ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2013, № 3, с. 85–100

КОМПЬЮТЕРНЫЕ МЕТОДЫ

УДК 004.4'242

ПРИМЕНЕНИЕ ЭВОЛЮЦИОННОГО ПРОГРАММИРОВАНИЯ НА ОСНОВЕ ОБУЧАЮЩИХ ПРИМЕРОВ ДЛЯ ГЕНЕРАЦИИ КОНЕЧНЫХ АВТОМАТОВ, УПРАВЛЯЮЩИХ ОБЪЕКТАМИ СО СЛОЖНЫМ ПОВЕДЕНИЕМ*

© 2013 г. А. В. Александров, С. В. Казаков, А. А. Сергушичев, Ф. Н. Царев, А. А. Шалыто

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

Поступила в редакцию 05.07.11 г., после доработки 21.09.12 г.

Для генерации автоматов управления объектами со сложным поведением предлагается применять эволюционное программирование. При этом вместо известного подхода, в котором оценка качества управляющего автомата производится на основе моделирования, занимающего обычно большое время, используется подход, в котором выполняется сравнение поведения автоматов с поведением, обеспечиваемым за счет управления человеком. Особенность рассматриваемого подхода состоит в том, что он позволяет работать с объектами управления, имеющими не только дискретные, но и непрерывные параметры. Применение подхода иллюстрируется на примере создания автомата, управляющего моделью самолета в режиме "мертвая петля".

DOI: 10.7868/S0002338813020029

Введение. В последнее время для программирования систем со сложным поведением все шире применяется автоматное программирование, в рамках которого поведение программ описывается с помощью конечных детерминированных автоматов (в дальнейшем – автоматов) [1]. В автоматном программировании программы предлагается строить в виде набора автоматизированных объектов управления. Каждый такой объект состоит из объекта управления и системы управления (системы управляющих автоматов). Система автоматов получает на вход события и переменные из внешней среды и от объекта управления. На основании этих данных система управления вырабатывает выходные воздействия для объекта управления. Парадигма управления, основанная на аналогичном подходе, носит название "автоматное управление" [2]. Для многих задач управляющие автоматы удается строить эвристически, но существуют задачи, для которых такое построение невозможно или затруднительно. К этому классу относятся, например, задача "Умный муравей" [3–5], задача "Умный муравей-3" [6] и задача об управлении моделью беспилотного летательного аппарата [7].

Существует несколько подходов к решению последней задачи. Один из них состоит в выделении "идеальной" траектории из нескольких полетов, выполненных человеком, и последующее следование ей. Такой подход рассмотрен в [8]. Другой подход – использование автоматов для управления беспилотным летательным аппаратом и построение таких автоматов с помощью генетических алгоритмов, описанных в [9–13].

В [14] для генерации конечного автомата верхнего уровня, управляющего моделью беспилотного самолета, применяется алгоритм генетического программирования, основанный на использовании метода сокращенных таблиц для представления конечных автоматов. При этом вычисление функции приспособленности базируется на моделировании поведения самолета во внешней среде, которое занимает достаточно большое время.

Цель настоящей работы – разработка лишенного указанного недостатка метода, основанного на эволюционном программировании, для построения автоматов, управляющих объектами со сложным поведением. Для этого предлагается строить автоматы управления таким объектами отдельно для каждого из их режимов работы с помощью эволюционного программирования, *ис-*

* Исследование проводится в рамках Федеральной целевой программы "Научные и научно-педагогические кадры инновационной России на 2009–2013 годы".

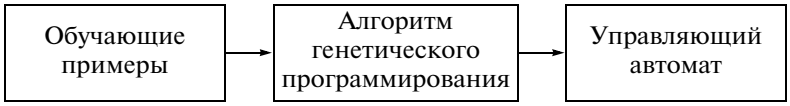

Рис. 1. Общая схема построения управляющего автомата

пользуя обучающие примеры, создаваемые для каждого режима. Задавая большое число обучающих примеров, можно, как и в работе [8], избавиться от неточностей, допускаемых человеком при управлении. Такой подход является развитием идей, предложенных в работе [15], в которой рассматривались только объекты, управляемые дискретными выходными воздействиями.

В настоящей статье указанный подход обобщается на объекты, которые управляются с помощью не только дискретных, но и непрерывных воздействий. При этом в качестве примера объекта со сложным поведением рассмотрена модель самолета в режиме "мертвая петля". Для объединения автоматов, управляющих режимами, с помощью метода сокращенных таблиц [14] или эвристического метода строится головной автомат, каждое из состояний которого соответствует одному из режимов. В результате формируется иерархическая система взаимодействующих автоматов. Вопрос о построении головного автомата в данной работе не рассматривается. Кроме более высокого быстродействия, метод обладает еще одним преимуществом по сравнению с вычислением функции приспособленности с помощью моделирования – в предложенном методе эту функцию не требуется изменять как при переходе от одного режима к другому, так и при переходе от одного объекта управления к другому.

1. Постановка задачи. На рис. 1 представлена общая схема построения управляющего автомата.

Исходными данными для построения управляющего конечного автомата является набор обучающих примеров (тестов), структура которых подробно описана ниже. Тесты, задающие эталонное поведение, создаются человеком. Задача алгоритма эволюционного программирования состоит в построении конечного автомата, который задает поведение объекта управления, наиболее близкое к эталонному. Если использовать достаточно большое число тестов, то дополнительно появляется возможность избавиться от мелких неточностей, которые допускает при управлении человек.

1.1. Органы управления. Объект управления характеризуется набором органов управления, с помощью воздействия на которые объектом можно управлять. Параметры, соответствующие органам управления, будем называть управляющими. Параметры некоторых органов управления могут принимать лишь конечное множество значений – такие органы называются дискретными. Параметры других органов управления характеризуются вещественными значениями – такие органы называются непрерывными. Будем также называть управляющие воздействия на непрерывные органы управления непрерывными, а управляющие воздействия на дискретные органы – дискретными.

Подход, предлагаемый в настоящей работе, ориентирован на объекты управления как с дискретными, так и с непрерывными органами управления. Если все органы дискретны, то следует использовать методику, изложенную в [15]. Непрерывное воздействие изменяет параметр органа управления на некоторую вещественную величину, а дискретное – устанавливает соответствующий орган управления в конкретное значение. Заметим, что последовательное выполнение действий с одним органом управления эквивалентно сумме воздействий на него в случае непрерывного воздействия и последнему воздействию – в случае дискретного.

Например, одним из непрерывных управляющих параметров является угол поворота руля самолета. Непрерывным воздействием на этот орган управления является **изменение** угла его поворота на некоторое значение. Тогда последовательность поворотов руля на x и y градусов эквивалентна повороту руля на $x + y$ град. В свою очередь примером дискретного исполнительного органа является стартер. Дискретное воздействие на него – включение или выключение. Тогда последовательность включений и выключений стартера эквивалентна последнему совершенному над ним действию.

1.2. Структура теста. Схема, отражающая структуру теста, приведена на рис. 2.

