Опубликовано в Научно-техническом вестнике СПбГУ ИТМО. 2008. Выпуск 53. Автоматное программирование, с. 60-79.

УДК 004.4'242

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ И МЕТОДОВ СОКРАЩЕННЫХ ТАБЛИЦ ПЕРЕХОДОВ И ДЕРЕВЬЕВ РЕШЕНИЙ ДЛЯ ПОСТРОЕНИЯ АВТОМАТОВ УПРАВЛЕНИЯ МОДЕЛЬЮ БЕСПИЛОТНОГО ЛЕТАТЕЛЬНОГО АППАРАТА

А.А. Давыдов, Д.О. Соколов, Ф.Н. Царев
(Санкт-Петербургский государственный университет информационных технологий, механики и оптики)

В работе рассматривается применение генетического программирования для построения конечных автоматов, управляющих системами со сложным поведением. Для представления конечных автоматов используются два метода: метод сокращенных таблиц переходов и метод представления автоматов деревьями решений. Применение этих методов иллюстрируется на примере задачи об управлении моделью беспилотного летательного аппарата.
Ключевые слова: генетическое программирование, конечный автомат, беспилотный летательный аппарат, автоматное программирование

Введение

В последнее время все чаще применяется автоматное программирование [1], в рамках которого поведение программ описывается с помощью конечных детерминированных автоматов. В ряде задач автомат удается построить эвристическими методами, однако часто такое построение требует больших затрат времени или вообще невозможно. Примером такой задачи является управление командой беспилотных летательных аппаратов [2, 3] в соревнованиях с другой командой. Полный перебор крайне трудоемок, а эвристическое построение не всегда дает приемлемые результаты. Поэтому для построения автоматов в задачах такого рода целесообразно применять генетические алгоритмы и генетическое программирование.

Целью настоящей работы является построение с помощью генетического программирования автоматов Мили для управления беспилотными летательными аппаратами.

Постановка задачи

Проводится соревнование [2, 3] между двумя командами беспилотных летательных аппаратов. Цель соревнований состоит в том, чтобы один из летательных аппаратов команды переместился на максимальное расстояние от линии старта. Состязание проходит на трассе, представляющей собой полубесконечную (бесконечную в одну сторону) полосу шириной 40 м. Маневры, связанные с изменением высоты полета, не допускаются (таким образом, трасса соревнования двухмерна).

Каждая команда состоит из N летательных аппаратов. В дальнейшем, кроме термина «соревнование», будем использовать термин «гонка». В начале гонки аппараты первой команды располагаются в воздухе случайным образом на некотором расстоянии от линии старта в левой половине трассы. Вторая команда размещается симметрично первой на правой половине трассы. Для каждого аппарата заданы начальная скорость и направление движения. В простейшем случае начальные скорости всех аппаратов одинаковы, а направления – строго вперед. Летательные аппараты в процессе полета могут поворачивать. Каждый летательный аппарат имеет определенный запас топлива, расхо-

дуемого в процессе движения. По команде «Старт» все аппараты начинают движение с целью максимально удалиться от линии старта. Они в процессе полета могут изменять скорость своего движения за счет изменения расхода топлива.

Беспилотные летательные аппараты, покинувшие трассу, считаются прекратившими гонку. Выходом за пределы коридора считается пересечение центром аппарата границы трассы. Аппарат также считается аварийно прекратившим соревнование еще в двух случаях: во-первых, если его скорость падает ниже определенной величины, а топливный бак не пуст; во-вторых, если при столкновении двух аппаратов их относительная скорость больше определенной величины; в противном случае происходит абсолютно упругое столкновение.

Динамика беспилотных летательных аппаратов подчиняется второму закону Ньютона и подробно описана в работе [3]. Для нас важно то, что на каждый летательный аппарат влияет расположение остальных аппаратов, замедляя (благодаря реактивной силе двигателя) или, наоборот, ускоряя его (благодаря уменьшению силы сопротивления воздуха). Это дает простор для разработки различных стратегий управления командой беспилотных летательных аппаратов, что и является решаемой задачей.

В работе [3] была предложена система управления беспилотными летательными аппаратами, основанная на мультиагентном подходе [4]. При таком подходе каждый аппарат рассматривается в отдельности от других, а искусственный интеллект каждого летательного аппарата реализуется на основе конечных автоматов (для всех аппаратов одной команды он одинаков). Целью данной работы является построение управляющего конечного автомата с помощью генетических алгоритмов [5–11].

В рамках работ по применению генетических алгоритмов для построения конечных автоматов, проводимых в СПбГУ ИТМО, предложено несколько методов представления управляющего автомата в виде хромосомы, используемой в генетическом алгоритме [12–13]. В настоящей работе применяются метод представления автоматов деревьями решений [12] и метод сокращенных таблиц [13].

Структура системы управления беспилотным летательным аппаратом

В течение соревнования каждый летательный аппарат испытывает на себе воздействие среды и других летательных аппаратов. Для описания этих воздействий служат следующие логические входные переменные:

1. Граница трассы справа,
2. Граница трассы слева,
3. Другой аппарат слева,
4. Другой аппарат справа,
5. Другой аппарат спереди,
6. Другой аппарат сзади.

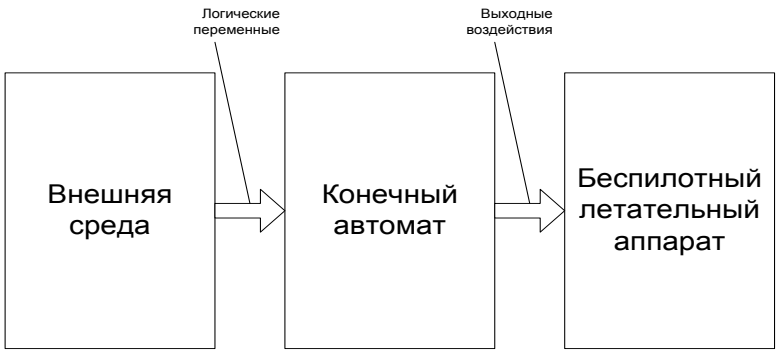

Рис. 1. Структурная схема системы управления беспилотным летательным аппаратом

Эти переменные подаются на вход управляющего автомата, который, в свою очередь, формирует последовательность выходных воздействий (рис. 1). Приведем список выходных воздействий:

1. установить нормальный расход топлива,
2. увеличить расход топлива на фиксированную величину,
3. уменьшить расход топлива на фиксированную величину,
4. сделать расход топлива максимально возможным,
5. изменить направление аэродинамического руля на фиксированный угол налево,
6. изменить направление аэродинамического руля на фиксированный угол направо,
7. лететь прямо.

Представление особи в генетическом алгоритме при помощи сокращенных таблиц переходов

При использовании метода сокращенных таблиц переходов каждая особь хранит следующие параметры:

1. массив состояний,
2. число состояний,
3. число обрабатываемых входных переменных,
4. число возможных действий.

Каждое состояние хранит сокращенную таблицу переходов, которая содержит:

1. массив значимых входных переменных.
2. массив состояний для всех переходов.
3. массив выполняемых действий для всех переходов.

На рис. 2 приведен пример сокращенной таблицы для одного из состояний автомата. Здесь столбец `S` – массив состояний, `Z1–Z4` массивы действий, `Z1[i]` равно единице, если данное действие выполняется. Массив `variables` указывает, какие переменные значимы, если переменной соответствует единица, то переменная значима.

