SCM'2010

International Conference
on Soft Computing and Measurements

Международная конференция по
мягким вычислениям и измерениям

Proceedings

Сборник докладов

of conference

Volume 1
Том 1

Санкт-Петербург
Издательство СПБГЭТУ «ЛЭТИ»
2010

ГЕНЕТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ НА ОСНОВЕ ОБУЧАЮЩИХ ПРИМЕРОВ ДЛЯ ПОСТРОЕНИЯ КОНЕЧНЫХ АВТОМАТОВ УПРАВЛЕНИЯ МОДЕЛЬЮ БЕСПИЛОТНОГО САМОЛЕТА

А. В. Александров, С. В. Казаков, А. А. Сергушичев, Ф. Н. Царев

Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Abstract. A construction method of finite-state machines with genetic programming based on testing is described in this paper. This method is applied to construction of control finite-state machine for a model of unmanned airplane. Method of automaton representation, algorithms of mutation and cross-over, fitness function and edges labeling algorithm are described. Results of application of the method to two problems of airplane control are described.

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 [1].

В автоматном программировании программы предлагается строить в виде набора автоматизированных объектов управления. Каждый такой объект состоит из системы управления (системы конечных автоматов) и объекта управления. Система конечных автоматов получает на вход события из внешней среды, а также входные переменные от объекта управления и внешней среды. На основании этих данных система управления вырабатывает выходные воздействия для объекта управления.

Для многих задач управляющие конечные автоматы удается строить эвристически, однако существуют задачи, для которых такое построение затруднительно. К задачам этого класса относятся, например, задача «Умный муравей» [2–4], задача «Умный муравей–3» [5] и задача об управлении моделью беспилотного летательного аппарата [6]. Для автоматического построения автоматов в таких задачах можно применять генетические алгоритмы [7–9].

В работе [10] для построения конечного автомата, реализующего автопилот верхнего уровня, который управляет моделью беспилотного самолета, было предложено применять алгоритм генетического программирования [9], основанный на использовании метода сокращенных таблиц для представления конечных автоматов. При этом вычисление функции приспособленности основано на моделировании поведения беспилотного аппарата во внешней среде, поэтому оно занимает достаточно большое время.

Целью настоящей работы является разработка метода, лишенного этого недостатка. Разрабатываемый метод основан на генетическом программировании и алгоритме расстановки пометок на переходах и является развитием идей, предложенных в работе [11].

Постановка задачи. Исходными данными для построения управляющего конечного автомата является набор обучающих примеров. Каждый из них состоит из двух последовательностей: последовательности входных данных (описывает состояние объекта управления и окружающей среды) и соответствующей ей эталонной последовательности выходных воздействий (описывает изменение

параметров объекта управления). Эти обучающие примеры создаются человеком.

Задача аллгоритма генетического программирования состоит в построении конечного автомата, который задает поведение на обучающих примерах, достаточно хорошо соответствующее эталонному. Если использовать много обучающих примеров (в настоящей работе их число достигало 40), то появляется возможность избавиться от мелких неточностей, которые допускает при управлении человек.

Моделирование беспилотного самолета. Для моделирования беспилотного самолета применяется свободный авиасимулятор *FlightGear* (http://www.flightgear.org), который позволяет осуществлять программное управление самолетом, а также сохранение параметров полета (скорость, направление полета и т.д.) и состояния самолета (положение руля, элеронов, состояние стартера и т.п.).

Отметим, что некоторые параметры могут принимать лишь конечное число значений (такие параметры будем называть дискретными), а другие параметры характеризуются вещественными значениями, которые будем называть непрерывными. Кроме того, будем называть непрерывными действия над непрерывными параметрами, а дискретными – над дискретными параметрами.

При этом непрерывные действия изменяют соответственные параметры на некоторую вещественную величину, а дискретные – устанавливают соответствующий параметр в конкретное значение. Заметим, что последовательное выполнение действий с одним параметром эквивалентно сумме действий в случае непрерывного параметра и последнему действию – в случае дискретного.

Хромосома алгоритма генетического программирования. Конечный автомат в алгоритме генетического программирования представляется в виде объекта, который содержит описания переходов для каждого из состояний и номер начального состояния, причем число состояний автомата фиксированно и является параметром генетического алгоритма. Для каждого состояния хранится список переходов. Каждый переход описывается событием, при поступлении которого этот переход выполняется.