Тест T_i имеет две части: I_i и O_i. Каждая из них является последовательностью длины L_i – первая из них состоит из значений входных параметров, а вторая – из соответствующих эталонных значений управляющих параметров, которые записываются в ходе экспериментов, проводимых человеком. Каждый элемент $I_{i,t}$ последовательности входных параметров состоит из p чисел –

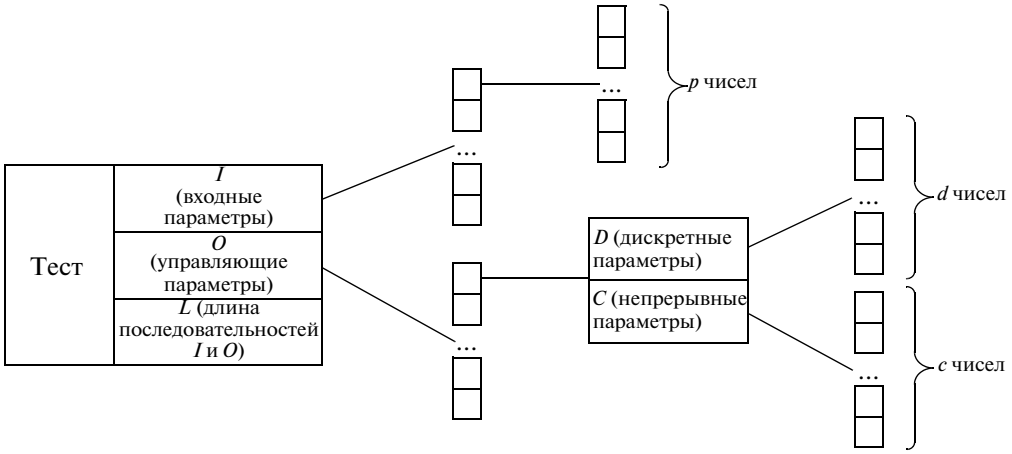

Рис. 2. Схема обучающего примера

значений этих параметров в момент времени t. Элемент $O_{i,t}$ включает в себя два набора: $D_{i,t}$ и $C_{i,t}$. Последовательность $D_{i,t}$ состоит из d дискретных, а $C_{i,t}$ – из c непрерывных параметров. Таким образом, $O_{i,t}$ состоит из $d + c$ чисел. Элементы указанных наборов будут обозначаться как $D_{i,t,k}$ и $C_{i,t,k}$ соответственно. Обозначим через $\tilde{O}_i$ набор управляющих параметров, выданных автоматом на тесте T_i. Структура $\tilde{O}_i$ совпадает со структурой O_i, а входящие в него элементы будем обозначать как $\tilde{D}_{i,t}$, $\tilde{C}_{i,t}$, $\tilde{D}_{i,t,k}$ и $\tilde{C}_{i,t,k}$ соответственно.

2. Предлагаемый алгоритм эволюционного программирования. Классический алгоритм эволюционного программирования состоит из следующих шагов: 1) создание начальной популяции; 2) вычисление значений функции приспособленности; 3) отбор особей для скрещивания; 4) скрещивание; 5) мутация. Поколение, полученное после шага 5, становится текущим, и шаги 2–5 повторяются, пока не будет достигнуто условие останова.

Предлагаемый алгоритм отличается от классического тем, что перед шагом 2 для максимизации значения функции приспособленности **введен дополнительный шаг** – расстановка значений выходных воздействий на переходах “скелета” автомата. Под “скелетом” понимается автомат, значения выходных воздействий которого будут определены в дальнейшем. “Скелет” задается в виде полной таблицы переходов – для каждого состояния и каждого условия истинности или ложности входной переменной задается переход. В “скелете” возможны два типа переходов. Для переходов первого типа символы выходных воздействий не указываются (предыдущие значения управляющих параметров не изменяются). Для переходов второго типа соответствующие символы указываются. Ниже приведено описание этого алгоритма в порядке, удобном для понимания.

2.1. Особь в предлагаемом алгоритме эволюционного программирования. Конечный автомат в предлагаемом алгоритме представляется в виде объекта, который содержит описания переходов для каждого из состояний и номер начального состояния автомата, причем число состояний автомата фиксировано и является параметром алгоритма. Для каждого перехода задается условие, при котором он выполняется. Это условие имеет вид “x_i” или “$\neg x_i$”, где x_i – i-е утверждение (предикат) о состоянии объекта управления (например, “двигатель включен” для объекта управления “самолет”). Эти условия составляются вручную до запуска эволюционного алгоритма и не изменяются в течение его работы. При этом на переходах в особи не задаются значения выходных воздействий z_j, где z_j – j-й кортеж **изменений** управляющих параметров. Таким образом, в особи кодируется только “скелет” автомата, а конкретные выходные воздействия, вырабатываемые на переходах, определяются с помощью алгоритма расстановки действий.

2.2. Работа автомата. Будем считать, что генерируемый автомат является синхронным – все такты его работы одинаковы, а их величина определяется инерционностью объекта управления. При этом под тактом его работы будем понимать запуск автомата на одном наборе входных параметров.

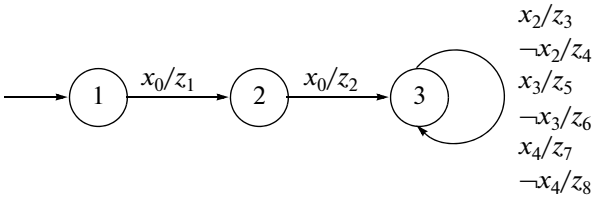

Рис. 3. "Скелет" автомата

На каждом такте система управления получает значения всех входных параметров. После этого вычисляются логические значения всех условий объекта управления в порядке увеличения их номеров. После вычисления соответствующего значения условия оно подается на вход автомата. Автомат в течение одного такта может совершить несколько переходов, и результирующее воздействие автомата в каждый момент времени t может быть составлено из нескольких последовательно выполненных воздействий на отдельных переходах. Напомним, что каждый параметр результирующего воздействия – сумма воздействий в случае непрерывного параметра и последнее воздействие – в случае дискретного.

На рис. 3 изображен возможный "скелет" автомата.

Для данного скелета x_0–x_4 – заданные предикаты, а значения выходных воздействий (кортежей) z_1, ..., z_8 определяются в дальнейшем с помощью алгоритма их расстановки.

2.3. Функция приспособленности. Для вычисления значения функции приспособленности на вход автомата подается каждая из N последовательностей I_i – входной набор i-го теста и определяется последовательность значений управляющих параметров $\tilde{O}_i$, которую генерирует автомат. После этого вычисляется значение функции. Выбранная функция приспособленности отражает близость поведения автомата к эталонному и имеет вид

$$Fitness = 1 - \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{\rho^2(O_i,\tilde{O}_i)}{\rho^2(O_i,0)}}.$$

Для упрощения дальнейших формул обозначим через A_i множитель $\dfrac{1}{\rho^2(O_i,0)}$, тогда предыдущее равенство примет вид

$$Fitness = 1 - \sqrt{\frac{1}{N}\sum_{i=1}^{N} A_i\rho^2(O_i,\tilde{O}_i)}. \qquad (2.1)$$

Здесь под $\rho(O_i,\tilde{O}_i)$ понимается "расстояние" между выходной и эталонной последовательностями управляющих параметров, которое вычисляется по формуле

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

Как предлагалось выше, перед вычислением функции приспособленности к "скелету" автомата применяется алгоритм расстановки воздействий. Этот алгоритм расставляет значения воздействий на переходах так, чтобы максимизировать значение функции приспособленности при заданном "скелете" автомата. Далее, когда говорится о максимизации или минимизации некоторой функции, ее аргументами являются значения выходных воздействия на переходах, а область их изменения зависит от того, каким управляющим параметрам – дискретным или непрерывным – соответствуют воздействия.

2.4. Алгоритм расстановки воздействий – дополнительный шаг в алгоритме эволюционного программирования. При использовании этого алгоритма на переходах "скелета" автомата на его вход подается набор параметров из каждого теста, который был ранее назван входным. При этом запоминаются переходы, совершаемые автоматом.

2.4.1. О д и н т е с т. Для упрощения понимания предлагаемого подхода рассмотрим случай, когда обучающий набор состоит из одного теста T ($N = 1$). Тогда функция приспособленности приобретает вид

$$Fitness = 1 - \sqrt{A\rho^2(O,\tilde{O})}.$$

Здесь и дальше в этом пункте будем опускать первый нижний индекс со значением 1 для упрощения формул (вместо A_1 будем писать A, вместо $O_1 - O$ и т.д.).