S	Z1	Z2	Z3	Z4
1	0	0	1	0
3	1	0	1	1
5	1	0	1	0
1	1	0	1	1

variables	A	B	C	D	E	F	G
	0	0	1	0	0	1	0

Рис. 2. Пример сокращенной таблицы переходов

Значимыми называются входные переменные, значение которых обрабатываются при нахождении в данном состоянии (значения других переменных игнорируются).

Восстановление связей между состояниями

В данном разделе описан алгоритм восстановления связей между состояниями автомата, который обеспечивает достижение большего числа состояний автомата из начального за счет изменения конечного состояния некоторых переходов. Для поиска достижимых состояний используется алгоритм *поиска в ширину (Breadth First Search)*. Данный алгоритм описан ниже с помощью псевдокода.

```
TableAutomaton repairedAutomaton() {
  for (повторить N раз) {
    Запустить поиск в ширину от стартового состояния.
    Массив пометок mark содержит информацию о том, было
    ли посещено состояние.
    for (для всех i: состояний автомата) {
      if (!mark[i]) {
        State st = случайное достижимое состояние;
        Установить случайный переход из st в
        состояние i;
      }
    }
    if (не изменилось ни одно из состояний) {
      выйти из цикла;
    }
  }
  return this;
}
```

Данная процедура не гарантирует достижимость всех состояний из начального состояния для графа переходов, а лишь восстанавливает ее с вероятностью, возрастающей с константой N (в данной работе N = 10). Рис. 3 иллюстрирует работу данного алгоритма, серым цветом отмечены состояния, достижимые из стартового.

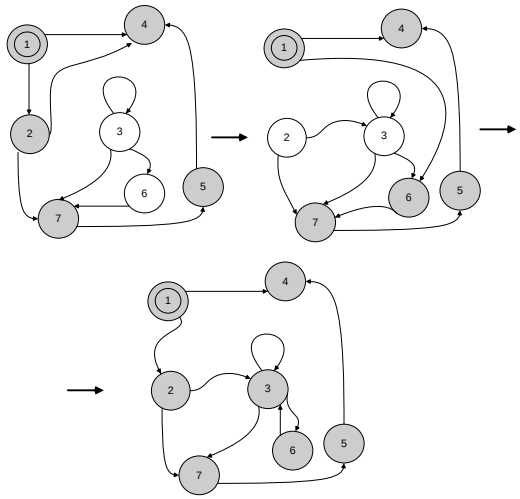

Рис. 3. Пример работы алгоритма восстановления связей между состояниями

Особенности применения сокращенных таблиц

Опишем преимущества метода сокращенных таблиц переходов применительно к рассматриваемой задаче:

- несовместность некоторых входных переменных (граница трассы слева, граница трассы справа) делает неиспользуемыми некоторые переходы. При использовании сокращенных таблиц вероятность такого события, по сравнению с полными таблицами, значительно снижается;
- сокращение используемой памяти, а также ускорение работы алгоритма.

При применении сокращенных таблиц все значимые переменные имеют равный приоритет, так как решение принимается сразу на основе всех переменных, которые используются в данном состоянии (альтернативой являются деревья решений [12], в которых решение принимается поэтапно).

Представление особи в генетическом алгоритме при помощи деревьев решений

При использовании метода представления автоматов деревьями решений каждая особь хранит следующие параметры:

- массив состояний;
- «рекомендуемая» высота дерева.

Каждое состояние представляет собой дерево решений для функции вида

$$f : \{0,1\}^n \to N \times \{0,1\}^k,$$

где n – число входных переменных, k – число возможных действий беспилотного летательного аппарата. Данное дерево по значениям входных переменных выдает номер состояния, в которое необходимо перейти автомату, а также вектор из нулей и единиц. При этом если i-ая компонента этого вектора равна единице, то аппарату необходимо выполнить i-ое действие. Далее будем называть этот вектор *вектором действий*. На рис. 4 приведен пример дерева решений для функции от двух булевых переменных (A, B).

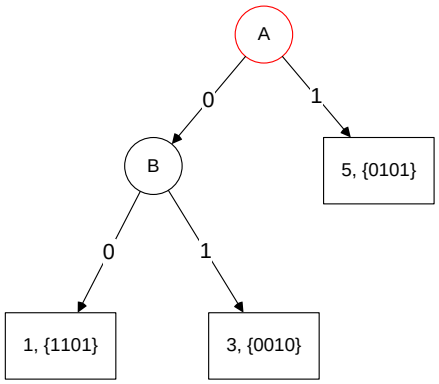

Рис. 4. Пример дерева решений

Каждое состояние представляется деревом решений, которое состоит из *узлов*. Среди узлов выделяется *корень* – стартовый узел дерева решений (на рис. 4 изображен сверху).

Каждый узел дерева решений состоит из следующих частей:

1. указателя на левого ребенка (для листьев этот указатель пуст);
2. указателя на правого ребенка (для листьев этот указатель пуст);
3. переменной, по которой происходит расщепление в данном узле (для листьев это значение не используется);
4. номера состояния автомата, в которое ведет переход из данной вершины (используется только в листьях);
5. вектор действий (для внутренних вершин этот вектор пуст).

Левый ребенок внутреннего узла соответствует нулевому значению переменной расщепления, а правый – единичному.

Алгоритм обрезки недостижимых ветвей дерева

Если по пути из корня до некоторого узла переменная, по которой происходит расщепление в этом узле, встречается дважды, то ветвь, соответствующая значению противоположному тому, которое было выбрано при первом расщеплении, будет *недостижимой*. Рис. 5 поясняет это утверждение. Светло-серым цветом на рис. 5 отмечены вершины с повторяющейся переменной расщепления, а темно-серым – недостижимая ветвь.

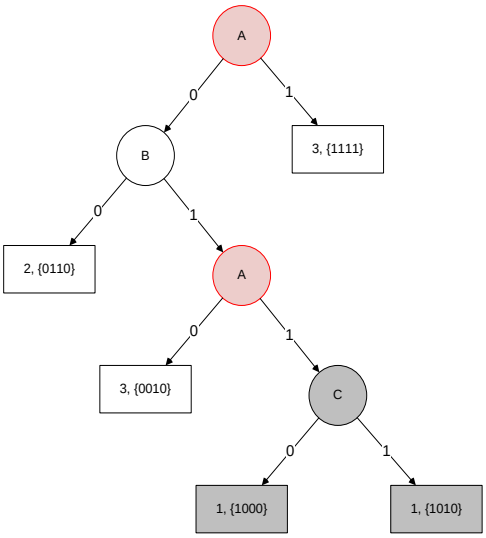

Рис. 5. Недостижимые ветви дерева решений

Для обрезки недостижимых ветвей используется модификация алгоритма *поиска в глубину* (*Depth First Search*). При рекурсивном спуске запоминаются переменные, по которым уже проводилось расщепление на пути из корня в текущий узел, а также значения этих переменных, соответствующие этому пути. Если переменная встречается второй раз, то текущий узел заменяется корнем достижимого поддерева этого узла. Решение о том, какое из поддеревьев достижимо, принимается на основании запомненных значений переменных. Работу данного алгоритма иллюстрирует рис. 6.