Таким образом, в особи кодируется только «скелет» управляющего конечного автомата, а конкретные выходные воздействия, вырабатываемые на переходах, определяются с помощью алгоритма расстановки пометок.

На рис. 1 изображен возможный «скелет» автомата.

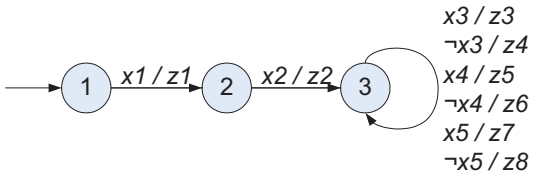

Рис. 1. «Скелет» автомата, поддерживающего постоянное направление движения

На этом рисунке входные переменные имеют следующий смысл:

- *x1* – начало такта;
- *x2* – двигатель включен;
- *x3* – отклонение меньше нуля;
- *x4* – скорость изменения отклонения меньше нуля;
- *x5* – ускорение изменения отклонения меньше нуля.

Выходные воздействия *z1*, …, *z8*, указанные на рис. 1, в «скелете» автомата не определены – конкретные выходные воздействия определяются при вычислении функции приспособленности.

Если для данного «скелета» определить выходные воздействия следующим образом:

- *z1* – включить стартер;
- *z2* – выключить стартер, поставить обороты на максимум;
- *z3/z4* – повернуть руль вправо/влево на 0.01;
- *z5/z6* – повернуть руль вправо/влево на 0,005;

- *z7/z8* – повернуть руль вправо/влево на 0,0025,

то получится автомат, поддерживающий постоянное направление движения.

Функция приспособленности. Вычисление функции приспособленности состоит из двух частей: расстановка выходных воздействий и оценка полученного автомата на обучающих примерах. Расстановка воздействий выполняется таким образом, чтобы максимизировать оценку при заданном «скелете» автомата.

Рассмотрим процесс вычисления оценки. На вход автомату подается каждая из последовательностей in[*i*] – входной набор *i*-ого обучающего примера. Затем записывается последовательность выходных воздействий, которую сгенерировал автомат. Обозначим ее как *out[i]*, а выходную последовательность из теста *i* – как *ans[i]*. После этого оценка адекватности вычисляется так: $Fitness = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \frac{D(out[i], ans[i])}{D(ans[i], 0)}\right)$, где

$$D(out[i], ans[i]) = \sum_{t=0}^{dlen[i]} \left(dout[i][t] - dans[i][t]\right)^2 + \sum_{t=0}^{clen[i]} \left[cout[i][t] \neq cans[i][t]\right],$$

где *dout[i]* и *dans[i]* – векторы дискретных выходных воздействий *i*-ого теста, dlen[*i*] – их длины, а cout[*i*] и cans[*i*] – векторы непрерывных выходных воздействий *i*-ого теста, clen[*i*] – их длины.

Алгоритм расстановки воздействий. Для того, чтобы расставить действия на переходах, скелету автомата на вход подается входной набор параметров из каждого обучающего примера, при этом отмечаются переходы, совершаемые им. Можно показать следующее:

- в случае дискретных действий максимум функции приспособленности достигается при выборе того действия, которое встречается на рассматриваемом переходе наибольшее число раз;
- так как в каждый момент времени значение каждого параметра определяется линейной комбинацией неизвестных воздействий, то значения отвечающих максимуму функции приспособленности непрерывных воздействий можно получить, дифференцируя функцию приспособленности и решая полученную систему линейных уравнений.

Операция мутации. При мутации с заданной вероятностью (по умолчанию, она равна 0.5) выполняется каждое из действий:

- изменение начального состояния;
- добавление, изменение или удаление конкретного перехода.

Операция скрещивания. В скрещивании принимают участие две особи. При этом результатом операции скрещивания 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* верно ровно 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].*

Результаты. Для проверки эффективности предложенного метода были поставлены две задачи – построение автомата, управляющего разгоном самолета, и автомата, выполняющего «мертвую петлю».

