ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ ГЕНЕРАЦИИ АВТОМАТА В ЗАДАЧЕ ОБ «УМНОМ МУРАВЬЕ»

Ф. Н. Царев, А. А. Шалыто

ВВЕДЕНИЕ

В последнее время все шире начинает применяться автоматное программирование, в рамках которого поведение программ описывается с помощью конечных детерминированных автоматов [1].

Для многих задач автоматы удается строить эвристически, однако существуют задачи, для которых такое построение автоматов затруднительно. К задачам этого класса относится, в частности, и задача об «Умном муравье» [2–4].

В этих работах генерация автоматов выполнялась с помощью генетических алгоритмов [5] на однопроцессорной ЭВМ. Однако построенные в этих работах автоматы содержат большое число состояний.

Цель настоящей работы – устранить этот недостаток, также используя однопроцессорную ЭВМ.

ПОСТАНОВКА ЗАДАЧИ

Приведем описание задачи об «Умном муравье» [2]. Игра происходит на поверхности тора размером 32 на 32 клетки (рис. 1). В некоторых клетках (обозначены на рис. 1 черным цветом) находится еда. Она расположена вдоль некоторой ломаной, но не во всех ее клетках. Клетки ломаной, в которых нет еды обозначены серым цветом. Белые клетки не содержат еду и не принадлежат ломаной. Всего на поле 89 клеток с едой.

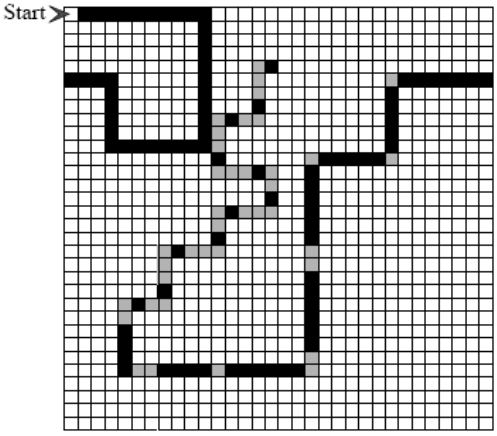

Рис. 1. Игровое поле

В клетке, помеченной меткой «Start», в начале игры находится муравей. Он занимает одну клетку и смотрит в одном из четырех направлений (север, юг, запад, восток). В начале игры муравей смотрит на восток.

Муравей умеет определять находится ли непосредственно перед ним еда. За один игровой ход муравей может совершить одно из четырех действий:

- сделать шаг вперед, съедая еду, если она там находится;
- повернуть налево;
- повернуть направо;
- ничего не делать.

Съеденная муравьем еда не восполняется, муравей жив на протяжении всей игры, еда не является необходимым ресурсом для его жизни. Ломаная не случайна, а строго фиксирована. Муравей может ходить по любым клеткам поля.

Игра длится 200 ходов, на каждом из которых муравей совершает одно из четырех действий. По истечении 200 ходов подсчитывается количество еды, съеденной муравьем. Это значение и есть результат игры.

Цель игры – создать муравья, который за 200 ходов съест как можно больше еды (желательно, все 89 единиц). При этом отметим, что муравьи, съедающие всю еду, заканчивают игру с одинаковым результатом, который не зависит от того, сколько ходов им на это потребовалось.

Один из способов описания поведения муравья – конечный автомат с действиями на переходах (автомат Мили), у которого есть одна входная переменная логического типа (находится ли еда перед муравьем), а множество выходных воздействий состоит из четырех упомянутых выше элементов.

Например, эвристически построенный автомат из [2], граф переходов которого изображен на рис. 2, описывает поведение муравья, который съедает 81 единицу еды за 200 ходов, а всю еду – за 314 ходов.

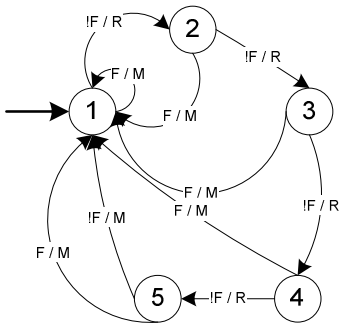

Рис. 2. Простой автомат, описывающий поведение муравья

Поясним используемые на рис. 2 обозначения. Пометки на переходах имеют формат `условие / действие`. Условия обозначаются следующим образом:

- `F` – перед муравьем есть еда;
- `!F` – перед муравьем нет еды.

Действия обозначаются следующим образом:

- `M` – «Сделать шаг вперед»;
- `L` – «Повернуть налево»;
- `R` – «Повернуть направо»;
- `N` – «Ничего не делать».

В работах [2–4] приведены графы переходов автоматов, которые сгенерированы с помощью генетических алгоритмов. Так, в [3] приведен автомат, который решает рассматриваемую задачу за 200 ходов и содержит 13 состояний. В [2] приведен автомат, граф переходов которого содержит 11 состояний, и позволяет, по утверждению авторов, съесть муравью всю еду за 193 хода. В [4] приведен автомат, содержащий 8 состояний и позволяющий муравью съесть всю еду за 193 хода.