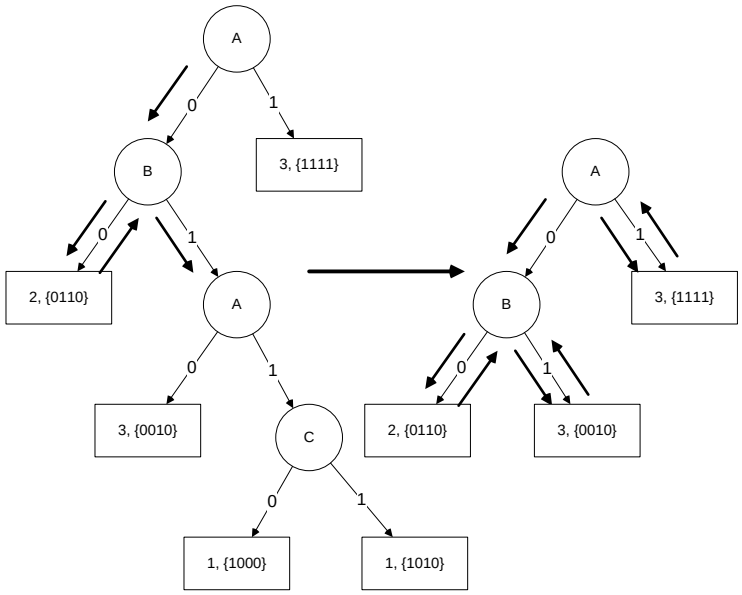

Рис. 6. Удаление недостижимых ветвей дерева решений

Генерация случайного дерева решений

Для генерации внутренней вершинц дерева применяется следующий алгоритм:

- выбрать переменную, по которой в данной вершине дерева будет проводиться расщепление;
- сгенерировать левого ребенка;
- сгенерировать правого ребенка.

Генерация листа происходит следующим образом:

- случайным образом выбрать номер состояния, в которое будет вести переход из данного листа;
- случайный образом сгенерировать вектор действий.

Для генерации случайного дерева решений применяется следующий алгоритм: с некоторой вероятностью (в работе использовалось значение 0.85) генерируется лист или внутренний узел. Данный алгоритм может продолжать свою работу бесконечно долго. Чтобы этого избежать введено ограничение на высоту дерева. Если при генерации высота дерева превысит удвоенное число возможных входных переменных, то обязательно генерируется лист.

Особенности представления при помощи деревьев решений

Дерево решений имеет те же достоинства, что и сокращенные таблицы:

- так же, как и при использовании сокращенных таблиц, могут использоваться не все входные переменные. Это позволяет с большей вероятностью, чем у представления с помощью битовых строк или полных таблиц переходов, исключить из рассмотрения переходы с несовместными переменными;
- еще большее сокращение объема требуемой памяти по сравнению с сокращенными таблицами. Даже если в конкретном дереве используется k входных переменных, то его размер может быть меньше чем $2k$ (вплоть до линейного от числа переменных).

Специфика деревьев решений указывает на то, что в каждом состоянии автомата задается свой приоритет входных переменных. Это связано с тем, что чем выше находится лист дерева, в котором происходит расщепление по данной переменной, тем больше вероятность, что в него можно попасть в зависимости от значений других входных переменных.

Генетический алгоритм

В работе для генерации системы управления беспилотным летательным аппаратом используется *островной генетический алгоритм* [8, 11]. Общая схема (рис. 7) алгоритма заключается в том, что существует несколько популяций – *островов*. Большую часть времени развитие популяций на каждом острове происходит независимо. При этом через заданное число поколений происходит *миграция* – часть особей с каждого острова передается другому острову.

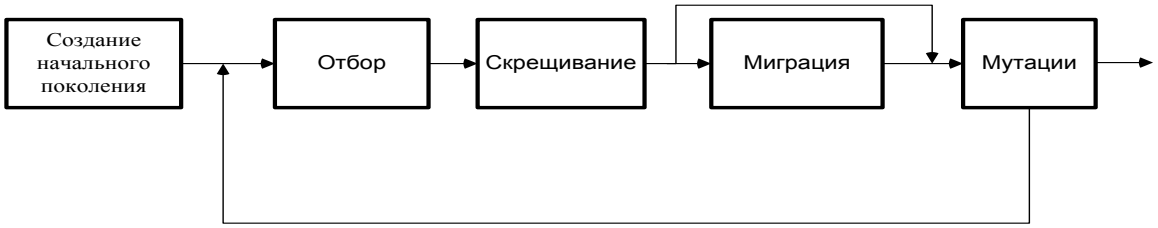

Рис. 7. Схема островного генетического алгоритма

Создание начального поколения

Все острова заполняются случайно сгенерированными особями. Все особи имеют заранее заданное число состояний. При применении сокращенных таблиц для хромосом всех особей задается число значимых переменных, которое одинаково для всех состояний.

Формирование следующего поколения

В качестве основной стратегии формирования следующего поколения используется элитизм. При рассмотрении текущего поколения отбрасываются все особи, кроме некоторой доли наиболее приспособленных – «элиты». «Элита» переходит в следующее поколение напрямую. После этого поколение дополняется в определенной пропорции случайными особями, особями из текущего поколения, которые мутировали, и результатами скрещивания особей из текущего поколения (отдельно отметим, что скрещиваться могут не только элитные особи, а все). Особи, «имеющие право» давать потомство, определяются «в поединке»: выбираются две случайные пары особей, и более приспособленная особь в каждой из них становится одним из родителей.

Оператор скрещивания особей, представленных при помощи сокращенных таблиц переходов

Оператор скрещивания особей для случая сокращенных таблиц переходов представляется следующим образом. Обозначим родительские особи – *P1* и *P2*, а детей – *S1* и *S2*. Обозначим *k*-ое состояние автомата *A* как *A.a[k]*. Как *c1[k]* и *c2[k]* обозначим результат скрещивания состояний *P1.a[k]* и *P2.a[k]*. Тогда для любого *k* будет верно утверждение *S1.a[k] = c1[k], S2.a[k] = c2[k]*.

Алгоритм скрещивания состояний представляет собой *одноточечное* скрещивание соответствующих столбцов таблицы переходов. Для каждого столбца выполняются следующие действия:

- случайный выбор границы разделения столбца;
- соответствующий столбец состояния *c1* составляется из первой части столбца состояния *P1.a[k]* и второй части столбца особи *P2.a[k]*;
- соответствующий столбец состояния *c2* составляется из второй части столбца особи *P1.a[k]* и первой части столбца особи *P2.a[k]*.

Данный алгоритм проиллюстрирован на рис. 8.

