Опубликовано в Научно-техническом вестнике СПбГУ ИТМО. 2008. Выпуск 53. Автоматное программирование, с. 24-42.

УДК 004.4'242

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ ГЕНЕРАЦИИ АВТОМАТОВ С БОЛЬШИМ ЧИСЛОМ ВХОДНЫХ ПЕРЕМЕННЫХ

Н.И. Поликарпова, В.Н. Точилин, А.А. Шалыто

(Санкт-Петербургский государственный университет информационных технологий, механики и оптики)

Эффективность известных методов автоматической генерации конечных автоматов на основе генетического программирования экспоненциально снижается с ростом числа входных воздействий автомата. В работе предложен метод, который не имеет указанного недостатка. Предпочтительность применения предложенного метода при большом числе входных воздействий обоснована теоретически. Метод был применен в инструментальном средстве для автоматизации разработки системы управления самолетом на высоком уровне абстракции.

Ключевые слова: генетический алгоритм, автоматное программирование

Введение

В литературе описывается ряд методов построения конечных автоматов с помощью генетических алгоритмов, генетического программирования и других эволюционных подходов. Большинство работ в этой области посвящено построению автоматов-*распознавателей*, описывающих грамматику некоторого языка. Задача такого автомата состоит в определении принадлежности заданной строки языку. Распознаватель не производит выходных воздействий, результат определяется состоянием автомата после обработки входной последовательности. В данном направлении как наиболее значимые можно выделить работы [1–8]. Более сложная форма конечного автомата – *преобразователь* – отображает множество входных строк на множество выходных, возможно над другим алфавитом. Примером преобразователя может служить компилятор, а примером распознавателя – синтаксический анализатор, определяющий соответствие кода программы грамматике языка программирования. Эволюционному построению преобразователей посвящена работа [9].

В распознающих и преобразующих автоматах все условия переходов имеют вид сравнения входного символа с заданным. Таким образом, подмножество входных воздействий, удовлетворяющее условию конкретного перехода, всегда состоит из единственного входного символа.

В области генетического программирования традиционно используются модели вычислений в виде графов, которые можно интерпретировать как графы переходов конечных автоматов. Построению программ в виде графов посвящено большое число работ, например [10–14]. Применение автоматов в играх описано в работах [15–17].

Небольшое число работ затрагивают вопросы построения *управляющих автоматов* – автоматов, описывающих логику сложного поведения сущности или системы. Например, в статьях [18–20] рассматривается автоматическое построение компонент программного обеспечения логических контроллеров в виде автоматов, а в работе [21] оценивается эффективность автоматных моделей применительно к различным задачам.

Практически во всех рассмотренных исследованиях автомат в каждый момент времени обрабатывает только одну входную переменную. Исключения составляют лишь работы [16] (четыре параллельных троичных входа) и [21] (в качестве условия перехода допускается сравнение значений двух регистров). Теоретически любое число параллельных входов сводится к одному, в качестве воздействий которого выступают

комбинации сигналов исходных параллельных входов. Однако размер алфавита полученного таким образом входа растет экспоненциально с увеличением числа исходных параллельных входов. В упомянутых работах параллельные входы не приводят к недопустимо большому алфавиту, но для реальных систем эта проблема крайне актуальна.

Также в рассмотренных работах автомат на каждом шаге может выполнить не более одного действия из заданного множества. В таком случае любая комбинация действий, которые может потребоваться выполнить одновременно, также должна считаться элементарным действием. При этом требуется априорная информация обо всех возможных комбинациях действий либо задание вместе с элементарными действиями всех их наборов, что приводит к экспоненциальному росту числа действий. Отметим, что в работе [21] допускаются действия с аргументами, а также параллельно выполняемые автоматы, отвечающие за различные действия, что значительно ослабляет проблему.

В настоящей работе рассматривается задача автоматического построения на основе генетического программирования управляющих автоматов, графы переходов которых в общем случае помечаются булевыми формулами. Из всех перечисленных работ наилучшие результаты по этой тематике получены в статьях [19, 20], посвященных созданию управляющей программы робота. Однако и эти работы не лишены упомянутых выше недостатков, относящихся к входным и выходным воздействиям. Насколько известно авторам, исследования в области эволюционного построения логики систем со сложным поведением, отличных от роботов, ранее не проводились.

При разработке предлагаемого авторами метода сокращенных таблиц наибольшее внимание уделялось специфике управляющих автоматов – возможности построения автоматов с произвольным числом параллельных входов и выходов. Кроме того, для сокращения пространства поиска метод использует *концепцию автоматизированного объекта управления* [22]: логика сложного поведения описывается автоматом или системой автоматов на высоком уровне абстракции, а объект управления порождает входные и выходные данные и оптимизации не подвергается. Объект управления может быть произвольным (и, вообще говоря, сколь угодно сложным). Таким образом, предлагаемый метод решает задачу об использовании сложных структур данных в рамках генетического программирования, поставленную основателем генетического программирования *J. Koza* в работе [23].

Постановка задачи

Сформулируем решаемую в настоящей работе *задачу построения управляющего автомата*. Пусть задан объект управления $O = \langle V, v_0, X, Z \rangle$, где V – множество вычислительных состояний (или *значений*), v_0 – *начальное значение*, $X = \{x_i : V \to \{0,1\}\}_{i=1}^{n}$ – множество предикатов, $Z = \{z_i : V \to V\}_{i=1}^{m}$ – множество действий. Также задана оценочная функция $\varphi : V \to \mathbf{R}^+$ и натуральное число k.

Объект O может управляться автоматом вида $A = \langle S, s_0, \Delta \rangle$, где S – конечное множество управляющих состояний, s_0 – *стартовое состояние*, $\Delta : S \times \{0,1\}^n \to S \times Z^*$ – *управляющая функция*. Управляющую функцию можно разложить на две компоненты: функцию выходов $\zeta : S \times \{0,1\}^n \to Z^*$ и функцию переходов $\delta : S \times \{0,1\}^n \to S$.

Пусть перед началом работы объекту управления соответствует начальное значение v_0, а автомат находится в стартовом состоянии s_0. Назовем *шагом работы* автоматизированного объекта следующую последовательность операций.

1. Объект управления вызывает все предикаты из множества X и формирует из их значений вектор *входного воздействия* $in \in \{0,1\}^n$.
2. Автомат вычисляет значение вектора *выходного воздействия* $out = \zeta(s, in)$, где s – текущее состояние автомата, и переходит в новое управляющее состояние $s_{new} = \delta(s, in)$.
3. Объект управления по очереди вызывает действия $z \in out$, изменяя при этом текущее вычислительное состояние [22].

Задача построения управляющего автомата состоит в том, чтобы найти автомат заданного вида такой, что за k шагов работы под управлением этого автомата объект O перейдет в вычислительное состояние с максимальной пригодностью ($\varphi(v) \to \max$).

В связи с использованием генетического программирования для этой задачи возникают следующие подзадачи: выбор представления конечного автомата в виде особи; адаптация генетических операторов (мутации и скрещивания) для выбранного представления; настройка параметров генетической оптимизации.