В [2] показано, что указанный выше автомат с 13 состояниями был найден в результате генерации 52 поколений, в течение которых было построено более трех миллионов автоматов, а построение автомата с 11 состояниями потребовало от 418 до 4051 поколений. При этом при переборе строилось от 63000 до 607950 автоматов.

Ниже описывается алгоритм генетического программирования [6], который строит автомат с **восемью** состояниями, перебирая при этом от 200000 до 1500000 автоматов.

Этот же алгоритм позволил построить автомат с **семью** состояниями за 130000 поколений. Перебор при этом осуществлялся среди 160 миллионов автоматов.

ОПИСАНИЕ АЛГОРИТМА ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Разработанный алгоритм генетического программирования состоит из пяти частей:

- создание начального поколения;
- мутация;
- скрещивание (кроссовер);
- отбор особей для формирования следующего поколения;
- вычисление функции приспособленности (фитнес-функции).

Каждая особь представляет собой некоторый автомат, описывающий поведение муравья. Хромосома особи состоит из номера начального состояния и описаний состояний. Описание состояния содержит описания двух переходов, соответствующих тому, что перед муравьем либо есть еда, либо ее нет. Описание перехода состоит из номера состояния, в которое он ведет, и действия, выполняемого при выборе этого перехода.

Хромосома представляется не в виде битовой строки, как в генетических алгоритмах, а в виде объекта в языке программирования *Java*. Этот объект имеет описанную структуру:

```
public class Automaton {
        public Transition[][] transitions;
        public int initialState;
        public int stateCount;
}
```

Создание начального поколения. Начальное поколение состоит из фиксированного числа случайно сгенерированных автоматов. Все автоматы в поколении имеют одинаковое наперед заданное количество состояний.

Мутация. При мутации случайно выбирается один из четырех равновероятных вариантов:

- изменение начального состояния – в этом случае новое начальное состояние выбирается случайно и равновероятно;
- изменение действия на переходе – случайно и равновероятно выбирается переход, и действие на нем изменяется на случайное. При этом все возможные действия равновероятны;
- изменение состояния, в которое ведет переход, – случайно и равновероятно выбирается переход. После этого состояние, в которое ведет переход, заменяется на случайно выбранное состояние;
- изменение условия на переходе – случайно и равновероятно выбирается состояние. После этого переходы из этого состояния, соответствующие условиям «Перед муравьем есть еда» и «Перед муравьем нет еды», меняются местами.

Скрещивание. Оператор скрещивания получает на вход две особи и выдает также две особи. Процесс скрещивания происходит следующим образом. Обозначим родительские особи `P1` и `P2`, а потомков – `S1` и `S2`.

Обозначим начальное состояние автомата `A` как `A.is`. Тогда для потомков `S1` и `S2` будет верно одно из двух: либо `S1.is = P1.is` и `S2.is = P2.is`, либо `S1.is = P2.is` и `S2.is = P1.is`, причем оба варианта равновероятны.

Опишем, как «устроены» переходы автоматов-потомков `S1` и `S2` –может быть реализован один из двух равновероятных вариантов.

Первый вариант скрещивания. Обозначим переход из состояния номер `i` в автомате `P1` по значению входной переменной «Перед муравьем есть еда» как `P1(i, 1)`, а по значению «Перед муравьем нет еды» как `P1(i, 0)`. Аналогичный смысл придадим обозначениям `P2(i, 0)` и `P2(i, 1)`. Тогда для переходов из состояния с номером `i` в автоматах-потомках `S1` и `S2` будет справедливо одно из четырех:

- либо `S1(i, 0) = P1(i, 0), S1(i, 1) = P2(i, 1)` и `S2(i, 0) = P2(i, 0), S2(i, 1) = P1(i, 1)`;
- либо `S1(i, 0) = P2(i, 0), S1(i, 1) = P1(i, 1)` и `S2(i, 0) = P1(i, 0), S2(i, 1) = P2(i, 1)`;
- либо `S1(i, 0) = P1(i, 0), S1(i, 1) = P1(i, 1)` и `S2(i, 0) = P2(i, 0), S2(i, 1) = P2(i, 1)`;
- либо `S1(i, 0) = P2(i, 0), S1(i, 1) = P2(i, 1)` и `S2(i, 0) = P1(i, 0), S2(i, 1) = P1(i, 1)`.

Все четыре варианта равновероятны.

Второй вариант скрещивания. В автоматах `P1` и `P2` найдем переходы, которые они выполняют в течение первых сорока ходов по игровому полю. Обозначим множество таких переходов автоматов `P1` и `P2` как `TF(P1)` и `TF(P2)` соответственно. Множество переходов некоторого автомата `A` обозначим `T(A)`. Возможны два равновероятных варианта:

- либо $T(S1) = TF(P1) \cup (T(P2) \setminus TF(P2))$ и $T(S2) = TF(P2) \cup (T(P1) \setminus TF(P1))$;
- либо $T(S2) = TF(P1) \cup (T(P2) \setminus TF(P2))$ и $T(S1) = TF(P2) \cup (T(P1) \setminus TF(P1))$.