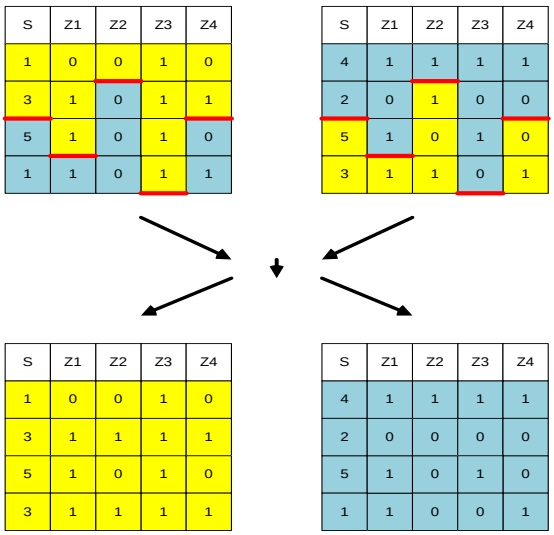

Рис. 8. Скрещивание сокращенных таблиц переходов

Для того чтобы выбрать значимые переменные *c1[k]* и *c2[k]*, выполняется просмотр списка значимых переменных родителей. Обозначим за *v[k]* *k*-ый элемент в списке значимых переменных. При этом справедливо одно из следующих утверждений:

1. *P1.v[k] = 0, P2.v[k] = 0* → *S1.v[k] = 0, S2.v[k] = 0*;
2. *P1.v[k] = 1, P2.v[k] = 1* → *S1.v[k] = 1, S2.v[k] = 1*;
3. *P1.v[k] = 0, P2.v[k] = 1* или *P1.v[k] = 1, P2.v[k] = 0* → возможен один из трех случаев:
   1. списки значимых переменных обоих детей еще не заполнены (число значимых переменных в обоих списках не превышает число значимых переменных у родителей), тогда с вероятностью *0.5* выполняются соотношения *P1.v[k] = 0, P2.v[k] = 1*, иначе *P1.v[k] = 1, P2.v[k] = 0*;
   2. списки значимых переменных первого ребенка заполнены, тогда *P1.v[k] = 0, P2.v[k] = 1*;
   3. списки значимых переменных второго ребенка заполнены, тогда *P1.v[k] = 0, P2.v[k] = 1*;

Отметим, что списки значимых переменных обоих потомков могут быть заполнены одновременно тогда и только тогда, когда в списках обоих родителей не осталось не просмотренных значимых переменных.

Оператор мутации особей, представленных при помощи сокращенных таблиц переходов

При мутации особи выполняются следующие действия:

- с вероятностью 0.5 случайное изменение начального состояния;
- мутация случайного состояния.

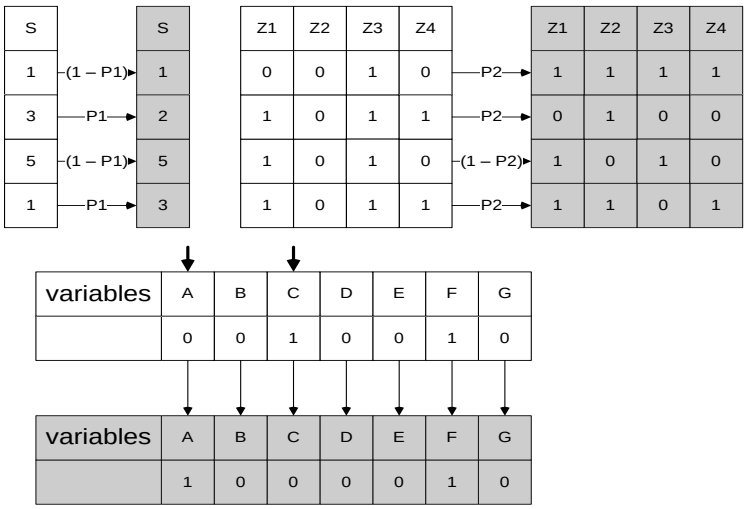

Рис. 9. Мутация сокращенных таблиц

Мутация состояния следующим образом: для каждой строки таблицы переходов выполнить:

- с некоторой заранее заданной вероятностью *p1* изменить переход из данного состояния по текущим значениям входных переменных на случайное значение (из допустимых);
- с некоторой заранее заданной вероятностью *p2* изменить все действия на данном переходе. Каждое действие выполняется на данном переходе с вероятностью *n1/ k*, где *n1* – число выполняемых действий на данном переходе до начала мутации, *k* – число возможных действий.

Выбор *значимых переменных* состояния осуществляется следующим образом:

- выбрать случайным образом две переменные;
- если одна из них значима в данном состоянии, а другая нет, то они меняются местами. На рис. 9 приведен пример мутации. Серым цветом отмечены получившиеся части состояния.

По окончании работы оператора мутации для получившейся особи запускается алгоритм восстановления связей между состояниями.

Оператор скрещивания особей, представленных при помощи деревьев решений

Оператор скрещивания особей представляется следующим образом. Обозначим родительские особи – *P1* и *P2*, а детей – *S1* и *S2*. Обозначим *k*-ое состояние автомата *A*, как *A.a[k]*. Обозначим *c1[k]* и *c2[k]* результат скрещивания состояний *P1.a[k]* и *P2.a[k]*. Тогда для любого *k* будет верно утверждение:

S1.a[k] = c1[k], S2.a[k] = c2[k].

Оператор скрещивания состояний фактически выбирает два поддерева – одно из первого дерева решений, второе из второго – а затем меняет их местами. Этот алгоритм, также как операция мутации и алгоритм обрезки недостижимых ветвей, представляет собой модификацию алгоритма поиска в глубину и имеет рекурсивную структуру. При каждом рекурсивном вызове этого алгоритма выполняются следующие действия:

- если текущий узел является листом, то обязательно, а для внутренних узлов с вероятностью P, вернуть данное поддерево;
- иначе равновероятно пойти в одно из поддеревьев текущего узла.

На рис. 11 приведен пример, в котором необходимо, как и в случае с оператором мутации, провести обрезку недостижимых ветвей. При этом для всех деревьев, соответствующих состояниям автомата, запускается алгоритм обрезки недостижимых ветвей.

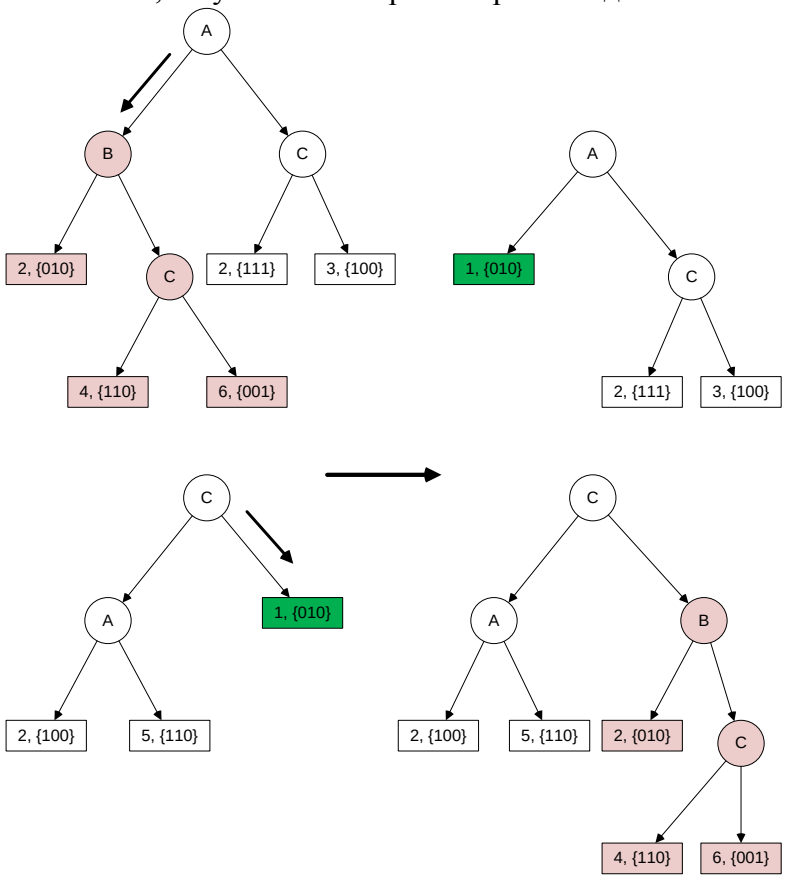

Рис. 11. Пример скрещивание деревьев решений

Оператор мутации особей, представленных при помощи деревьев решений

Оператор мутации выполняет:

- с вероятностью *0.5* случайное изменение стартового состояния;
- мутацию случайного состояния.

