Санкт-Петербургский государственный университет
информационных технологий, механики и оптики
Факультет информационных технологий и программирования
Кафедра «Компьютерные технологии»

А. А. Чебатуркин

Отчет по лабораторной работе «Построение управляющих автоматов с помощью генетических алгоритмов»

Вариант № 5

Санкт-Петербург
2009

Оглавление

Введение

В данной лабораторной работе изучается применение генетических алгоритмов для генерации конечных автоматов. В качестве примера взята задача об «Умном муравье». Результатом работы является автомат Мили, построенный с помощью генетического алгоритма и представляющий логику муравья.

При выполнении лабораторной работы использовалась программа «Виртуальная лаборатория» [1], написанная студентами кафедры «Компьютерные технологии» СПбГУ ИТМО и позволяющая реализовывать генетические алгоритмы и особи для них в виде подключаемых модулей.

1. Постановка задачи

Задача данной лабораторной работы – построить автомат Мили, близкий к оптимальному, решающий задачу об «Умном муравье» [2]. Оптимальность заключается в том, что автомат должен иметь минимальное число состояний, и муравей, управляемый данным автоматом, должен съесть как можно больше еды, выполнив при этом как можно меньше шагов.

1.1. Автомат Мили

Автомат Мили – это конечный автомат, генерирующий выходные воздействия в зависимости от текущего состояния и входного воздействия. Пример диаграммы переходов автомата приведен на рис. 1.

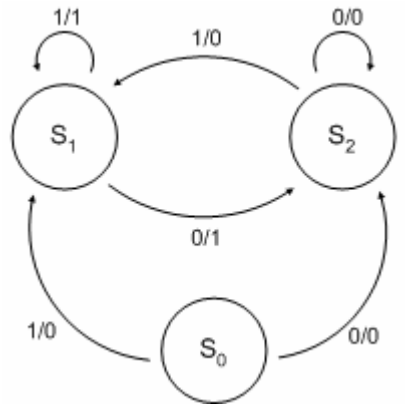

Рис. 1

Как видно из приведенной диаграммы, над каждой дугой расположена пара значений – входное и выходное воздействия. При этом выходное воздействие зависит не только от состояния, в котором находится автомат, но и от входного воздействия.

1.2. Задача об «Умном муравье»

В задаче об «Умном муравье» рассматривается поле, состоящее из клеток. Поле имеет размеры 32x32 клеток и располагается на поверхности тора. Некоторые клетки поля пусты, некоторые содержат по одному яблоку. Всего на поле 89 яблок. Муравей начинает свое движение из клетки, помеченной как «`Start`».

За один ход муравей может определить, есть ли в клетке перед ним яблоко, и выполнить одно из следующих действий:

- повернуть направо;
- повернуть налево;
- сделать шаг вперед, и если в новой клетке есть яблоко, то съесть его;
- ничего не делать.

Максимальное число ходов – 200. Цель работы – создать муравья с фиксированным числом состояний, который съест как можно больше яблок.

2. Реализация

Виртуальная лаборатория состоит из ядра и подключаемых модулей. Для решения поставленной задачи требуется создать два модуля. Первый из них – модуль генетического алгоритма, использующий клеточный алгоритм (Cellular Genetic Algorithm [3]). Второй модуль должен реализовывать «особь» – конечный автомат Мили, решающий задачу об «Умном муравье». Для «особи» необходимо реализовать операции мутации и скрещивания.

2.3. Представление автоматов

Каждый автомат представляется в виде битовой строки, реализованной классом `BitSet` из стандартной библиотеки *Java*. Пример задания автомата из N состояний представлен на рис. 2.

Рис. 2

В описании каждого состояния для каждого из двух входных воздействий («есть еда», «нет еды») закодировано сначала состояние, в которое будет выполняться переход, а затем действие при переходе. На рис. 3 приведен пример кодирования переходов из состояния. В данном случае при входном воздействии «есть еда» автомат переходит в четвертое состояние, совершая при этом поворот налево (действие L), при входном воздействии «нет еды» выполняется переход в седьмое состояние, совершая шаг вперед (действие M).

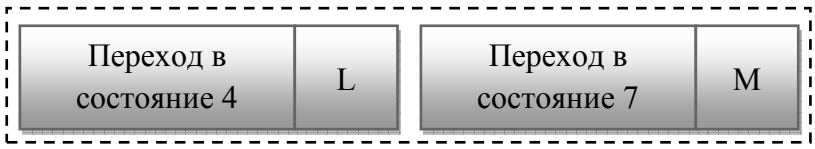

Рис. 3

Выходное воздействие, совершаемое автоматом при переходе, кодируется двумя битами:

- 00 – ничего не делать (N);
- 01 – поворот налево (L);
- 11 – поворот направо (R);
- 10 – шаг вперед (M).

Входное воздействие – это содержимое клетки поля, которое видит муравей.