Формирование следующего поколения. В качестве основной стратегии формирования следующего поколения используется элитизм [7]. При обработке текущего поколения отбрасываются все особи, кроме нескольких наиболее приспособленных. Доля выживающих особей постоянна для каждого поколения и является одним из параметров алгоритма.

Эти особи переходят в следующее поколение. После этого оно дополняется до требуемого размера следующим образом: пока оно не заполнено выбираются две особи из текущего поколения, и они с некоторой вероятностью скрещиваются или мутируют. Обе особи, полученные в результате мутации или скрещивания, добавляются в новое поколение.

Кроме этого, если на протяжении достаточно большого числа поколений не происходит увеличения приспособленности, то применяются «малая» и «большая» мутации поколения. При «малой» мутации поколения ко всем особям, кроме 10% лучших, применяется оператор мутации. При «большой» мутации каждая особь либо мутирует, либо заменяется на случайно сгенерированную.

Количество поколений до «малой» и «большой» мутации постоянно во время работы алгоритма, но может быть различным для разных его запусков.

Вычисление функции приспособленности. Функция приспособленности особи (автомата) равна $F + \frac{200 - T}{200}$, где F –количество еды, которое съедает за 200 ходов муравей, поведение которого задается этим автоматом, а T – номер хода, на котором муравей съедает последнюю единицу еды. Она вычисляется моделированием и запоминается. Таким образом, для каждой особи функция приспособленности вычисляется один раз.

Настраиваемые параметры алгоритма. Следующие параметры алгоритма генетического программирования могут быть изменены:

- размер поколения;
- доля особей, переходящих в следующее поколение;
- количество состояний;
- вероятность мутации;
- время до «малой» мутации поколения;
- время до «большой» мутации поколения.

Как показывают вычислительные эксперименты, изменение некоторых из этих параметров может существенно влиять на время поиска автомата, позволяющего муравью съесть всю еду.

РЕЗУЛЬТАТЫ

Вычислительные эксперименты проводились с различными значениями параметров алгоритма. При этом в перебираемых автоматах не разрешалось использовать действие «Ничего не делать». Заметим, что это действие, на самом деле, бесполезно. Оно подобно ε-переходам в конечных недетерминированных автоматах, используемых для распознания регулярных языков, и может быть устранено алгоритмом, подобным алгоритму ε-замыкания [8].

На рис. 3 изображен граф переходов построенного разработанным алгоритмом автомата с **семью** состояниями, который позволяет муравью съесть всю еду за 190 ходов.

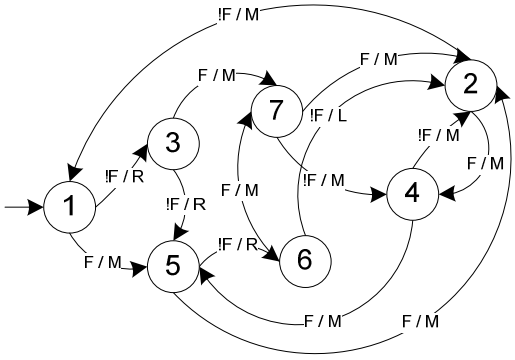

Рис. 3. Автомат, позволяющий муравью съесть всю еду

На рис. 3 используются те же обозначения, что и на рис. 2.

ЗАКЛЮЧЕНИЕ

Во многих работах, например в [9], генетические алгоритмы применяются для настройки нейронных сетей. Однако нейронная сеть обладает существенным недостатком: после того, как она настроена, ни понять, как она работает, ни изменить ее так, чтобы она стала работать лучше человек не в состоянии. Авторы предполагают, что конечные автоматы этого недостатка лишены, так как они лучше структурированы за счет применения концепции состояний [10].

Настоящая работа вместе с программной реализацией описанного алгоритма будет выложена на сайте http://is.ifmo.ru в разделе *Статьи*.

ИСТОЧНИКИ

1. Шалыто А.А. Технология автоматного программирования / Труды первой Всероссийской научной конференции "Методы и средства обработки информации" М.: МГУ. 2003. http://is.ifmo.ru/works/tech_aut_prog/

2. Angeline P. J., Pollack J. Evolutionary Module Acquisition // Proceedings of the Second Annual Conference on Evolutionary Programming. 1993. 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. 1992. www.cs.ucla.edu/~dyer/Papers/AlifeTracker/Alife91Jefferson.html

4. Chambers L. Practical Handbook of Genetic Algorithms. Complex Coding Systems. Volume III. CRC Press, 1999.

5. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. М.: Физматлит, 2006.

6. Koza J. R. Genetic programming: on the programming of computers by means of natural selection. MIT Press, 1992.

7. De Jong K. An analysis of the behavior of a class of genetic adaptive systems. PhD thesis. Univ. Michigan. Ann Arbor, 1975.
8. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002.
9. Рассел С., Норвиг П. Искусственный интеллект: современный подход. М.: Вильямс, 2006.
10. SWITCH-технология. Википедия. http://ru.wikipedia.org/wiki/Switch-%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F