В классической интерпретации генетического алгоритма особь представляется в виде набора хромосом. Управляющий автомат можно представить как набор состояний, в каждом из которых его поведение определяется сужением управляющей функции $\Delta_s : \{0,1\}^n \to S \times Z^*$, $s \in S$. Таким образом, удобно сопоставить каждому состоянию хромосому. Следовательно, от задачи представления автомата в виде особи перейдем к более конкретной постановке проблемы – представлению управляющего состояния автомата в виде хромосомы.

Представление в виде хромосомы

Основная проблема, возникающая при использовании представления состояния автомата в виде таблицы переходов и действий – это экспоненциальный рост размерности хромосомы с увеличением числа предикатов объекта управления (напомним, что число строк в таблице 2^n, где n – число предикатов). Опыт показывает, что в реальных задачах управляющие автоматы, построенные вручную, имеют гораздо меньше переходов. Причина этого состоит, видимо, 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 использовать для сокращения описания управляющего состояния разными способами. Авторами выбран один из подходов, при котором число значимых в состоянии предикатов ограничивается некоторой константой r. К стандартной таблице, задающей сужение управляющей функции на данное состояние, в этом случае добавляется битовый вектор, описывающий множество значимых предикатов (рис. 1). Число строк таблицы в этом случае 2^r. При этом 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.

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. 1. Хромосома состояния: сокращенная таблица (n = 6, r = 2).

Генетические операции

Опишем генетические операторы над хромосомами состояний, записанными в виде сокращенных таблиц.

Алгоритм 1. Мутация сокращенных таблиц. По сравнению со стандартным представлением добавилась возможность мутации множества значимых предикатов. При этом каждый из значимых предикатов с некоторой вероятностью заменяется другим, который не принадлежит множеству (рис. 2).

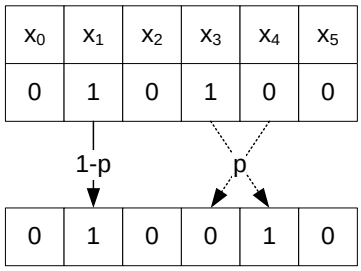


Рис. 2. Пример мутации множества значимых предикатов.

Мутация самой сокращенной таблицы происходит так же, как мутация полной таблицы. Описание алгоритма приведено в листинге 1 приложения.

Алгоритм 2. Скрещивание сокращенных таблиц. Это наиболее сложный из предлагаемых алгоритмов. Основная последовательность его шагов отражена в листинге 2 приложения.

Поскольку родительские хромосомы, представленные сокращенными таблицами, могут иметь разные множества значимых предикатов, сначала необходимо выбрать, какие из этих предикатов будут значимы для хромосом детей. Функция `ChoosePreds`, осуществляющая этот выбор, представлена в листинге 3 приложения. В результате работы функции `ChoosePreds` предикаты, значимые для обоих родителей, наследуются обоими детьми, а каждый из тех предикатов, которые были значимы лишь для одной родительской особи, равновероятно достанется любому из двух детей. Пример работы функции для родительских хромосом, представленных на рис. 3, проиллюстрирован на рис. 4.

После выбора значимых предикатов заполняются таблицы обоих детей. Алгоритм заполнения представлен в листинге 4 приложения.

Иллюстрация примера заполнения первой строки таблицы одного из детей приведена на рис. 5. В данной реализации оператора скрещивания на значения каждой строки таблицы ребенка влияют значения нескольких строк родительских таблиц. При этом конкретное значение, помещаемое в ячейку таблицы ребенка, определяется «голосованием» всех влияющих на нее ячеек родительских таблиц.

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

x_0	x_1	x_2	x_3	x_4	x_5
1	1	0	0	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

x_0	x_1	s	z_0	z_1	z_2
0	0	1	1	1	0
0	1	2	0	0	1
1	0	2	0	1	0
1	1	0	0	1	0

Рис. 3. Родительские хромосомы, представленные сокращенными таблицами.

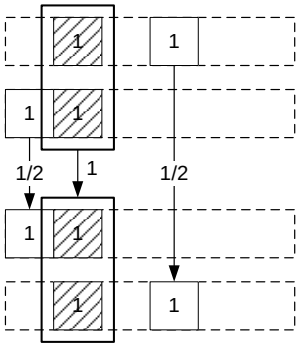


Рис. 4. Пример выбора значимых предикатов детей.

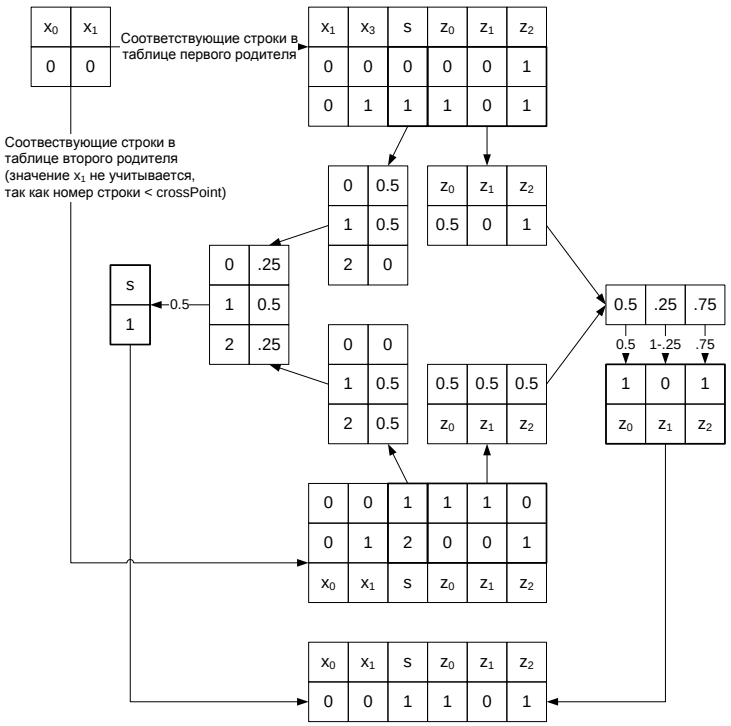


Рис. 5. Пример заполнения строки таблицы ребенка при скрещивании сокращенных таблиц

В описанном выше варианте алгоритма все состояния автомата имеют равное число значимых предикатов (r – константа для всего процесса оптимизации). Однако

предложенный алгоритм скрещивания легко расширяется на случай разного числа значимых предикатов у пары родителей.

Эффективность метода сокращенных таблиц

Для оценки характеристик метода сокращенных таблиц был проведен ряд экспериментов, в которых эффективность данного метода сравнивалась с эффективностью метода полных таблиц [24]. Сравнение проводилось по следующим критериям: объем занимаемой памяти, время, затрачиваемое на обработку каждого поколения, скорость роста функции пригодности (от номера поколения и от времени).

Как упоминалось выше, основное достоинство метода сокращенных таблиц состоит в том, что он решает проблему экспоненциального роста размера описания автомата с увеличением числа входных переменных (предикатов объекта управления). Это свойство метода подтвердилось при экспериментальной проверке. Во время эксперимента описания автоматов хранились в памяти компьютера. Поэтому объем занимаемой памяти в этом случае прямо пропорционален размеру описания автомата. Результаты эксперимента приведены на рис. 6, a.

