Опубликовано в Научно-техническом вестнике СПбГУ ИТМО. 2008. Выпуск 53. Автоматное программирование, с. 79-88.

УДК 004.4'242

ПОСТРОЕНИЕ АВТОПИЛОТА ДЛЯ УПРОЩЕННОЙ МОДЕЛИ ВЕРТОЛЕТА С ПОМОЩЬЮ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

П.Г. Лобанов, С.А. Сытник, А.А. Шалыто

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

Предложен генетический алгоритм построения автопилота для упрощенной модели вертолета. Задачей вертолета является прохождение заданного множества целей в заданном порядке. алгоритм реализован на языке программирования *Java*. Выполнены вычислительные эксперименты, демонстрирующие эффективность этого алгоритма.

Ключевые слова: генетические алгоритмы, конечный автомат, автоматное программирование

Введение

Существует ряд задач, которые эффективно решаются с помощью конечных автоматов. В большинстве случаев построение автоматов выполняется эвристически [1], что достаточно трудоемко. В связи с этим актуальна задача разработки методов автоматизированной генерации автоматов. Генетические алгоритмы [2] представляют собой мощный подход, позволяющий получать точные или достаточно близкие к ним решения для широкого спектра задач. Они применимы и в тех случаях, когда решением задачи является конечный автомат. Для таких задач, как «умный муравей» [2], итерированная дилемма узников [3], задача о флибах [4], синхронизация [5] и классификация плотности для клеточных автоматов [6, 7], известны генетические алгоритмы [2], которые позволяют автоматически строить автоматы. В данной работе рассматривается одна из таких задач – задача построения автопилота для простейшей модели вертолета с помощью генетического алгоритма.

Постановка задачи

Требуется построить автопилот для простейшего вертолета, перемещающегося в двумерной плоскости. За один шаг вертолет может повернуться на некоторый фиксированный угол и изменить скорость своего движения (ускориться или замедлиться). Минимальная скорость, с которой может перемещаться модель вертолета, $V_{min} = 10^{-4}$, максимальная – $V_{max} = 2$, ускорение $a = 0.1$ (все величины и графики в работе приведены в условных единицах). Требуется построить автопилот, который за отведенное время полета проведет вертолет в определенном порядке через максимальное число заранее заданных целей. Цель – это точка, через которую должен пройти вертолет. Цели нумеруются, а вертолет проходит цели в порядке возрастания номеров. При этом вертолет не может пропускать цели, но, так как время полета ограничено, он может не успеть пройти все из них. Считается, что вертолет прошел цель, если он оказался от нее на расстоянии не более 0.5.

Вертолет и окружающая его среда представляются моделями, образующими взаимодействующую систему (рис. 1).

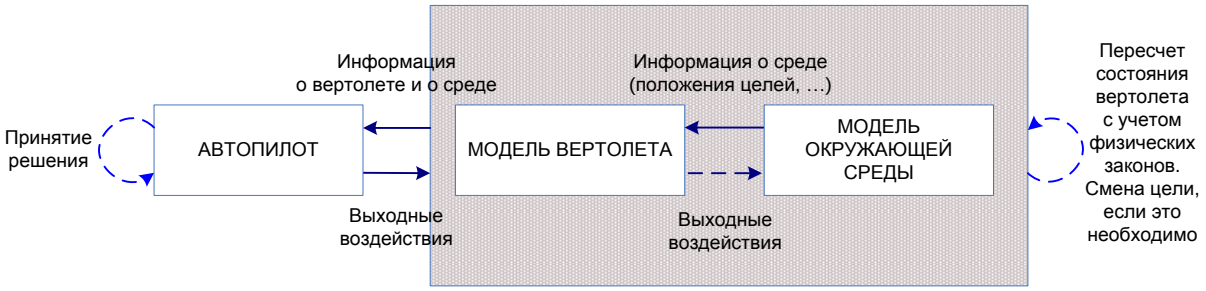

Рис. 1. Структурная схема системы

Модель вертолета, используя полученную от среды информацию, вычисляет необходимые ей данные о текущем состоянии системы. Она использует часть этих данных для их передачи автопилоту в качестве входных воздействий с целью анализа текущей ситуации и принятия решений. Модель окружающей среды «знает», что представляют собой эти воздействия с точки зрения определенных в ней физических законов, и в соответствии с ними пересчитывает положение и скорость вертолета.