Функция приспособленности достигает максимума, когда квадрат расстояния $\rho^2(O,\tilde{O})$ минимален. После подстановки и изменения порядка суммирования получим

$$\sum_{k=1}^{d}\sum_{t=1}^{L}\left[D_{t,k} \neq \tilde{D}_{t,k}\right] + \sum_{k=1}^{c}\sum_{t=1}^{L}\left(C_{t,k} - \tilde{C}_{t,k}\right)^2.$$

Заметим, что сумму для каждого параметра можно минимизировать независимо. Следовательно, достаточно минимизировать сумму

$$\sum_{t=1}^{L}\left[D_{t,k} \neq \tilde{D}_{t,k}\right] \tag{2.2}$$

для каждого k от 1 до d и сумму

$$\sum_{t=1}^{L}\left(C_{t,m} - \tilde{C}_{t,m}\right)^2 \tag{2.3}$$

для каждого m от 1 до c. Далее будем считать индексы k и m фиксированными.

2.4.1.1. Р а с с т а н о в к а д и с к р е т н ы х в о з д е й с т в и й. Пусть $\tilde{D}_{t,k}$ – это еще не определенное значение k-го дискретного параметра воздействия на последнем выполненном переходе в момент времени t. Поэтому выражение (2.2) можно переписать следующим образом:

$$\sum_{i=1}^{n}\sum_{t\in\Psi_i}\left[D_{t,k} \neq \tilde{D}_{t,k}\right]. \tag{2.4}$$

Здесь Ψ_i – множество значений моментов времени t, в которых номер последнего выполненного перехода был равен i, а n – число переходов в автомате. Для минимизации суммы (2.4) достаточно минимизировать сумму вида

$$\sum_{t\in\Psi_i}\left[D_{t,k} \neq \tilde{D}_{t,k}\right]$$

для каждого i от 1 до n. Зафиксируем i. При $t \in \Psi_i$ значение $\tilde{D}_{t,k}$ равно значению k-го дискретного параметра на переходе i, которое не зависит от t. Обозначим его через u ($u = \tilde{D}_{t,k}$ при $t \in \Psi_i$). После введения этого обозначения получим

$$\sum_{t\in\Psi_i}\left[D_{t,k} \neq u\right].$$

Пусть $\tilde{\Psi}_{i,h}$ – множество тех моментов времени t из Ψ_i, в которые $D_{t,k}$ равно v_h. Здесь v_h – h-е возможное значение k-го дискретного параметра. Заметим, что $\tilde{\Psi}_{i,h}$ – непересекающиеся подмножества и при этом выполняется соотношение

$$\bigcup_{h=1}^{G}\tilde{\Psi}_{i,h} = \Psi_i.$$

Учитывая сказанное и обозначив как G число возможных значений k-го дискретного параметра, получим

$$\sum_{h=1}^{G}\sum_{t\in\tilde{\Psi}_{i,h}}[u\neq D_{t,k}]=\sum_{h=1}^{G}\sum_{t\in\tilde{\Psi}_{i,h}}[u\neq v_h]=\sum_{h=1}^{G}[u\neq v_h]\sum_{t\in\tilde{\Psi}_{i,h}}1=\sum_{h=1}^{G}[u\neq v_h]\left|\tilde{\Psi}_{i,h}\right|. \quad (2.5)$$

Выбрав в качестве значения дискретного параметра на i-м переходе $u = v_g$, получим значение суммы (2.5), равное $|\Psi_i| - |\tilde{\Psi}_{i,g}|$, так как все коэффициенты $[u \neq v_h]$ равны единице тогда и только тогда, когда $g \neq h$, а

$$\sum_{h=1}^{G}\left|\tilde{\Psi}_{i,h}\right|=|\Psi_i|.$$

Поэтому для минимизации суммы (2.5) необходимо выбрать то значение v_g, при котором $|\tilde{\Psi}_{i,g}|$ максимально, – наиболее часто встречающееся значение среди $D_{t,k}$ при $t \in \Psi_i$. Если таких значений несколько, то можно выбрать любое из них.

2.4.1.2. Расстановка непрерывных воздействий. Как упоминалось выше, величина изменения m-го непрерывного параметра равна сумме изменений этого параметра на всех выполненных переходах. При этом на каждом из этих переходов рассматриваемый параметр изменяется на определенную (постоянную для этого перехода), но неизвестную величину. Обозначим через u_i величину изменения параметра на i-м переходе. Тогда окончательное значение параметра в момент времени t равно сумме его начального значения (равного нулю) и всех его изменений на каждом выполненном переходе:

$$\tilde{C}_{t,k}=\sum_{i=1}^{n}\alpha_i[t]u_i.$$

Здесь $\alpha_i[t]$ – число выполнений i-го перехода к моменту времени t. Таким образом, сумму (2.3) можно переписать в виде

$$S=\sum_{t=1}^{L}\left(C_{t,m}-\sum_{i=1}^{n}\alpha_i[t]u_i\right)^2.$$

Для того чтобы найти точку, в которой S принимает минимальное значение, необходимо найти точку, в которой частные производные S по u_j для всех j были равны нулю. Эти производные имеют вид

$$\frac{\partial S}{\partial u_j}=-\sum_{t=1}^{L}2\alpha_j[t]\left(C_{t,m}-\sum_{i=1}^{n}\alpha_i[t]u_i\right)=\sum_{t=1}^{L}2\alpha_j[t]\left(\sum_{i=1}^{n}\alpha_i[t]u_i-C_{t,m}\right).$$

После преобразования получим систему линейных уравнений:

$$\begin{cases}\displaystyle\sum_{i=1}^{n}\left(\sum_{t=1}^{L}\alpha_1[t]\alpha_i[t]\right)u_i=\sum_{t=1}^{L}C_{t,m}\alpha_1[t];\\ \displaystyle\sum_{i=1}^{n}\left(\sum_{t=1}^{L}\alpha_2[t]\alpha_i[t]\right)u_i=\sum_{t=1}^{L}C_{t,m}\alpha_2[t];\\ \ldots\\ \displaystyle\sum_{i=1}^{n}\left(\sum_{t=1}^{L}\alpha_n[t]\alpha_i[t]\right)u_i=\sum_{t=1}^{L}C_{t,m}\alpha_n[t].\end{cases}$$

В данной работе эти c систем (для m от 1 до c) решаются методом Гаусса с целью получения искомых величин u_j изменения m-го параметра на переходе j.

2.4.2. Несколько тестов. Обобщим метод на случай нескольких тестов. При этом функция (2.1) достигает максимума, когда минимальна сумма

$$\sum_{i=1}^{N} A_i \rho^2(O_i, \tilde{O}_i).$$

Как и ранее, суммы по каждому параметру можно минимизировать независимо друг от друга. Получим, что необходимо минимизировать сумму

$$\sum_{i=1}^{N} A_i \sum_{t=1}^{L_i} \left[D_{i,t,k} \neq \tilde{D}_{i,t,k} \right] \tag{2.6}$$

для каждого k от 1 до d и сумму

$$\sum_{i=1}^{N} A_i \sum_{t=1}^{L_i} \left(C_{i,t,m} - \tilde{C}_{i,t,m} \right)^2 \tag{2.7}$$

для каждого m от 1 до c. Далее считаем индексы k и m зафиксированными.

2.4.2.1. Расстановка дискретных воздействий. Выделим из суммы (2.6) слагаемые, соответствующие каждому переходу:

$$\sum_{i=1}^{N} A_i \sum_{j=1}^{n} \sum_{t \in \Psi_{i,j}} \left[D_{i,t,k} \neq \tilde{D}_{i,t,k} \right] = \sum_{j=1}^{n} \sum_{i=1}^{N} A_i \sum_{t \in \Psi_{i,j}} \left[D_{i,t,k} \neq \tilde{D}_{i,t,k} \right].$$

Здесь $\Psi_{i,j}$ – множество таких моментов времени t, при которых номер последнего выполненного перехода автомата, запущенного на тесте i, равен j, а n – число переходов в автомате. Таким образом, для каждого j от 1 до n необходимо минимизировать выражение вида

$$\sum_{i=1}^{N} A_i \sum_{t \in \Psi_{i,j}} \left[D_{i,t,k} \neq \tilde{D}_{i,t,k} \right].$$

Зафиксируем j. Как и в случае, когда обучающий пример состоит из одного теста, $\tilde{D}_{i,t,k}$ при $t \in \Psi_{i,j}$ равно значению k-го дискретного параметра на переходе j, которое не зависит от t. Снова обозначим его через u.