Оператор мутации состояния представляет собой модификацию алгоритма поиска в глубину. Выполняются следующие действия:

- если текущий узел является листом, то обязательно, а для внутренних узлов с вероятностью P, вернуть случайно сгенерированное дерево решений;
- иначе равновероятно пойти в одно из поддеревьев.

Фактически данный алгоритм случайно выбирает некоторое поддерево и заменяет его на случайно сгенерированное. Выбор поддерева происходит не равновероятно – чем выше узел, тем больше вероятность выбора его поддерева. По окончании мутации

для мутированной особи необходимо запустить алгоритм обрезки недостижимых ветвей дерева. Это иллюстрирует рис. 10.

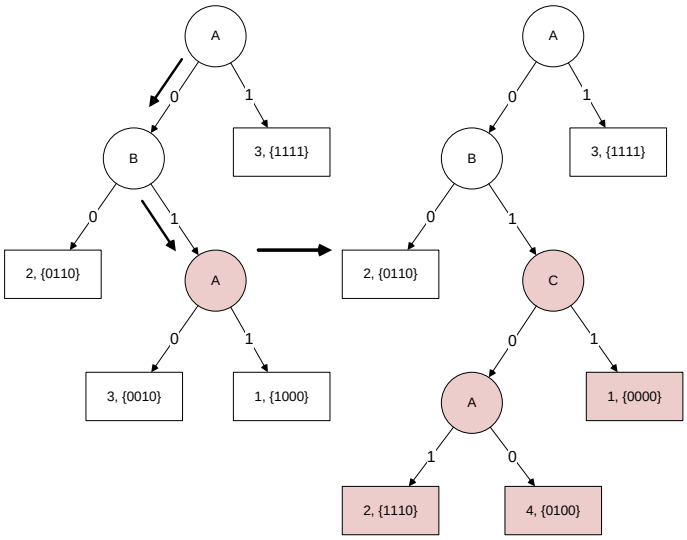

Рис. 10. Мутация деревьев решений

Миграции

Через фиксированное число поколений каждый остров меняется с другим случайным числом случайно выбранных элитных особей. Также через некоторое число поколений происходит *большая мутация*, называемая также *мутацией острова*.

Большие мутации

Для того чтобы избежать «вырождения» автоматов (попадания в локальный максимум функции приспособленности), предлагаемый генетический алгоритм использует описанную ниже процедуру. Через заранее заданное число поколений фиксированная доля островов заменяется островами со случайными особями. Это является причиной провала (рис. 12) на 41-ом поколении.

Проведение мутации в момент, когда функция приспособленности (имеются в виду только элитные особи) изменяется незначительно, невозможно, так как за счет миграции особей между островами среднее значение функции приспособленности постоянно изменяет свое значение.

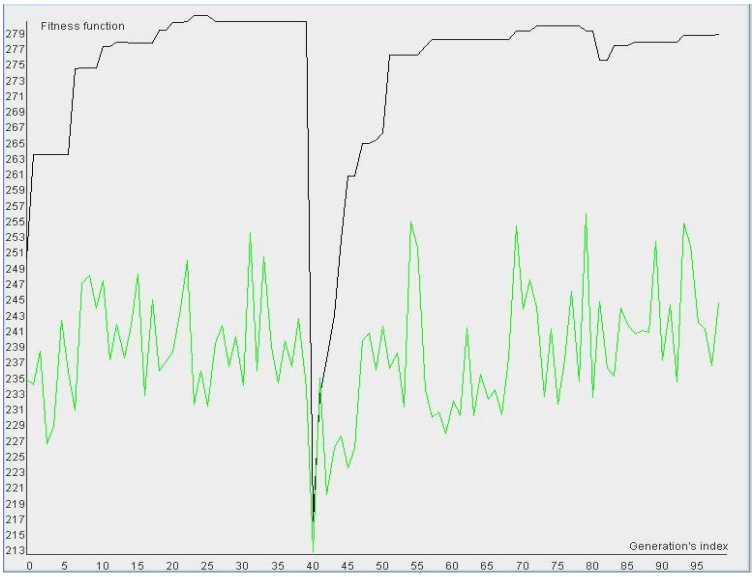

Рис. 12 Большая мутация

Вычисление функции приспособленности

Функция приспособленности особи должна зависеть от результата команды беспилотных летательных аппаратов, использующей стратегию, описываемую этим автоматом. Чтобы задать вид зависимости, требуется определить условия соревнования: начальные координаты летательных аппаратов и команду соперника. В этом-то и состоит проблема, так как, если при каждом запуске автомата начальные координаты и соперника выбирать случайным образом, то функция приспособленности будет «необъективна». Противоположный вариант – зафиксировать соперника и начальные координаты.

Предлагается вычислять функцию приспособленности как среднее арифметическое результатов нескольких соревнований. При этом начальные координаты – случайные, но одинаковые для всех команд, реализующих стратегию, заданную генерируемыми особями. Число соревнований выбиралось равным 10 из соображений уменьшения времени работы алгоритма.

$$F = \frac{\sum_{i=0}^{k} r_k}{k}.$$

Для особей, представленных в виде деревьев решений, формула была несколько изменена:

$$F = \frac{\sum_{i=0}^{k} r_k}{k} - C * Z(h_{\max}, height),$$

где C – некоторая константа, h_{max} – максимальная из высота деревьев решений, соответствующих состояниям данного автомата, *height* – «рекомендуемая» высота дерева решений, а Z – функция, определяемая следующим образом:

$$Z(a,b) = \begin{cases} a-b, a \geq b; \\ 0, a < b. \end{cases}$$

Таким образом, данная функция отвечает за то чтобы дерево решений не сильно «разрасталось». Если дерево решений имеет высоту больше *height*, функция приспособленности такой особи может быть ниже, даже если средний результат ее соревнований выше.

В качестве соперников были выбраны две команды. Первая – команда с «агрессивной» стратегией. Эта стратегия заключается в том, что один летательный аппарат летит только вперед с постоянным нормальным расходом топлива (нормальный расход топлива – такой расход топлива, при котором достигается наибольшая дальность полета). Остальные аппараты пытаются сбить летательные аппараты соперника. В качестве второго соперника была выбрана команда со стратегией, заданной автоматом, построенным с помощью описываемого генетического алгоритма, но при вычислении функции приспособленности, в которой в качестве соперника была взята только команда с «агрессивной» стратегией.