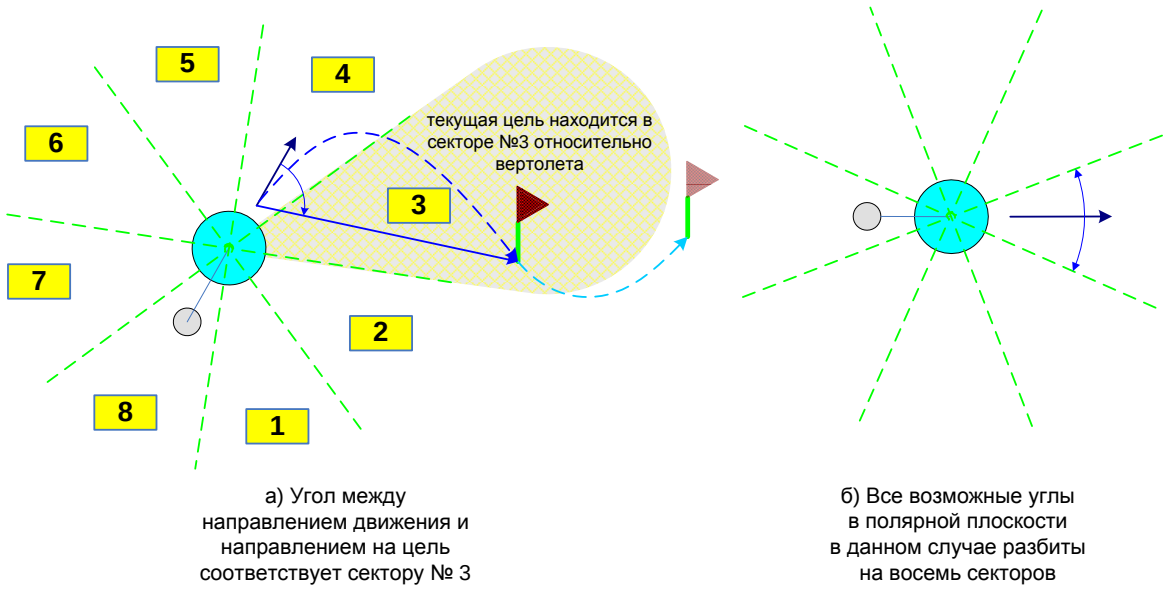

Рис. 2. Входные данные

Входные данные (воздействия) (рис. 2) автопилота представляют собой единственную переменную: положение текущей цели относительно вертолета, заданное углом между направлением движения вертолета и направлением на цель (рис. 2, а). Сектора всегда неподвижны относительно вертолета. Необходимо отметить, что вертолет летит не по границе двух секторов, а посередине одного сектора (рис. 2, б).

В качестве выходных в работе выбраны два воздействия, которые действуют относительно текущего вектора движения вертолета: «Изменить скорость» (придать ускорение в направлении вектора движения) и «Повернуть» (повернуть вектор движения на некоторый угол). Текущие значения ускорения и угла поворота никак не связаны с их предыдущими значениями. В каждый момент времени вертолет знает о положении только одной цели. Таким образом, следующее состояние вертолета зависит от его положения в пространстве, скорости движения и текущей цели. При достижении вертолетом цели текущей становится следующая по порядку. В результате направление на цель изменяется.