Пусть $\tilde{\Psi}_{i,j,h}$ – это множество тех моментов времени t из $\Psi_{i,j}$, при которых $D_{i,t,k}$ равно v_h. Как и раньше, v_h – h-е возможное значение k-го дискретного параметра, подмножества $\tilde{\Psi}_{i,j,h}$ являются непересекающимися, выполняется соотношение

$$\bigcup_{h=1}^{G} \tilde{\Psi}_{i,j,h} = \Psi_{i,j}.$$

Обозначая как G число возможных значений этого параметра, получим

$$\sum_{i=1}^{N} A_i \sum_{t \in \Psi_{i,j}} [D_{i,t,k} \neq u] = \sum_{i=1}^{N} A_i \sum_{h=1}^{G} \sum_{t \in \tilde{\Psi}_{i,j,h}} [D_{i,t,k} \neq u] =$$

$$= \sum_{i=1}^{N} A_i \sum_{h=1}^{G} [v_h \neq u] \sum_{t \in \tilde{\Psi}_{i,j,h}} 1 = \sum_{i=1}^{N} A_i \sum_{h=1}^{G} [u \neq v_h] \left| \tilde{\Psi}_{i,j,h} \right|.$$

Выбрав v_g в качестве значения дискретного параметра на j-м переходе ($u = v_g$) и подставив в предыдущую сумму, запишем ее в виде:

$$\begin{aligned} &\sum_{i=1}^{N} A_i \sum_{h=1}^{G} [v_g \neq v_h] \left| \tilde{\Psi}_{i,j,h} \right| = \sum_{i=1}^{N} A_i \sum_{h=1}^{G} [g \neq h] \left| \tilde{\Psi}_{i,j,h} \right| = \sum_{i=1}^{N} A_i \left(\sum_{h=1}^{G} \left| \tilde{\Psi}_{i,j,h} \right| - \left| \tilde{\Psi}_{i,j,g} \right| \right) = \\ &= \sum_{i=1}^{N} A_i \left(\left| \Psi_{i,j} \right| - \left| \tilde{\Psi}_{i,j,g} \right| \right) = \sum_{i=1}^{N} A_i \left| \Psi_{i,j} \right| - \sum_{i=1}^{N} A_i \left| \tilde{\Psi}_{i,j,g} \right|. \end{aligned} \tag{2.8}$$

Для минимизации суммы (2.8) необходимо выбрать то значение v_g, при котором

$$\sum_{i=1}^{N} A_i \left|\tilde{\Psi}_{i,j,g}\right|$$

максимально. Таким образом, на j-м переходе в качестве значения k-го дискретного параметра следует выбрать v_g, где

$$g = \arg\max_{g} \sum_{i=1}^{N} A_i \left|\tilde{\Psi}_{i,j,g}\right|.$$

2.4.2.2. Расстановка непрерывных воздействий. Ввиду того, что величина изменения m-го непрерывного параметра в момент времени t ($\tilde{C}_{i,t,m}$) равна сумме изменений этого параметра на всех выполненных переходах к этому моменту, из формулы (2.7) получим

$$S = \sum_{i=1}^{N} A_i \sum_{t=1}^{L_i} \left(C_{i,t,m} - \sum_{j=1}^{n} \alpha_{i,j}[t] u_j \right)^2.$$

Здесь, как и раньше, u_j – неизвестная величина изменения m-го непрерывного параметра на j-м переходе, а $\alpha_{i,j}[t]$ – число выполнений j-го перехода к моменту времени t автомата, запущенного на тесте i. Производная S по u_h имеет вид

$$\frac{\partial S}{\partial u_h} = \sum_{i=1}^{N} A_i \sum_{t=1}^{L_i} 2\alpha_{i,h}[t] \left(\sum_{j=1}^{n} \alpha_{i,j}[t] u_j - C_{i,t,m} \right).$$

Приравняв все производные к нулю, для каждого непрерывного m-го параметра получим систему из n уравнений:

$$\begin{cases} \sum\limits_{j=1}^{n} \left(\sum\limits_{i=1}^{N} A_i \sum\limits_{t=1}^{L_i} \alpha_{i,1}[t]\alpha_{i,j}[t] \right) u_j = \sum\limits_{i=1}^{N} A_i \sum\limits_{t=1}^{L_i} C_{i,t,m}\alpha_{i,1}[t]; \\ \sum\limits_{j=1}^{n} \left(\sum\limits_{i=1}^{N} A_i \sum\limits_{t=1}^{L_i} \alpha_{i,2}[t]\alpha_{i,j}[t] \right) u_j = \sum\limits_{i=1}^{N} A_i \sum\limits_{t=1}^{L_i} C_{i,t,m}\alpha_{i,2}[t]; \\ \dots \\ \sum\limits_{j=1}^{n} \left(\sum\limits_{i=1}^{N} A_i \sum\limits_{t=1}^{L_i} \alpha_{i,n}[t]\alpha_{i,j}[t] \right) u_j = \sum\limits_{i=1}^{N} A_i \sum\limits_{t=1}^{L_i} C_{i,t,m}\alpha_{i,n}[t]. \end{cases}$$

Каждая из этих систем решается методом Гаусса. Полученные u_j – искомые величины изменений рассматриваемого непрерывного m-го параметра на переходе j. Отметим, что линейность системы уравнений зависит от вида функции приспособленности. Первоначально авторами была выбрана функция

$$Fitness = \frac{1}{N} \sum_{i=1}^{N} \sqrt{1 - A_i \rho^2(O_i, \tilde{O}_i)},$$

что приводило к системе нелинейных уравнений и резко усложняло решение задачи.

2.5. Теоретическая оценка временной сложности вычисления функции приспособленности. Оценим время работы алгоритма расстановки воздействий в случае нескольких тестов. Обозначим за L сумму $\sum\limits_{i=1}^{N} L_i$. При расстановке дискретных воздействий необходимо совершить $O(n + L)$ действий для каждого k от 1 до d. Общее число действий – $O(dn + dL) = O(n + L)$, если считать d константой. Заметим, что в случае расстановки непрерывных воздействий матрица левой части системы имеет размер $n \times n$. При выбранной функции

приспособленности система может быть решена методом Гаусса, который требует $O(n^3)$ действий. Поскольку матрица системы одна и та же для каждого значения k, то суммарно все системы могут быть решены с помощью $O(n^2(n+c))$ действий. Для формирования матрицы системы требуется $O(n^2L)$ действий. Для формирования k-го столбца свободных членов – $O(nL)$ действий для k от 1 до c. Суммарно на расстановку непрерывных воздействий требуется число действий, равное $O(n(n+c)(n+L)) = O(n^3 + n^2L)$, если c считать константой. Суммарно на расстановку всех воздействий требуется $O(n^3 + n^2L)$ действий.

По определению "О" существуют константы α и β, такие, что расстановка всех воздействий на переходах "скелета" требует не более $\alpha(n^3 + n^2L) + \beta$ простых арифметических операций. Эта оценка линейна по суммарной длине тестов (которая пропорциональна времени управления объектом). Для того чтобы она была приемлемой для практического применения, необходимо выбрать относительно небольшое n. Отметим также, что при больших значениях n время работы алгоритма эволюционного программирования существенно повышается, так как его пространство поиска экспоненциально увеличивается с ростом n. В настоящей работе n не превышало 130. Конкретные значения этих констант α и β зависят от реализации алгоритма вычисления функции приспособленности (при аккуратной реализации алгоритма они невелики). Заметим, что в подходе, предложенном в [14], с которым производится сравнение, вычисление функции приспособленности также имеет линейную относительно времени управления объектом оценку сложности, но в ней константы обычно значительно больше из-за использования моделирования.