Этот автомат был выбран по двум причинам. Во-первых, он «продемонстрировал свою силу» в соревнованиях с «агрессивной» командой и командой, реализующей стратегию, заданную системой автоматов, построенных вручную (описана в работе [3]). Во-вторых, в отличие от этих двух команд, он реализует детерминированную стратегию. И, наконец, этот автомат реализует «дружелюбную» стратегию, которая заключается в том, что летательные аппараты помогают лететь друг другу, а не пытаются сбить аппараты команды соперника. Это позволяет «тренировать» новые управляющие автоматы сразу на двух принципиально различных стратегиях: «агрессивной» и «дружелюбной».

Проблемой является принципиальная невозможность оценить полученную особь по абсолютной шкале, как уже упоминалось ранее. Это происходит из-за недетерминированности начальных параметров задачи, а также противников. Этот эффект можно наблюдать на рис. 12, где изображены графики функции приспособленности особи, представляющей собой систему управления беспилотными летательными аппаратами от индекса поколения. Светлым цветом на этих графиках изображена средняя дальность летательного аппарата при проведении 30 соревнований (против десяти при вычислении функции приспособленности), а черным – функция приспособленности.

Число поколений работы генетического алгоритма ограничено лишь теми соображениями, что генерируемые на больших поколениях особи не демонстрируют универсального поведения, а приспосабливаются к соперникам и начальным условиям. Однако, если задан конкретный противник, то вырастить автомат, который, по крайней мере, играет не хуже его, не составляет особого труда. Естественно, результат соревнования все равно будет зависеть от начальных условий, но эта зависимость будет не слишком значимой.

Особенности применения островного генетического алгоритма

Островной генетический алгоритм имеет следующие достоинства (перед алгоритмами, в которых поколение развивается как единое целое):

- более быстрая сходимость к максимуму за счет миграций;
- возможность выхода из локального максимума без потери результатов предыдущих вычислений за счет «изолированности» островов.

К недостаткам островного алгоритма можно отнести более частое попадание в локальные максимумы по сравнению с традиционным генетическим алгоритмом.

Результаты

При помощи алгоритма генетического программирования с использованием метода представления автоматов с помощью сокращенных таблиц переходов построен конечный автомат управления моделью беспилотного летательного аппарата. Его граф переходов приведен на рис. 13. Построение этого автомата заняло 62 поколения при следующих значениях параметров алгоритма.

1. Число островов – 10.

2. Размер острова – 100 особей.
3. Процент «элиты» – 10%.
4. Вероятность мутации – 10%.
5. Процент особей, участвующих в миграции между островами – 3%.
6. Число поколений, проходящее между миграциями особей между островами – 10.
7. Число поколений, проходящее между «большими» мутациями – 41.
8. Процент островов, на которых все особи заменяются случайно сгенерированными в процессе «большой» мутации – 85%.

На рис. 13 используются указанные ниже обозначения. Пометки на вершинах имеют вид *номер/массив значимых переменных*. Переменные перечисляются в следующем порядке.

1. Граница трассы справа.
2. Граница трассы слева.
3. Другой аппарат слева.
4. Другой аппарат справа.
5. Другой аппарат спереди.
6. Другой аппарат сзади.

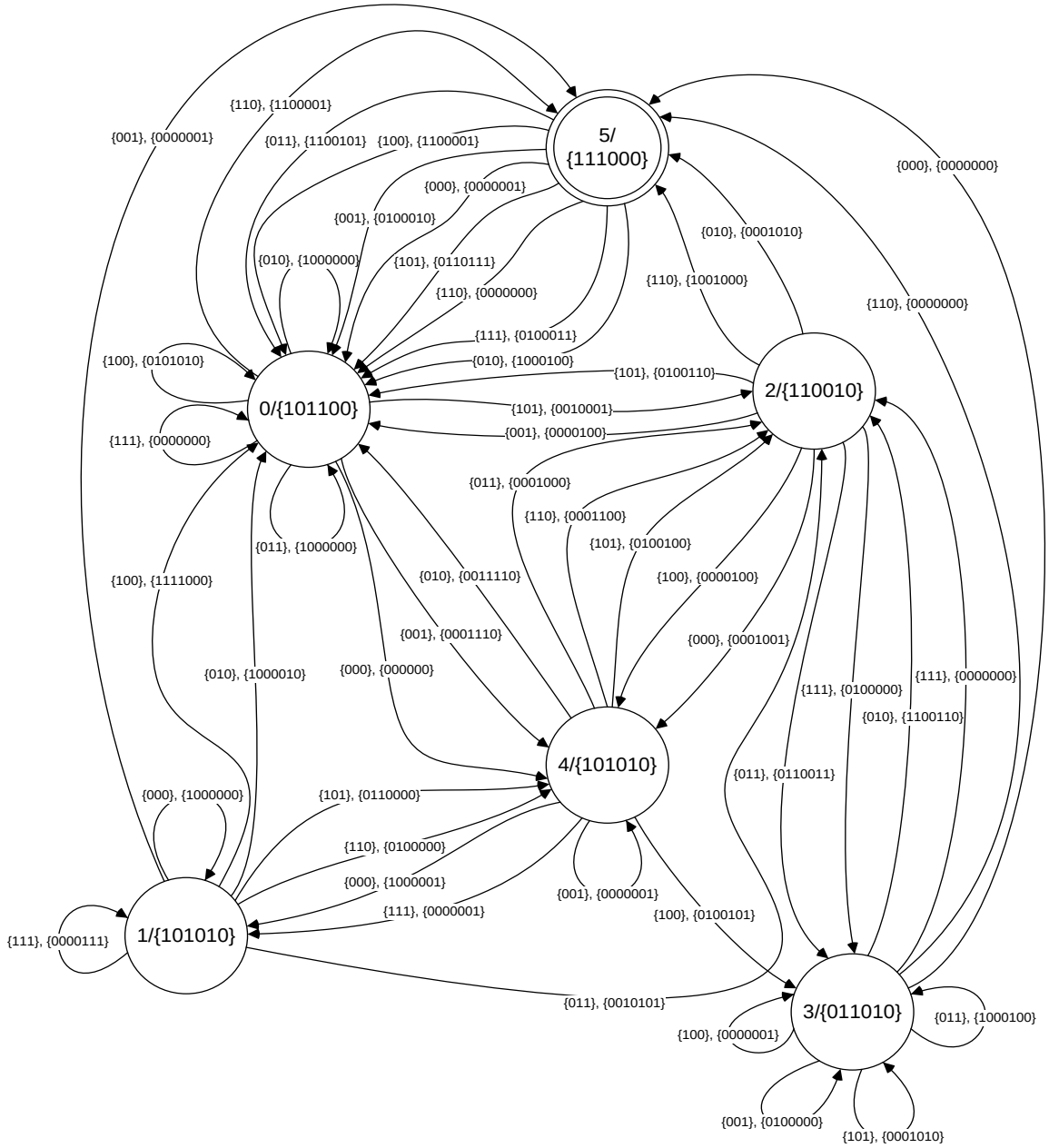

Рис. 13. Управляющий автомат, построенный алгоритмом генетического программирования

Пометки на переходах имеют вид *{значения переменных}/{вектор действий}*. Действия перечисляются в следующем порядке.

1. Установить нормальный расход топлива.
2. Изменить направление аэродинамического руля на фиксированный угол налево.
3. Изменить направление аэродинамического руля на фиксированный угол направо.
4. Лететь прямо.
5. Сделать расход топлива максимально возможным.
6. Увеличить расход топлива на фиксированную величину.
7. Уменьшить расход топлива на фиксированную величину.
8. Двойной окружностью показано начальное состояние автомата.