В качестве модели автопилота используется конечный автомат. Его входные и выходные воздействия должны быть дискретными. Для этого пространство вокруг вертолета разбивается на сектора обзора вертолета, число которых определяется заранее и является настраиваемым параметром. Автомат содержит некоторое число состояний, ограниченное сверху. Из каждого состояния графа переходов исходят дуги (переходы), число которых равно числу секторов. В качестве значения входной переменной, помечающей соответствующий переход, выступает номер сектора, в котором находится текущая цель вертолета. Каждый переход переводит автомат в новое состояние. Он помечен конкретным набором выходных воздействий. Поскольку автомат не использует памяти для хранения информации о глубокой предыстории (зависит только от предыдущего состояния), выбор следующего состояния производится автоматом лишь на основе входного воздействия и текущего состояния. Состояния *косвенно* отражают информацию о текущем положении вертолета, его скорости и предыстории переходов между состояниями. Рис. 3 иллюстрирует работу автопилота, заданного конечным автоматом.

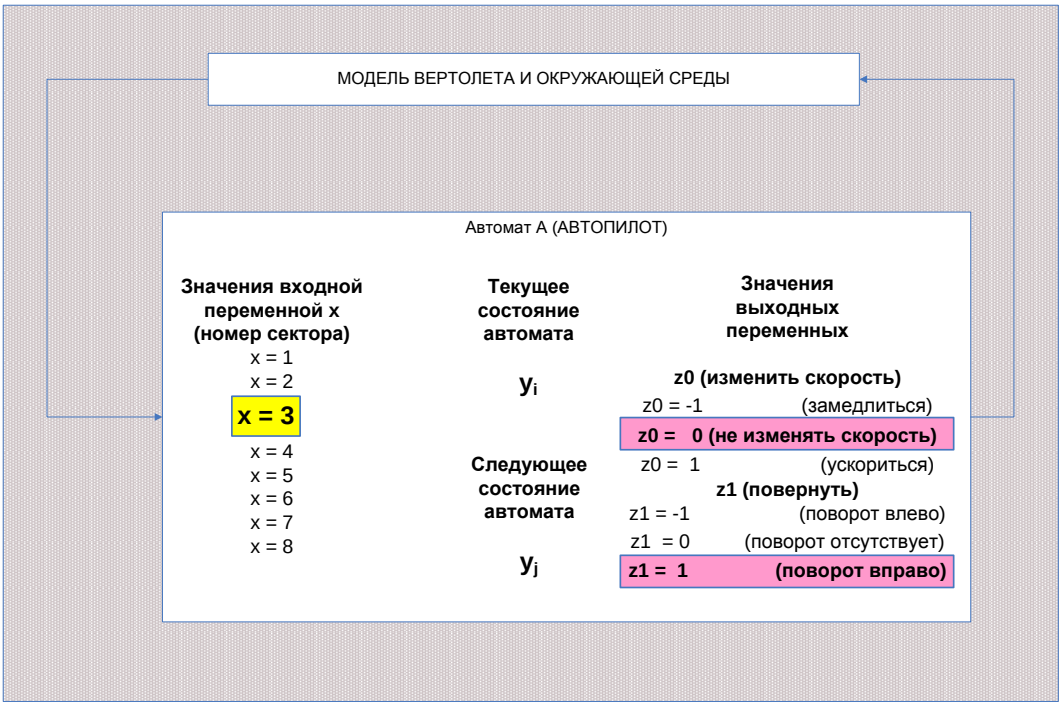

Рис. 3. Работа автопилота, заданного конечным автоматом

Модель вертолета

Модель вертолета содержит данные о положении вертолета в пространстве, его скорости и направления движения. Положение вертолета в пространстве хранится в виде точки $|x_h, y_h|$. Скорость и направление движения вертолета задаются вектором $|v_x, v_y|$, который называется вектором скорости. При этом скорость вертолета определяется соотношением $V = \sqrt{v_x^2 + v_y^2}$. Так как по условию задачи скорость вертолета не может быть меньше 10^{-4}, то вектор скорости всегда однозначно определяет направление движение вертолета.