2.6. Операторы алгоритма эволюционного программирования. В настоящем разделе описаны операторы отбора, мутации и скрещивания, применяемые в разработанном алгоритме эволюционного программирования.

2.6.1. Оператор отбора. Этот оператор необходим для выделения из текущего поколения наиболее подходящих для решения задачи особей и добавления их в промежуточное поколение. Для сравнения особей применяется функция приспособленности, сопоставляющая каждой особи число, характеризующее, насколько хорошо автомат, соответствующий особи, подходит для решения задачи. При этом отметим, что чем больше это число, тем лучшей считается особь. В данной работе используется *турнирный метод* отбора [9]. В этом методе случайным образом выбирается k особей из текущего поколения. Среди них определяется лучшая особь, которая и добавляется в промежуточное поколение. Турнир проводится столько раз, сколько особей в поколении.

2.6.2. Оператор мутации. При применении этого оператора выполняется с заданной вероятностью каждое из действий: изменение начального состояния; добавление, удаление или изменение случайного перехода в "скелете" автомата. Под изменением перехода понимается изменение состояния, в которое выполняется переход по условию, на случайно выбранное.

2.6.3. Оператор скрещивания. В скрещивании принимают участие две особи. Результатом применения оператора также являются две особи. Обозначим "родительские" особи как *P1* и *P2*, а "дочерние" – *C1* и *C2*. Тогда для стартовых состояний *C1.is* и *C2.is* справедливо одно из следующих утверждений: *C1.is* = *P1.is* и *C2.is* = *P2.is*; *C1.is* = *P2.is* и *C2.is* = *P1.is*. Далее для каждой ячейки таблицы переходов $t[i][j]$ с равной вероятностью выбирается один из следующих вариантов: $C1.t[i][j] = P1.t[i][j]$ и $C2.t[i][j] = P2.t[i][j]$; $C1.t[i][j] = P2.t[i][j]$ и $C2.t[i][j] = P1.t[i][j]$.

2.7. Предлагаемый метод. Обобщая изложенное, сформулируем предлагаемый метод генерации управляющих автоматов с помощью эволюционного программирования на основе обучающих примеров.

1. Задается объект управления (например, в виде эмулятора) с известным интерфейсом управления и возможностью записи входных и управляющих параметров.

2. Система управления имеет иерархическую структуру. Выделяются режимы, для каждого из которых с помощью предлагаемого метода строится автомат. Автомат верхнего уровня, отвечающий за переключение режимов, может быть построен вручную или сгенерирован с помощью метода, изложенного в [14].

3. Для каждого режима многократно выполняются следующие действия.

3.1. Выбор параметров. В качестве входных *воздействий* для автомата используются предикаты от входных параметров, а в качестве выходных воздействий – изменения управляющих параметров. На каждом переходе автомата располагается кортеж из стольких чисел, сколько было выбрано управляющих параметров.

3.2. Создание тестов.

3.2.1. Запуск объекта управления с максимально возможной частотой записи параметров. Для разных режимов в качестве тестов используется различное число наборов параметров. Совокупность этих наборов для каждого запуска является одним тестовым примером. Записанные человеком последовательности будем называть эталонными.

3.2.2. Эвристический выбор числа тестов. При увеличении их числа растет качество результата, но увеличивается время сбора данных и время работы алгоритма эволюционного программирования для построения автомата.

3.3. Использование эволюционного программирования.

3.3.1. Особью является "скелет" автомата – автомат с фиксированным числом состояний, содержащий переходы, на которых расставлены условия переходов и действия, но значения действий не определены.

3.3.2. Выбор функции приспособленности. Она должна отражать степень схожести эталонных (полученных человеком) и сгенерированных автоматом выходных воздействий. От вида этой функции зависит сложность работы алгоритма – придется решать системы линейных или нелинейных уравнений, каждая из которых соответствует одному непрерывному управляющему параметру.

3.3.3. Формирование набора предикатов, которые образуют условия на переходах автоматов, выполняется эвристически.

3.3.4. Написание вручную для каждого предиката кода (на одном из языков программирования), который его реализует на основе входных параметров.

3.3.5. Создание оператора, преобразующего "скелет" в автомат таким образом, чтобы значение функции приспособленности для этого автомата было максимальным из всех возможных для автоматов, соответствующих этому "скелету".

3.3.5.1. Запуск "скелета" автомата на тестовых примерах и запись последовательности выполненных переходов.

3.3.5.2. Определение для каждого такта работы объекта управления значений управляющих параметров, выраженных через начальные значения и все неизвестные изменения на переходах. Для непрерывных действий это значение – сумма значения на предыдущем такте и изменений, выполненных на текущем такте, а для дискретных – значение на последнем из выполненных переходов.

3.3.5.3. Подстановка выражений для значений управляющих параметров в функцию приспособленности. Таким образом, функция приспособленности параметризуется значениями действий на переходах "скелета".

3.3.5.4. Решение задачи максимизации функции приспособленности. При хорошем выборе функции приспособленности выражение для значений воздействий над дискретными параметрами можно получить в явном виде. Для непрерывных воздействий их значения определяются в результате решения системы уравнений. Эта система получается путем приравнивания к нулю частных производных функции приспособленности по неизвестным воздействиям. При этом если функция приспособленности выбрана хорошо, то получающаяся система линейна. Таким образом, функция приспособленности должна быть "хорошей" как для дискретных, так и для непрерывных параметров. Именно такой и является функция приспособленности (2.1).

3.3.6. Выбор модели алгоритма эволюционного программирования и выбор его характеристик (число особей в поколении, размер элиты, выбор функций отбора и скрещивания, величина такта и т.п.).

3.3.7. Запуск алгоритма.

3.4. Запуск полученных автоматов и оценка их поведения.

3. Проверка работоспособности метода. Для проверки работоспособности предложенного метода была поставлена задача генерации автомата, обеспечивающего управление самолетом в режиме "мертвая петля".

3.1. Взаимодействие сгенерированного автомата с моделью беспилотного самолета. Существуют симуляторы, моделирующие полет самолета. Авторами был выбран свободный кроссплатформенный симулятор *FlightGear* (http://www.flightgear.org), который применительно к настоящей работе позволяет осуществлять как ручное, так и автоматное управление моделью самолета. На рис. 4 представлен снимок экрана симулятора *FlightGear* в момент начала разгона.

Симулятор позволяет осуществлять сохранение параметров полета (скорость, направление полета и т.д.) и параметров самолета (положение руля, элеронов, состояние стартера и т.п.). Па-

Рис. 4. Снимок экрана симулятора *FlightGear* в момент начала разгона

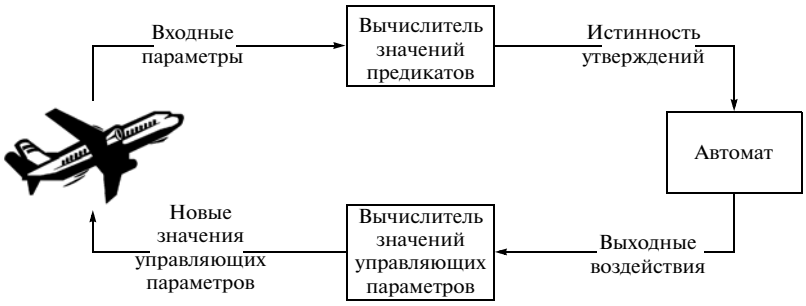

Рис. 5. Управление самолетом с помощью автомата

раметры полета являются входными параметрами для системы управления, а параметры самолета – управляющими, так как за счет их изменений выполняется управление самолетом. Значения управляющих параметров формируются автоматом (рис. 5).