Из рассмотрения графика следует, что объем занимаемой памяти при использовании метода полных таблиц растет экспоненциально, в то время как при использовании метода сокращенных таблиц объем занимаемой памяти с ростом числа предикатов практически не изменяется. В действительности он зависит от числа предикатов линейно, однако коэффициент линейной зависимости настолько мал, что в экспериментах она не отражается. Аналогичный характер имеет зависимость времени, требуемого на обработку каждого поколения, от числа предикатов (рис. 6, b).

Теперь перейдем к оценке скорости роста функции пригодности. На рис. 6, c, приведены зависимости значения оценочной функции от номера поколения при использовании метода полных таблиц и метода сокращенных таблиц (измерения приводились при небольшом числе предикатов). Из последнего графика следует, что оптимизация методом полных таблиц требует вычисления меньшего числа поколений. Это означает, что при небольшом числе предикатов метод полных таблиц обладает более высоким быстродействием. Однако с ростом числа предикатов стремительно растет время обработки одного поколения методом полных таблиц. Кроме того, из-за экспоненциального роста объема требуемой памяти применение метода полных таблиц, начиная с некоторого числа предикатов, становится не просто неэффективным, а практически невозможным. В экспериментах авторам не удалось построить методом полных таблиц автомат с более чем 14 входными переменными. Отметим также, что автоматы, которые построены методом полных таблиц, практически невозможно изобразить и понять, так как в них присутствует большое число избыточных переходов, а условия на переходах громоздки. Напротив, автоматы, построенные методом сокращенных таблиц, могут быть сравнительно легко поняты человеком.

Чтобы адекватно оценить быстродействие обоих методов, необходимо установить зависимость значения оценочной функции, достигаемого с помощью каждого метода за определенное время, от числа предикатов. Такая зависимость для промежутка времени, равного пяти минутам, приведена на рис. 6, d. Как и ожидалось, приведенные зависимости показывают, что при небольшом числе предикатов метод полных таблиц имеет более высокое быстродействие, однако с ростом числа предикатов его быстродействие резко падает. В то же время быстродействие метода сокращенных таблиц с ростом числа предикатов уменьшается незначительно.

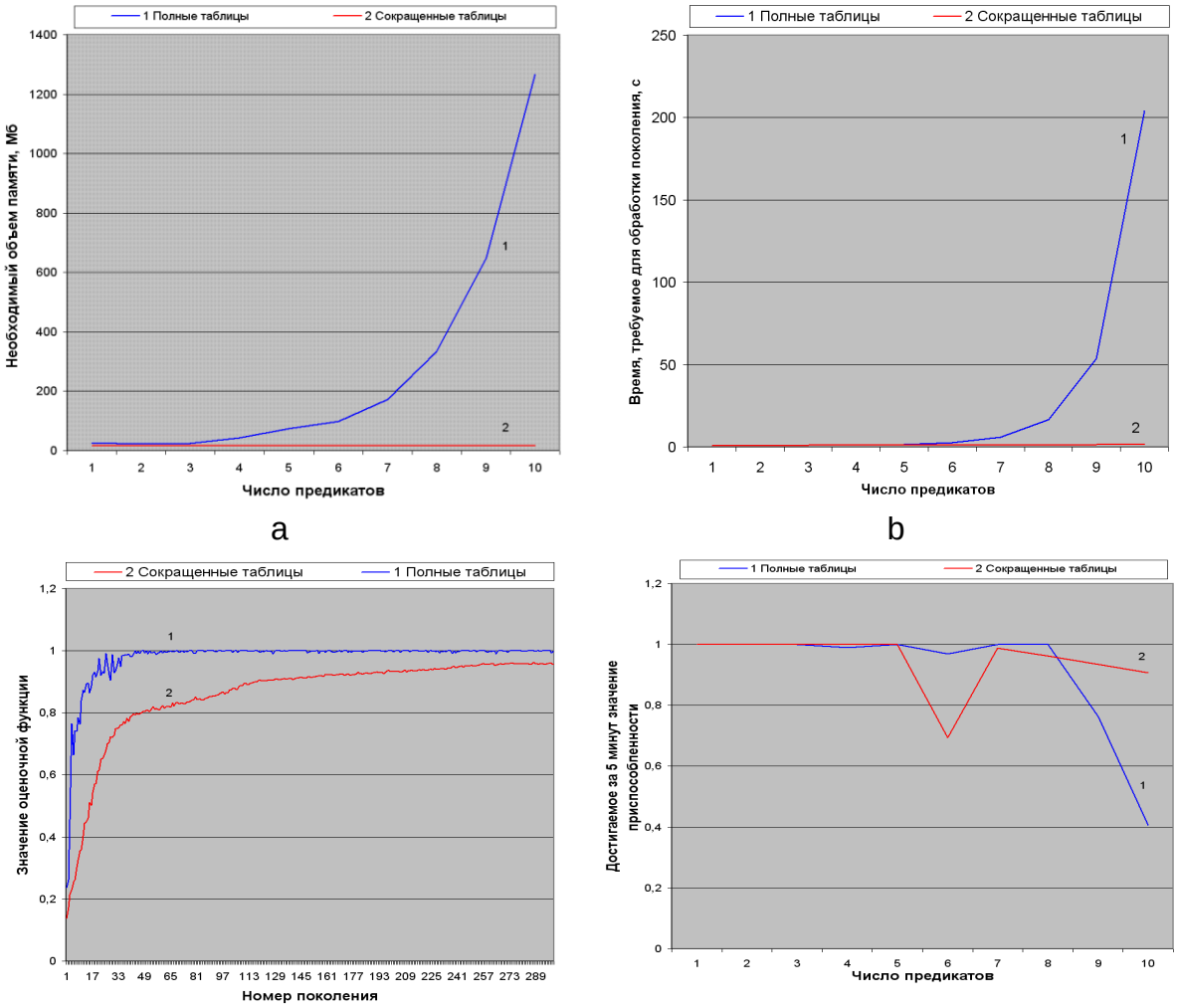


Рис. 6. Зависимости объема занимаемой памяти (a), времени обработки поколения (b), значения оценочной функции для заданной продолжительности работы метода (d) от числа предикатов и значения оценочной функции от номера поколения (c)

.

Из приведенного исследования характеристик методов полных и сокращенных таблиц можно сделать следующие выводы:

- при небольшом числе предикатов метод полных таблиц является более эффективным по времени и достаточно эффективным по памяти, однако построенные этим методом автоматы непонятны человеку;
- начиная с некоторого числа предикатов применение метода полных таблиц практически невозможно, в то время как метод сокращенных таблиц остается достаточно эффективным и по времени и по памяти.

Для генерации автоматов на основе предложенного метода авторами было разработано инструментальное средство, которое опубликовано на сайте http://is.ifmo.ru в разделе «Работы».

Применение метода сокращенных таблиц для автоматического управления самолетом

Предложенный метод был применен для разработки системы автоматического управления моделью самолета на высоком уровне абстракции. Эта система должна быть реализована конечным автоматом. При этом предполагается, что на более низком

уровне абстракции используется автопилот, который управляет самолетом в каждом состоянии конечного автомата. Этот автопилот в настоящей работе не строится, а считается заданным. Разработанная система должна обеспечить выполнение полета без участия пилота и не должна была предъявлять нестандартных требований к наземному оборудованию.

Ниже представлена последовательность действий, выполняемых при применении метода сокращенных таблиц для автоматического управления самолетом.