Модель вертолета отвечает за изменение его положения в пространстве и пересчет скорости. Положение вертолета в следующий момент времени $\left|x'_h, y'_h\right|$ вычисляется по формуле $\left|x'_h, y'_h\right| = \left|x_h + v_x, y_h + v_y\right|$.

Определение вектора скорости $\left|v'_x, v'_y\right|$ для следующего момента времени выполняется за два шага. Сначала с помощью формулы

$$\left|v''_x, v''_y\right| = \left|v_x \frac{V + 0.1z_0}{V}, v_y \frac{V + 0.1z_0}{V}\right|$$

определятся *промежуточный вектор*, учитывающий изменение скорости вертолета, задаваемое значением выходной переменной z0. Затем определяется искомый вектор с помощью поворота *промежуточного вектора*

$$\left|v'_x, v'_y\right| = \left|v''_x + \frac{0.1z_1 v''_y}{V''}, v''_y - \frac{0.1z_1 v''_x}{V''}\right|,$$

где z_1 – значение выходной переменной z1, а V'' – длина промежуточного вектора $\left|v''_x, v''_y\right|$. Если длина полученного вектора V'' оказывается меньше минимальной скорости V_{min}, то вектор скорости $\left|v'_x, v'_y\right|$ вычисляется по формуле

$$\left|v'_x, v'_y\right| = \left|v'_x \frac{V_{min}}{V'}, v'_y \frac{V_{min}}{V'}\right|.$$

Если V' больше максимальной скорости V_{max}, то $\left|v'_x, v'_y\right|$ изменяется по формуле

$$\left|v'_x, v'_y\right| = \left|v'_x \frac{V_{max}}{V'}, v'_y \frac{V_{max}}{V'}\right|.$$

Модель окружающей среды

В модели окружающей среды хранятся координаты целей вертолета $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ и номер текущей цели k. Когда расстояние от вертолета до текущей цели становится меньше 0.5 ($\sqrt{(x_k - x_h) + (y_k - y_h)} < 0.5$), номер k увеличивается на единицу.

Модель среды отвечает за определение номера сектора, в котором находится текущая цель относительно вертолета. Сначала вычисляется угол α между направлением движения вертолета и направлением на текущую цель. Для этого используется формула

$$\alpha = \operatorname{arctg}\left(\frac{v_x(y_k - y_h) - v_y(x_k - x_h)}{v_x(x_k - x_h) + v_y(y_k - y_h)}\right).$$

Номер сектора n вычисляется по формуле:

$$n = \left\lfloor \frac{\frac{\alpha}{\pi} + 1}{2m} \right\rfloor + 1,$$

где m – число секторов обзора.

Алгоритм построения автопилота

Задача автопилота – провести вертолет через максимальное число наперед заданных целей в определенном порядке. Как было отмечено выше, время полета ограничено. Лучшим является автопилот, который провел вертолет через большее число целей за отведенное время. Оценочная функция (функция *fitness*) [2] должна быть максимально гладкой, что позволяет улучшить точность сравнения особей. Если число пройденных целей одинаково, то лучшим считается автопилот, для которого расстояние до следующей цели к моменту завершения полета минимально.

В настоящей работе хромосому будем задавать в виде графа переходов конечного автомата. Такое задание хромосомы определяется тем, что в рассматриваемой задаче переходы помечаются только одной входной переменной. При этом гены хромосомы – компоненты графа переходов (например, значения входной переменной).

Общая схема генетического алгоритма

На рис. 4 приведена общая схема работы генетического алгоритма [2]. В данной работе в качестве стратегии отбора особей для построения следующего поколения используется отбор отсечением [8]. При применении такой стратегии отбора родительские решения выбираются из группы лучших решений текущего поколения. Размер этой группы (порог отсечения) обычно составляет от 1/100 до 1/3 от размера поколения. Все автоматы из группы лучших решений имеют одинаковые шансы быть выбранными в качестве родительских особей.