3.2. Задача генерации автомата, выполняющего "мертвую петлю". Эта задача может быть сформулирована следующим образом: требуется построить автомат, под управлением которого модель самолета делает мертвую петлю, а затем ровно пролетает несколько секунд.

3.2.1. Использование предлагаемого метода. Кратко рассмотрим основные шаги применения предлагаемого метода на рассматриваемом примере: сформирован необходимый и достаточный набор утверждений о состоянии самолета; записаны три набора тестов, которые рассматривались независимо друг от друга (каждый набор состоял из 10 тестов; каждый тест состоял из нескольких тысяч наборов входных и управляющих параметров; опрос параметров производился 10 раз в секунду); алгоритм эволюционного программирования запускался для каждого из трех наборов тестов и каждого набора параметров алгоритма.

Перечислим параметры алгоритма и их значения: размер поколения – 100 особей; число состояний автомата – от двух до пяти; вероятность мутации – 0.5; метод отбора – турнирный; размер элиты – две особи; величина такта – 0.1 с.

Вычисления производились на одном ядре компьютера с процессором *Intel Core 2 Duo T7250* с тактовой частотой 2 ГГц под управлением операционной системы *Microsoft Windows XP*. Язык программирования – *Java*.

В среднем время работы алгоритма составляло около 10 ч для одного набора параметров и одного набора тестов. При этом было вычислено около двух тысяч поколений. Таким образом, создание и обработка одного поколения в среднем занимали около 20 с или 0.2 с на один автомат, что значительно меньше, чем в работе [14], где это занимало около 5 мин.

Выбранные утверждения: x_0 – двигатель включен; x_1 – ускорение изменения направления движения самолета больше нуля; x_2 – скорость изменения направления движения самолета больше нуля; x_3 – величина отклонения от начального направления меньше одного градуса; x_4 – величина отклонения от начального направления больше нуля (отклонение влево); x_5 – ускорение изменения крена (угла наклона) больше нуля; x_6 – скорость изменения крена больше нуля; x_7 – крен самолета маленький (меньше 1°); x_8 – крен больше нуля (самолет завалился на правый

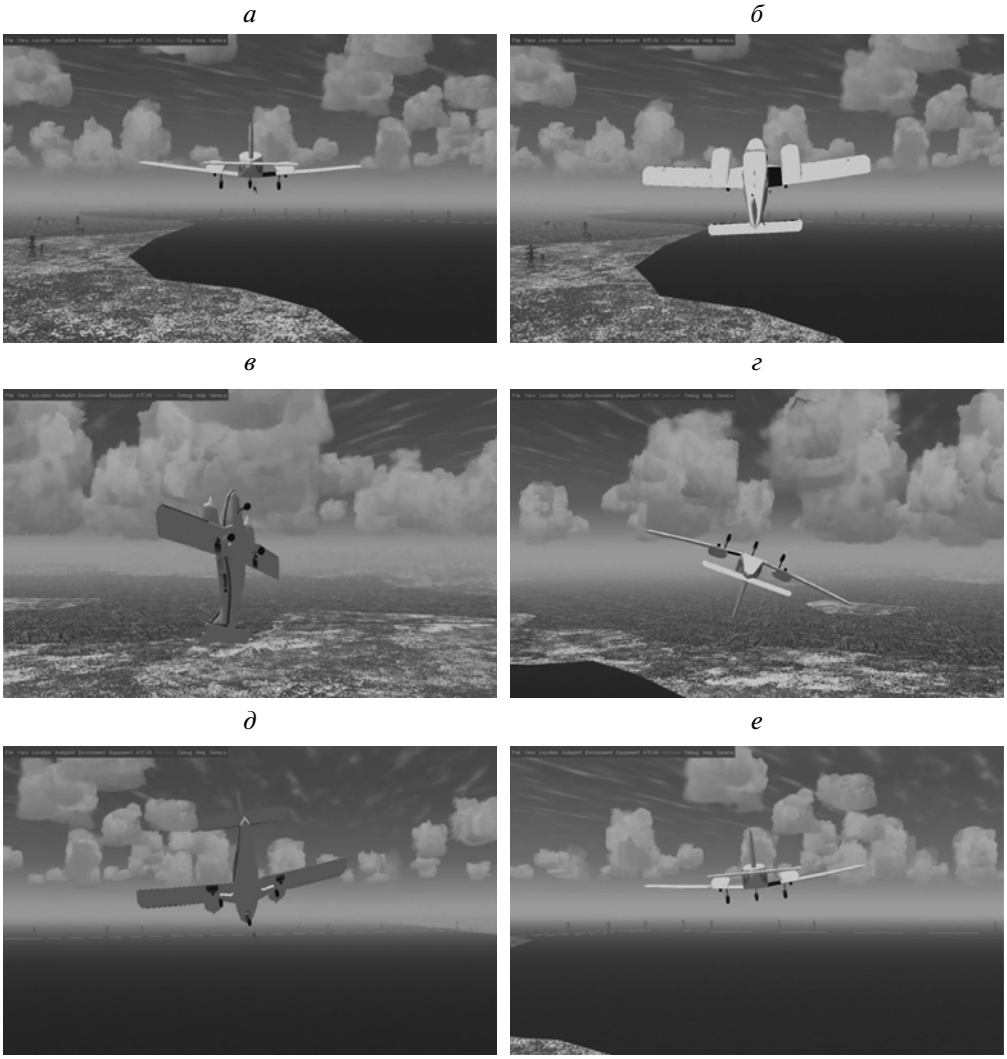

Рис. 6. Кадры видеозаписи одного из обучающих примеров выполнения “мертвой петли”

бок); x_9 – ускорение изменения вертикальной скорости самолета больше нуля; x_{10} – скорость изменения вертикальной скорости больше нуля; x_{11} – вертикальная скорость маленькая (меньше 0.1 м/с); x_{12} – вертикальная скорость больше нуля (самолет поднимается).

Управляющими органами являются: магнето, стартер, дроссель, элероны, руль высоты и руль направления. Первые два органа являются дискретными, а остальные – непрерывными. При этом, например, воздействие “включить стартер” является дискретным, а “повернуть руль на 0.5° вправо” – непрерывным. В результате запусков алгоритма эволюционного программирования было установлено, что автоматы с тремя-четырьмя состояниями “ведут себя” достаточно хорошо, а также, что с увеличением числа состояний автомата его структура становится все более сложной, а поведение ухудшается.

3.2.2. З а п и с а н н ы е т е с т ы. На рис. 6, *а*–*е* приведены 6 кадров видеозаписи одного из обучающих примеров выполнения “мертвой петли”.

Следует отметить, что данный трюк является классической “мертвой петлей” Нестерова, но из-за того, что камера, с которой записывается вид на самолет, меняет свое положение, может создаться впечатление, что выполняется другой более сложный трюк.

Приведем ссылки на некоторые видеозаписи полетов под управлением человека: успешное выполнение “мертвой петли” (этот полет вошел в обучающий набор) – http://www.youtube.com/watch?v=G5Kcx0ohpNo; неудачное выполнение (этот полет не вошел в обучающий набор) – http://www.youtube.com/watch?v=OGVTch-a97A.