1. Разработка (выбор) эмулятора самолета.
2. Построение преобразователя интерфейсов.
    1. Выбор и реализация входных воздействий.
    2. Выбор и реализация выходных воздействий.
3. Выбор функции приспособленности.
4. Генерация автомата.
    1. Настройка эмулятора.
    2. Порождение популяции псевдослучайных автоматов.
    3. Взаимодействие автоматов с эмулятором через преобразователь интерфейсов.
    4. Оценка каждого автомата с помощью функции приспособленности.
    5. Использование генетических операций для генерации автоматов с вероятно более высоким значением функции приспособленности.
    6. Если максимальная полученная оценка функции приспособленности меньше целевого значения, то переход к пункту 4.4.
    7. Запись автомата с максимальной оценкой функции приспособленности.
    8. Ручное упрощение графа переходов для повышения «понятности» автомата.
5. Проведение эксперимента по управлению моделью самолета сгенерированным автоматом.
    1. Выполнение написанных вручную блоков прокладки маршрута и настройки навигации.
    2. Запуск системы автомат – преобразователь интерфейсов – эмулятор.
    3. Формирование результатов моделирования.
    4. Анализ результатов моделирования.

Разработка (выбор) эмулятора самолета

В рассматриваемой задаче объектом управления является самолет, находящийся в некоторой среде, включающей воздух, возможно движущийся, взлетно-посадочные полосы, ландшафт, службу управления полетами, радиомаяки и многое другое. Требуется эмулировать аэродинамику, механику и работу оборудования самолета. Ввиду трудоемкости разработки необходимого эмулятора представляется естественным использование эмулятора стороннего производителя. Для этой цели был выбран авиационный симулятор *X-Plane* [25] . Данный симулятор представляет собой игру для персонального компьютера, обладающую достаточной точностью физической эмуляции, высокой скоростью работы, возможностью взаимодействия с другими программами и недорогой лицензией, не препятствующей ее использованию в данном проекте.

Выбор и реализация входных воздействий

Выбранные входные воздействия приведены в табл. 1.

Таблица 1. Входные воздействия автомата

Идентификатор	Описание значения
x1	Самолет движется
x2	Скорость самолета достаточна для полета с выпущенными за-

	крылками
x3	Скорость самолета достаточна для полета с убранными закрылками
x4	Самолет летит
x5	Самолет находится рядом с поверхностью земли
x6	Высота полета соответствует рекомендованной высоте эшелона
x7	Самолет находится рядом с опорной точкой GPS-приемника

Выбор и реализация выходных воздействий

Выбранный набор действий приведен в табл. 2.

Таблица 2. Выходные воздействия автомата

Идентификатор	Описание действия
z1	Настройка навигационного приемника на частоту посадочного маяка аэропорта отправления
z2	Настройка навигационного приемника на частоту посадочного маяка аэропорта прибытия
z3	Перевод GPS-приемника в режим следования к опорной точке
z4	Установка переключателя управления полетом в положение “АВТО”
z5	Переключение автопилота в режим полета на точку в горизонтальной плоскости
z6	Переключение автопилота в режим полета по курсу в горизонтальной плоскости
z7	Переключение автопилота в режим снижения по маяку
z8	Переключение автопилота в режим постоянной высоты
z9	Переключение автопилота в режим постоянной вертикальной скорости
z10	Включение колесного тормоза
z11	Выключение колесного тормоза
z12	Установка максимальной подачи топлива
z13	Установка средней подачи топлива
z14	Установка минимальной подачи топлива
z15	Убирание закрылков
z16	Установка закрылков во взлетное положение
z17	Установка закрылков в посадочное положение
z18	Выпуск шасси
z19	Убирание шасси
z20	Использование навигационного приемника в качестве источника сигнала индикатора горизонтальной ситуации
z21	Использование GPS-приемника в качестве источника сигнала индикатора горизонтальной ситуации
z22	Установка скорости набора высоты и высоты полета на автопилоте
z23	Управление рулем направления в соответствии с сигналами автопилота
z24	Установка руля направления в центральное положение

Соответствующие входным переменным приборы и соответствующие действиям органы управления эмулятором самолета показаны на рис. 7. Неотмеченные приборы не используются.

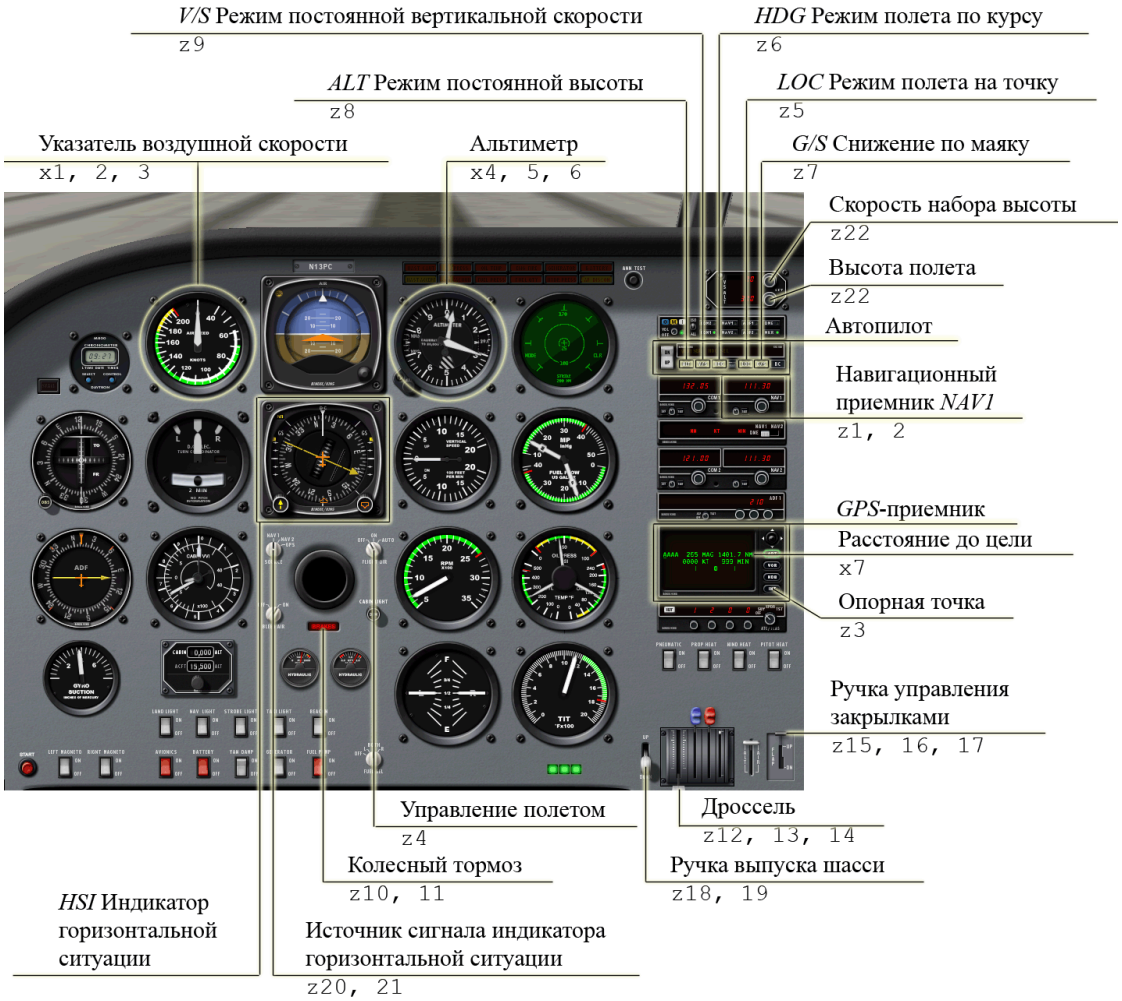


Рис. 7. Используемые приборы и органы управления самолетом.