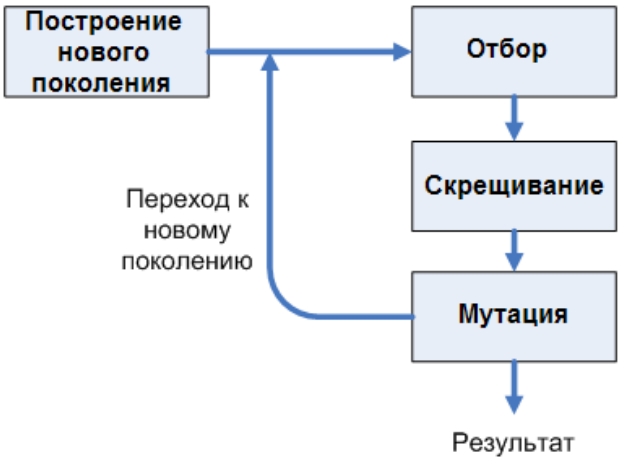

Рис. 4. Общая схема генетического алгоритма

Приведем описание генетического алгоритма, использующего отбор отсечением.

1. Начальное поколение заполняется случайными решениями.
2. Начальное поколение решений становится текущим.
3. Из текущего поколения отбирается группа лучших решений.
4. Формируется новое поколение решений:
   a. Создается пустое новое поколение.
   b. Из группы лучших решений, выбранных на основе порога отсечения, случайным образом выбирается некоторая пара.
   c. Формируется новое решение с помощью применения оператора скрещивания [2] к двум выбранным решениям.
   d. К новому решению *последовательно* применяются операторы мутации [2].
   e. Полученное решение добавляется в новое поколение решений.

f. Если размер нового поколения меньше размера текущего поколения, то переходим к пункту b.

5. Новое поколение становится текущим.
6. Если число созданных поколений меньше заданного пользователем, то переходим к шагу 3.

В приведенном алгоритме на шаге 4, d, указано, что к новому решению последовательно применяются операторы мутации, в качестве которых выступают *n*-точечный оператор и предлагаемый авторами оператор мутации – сортировка состояний в порядке использования.

Оператор скрещивания

В данной работе первым используется *n*-точечный оператор *скрещивания*. Для этого оператора задается вероятность, с которой он применения к каждому гену в хромосоме. Приведем описание алгоритма работы оператора скрещивания.

1. В качестве нового автомата формируется копия первого из выбранных решений (шаг 4, b приведенного выше алгоритма).
2. Осуществляется цикл по всем его состояниям. Для каждого состояния:
   a. Выполняется цикл по всем переходам, исходящим из состояния. Для каждого перехода:
      i. Определяется с заданной вероятностью, требуется ли изменить номер состояния, в которое переходит автомат. Если это требуется, то он изменятся на номер состояния из соответствующего перехода второго автомата.
      ii. Определяется с заданной вероятностью, требуется ли изменить значение выходной переменной, генерируемое при переходе автомата. Если это требуется, то значение указанной переменной заменяется значением выходной переменной, полученной из соответствующего перехода второго автомата.

N-точечный оператор мутации

Этот оператор применяется к каждому гену в хромосоме с некоторой заранее заданной вероятностью, в отличие от обычного оператора мутации. Такой оператор может изменить сразу несколько генов. При этом отметим, что значения этой вероятность и вероятности используемой в операторе скрещивания в общем случае отличаются. Приведем описание алгоритма работы рассматриваемого оператора мутации.

3. Осуществляется цикл по всем состояниям автомата:
   a. Выполняется цикл по всем переходам из состояния:
      i. С заданной вероятностью определяется необходимость изменение номера состояния, в которое автомат переходит.
      ii. Если требуется изменить номер состояния, то он изменяется на номер состояния, выбранный случайным образом.
      iii. С заданной вероятностью определятся необходимость изменения значения выходной переменной, генерируемой автоматом при переходе.
      iv. Если требуется изменить значение выходной переменной, то оно изменяется на одно из возможных значений, выбранное случайным образом.