Описание переходов лучшего автомата

Дуга	Предикаты	Дискретные выходные воздействия		Непрерывные выходные воздействия			
		магнето	стартер	дроссель	элероны	руль высоты	руль направления
0 → 0	not x_0	0	0	1.00000	0.05239	0.01043	0.00000
	not x_2	0	0	0.00000	−0.01315	0.02117	0.00000
	not x_4	0	0	0.00000	0.00199	−0.09535	0.00000
	not x_6	0	0	0.00000	0.01336	−0.07833	0.00000
	not x_8	0	0	0.00000	0.00178	−0.10299	0.00000
	not x_{11}	3	0	0.00000	−0.00178	0.01466	0.00000
0 → 1	not x_5	0	0	0.00000	−0.01119	0.09089	0.00000
	x_9	0	0	0.00000	−0.01622	0.20543	0.00000
	not x_9	0	0	0.00000	0.00000	0.00000	0.00000
0 → 2	x_1	0	0	0.00000	0.00217	0.01965	0.00000
	x_3	0	0	0.00000	0.02432	0.01225	0.00000
	x_5	0	0	0.00000	0.00000	0.00000	0.00000
	x_6	0	0	0.00000	0.00000	0.00000	0.00000
	not x_{12}	3	0	0.00000	0.00018	−0.00087	0.00000
0 → 3	x_4	0	0	0.00000	−0.01352	−0.08765	0.00000
	not x_7	0	0	0.00000	−0.00394	0.06189	0.00000
	not x_{10}	0	0	0.00000	−0.00688	0.05064	0.00000
1 → 0	not x_0	0	0	1.00000	0.05239	0.01043	0.00000
	x_2	0	0	0.00000	−0.02552	−0.01522	0.00000
	x_4	0	0	0.00000	0.00000	0.00000	0.00000
	x_{11}	3	0	0.00000	0.00000	0.00000	0.00000
1 → 1	not x_1	0	0	0.00000	−0.00132	0.00866	0.00000
	not x_2	0	0	0.00000	0.00000	0.00000	0.00000
	x_5	0	0	0.00000	−0.02809	−0.01434	0.00000
	x_7	0	0	0.00000	−0.00054	−0.00506	0.00000
	not x_{12}	3	0	0.00000	−0.01224	−0.03468	0.00000
1 → 2	x_0	0	0	0.00000	−0.00161	−0.00478	0.00000
	x_3	0	0	0.00000	0.00000	0.00000	0.00000
	not x_3	0	0	0.00000	−0.00274	−0.00202	0.00000
	x_6	0	0	0.00000	−0.01154	−0.11022	0.00000
	x_8	0	0	0.00000	−0.01082	−0.06436	0.00000
	not x_9	0	0	0.00000	−0.00993	−0.06966	0.00000
1 → 3	not x_5	0	0	0.00000	−0.03879	−0.09682	0.00000
	not x_{10}	0	0	0.00000	0.00172	0.03226	0.00000
	x_{12}	3	0	0.00000	0.00497	−0.01437	0.00000
2 → 0	not x_1	0	0	0.00000	0.03000	0.06766	0.00000
	not x_3	0	0	0.00000	0.00000	0.00000	0.00000
	x_9	3	0	0.00000	0.01939	0.04064	0.00000
2 → 1	x_0	0	0	0.00000	0.02731	0.02049	0.00000
	not x_0	0	0	0.00000	0.02047	−0.02413	0.00000
	x_4	0	0	0.00000	0.00110	−0.00006	0.00000
	not x_4	0	0	0.00000	0.00000	0.00000	0.00000
	x_8	0	0	0.00000	0.01159	0.06208	0.00000

Таблица. Окончание

Дуга	Предикаты	Дискретные выходные воздействия		Непрерывные выходные воздействия			
		магнето	стартер	дроссель	элероны	руль высоты	руль направления
2 → 2	x_2	0	0	0.00000	0.01880	0.00322	0.00000
	x_6	0	0	0.00000	−0.02062	−0.01551	0.00000
	not x_7	0	0	0.00000	0.00309	0.05025	0.00000
	not x_{10}	0	0	0.00000	−0.00054	−0.01195	0.00000
	not x_{11}	3	0	0.00000	0.01573	0.06745	0.00000
	not x_{12}	3	0	0.00000	−0.00140	−0.00006	0.00000
2 → 3	not x_3	0	0	0.00000	−0.00889	−0.04245	0.00000
	x_5	0	0	0.00000	0.00000	0.00000	0.00000
	x_7	0	0	0.00000	0.00239	0.03443	0.00000
	x_{11}	0	0	0.00000	0.00000	0.00000	0.00000
3 → 0	x_1	0	0	0.00000	0.00661	−0.00695	0.00000
	not x_{10}	0	0	0.00000	0.01817	0.04933	0.00000
3 → 1	x_3	0	0	0.00000	0.02187	0.03925	0.00000
	x_5	0	0	0.00000	0.00905	0.09217	0.00000
	not x_6	0	0	0.00000	0.00664	0.08602	0.00000
	not x_{12}	3	0	0.00000	0.00556	−0.07512	0.00000
3 → 2	x_4	0	0	0.00000	−0.01550	−0.09491	0.00000
	x_8	0	0	0.00000	−0.00498	0.01483	0.00000
	x_{10}	0	0	0.00000	0.00000	0.00000	0.00000
	not x_{11}	3	0	0.00000	0.00177	−0.03823	0.00000
3 → 3	not x_0	0	0	1.00000	0.04394	0.11148	0.00000
	not x_2	0	0	0.00000	−0.02054	−0.17037	0.00000
	not x_7	0	0	0.00000	−0.00315	0.00256	0.00000
	not x_9	0	0	0.00000	−0.01064	0.00960	0.00000
	x_{12}	3	0	0.00000	0.00757	0.00196	0.00000

3.2.3. Полученные результаты. Было проведено около 50 запусков алгоритма эволюционного программирования, в каждом из которых выбирался автомат с наибольшим значением функции приспособленности. Эти запуски отличались друг от друга используемыми параметрами и наборами тестов. Полеты моделей самолетов, управляемых выбранными автоматами, были просмотрены авторами. После этого автомат, используемый в полете, больше других похожем на идеальную "мертвую петлю", был назван лучшим. Этот автомат имеет четыре состояния и 68 переходов (см. таблицу). При этом отметим, что идеальная "мертвая петля" может отличаться от эталонной, которая выполняется вручную.

В процессе наблюдения за ходом выполнения "мертвом петли" под управлением лучшего автомата было установлено, что в зависимости от параметров среды и самолета при запуске автомата возможны три варианта выполнения "петли".

1. В большинстве случаев модель самолета, как и при управлении вручную, выполняет одну "мертвую петлю" и летит дальше.

2. Иногда модель самолета может выполнить несколько "мертвых петель" с некоторым интервалом. Это происходит в случае, когда значения параметров модели самолета в конце выполнения "мертвой петли" схожи со значениями параметров в начале ее выполнения. Это приводит к тому, что если автомат после осуществления "петли" находится в том же состоянии, в котором был в начале, то поведение модели зацикливается. В ходе экспериментов авторы неоднократно наблюдали выполнение двух "мертвых петель" подряд. Выполнение большего числа "петель" не наблюдалось.