Функция приспособленности

Переформулируем задачу в виде функции приспособленности. Выберем существенные характеристики, которые будут влиять на значение функции приспособленности и использоваться при ее вычислении. От автопилота требуется провести самолет по маршруту и не разбить его. Следовательно, можно выделить два значимых фактора: отклонение от маршрута и сохранность самолета. Кроме того, важно, чтобы после прохождения маршрута самолет остановился (или снизил скорость до безопасной) на взлетно-посадочной полосе. Совокупное отклонение от маршрута как по положению, так и по скорости будем вычислять в виде интегралов модулей соответствующих отклонений по времени:

$$P = \int_{t_0}^{t_1} |p(t)| * dt,$$

где P – совокупное отклонение от маршрута по положению, t_0 – время начала эмуляции, t_1 – время окончания эмуляции, $p(t)$ – отклонение по положению в момент времени t;

$$V = \int_{t_0}^{t_1} |v(t)| * dt ,$$

где V – совокупное отклонение от оптимальной скорости, а $v(t)$ – отклонение от оптимальной скорости в момент времени t. Требуется оптимизировать систему управления по трем параметрам. Для того чтобы свести многокритериальную оптимизацию к однокритериальной, выберем функцию приспособленности вида

$$f = \frac{t_1 - t_0}{\left(100 + \frac{P+V}{t_1 - t_0}\right) * 0{,}77 * (1 + b * 9)},$$

где b – переменная, равная нулю при целом самолете и единице при разбитом.

Построение управляющего автомата

После описания всех необходимых элементов программы можно проверить ее работоспособность. Основными показателями являются динамика значений функции приспособленности в процессе работы программы и качество автомата, получаемого после завершения работы. График изменения оценки приспособленности с ростом номера поколения автоматов приведен на рис. 8. Единичное значение функции приспособленности на двух последних поколениях свидетельствует об успешном завершении процесса оптимизации.

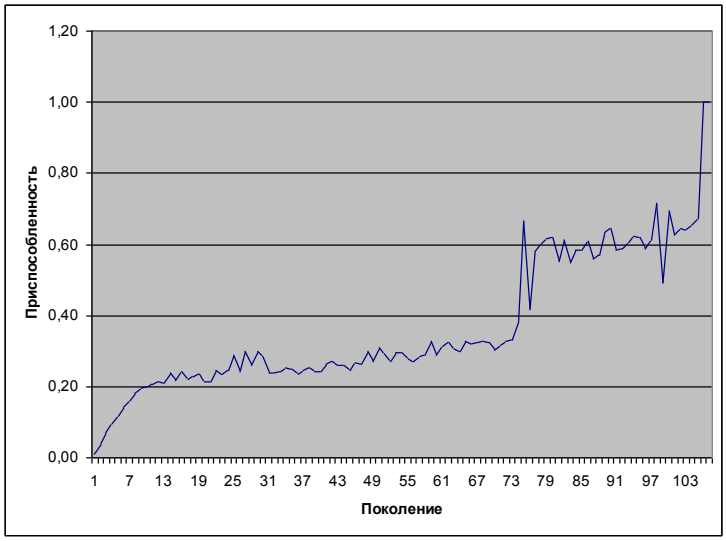


Рис. 8. Изменение приспособленности в процессе эволюции

При генерации автомата было наложено ограничение на число переменных `n`, проверяемых на переходах из состояний. Даже при `n` = 1 четырехпроцессорный кластер строил автомат около месяца (значение функции приспособленности вычислялось около пяти минут). При этом построение автомата, корректно управляющего самолетом, при принятом ограничении не гарантировалось.

Полученный в результате работы автомат, несмотря на небольшое число состояний и переходов, достаточно сложен для восприятия человеком ввиду большого числа вызываемых на переходах действий. Для упрощения автомата было произведено формальное, но не изменяющее поведение автомата, сокращение числа вызываемых на переходах действий. Кроме того, состояния были перенумерованы и снабжены названиями. Полученный автомат приведен на рис. 9.