Новый оператор мутации – сортировка состояний в порядке использования

В большинстве решений, которые перебирает генетический алгоритм, автомат в процессе работы не переходит в часть состояний, число которых в задается исходно. На

поведение автомата влияют только те состояния, из которых осуществляется переход по дуге хотя бы один раз. Далее такие состояния будем называть *используемыми*, а все остальные состояния – *неиспользуемыми*.

Приведем алгоритм сортировки состояний в порядке их использования.

1. Создается пустой словарь пар номеров [«старый номер состояния» – «новый номер состояния»].
2. Моделируется полет вертолета. Перед каждым переходом по дуге, если в словаре нет пары, в которой первый элемент равен текущему номеру состояния, то в него добавляется пара [текущий номер состояния – число пар в словаре].
3. Выполняется цикл по всем состояниям автомата. Для каждого состояния, если в словаре нет пары, в которой первый элемент равен номеру состояния, то в него добавляется пара [номер состояния – число пар в словаре].
4. Согласно словарю изменяется порядок состояний.

Состояния, для которых в словарь добавляются пары на шаге 2 – *используемые состояния*. Состояния, для которых в словарь добавляются пары на шаге 3 – *неиспользуемые состояния*.

Построенный автомат в большинстве случаев может быть упрощен за счет исключения *неиспользуемых* состояний и замены переходов в них переходами в начальное состояние (выполняется при необходимости компактного представления).

Описание программы

Для проведения экспериментов по генерации автопилотов была написана программа на языке *Java*, исходные коды которой будут приведены на сайте *http://is.ifmo.ru* в разделе «Статьи». Программа позволяет задавать время полета, верхний предел числа состояний автомата, размер поколения, максимальное число поколений, число секторов, вероятности применения для *n*-точечных операторов мутации и скрещивания. Также в ней можно изменить порог для стратегии отбора отсечением. При проведении экспериментов при необходимости можно изменять расположение целей вертолета.

Для кодирования графов переходов автоматов используется три класса: `StateMachine`, `State` и `Branch`. В качестве главного класса автомата применяется класс `StateMachine`. Классы `State` и `Branch` реализуют его состояния и переходы соответственно. Массив `states` в классе `StateMachine` содержит состояния автомата автопилота. Поле `currentStateIndex` используется для хранения номера текущего состояния автомата. В массив `brunches` класса `State` включены переходы из данного состояния. Поле `stateIndex` в классе `Branch` содержит номер состояния, в которое переходит автомат. В массиве `outputs` хранятся значения выходных переменных, генерируемых при переходе.

При запуске программа создает два файла. В первом из них содержится траектория полета вертолета для автопилота, построенного с помощью генетического алгоритма. Во второй файл выводится таблица переходов и выходов для лучшего автомата. Перед формированием таблицы производится сортировка состояний автомата в порядке использования и удаляются *неиспользуемые состояния*.

Эксперименты

Все эксперименты проводились при размере поколения 300 и пороге отсечения 90. Максимальное число поколений – 200. Вероятность применения оператора скрещи-

вания к переходу автомата – 0.04, а вероятность применения *n*-точечного оператора мутации – 0.02. Число состояний автомата не больше 15. Время полета – 200.

На рис. 5 изображены двадцать целей, пронумерованных в том порядке, в котором через них должен пролететь вертолет. Точка с координатами (0; 0) является исходной. В начале движения вертолет ориентирован вправо и движется в этом направлении с минимальной возможной скоростью. При проведении экспериментов изменялось число секторов обзора вертолета (четыре или шесть), а также использовалась или не использовалась сортировка состояний. Эксперименты с каждым набором этих параметров алгоритма проводились по 10 раз.