Для решения первой задачи:

- записано пять наборов тестов (по 10 обучающих примеров в каждом), которые рассматривались независимо друг от друга;

- число состояний автомата варьировалось от 2 до 10;
- вероятность мутации менялась от 0,02 до 0,5;
- в качестве методов отбора использовались турнирный метод и метод рулетки [12];
- размер элиты составлял от нуля до двух особей;
- в среднем время работы генетического алгоритма составляло несколько дней для каждого набора параметров.

Было установлено следующее:

- с увеличением числа состояний структура управляющего автомата становилась все менее логичной;
- наиболее оптимальными оказались автоматы с тремя–четырьмя состояниями;
- при вероятности мутации, равной 0,5, результаты достигались быстрее всего.

Аналогичная работа была проведена для решения второй задачи.

График зависимости максимального значения функции приспособленности от номера поколения для первой задачи приведены на рис. 2.

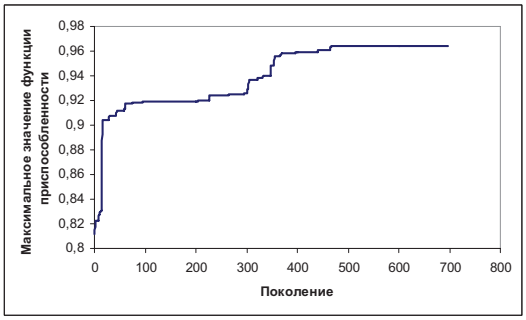

Рис. 2. График значения функции приспособленности

Для первой задачи в результате был получен автомат, поведение которого очень близко к записанным тестам. Для второй задачи пока не получено столь хорошего решения, хотя получен автомат, в целом справляющийся с задачей.

Видео одного из обучающих примеров для первой задачи можно посмотреть по адресу http://www.youtube.com/watch?v=BfG90yFiGUo, а запись разгона, выполняемого одним из полученных автоматов, – по адресу http://www.youtube.com/watch?v=AiosTU7HWb4.

Выводы. В работе был исследован метод построения с помощью генетического программирования автоматов для управления самолетом. Эффективность применения этих методов была проверена на вышеуказанных задачах.

Исследования финансируются по гранту РФФИ № 10-01-00654-а.

Литература

1. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. СПб: Питер, 2009.
2. Angeline P., Pollack J. Evolutionary Module Acquisition / Proceedings of the Second Annual Conference on Evolutionary Programming. Cambridge: MIT Press. 1993, pp.154–163. http://www.demo.cs.brandeis.edu/papers/ep93.pdf
3. Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C., Wang A. The Genesys System: Evolution as a Theme in Artificial Life /Proceedings of Second Conference on Artificial Life. MA: Addison-Wesley. 1992, pp.549–578.
4. Царев Ф. Н., Шалыто А. А. Применение генетического программирования для генерации автомата в задаче об «Умном муравье» / Сборник трудов IV-ой Международной научно-практической конференции «Интегрированные модели и мягкие вычисления в искусственном интеллекте». Том 2. М.: Физматлит. 2007, с. 590–597. http://is.ifmo.ru/genalg/_ant_ga.pdf
5. Бедный Ю. Д., Шалыто А. А. Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». http://is.ifmo.ru/works/ant
6. Парашенко 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. СПбГУ ИТМО. 2006. http://is.ifmo.ru/unimod-projects/plates/
7. Гладков Л. А., Курейчик В. В., Курейчик В. М. Генетические алгоритмы. М.: Физматлит, 2006.
8. Рассел С., Норвиг П. Искусственный интеллект: современный подход. М.: Вильямс, 2006.
9. Koza J. R. Genetic programming: on the programming of computers by means of natural selection. MIT Press, 1992.
10. Поликарпова Н. И., Точилин В. Н., Шалыто А. А. Применение генетического

программирования для генерации автоматов с большим числом входных переменных // Научно-технический вестник СПбГУ ИТМО. Выпуск 53. 2008. Автоматное программирование, с. 24–42.

11. Царев Ф. Н. Построение автоматов управления системами со сложным поведением на основе тестов с помощью генетического программирования / Сборник докладов XII Международной конференции по мягким вычислениям и измерениям (SCM'2009). Том 1. СПбГЭТУ «ЛЭТИ». 2009, с. 231–234.