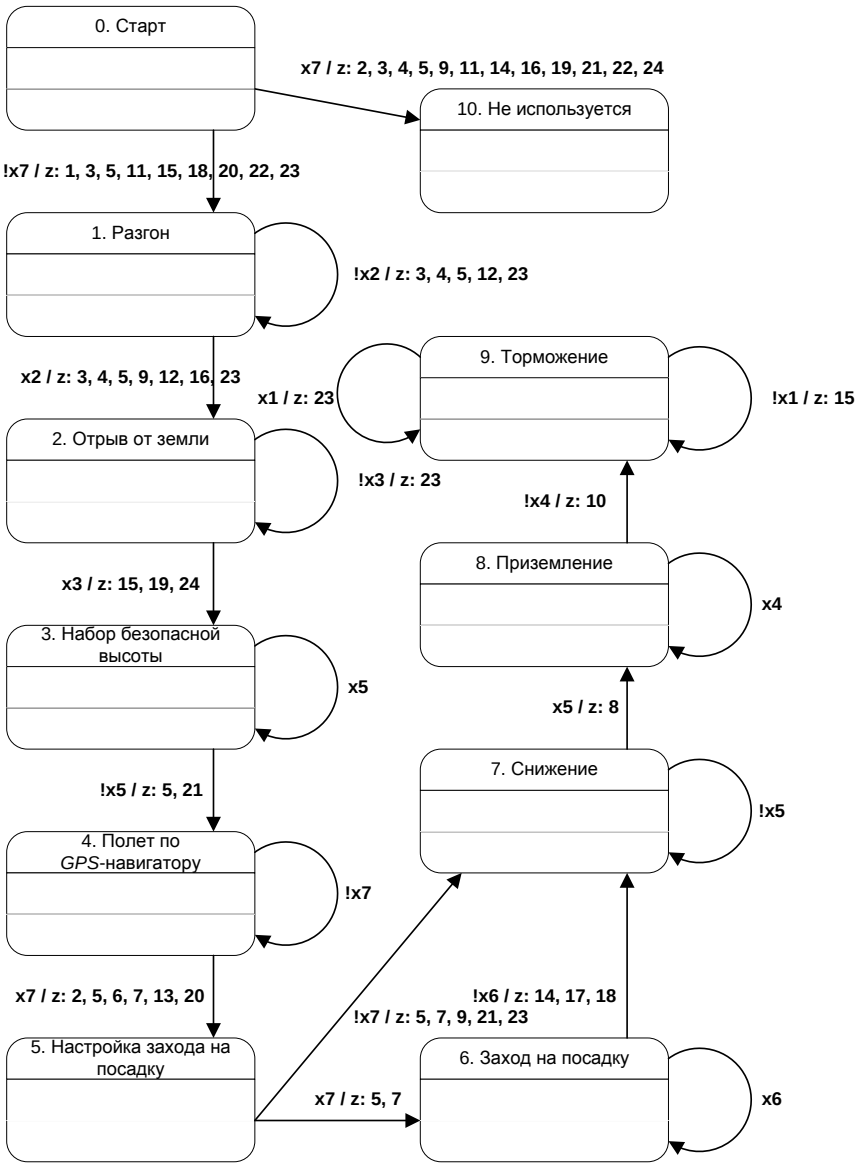
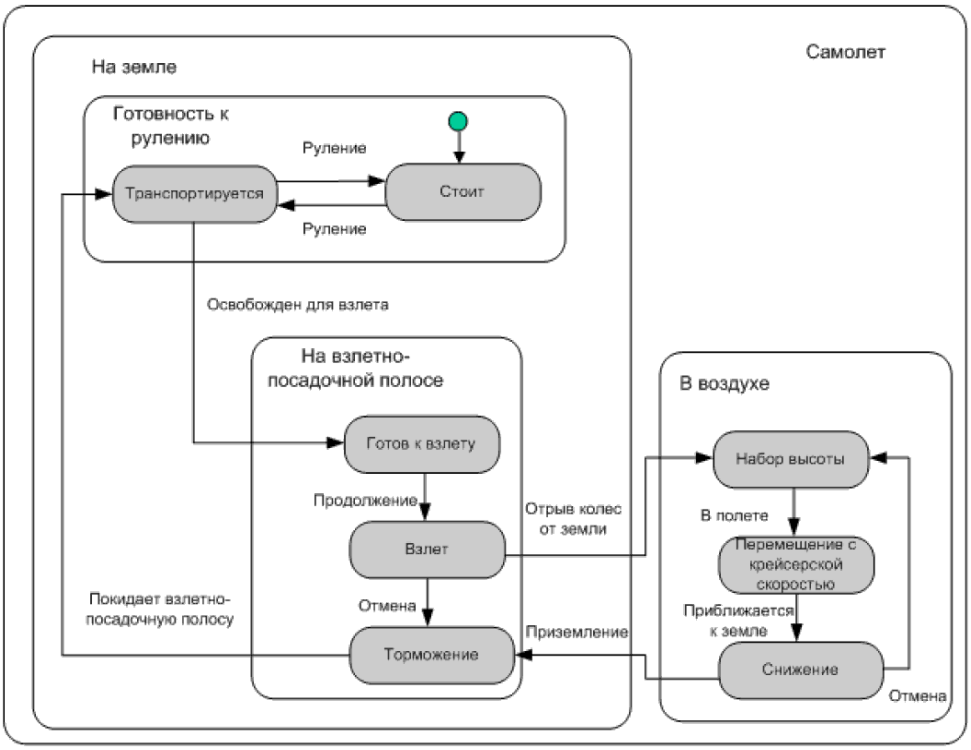


Рис. 9. Граф переходов полученного автомата

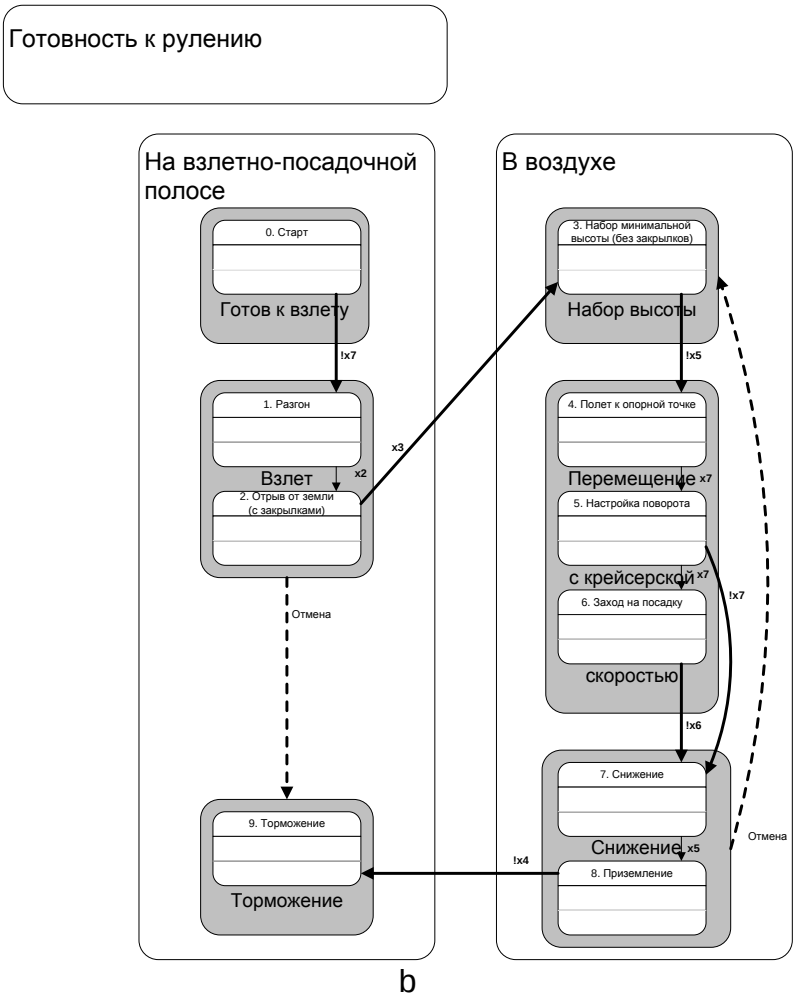
Простота структуры полученного управляющего автомата объясняется использованной конфигурацией метода сокращенных таблиц. Число анализируемых в каждом состоянии входных воздействий, как отмечалось в предыдущем разделе, было выбрано равным единице. Следовательно, из каждого состояния могло быть не более двух переходов. В рассматриваемом случае выразительной силы практически линейной структуры автомата было достаточно в результате двух факторов:

- высокоуровневые входные и выходные воздействия автомата;
- ограниченные требования к функциональности системы управления.

Изображенный на рис. 9 автомат может быть сравнительно легко построен эвристически (без использования генетического программирования). Однако в общем случае сложность результата не всегда находится в прямой зависимости от сложности процесса его получения. Построенный эвристически автомат часто оказывается неоправданно сложным по сравнению с полученным автоматически. Кроме того, подход на основе автоматической генерации характеризуется лучшей масштабируемостью.



a



b

Рис. 10. Диаграмма состояний для полета самолета из книги [27] (a) и ее сопоставление с автоматически построенным автоматом (b)

Сравним полученный автомат с автоматом, построенным эвристически. В книге [27] в качестве примера диаграммы состояний приведен граф переходов, описывающий полет самолета (рис. 10, a). На рис. 10, b, состояниям и переходам этого автомата сопоставляются состояния и переходы автоматически построенного автомата с рис. 9. Автоматически построенный автомат отличается отсутствием состояния готовности к рулению и связанных с ним переходов, а также отсутствием переходов отмены между состояниями «Взлет» и «Торможение» и между состояниями «Снижение» и «Набор высоты». Отсутствующие элементы выделены на рисунке пунктиром. Различия между автоматами объясняются отсутствием этапа руления и возможности отмены взлета или снижения в процессе обучения.

