик В.М.* Генетические алгоритмы. Состояние. Проблемы. Перспективы // Изв. РАН. ТиСУ. 1999. № 1. С. 144–160.
9. *Курейчик В.М., Родзин С.И.* Эволюционные алгоритмы: генетическое программирование // Изв. РАН. ТиСУ. 2002. № 1. С. 127–137.
10. *Heule M., Verwer S.* Exact DFA Identication Using SAT Solvers // Grammatical Inference: Theoretical Results and Applications. 10th International Colloquium (ICGI 2010). Lecture Notes in Computer Science. V. 6339. P. 66–79.
11. *Ulyantsev V., Tsarev F.* Extended Finite-state Machine Induction Using SAT-solver // Proc. of the 14th IFAC Symp. "Information Control Problems in Manufacturing – INCOM'12". 2012. P. 512–517.
12. *Кормен Т., Лейзерсон Ч., Ривест Р. и др.* Алгоритмы: построение и анализ. М.: Вильямс, 2005.
13. *Поликарпова Н.И., Точилин В.Н., Шалыто А.А.* Метод сокращенных таблиц для генерации автоматов с большим числом входных переменных на основе генетического программирования // Изв. РАН. ТиСУ. 2010. № 3. С. 100–117.
14. *Александров А.В., Казаков С.В., Сергушичев А.А. и др.* Применение эволюционного программирования на основе обучающих примеров для генерации конечных автоматов, управляющих объектами со сложным поведением // Изв. РАН. ТиСУ. 2013. № 3. С. 85–100; *Aleksandrov A.V., Kazakov S.V., Sergushichev A.A., Tsarev F.N., Shalyto A.A.* The Use of Evolutionary Programming Based on Training Examples for the Generation of Finite State Machines for Controlling Objects with Complex Behavior // J. Computer ans System Sciences International. 2013. V. 52. № 3. P. 410–425.
15. *Dorigo M.* Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano. Italy, 1992.
16. *Dorigo M., Stützle T.* Ant Colony Optimization. MIT Press, US, 2004.
17. *Chivilikhin D., Ulyantsev V.* Learning Finite-State Machines with Ant Colony Optimization // Lecture Notes in Computer Science. 2012. V. 7461/2012. P. 268–275.
18. *FlightGear* [Электронный ресурс]. Режим доступа http://www.flightgear.org/ свободный. Яз. англ. (дата обращения 14.02.13).