Команда, реализующая стратегию, описываемую полученным автоматом, из 50 соревнований с «агрессивной» командой выиграла 45, а с командой, построенной вручную [3] – 43.

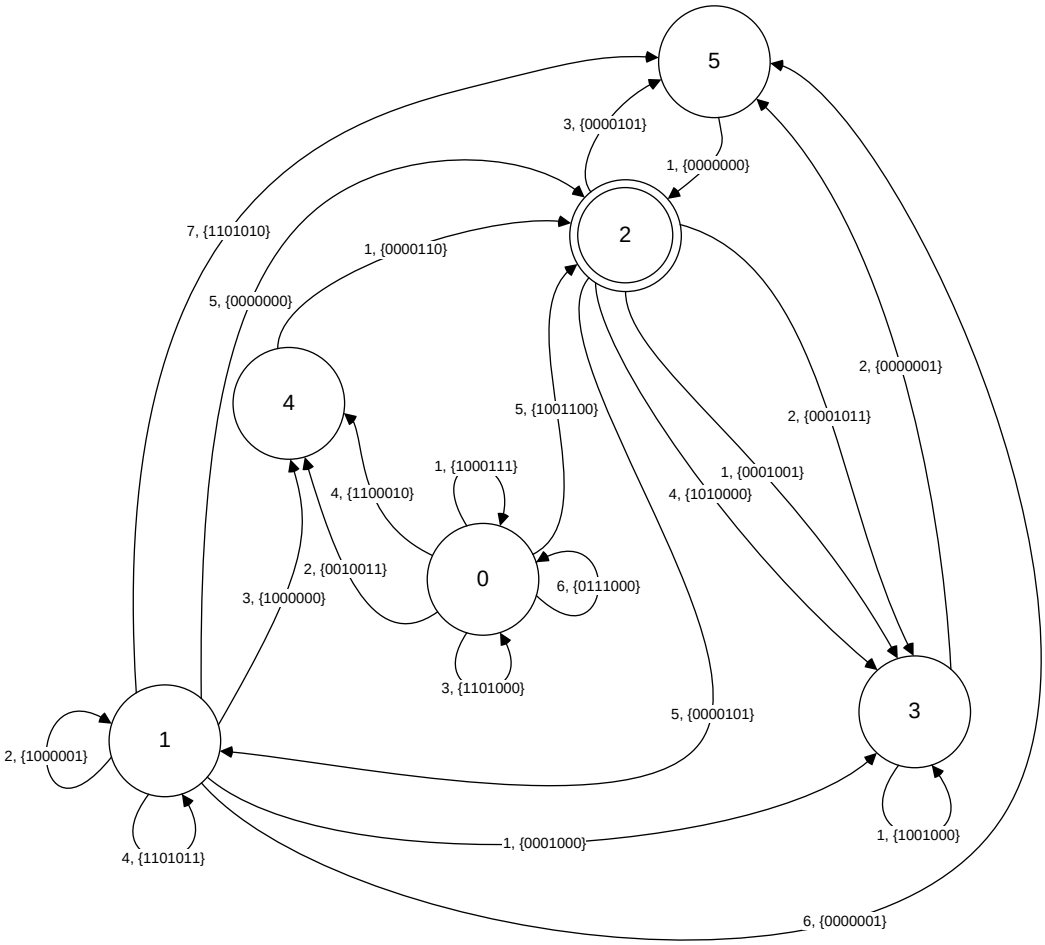

Рис. 14. Граф переходов конечного автомата, построенного с применением метода представления автоматов с помощью деревьев решений

При помощи алгоритма генетического программирования с использованием метода представления автоматов с помощью деревьев решений построен конечный автомат управления беспилотным летательным аппаратом. Его граф переходов приведен на рис. 14. Пометки на переходах имеют формат *номер перехода, {вектор действий}*. Построение этого автомата заняло 32 поколения при тех же значениях параметров, что и в случае с сокращенными таблицами, со следующими специфическими параметрами:

- рекомендуемая высота дерева решений – 4;
- вероятность генерации внутреннего узла дерева решений – 75%.

Опишем семантику каждой компоненты вектора действий.

1. Установить нормальный расход топлива.
2. Изменить направление аэродинамического руля на фиксированный угол налево.
3. Изменить направление аэродинамического руля на фиксированный угол направо.
4. Лететь прямо.
5. Сделать расход топлива максимально возможным.
6. Увеличить расход топлива на фиксированную величину.
7. Уменьшить расход топлива на фиксированную величину.

Переходы из каждого состояния пронумерованы натуральными числами, начиная с единицы. Номера переходов соответствуют номерам листов деревьев решений, показанных на рисунках ниже. Заметим, что в этом автомате состояние с номером 0 недостижимо и может быть удалено.

На следующих рисунках внутренние узлы дерева решений показаны кругами, а листья – прямоугольниками. Внутренние узлы помечены номером переменной, по которой осуществляется ветвление. Левое поддерево соответствует значению «ложь» переменной, которой помечен узел, а правое – значению «истина». Листья помечены номером перехода, который выполняется при достижении этого листа. Эти номера переходов соответствуют номерам переходов на рис. 15. Приведем перечень переменных, используемых на приведенных в настоящем разделе деревьях.

1. Граница трассы справа.
2. Граница трассы слева.
3. Другой аппарат слева.
4. Другой аппарат справа.
5. Другой аппарат спереди.
6. Другой аппарат сзади.

На рис. 15 показано дерево решений, соответствующее состоянию с номером 2 построенного автомата, на рис. 16 – дерево решений, соответствующее третьему состоянию автомата, на рис. 17 – дерево решений, соответствующее четвертому и пятому состояниям автомата. Наличие в этом дереве только одного узла, который является одновременно и корнем и листом, означает, что из этих состояний всегда выполняется один и тот же безусловный переход.

Приведем результаты соревнований, показанных системами управления, построенными с помощью метода сокращенных таблиц переходов, метода представления автоматов деревьями решений и построенной вручную. На рис. 18 показано распределение результатов, обеспеченных системой управления, построенной при помощи метода сокращенных таблиц переходов. На рис. 19 показано распределение результатов, обеспеченных системой управления, построенной при помощи метода представления автоматов деревьями решений. На рис. 20 показано распределение результатов, обеспеченных системой управления, построенной вручную в работе [3].