Проведение эксперимента по управлению моделью самолета сгенерированным автоматом

Работоспособность автомата в качестве системы управления моделью самолета проверяетс0я с помощью того же инструментального средства, на котором был сгенерирован автомат. Работа инструментального средства в этом режиме аналогична режиму генерации автомата с единственным автоматом в поколении и без применения генетических модификаций. Автомат, положение самолета и показания приборов в каждом состоянии показаны на рис. 11.

1

2


10.
Не используется
x7 / z: 2, 3, 4, 5, 9, 11, 14, 16, 19, 21, 22, 24
0. Старт
!x7 / z: 1, 3, 5, 11, 15, 18, 20, 22, 23
1. Разгон
!x2 / z: 3, 4, 5, 9, 12, 16, 23
x2 / z: 3, 4, 5, 9, 12, 16, 23
2. Отрыв от земли
!x3 / z: 23
x3 / z: 15, 19, 24
3. Набор безопасной высоты
x5
!x5 / z: 5, 21
4. Полет по GPS-навигатору
!x7
x7 / z: 2, 5, 6, 7, 13, 20
5. Настройка захода на посадку
x7 / z: 5, 7
6. Заход на посадку
x6
!x6 / z: 14, 17, 18
7. Снижение
!x5
x5 / z: 8
8. Приземление
x4
!x4 / z: 10
x1 / z: 23
9. Торможение
!x1 / z: 15
!x7 / z: 5, 7, 9, 21, 23

3


10.
Не используется
x7 / z: 2, 3, 4, 5, 9, 11, 14, 16, 19, 21, 22, 24
0. Старт
!x7 / z: 1, 3, 5, 11, 15, 18, 20, 22, 23
1. Разгон
!x2 / z: 3, 4, 5, 9, 12, 16, 23
x2 / z: 3, 4, 5, 9, 12, 16, 23
2. Отрыв от земли
!x3 / z: 23
x3 / z: 15, 19, 24
3. Набор безопасной высоты
x5
!x5 / z: 5, 21
4. Полет по GPS-навигатору
!x7
x7 / z: 2, 5, 6, 7, 13, 20
5. Настройка захода на посадку
x7 / z: 5, 7
6. Заход на посадку
x6
!x6 / z: 14, 17, 18
7. Снижение
!x5
x5 / z: 8
8. Приземление
x4
!x4 / z: 10
x1 / z: 23
9. Торможение
!x1 / z: 15
!x7 / z: 5, 7, 9, 21, 23

4


10.
Не используется
x7 / z: 2, 3, 4, 5, 9, 11, 14, 16, 19, 21, 22, 24
0. Старт
!x7 / z: 1, 3, 5, 11, 15, 18, 20, 22, 23
1. Разгон
!x2 / z: 3, 4, 5, 9, 12, 16, 23
x2 / z: 3, 4, 5, 9, 12, 16, 23
2. Отрыв от земли
!x3 / z: 23
x3 / z: 15, 19, 24
3. Набор безопасной высоты
x5
!x5 / z: 5, 21
4. Полет по GPS-навигатору
!x7
x7 / z: 2, 5, 6, 7, 13, 20
5. Настройка захода на посадку
x7 / z: 5, 7
6. Заход на посадку
x6
!x6 / z: 14, 17, 18
7. Снижение
!x5
x5 / z: 8
8. Приземление
x4
!x4 / z: 10
x1 / z: 23
9. Торможение
!x1 / z: 15
!x7 / z: 5, 7, 9, 21, 23

5


10.
Не используется
x7 / z: 2, 3, 4, 5, 9, 11, 14, 16, 19, 21, 22, 24
0. Старт
!x7 / z: 1, 3, 5, 11, 15, 18, 20, 22, 23
1. Разгон
!x2 / z: 3, 4, 5, 9, 12, 16, 23
x2 / z: 3, 4, 5, 9, 12, 16, 23
2. Отрыв от земли
!x3 / z: 23
x3 / z: 15, 19, 24
3. Набор безопасной высоты
x5
!x5 / z: 5, 21
4. Полет по GPS-навигатору
!x7
x7 / z: 2, 5, 6, 7, 13, 20
5. Настройка захода на посадку
x7 / z: 5, 7
6. Заход на посадку
x6
!x6 / z: 14, 17, 18
7. Снижение
!x5
x5 / z: 8
8. Приземление
x4
!x4 / z: 10
x1 / z: 23
9. Торможение
!x1 / z: 15
!x7 / z: 5, 7, 9, 21, 23

6

7

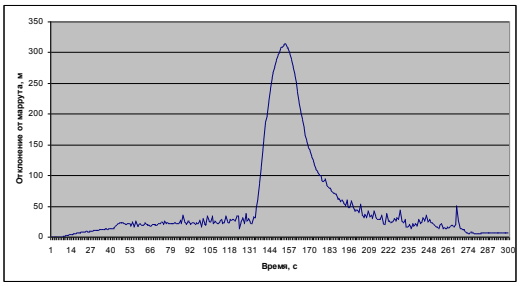
8

9

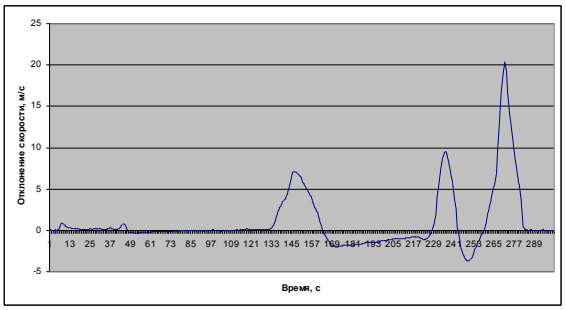
Рис. 11. Автомат и самолет в различных состояниях

Формирование результатов моделирования

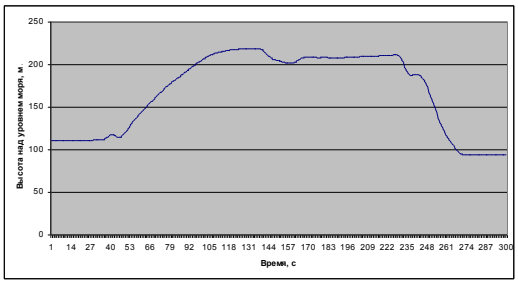
В процессе тестового использования сгенерированного автомата для управления самолетом фиксировались отклонения от маршрута и отклонения от рекомендованной скорости. Так как допустимые отклонения зависят от этапа полета, также фиксировалась высота полета, позволяющая определить моменты отрыва от земли при взлете и касания земли при приземлении. Кроме того, записывались действующие в процессе полета перегрузки, которые не учитывались при выводе автомата. Все записанные данные показаны на рис. 12.