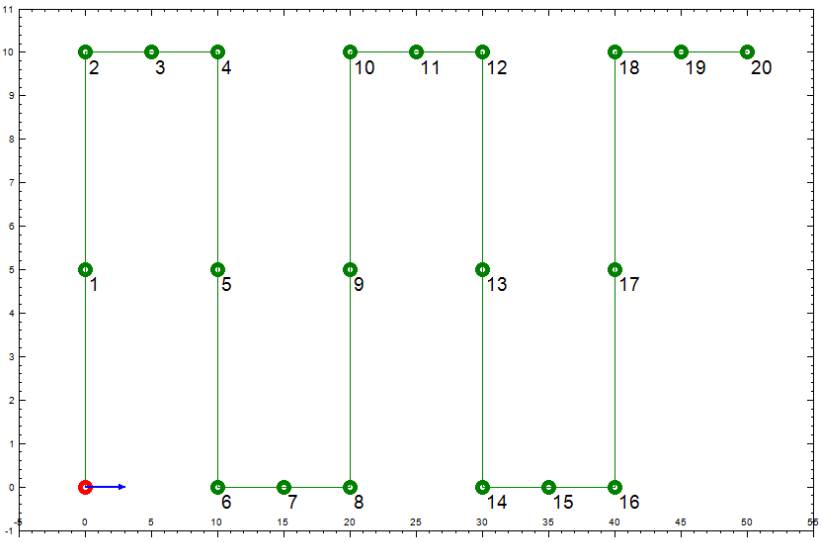

Рис. 5. Цели вертолета

Таблица 1. Результаты экспериментов

Число секторов	Сортировка состояний	Результат		
		худший	средний	Лучший
4	-	12	14.1	16
	+	12	14.7	17
6	-	11	15.6	18
	+	14	16.6	18

В табл. 1 приведены результаты экспериментов. Под результатом эксперимента понимается число целей, через которые пролетел вертолет с автопилотом, построенным с помощью предложенного генетического алгоритма. Столбец «худший» – худший результат из десяти экспериментов, колонка «лучший» – лучший результат этих экспериментов, а столбец «средний» – усредненный результат. Из табл. 1 видно, что использование алгоритма сортировки состояний улучшает эффективность работы генетического алгоритма.

Таблица 2. Лучший автопилот

	1			2			3			4			5			6		
	s	z0	z1	s	z0	z1	s	z0	z1	s	z0	z1	s	z0	z1	s	z0	Z1
1	2	1	-1	6	0	1	2	-1	-1	12	0	1	2	1	-1	10	1	0
2	8	-1	1	1	0	1	10	1	1	3	1	0	1	1	-1	3	-1	0
3	11	0	0	4	-1	1	3	-1	1	3	1	-1	7	-1	0	8	1	0
4	7	0	0	3	0	1	5	-1	1	2	-1	0	2	1	0	10	-1	-1

5	10	0	-1	2	0	-1	6	1	1	1	1	-1	2	0	0	2	0	1
6	7	0	-1	10	0	0	3	1	1	2	1	-1	1	-1	0	1	-1	-1
7	1	1	0	8	0	-1	8	0	1	7	1	-1	7	-1	-1	1	-1	-1
8	3	-1	-1	1	0	0	6	0	1	9	0	-1	11	0	1	4	0	0
9	1	1	0	7	-1	1	6	0	1	4	0	-1	12	-1	1	7	1	-1
10	1	-1	1	6	-1	-1	7	-1	1	11	-1	1	1	0	0	7	0	0
11	1	1	-1	6	0	0	7	-1	1	11	-1	1	1	0	0	7	0	0
12	1	0	-1	4	1	1	6	0	1	9	0	-1	4	1	-1	12	0	0

Приведем таблицу переходов и выходов автомата для лучшего автопилота (табл. 2). Столбцам этой таблицы соответствуют номера секторов обзора, строкам – состояния автомата. Каждый столбец состоит из трех колонок: колонка `s` – новое состояние, в которое перейдет автомат, колонки `z0`, `z1` – значения выходных переменных. На рис. 6 изображена траектория полета вертолета, управляемого лучшим из построенных автопилотов. Авторами также был выполнен эксперимент по использованию восьми секторов обзора вертолета, однако, несмотря на увеличение времени работы генетического алгоритма, существенного улучшения работы автопилота не наблюдалось