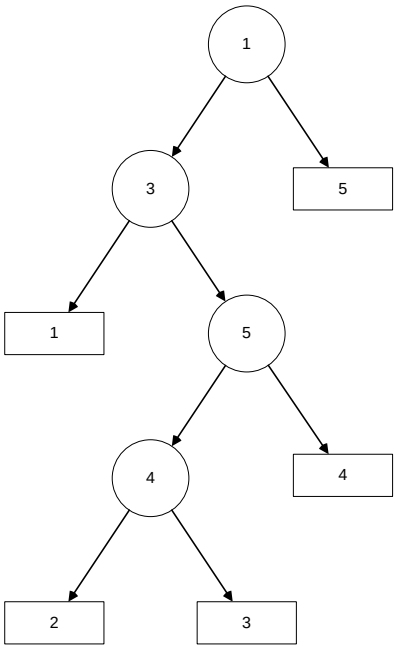

Рис. 15. Дерево решений, соответствующее второму состоянию автомата

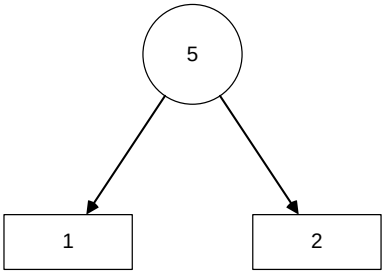

Рис. 16. Дерево решений, соответствующее третьему состоянию автомата

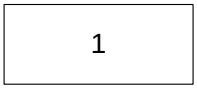

Рис. 17. Дерево решений, соответствующее четвертому и пятому состояниям автомата

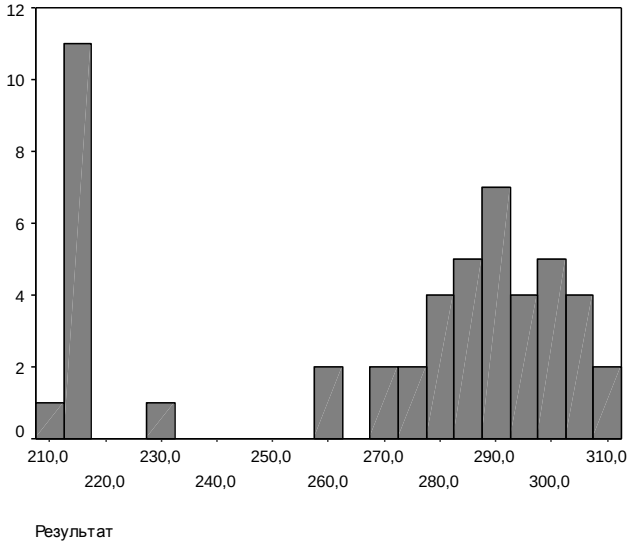

Рис. 18. Распределение результатов, обеспеченных системой, построенной при помощи метода сокращенных таблиц переходов

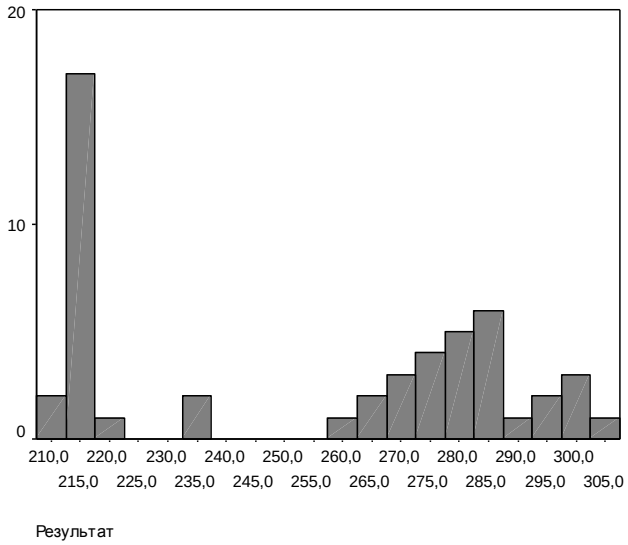

Рис. 19. Распределение результатов, обеспеченных системой, построенной при помощи метода представления автоматов деревьями решений

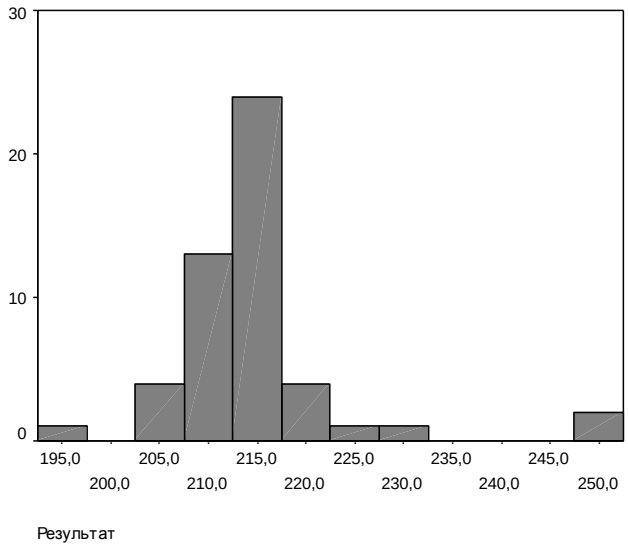

Рис. 20. Распределение результатов, обеспеченных системой из работы [3]

Выводы

В данной работе описано применение островного генетического алгоритма, а также двух способов представления особей (сокращенных таблиц переходов и деревьев решений), для создания управляющего автомата для модели беспилотного летательного аппарата. Проведено сравнение построенных автоматов управления с системой автоматов, построенной в работе [3]. Результаты этого сравнения позволяют сделать вывод о том, что построенные управляющие автоматы показали результаты, значительно превосходящие результаты, полученные с помощью системы автоматов, построенной вручную в работе [3]. Это доказывает эффективность используемых методов представления автоматов и их оптимизации применительно к рассматриваемой задаче.

Литература

1. Шалыто А.А. Технология автоматного программирования // Труды первой Всероссийской научной конференции «Методы и средства обработки информации». – М.: МГУ, 2003. – Режим доступа: http://is.ifmo.ru/works/tech_aut_prog/
2. Заочный тур всесибирской олимпиады 2005 по информатике. – Режим доступа: http://olimpic.nsu.ru/widesiberia/archive/wso6/2005/rus/1tour/problem/problem.html
3. Параценко 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. – Режим доступа: http://is.ifmo.ru/unimod-projects/plates/
4. Рассел С., Норвиг П. Искусственный интеллект. Современный подход. – М.: Вильямс. 2006.
5. Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C., Wang A. The Genesys System. 1992. – Режим доступа: www.cs.ucla.edu/~dyer/Papers/AlifeTracker/Alife91Jefferson.html
6. Angeline P. J., Pollack J. Evolutionary Module Acquisition // Proceedings of the Second Annual Conference on Evolutionary Programming. 1993. – Режим доступа: http://www.demo.cs.brandeis.edu/papers/ep93.pdf

7. Chambers L. Practical Handbook of Genetic Algorithms. Complex Coding Systems. Volume III. CRC Press, 1999.
8. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. – М.: Физматлит, 2006.
9. Koza J. R. Genetic programming: on the programming of computers by means of natural selection. – MIT Press, 1992.
10. Царев Ф.Н., Шалыто А.А. О построении автоматов с минимальным числом состояний для задачи об «умном муравье» // Сборник докладов X международной конференции по мягким вычислениям и измерениям. – СПбГЭТУ "ЛЭТИ". – Т.2. – 2007. – С.88–91. – Режим доступа: http://is.ifmo.ru/download/ant_ga_min_number_of_state.pdf
11. Яминов Б. Генетические алгоритмы. – Режим доступа: http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005
12. Данилов В.Р. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. СПбГУ ИТМО. 2007. Бакалаврская работа. http://is.ifmo.ru/papers/danilov_bachelor/
13. Поликарпова Н.И., Точилин В.Н. Применение генетического программирования для реализации систем со сложным поведением // Научно-технический вестник СПбГУ ИТМО. – 2007. – Выпуск 39. – С.276–293. – Режим доступа: http://vestnik.ifmo.ru/ntv/39/ntv_39.3.3.pdf