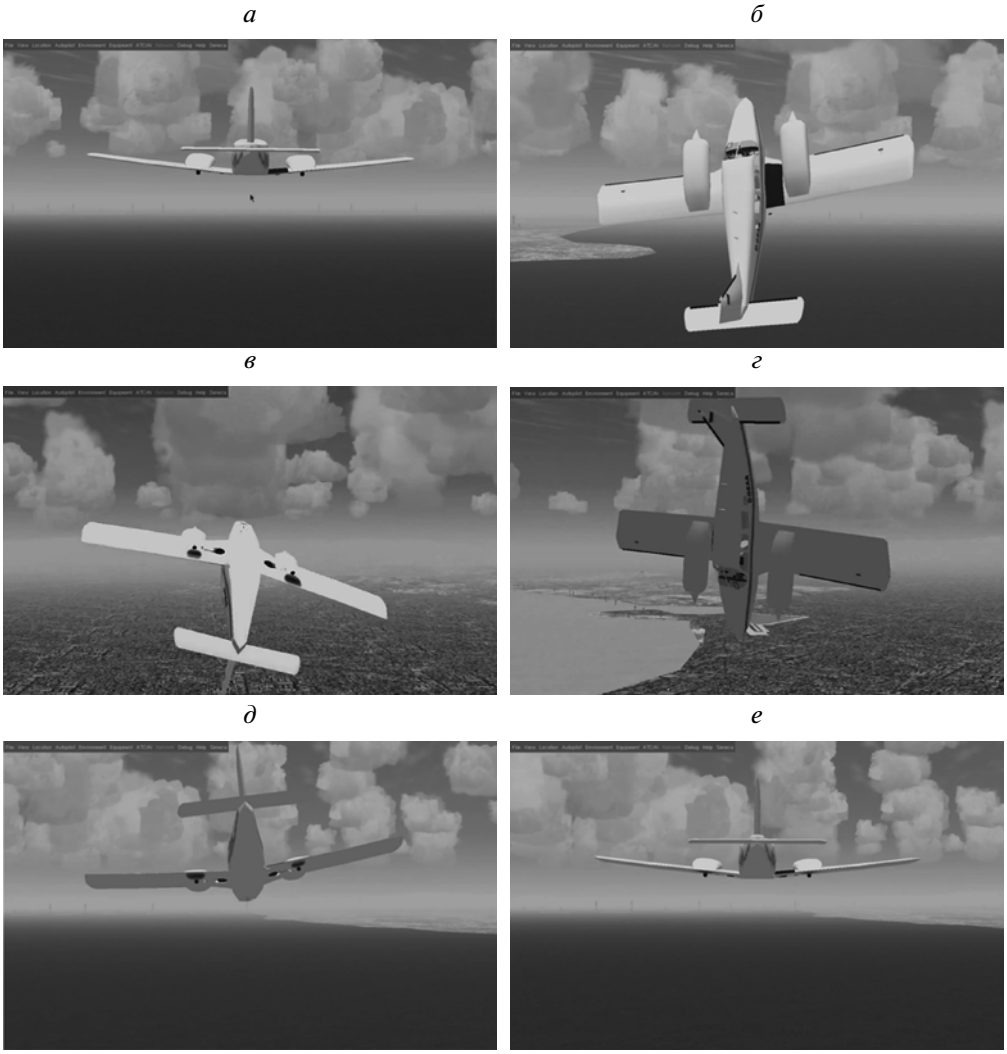

Рис. 7. Кадры видеозаписи выполнения “мертвой петли” под управлением лучшего автомата

3. Модель может вообще не выполнить “мертвую петлю”, так как автомат не справляется с управлением, что, правда, бывает крайне редко.

Определение условий, при которых выполняется каждый из этих вариантов полета, требует дальнейших исследований.

3.3. Пример полета самолета под управлением лучшего из полученных автоматов. Ниже приведены ссылки на видеозаписи трех вариантов реализации “мертвой петли” под управлением лучшего автомата: выполнение “мертвой петли”, близкое к “идеальному” (http://www.youtube.com/watch?v=TzrLoJjjVTA); выполнение “мертвой петли” с заваливанием на левый борт с последующим выравниванием (http://www.youtube.com/watch?v=C6WV7x2bqE8); последовательное выполнение двух “мертвых петель” (http://www.youtube.com/watch?v=yFiG4yz67Ks).

На рис. 7, *а*–*е* даны 6 кадров первой из этих видеозаписей.

В таблице дано описание переходов лучшего из сгенерированных автоматов для задачи выполнения “мертвой петли” (число состояний – 4, начальное состояние – 0, число дуг – 68).

Обратим внимание на то, что в используемом методе применяется полная таблица переходов. При этом для каждого состояния могут быть определены $2m$ переходов, где m – число предикатов. Для рассматриваемого примера число предикатов равно 13. Однако ни одно из четырех состояний не содержит их все. Отсутствие перехода в таблице свидетельствует о том, что на этом переходе автомат сохраняет свое состояние и параметры органов управления.

Перед осуществлением “мертвой петли” двигатель включен. Поэтому в ходе ее выполнения в использовании стартера нет необходимости – столбец “стартер” содержит только нули. Этот орган управления не был исключен при генерации автоматов, так как возможны дополнительные тесты, в которых требуется переключение стартера (например, если двигатель может заглохнуть). Последний столбец также оказался нулевым, но и соответствующий ему орган управления может быть использован при других тестах.

Заключение. Разработан метод эволюционного программирования на основе обучающих примеров для построения конечных автоматов, управляющих объектом со сложным поведением. Выполнено экспериментальное исследование предложенного метода и показано, что автоматически сгенерированный автомат может обеспечивать лучшее управления, чем ручное. В большинстве случаев автоматически сгенерированный автомат обеспечивает корректное выполнение задания. Показано, что предложенный метод позволяет значительно быстрее оценивать генерируемые автоматы по сравнению с прототипом [14]. Предложенный метод был применен к задаче управления моделью беспилотного самолета в режиме “мертвая петля”. Кроме более высокого быстродействия метода еще одно его преимущество по сравнению с вычислением функции приспособленности с помощью моделирования состоит в том, что в предложенном методе эту функцию не требуется изменять как при переходе от одного режима к другому, так и при переходе от одного объекта к другому.

СПИСОК ЛИТЕРАТУРЫ

1. *Поликарпова Н.И., Шалыто А.А.* Автоматное программирование. СПб: Питер, 2011. http://is.ifmo.ru/books/_book.pdf.
2. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998.
3. *Angeline P., Pollack J.* Evolutionary Module Acquisition // Proc. Second Annual Conf. on Evolutionary Programming. Cambridge: MIT Press, 1993. P. 154–163. http://www.demo.cs.brandeis.edu/papers/ep93.pdf.
4. *Jefferson D., Collins R., Cooper C. et al.* The Genesys System: Evolution as a Theme in Artificial Life // Proc. Second Conf. on Artificial Life. MA: Addison-Wesley, 1992. P. 549–578. www.cs.ucla.edu/~dyer/Papers/AlifeTracker/Alife91Jefferson.html.
5. *Царев Ф.Н., Шалыто А.А.* Применение генетического программирования для генерации автомата в задаче об “Умном муравье” // Сб. 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.
6. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. Промежуточный отчет по III этапу “Экспериментальные исследования поставленных перед НИР задач”. СПб: СПбГУ ИТМО, 2008. http://is.ifmo.ru/genalg/_2007_03_report-genetic.pdf.
7. *Паращенко Д.А., Царев Ф.Н., Шалыто А.А.* Технология моделирования одного класса мультиагентных систем на основе автоматного программирования на примере игры “Соревнование летающих тарелок” // Проектная документация. СПб: СПбГУ ИТМО, 2006. http://is.ifmo.ru/unimod-projects/plates.
8. *Coates A., Abbeel P., Ng A.Y.* Learning for Control from Multiple Demonstrations // Proc. 25th Intern. Conf. on Machine Learning. Helsinki: 2008. P. 144–151.
9. *Гладков Л.А., Курейчик В.В., Курейчик В.М.* Генетические алгоритмы. М.: Физматлит, 2006.
10. *Рассел С., Норвиг П.* Искусственный интеллект: современный подход. М.: Вильямс, 2006.
11. *Koza J.R.* Genetic Programming: on the Programming of Computers by Means of Natural Selection. Cambridge: MIT Press, 1992.
12. *Курейчик В.М.* Генетические алгоритмы. Состояние. Проблемы. Перспективы // Изв. РАН. ТиСУ. 1999. № 1. С. 144–160.
13. *Курейчик В.М., Родзин С.И.* Эволюционные алгоритмы: генетическое программирование // Изв. РАН. ТиСУ. 2002. № 1. С. 127–137.
14. *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. № 2. С. 100–117.
15. *Царев Ф.Н.* Метод построения управляющих конечных автоматов на основе тестовых примеров с помощью генетического программирования // Информационно-управляющие системы. 2010. № 5. С. 31–36.