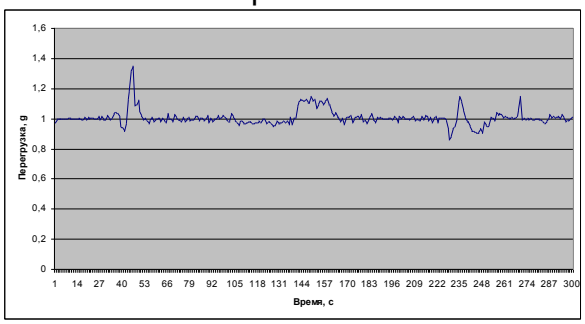
Отклонения от маршрута



Отклонения от рекомендованной скорости



Высота полета над уровнем моря



Перегрузки

Рис. 12. Результаты тестирования

Анализ результатов моделирования

Из графика следует, что максимальное отклонение самолета от маршрута за все время полета составило 314 м. При этом отклонение от маршрута не превышало 60 м на протяжении всего полета, кроме минутного участка в середине графика, когда отклонения наименее критичны. По графику высоты полета можно определить время отрыва и касания земли: 35 с и 271 с соответственно. Из установленного времени и графика отклонений от маршрута следует, что максимальное отклонение при разгоне составило 12 м, а при торможении – 10 м. Существенные, но не выходящие за пределы взлетно-посадочных полос отклонения позволяют считать, что сгенерированный автомат провел самолет по маршруту с достаточной точностью.

Из графика рекомендованной скорости следует, что скоростной режим соблюден, за исключением превышения скорости на 20,34 м/с в момент касания земли, которое было компенсировано при торможении и не привело к существенному превышению тормозного пути. Совокупное отклонение положения самолета после остановки, включающее поперечное отклонение от центра полосы и превышение тормозного пути, составляет 6,5 м. Из графика перегрузок следует, что ускоренное торможение не привело к существенному дискомфорту пассажиров. Проведенный анализ позволяет сделать заключение об удовлетворительном качестве автоматически построенного автомата.

Заключение

В работе была показана низкая эффективность существующих методов генерации автоматов с большим числом входных воздействий и предложен метод, который не имеет указанных недостатков. Изменение соотношения эффективностей известного и предложенного методов с ростом числа входных воздействий в пользу предложенного метода было обосновано теоретически и проверено на модельной задаче. Получение работоспособного и эффективного автомата в результате применения предложенного

метода для генерации системы управления самолетом показало его применимость для решения практических задач.

Литература

1. Gold E. M. Language Identification in the Limit // Information and Control. – 1967. – №10. – P.447–474.
2. Belz A. Computational Learning of Finite-State Models for Natural Language Processing. PhD thesis. – University of Sussex. 2000.
3. Clelland C. H., Newlands D. A. Pfsa modelling of behavioural sequences by evolutionary programming // Complex'94 – Second Australian Conference on Complex Systems. – IOS Press, 1994. – P.165–172.
4. Das S., Mozer M. C. A Unified Gradient-Descent/Clustering Architecture for Finite State Machine Induction. // Advances in Neural Information Processing Systems. – 1994.
5. Lankhorst M. M. A Genetic Algorithm for the Induction of Nondeterministic Pushdown Automata. // Computing Science Report. – University of Groningen Department of Computing Science. – 1995.
6. Belz A., Eskikaya B. A genetic algorithm for finite state automata induction with an application to phonotactics // ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing. – Saarbruecken. – 1998. – P. 9–17.
7. Ashlock D., Wittrock A., Wen T-J. Training finite state machines to improve PCR primer design // Congress on Evolutionary Computation (CEC'02). – 2002. – P.13–18.
8. Ashlock D. A., Emrich S. J., Bryden K. M. et al. A comparison of evolved finite state classifiers and interpolated markov models for improving PCR primer design // 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'04). – 2004. – P.190–197.
9. Lucas S. M. Evolving Finite State Transducers: Some Initial Explorations // Genetic Programming: 6th European Conference (EuroGP'03). – Berlin: Springer, 2003. – P.130–141.
10. Teller A., Veloso M. PADO: A New Learning Architecture for Object Recognition. Symbolic Visual Learning. – New York: Oxford University Press. 1996. P. 81–116.
11. Banzhaf W., Nordin P., Keller R. E., Francone F. D. Genetic Programming – An Introduction. // On the automatic Evolution of Computer Programs and its Application. – San Francisco: Morgan Kaufmann Publishers, 1998.
12. Kantschik W., Dittrich P., Brameier M. Empirical Analysis of Different Levels of Meta-Evolution // Congress on Evolutionary Computation. – 1999.
13. Kantschik W., Dittrich P., Brameier M., Banzhaf W. Meta-Evolution in Graph GP // Genetic Programming: Second European Workshop (EuroGP'99). – 1999.
14. Teller A., Veloso M. Internal Reinforcement in a Connectionist Genetic Programming Approach // Artificial Intelligence. North-Holland Pub. Co. – 1970. – P.161.
15. Miller J. H. The Coevolution of Automata in the Repeated Prisoner's Dilemma. // Working Paper. Santa Fe Institute. – 1989.
16. Spears W. M., Gordon D. F. Evolving Finite-State Machine Strategies for Protecting Resources // International Syposium on Methodologies for Intelligent Systems. – 2000.
17. Ashlock D. Evolutionary Computation for Modeling and Optimization. – New York: Springer. 2006.
18. Frey C., Leugering G. Evolving Strategies for Global Optimization. A Finite State Machine Approach // Genetic and Evolutionary Computation Conference (GECCO-2001). – Morgan Kaufmann. – 2001. – P. 27–33.
19. Petrovic P. Simulated evolution of distributed FSA behaviour-based arbitration // The Eighth Scandinavian Conference on Artificial Intelligence (SCAI'03). – 2003.

20. Petrovic P. Evolving automatons for distributed behavior arbitration. // Technical Report. – Norwegian University of Science and Technology. – 2005.
21. Petrovic P. Comparing Finite-State Automata Representation with GP-trees. // Technical report. – Norwegian University of Science and Technology. – 2006.
22. Поликарпова Н.И., Шалыто А.А. Учебно-методическое пособие по дисциплине «Автоматное программирование». – СПб: СПбГУ ИТМО, 2007. – Режим доступа: http://is.ifmo.ru/books/_umk.pdf
23. Koza J. R. Future Work and Practical Applications of Genetic Programming. Handbook of Evolutionary Computation. – Bristol: IOP Publishing Ltd. 1997.
24. Поликарпова Н.И., Точилин В.Н., Шалыто А.А. Применение генетического программирования для реализации систем со сложным поведением // Сборник трудов IV-ой Международной научно-практической конференции «Интегрированные модели и мягкие вычисления в искусственном интеллекте». – Том 2. – М.: Физматлит, 2007. – С. 598–604. – Режим доступа: http://is.ifmo.ru/genalg/_polikarpova.pdf
25. Сайт X-Plane by Laminar Research – Режим доступа: http://www.x-plane.com/
26. Koza J.R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. – The MIT Press. 1992.
27. Халл Э., Джексон К., Дик Д. Разработка и управление требованиями. Практическое руководство пользователя. – 2005.
28. Халл Э., Джексон К., Дик Д. Разработка и управление требованиями. – Режим доступа: http://download.telelogic.com/download/article/eBook_RU_Requirements_Engineering.pdf
29. Поликарпова Н.И., Точилин В.Н., Шалыто А.А. Разработка библиотеки для генерации управляющих автоматов методом генетического программирования // Сборник докладов X Международной конференции по мягким вычислениям и измерениям. – Том 2. – СПбГЭТУ «ЛЭТИ». – 2007. – С. 84–87. – Режим доступа: http://is.ifmo.ru/download/polikarpova(LETI).pdf