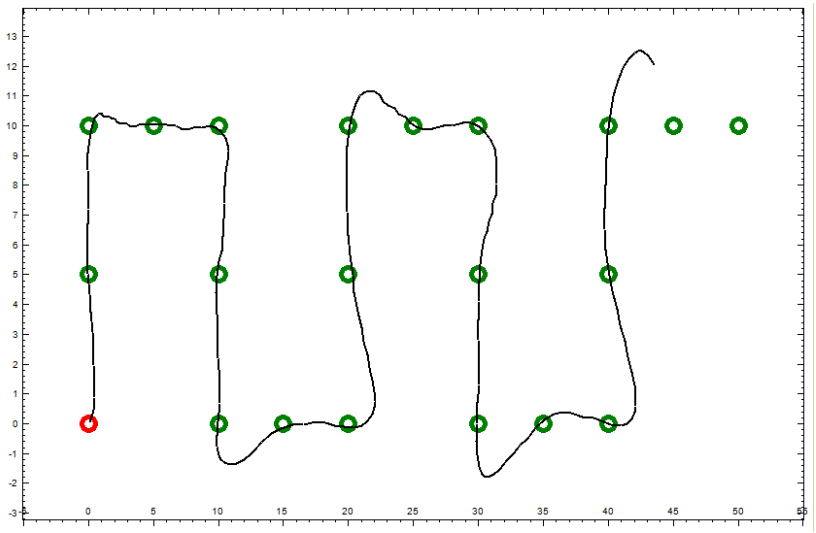

Рис. 6. Траектория полета лучшего вертолета

Заключение

В работе предложен генетический алгоритм, позволяющий *автоматически* построить автопилот для упрощенной модели вертолета. Разработана программа на языке *Java*, реализующая указанный алгоритм и позволяющая проводить эксперименты с моделями вертолета и окружающей среды. Предложен дополнительный оператор мутации, позволяющий улучшить генетический алгоритм. Приведены экспериментальные данные, подтверждающие эффективность этого алгоритма.

Построен автопилот с 12 состояниями, который проводит вертолет через первые 18 целей из 20 за отведенное время. Установлено, что при увеличении времени полета на 33 условных единицы полученный автопилот обеспечит прохождение всех 20 целей.

Литература

1. Шалыто А.А. Технология автоматного программирования / Труды первой Всероссийской конференции «Методы и средства обработки информации». – М.: МГУ. 2003. – Режим доступа: http://is.ifmo.ru/works/tech_aut_prog
2. Koza J. R. Genetic programming. On the Programming of Computers by Means of Natural Selection. – The MIT Press, 1998. – Режим доступа: www.ru.lv/~peter/zinatne/ebooks/MIT%20-%20Genetic%20Programming.pdf
3. Mitchell M. An Introduction to Genetic Algorithms. Cambridge. – MA.: MIT Press, 1996.
4. Лобанов П.Г., Шалыто А.А. Использование генетических алгоритмов для автоматического построения конечных автоматов в задаче о флибах // Известия РАН. – Теория и системы управления. – 2007. – № 5. – С127–136.
5. Wolfram S. A New Kind of Science. Champaign. – Wolfram Media, 2002.
6. Mitchell M., Crutchfield J., Hraber P. Evolving cellular automata to perform computations // Physica D. – 1993. – V. 75. – P. 361–391.
7. Бедный Ю.Д. Применение генетических алгоритмов для решения одной задачи на клеточных автоматах. Задача классификации плотности для клеточных автоматов. – Бакалаврская работа / СПбГУ ИТМО. – 2006. – Режим доступа: http://is.ifmo.ru/papers/genalgcelaut/
8. 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. // Artificial Life II: Proceedings of the Workshop on Artificial Life. – NJ: Addison-Wesley, 1992. – P. 549–578. – Режим доступа: www.cs.ucla.edu/~dyer/Papers/AlifeTracker/Alife91Jefferson.html