- F – впереди пусто;
- T – впереди есть еда;

Состояния конечного автомата нумеруются, начиная с нуля. Число бит для представления номера состояния вычисляется как наименьшая степень двух, большая или равная числу состояний в автомате.

2.4. Метод скрещивания

Оператор скрещивания получает на вход две особи и выдает также две особи. Число состояний в автоматах фиксировано и одинаково для всех особей. Процесс скрещивания происходит следующим образом:

1. Создаем два новых автомата с тем же числом состояний в каждом.
2. Случайным образом либо первый сын получает номер начального состояния от первого родителя, второй – от второго, либо первый – от второго родителя, второй – от первого.
3. Рассмотрим состояние с номером **i** в каждом родителе. Для каждого состояния есть два элемента **Transition** (индексы **[i][0]** и **[i][1]**). Таким образом, от двух родителей получаем четыре элемента **Transition**: **p10** – нулевой элемент первого родителя, **p11** – первый элемент первого родителя, **p20** – нулевой элемент второго родителя, **p21** – первый элемент второго родителя. Аналогично для того же **i** для детей имеем: **s10, s11, s20, s21**. Теперь выбираем случайно один набор из четырех: **(p10, p01, p00, p11) (p00, p11, p10, p01) (p10, p11, p00, p01) (p00, p01, p10, p11)**, и присваиваем **s00, s01, s10, s11** этим набором.
4. Повторяем шаг 3 для всех состояний.

2.5. Метод мутации

С заданной вероятностью для случайно выбранного состояния оператор мутации изменяет действие при переходе в другое состояние, также определяемое случайным образом. При мутации с вероятностью 10% также может измениться начальное состояние.

2.6. Метод генерации очередного поколения

Генерация нового поколения осуществляется с использованием клеточного генетического алгоритма.

Суть алгоритма состоит в следующем – пусть имеется замкнутая с обоих концов сетка (тор) в каждой клетке которой размещена особь. Каждая особь может взаимодействовать только с одним из четырех своих соседей (верхняя, нижняя, правая и левая клетки относительно данной).

В ходе процесса генерации очередного поколения, для каждой особи выбирается наиболее приспособленный (с наибольшим значением fitness-функции) из ее соседей. Далее осуществляется скрещивание этих двух особей, в результате которого либо один из детей помещается в клетку родителя, либо родитель остается на месте.

2.7. Вычисление функции приспособленности

В качестве функции приспособленности (*Fitness-функции*) будем использовать выражение вида:

$$\texttt{Fitness} = \texttt{Apples} - \texttt{Steps} / 200,$$

где `Apples` – число съеденных муравьем яблок за 200 шагов, `Steps` – ход, на котором муравей последний раз съел яблоко.

Эмуляция действий муравья выполняется до тех пора пока он либо не съест все яблоки на поле, либо не превысит максимально допустимое число шагов.

3. Результаты работы плагина

В результате работы генетического алгоритма был построен автомат Мили, состоящий из пяти состояний. Значение функции приспособленности для данного автомата равно 82.02. Из приведенного выше соотношения следует, что муравей съедает 83 яблока за 196 ходов.

3.1. Граф переходов

На рис. 4 представлен граф переходов автомата, описанного выше. Граф переходов получен с помощью визуализатора «Виртуальной лаборатории для генерации автоматов с помощью генетических алгоритмов» [1].

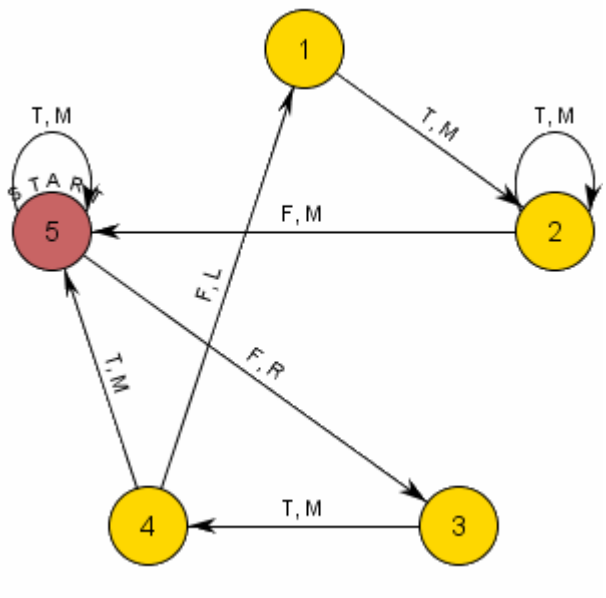

Рис. 4

Вершины графа обозначают состояния автомата. Ребра графа – переходы между состояниями. На ребрах написано входное и выходное воздействия при переходе.

3.2. График максимального значения функции приспособленности

На рис. 5 приведен график зависимости максимального значения функции приспособленности среди особей поколения.

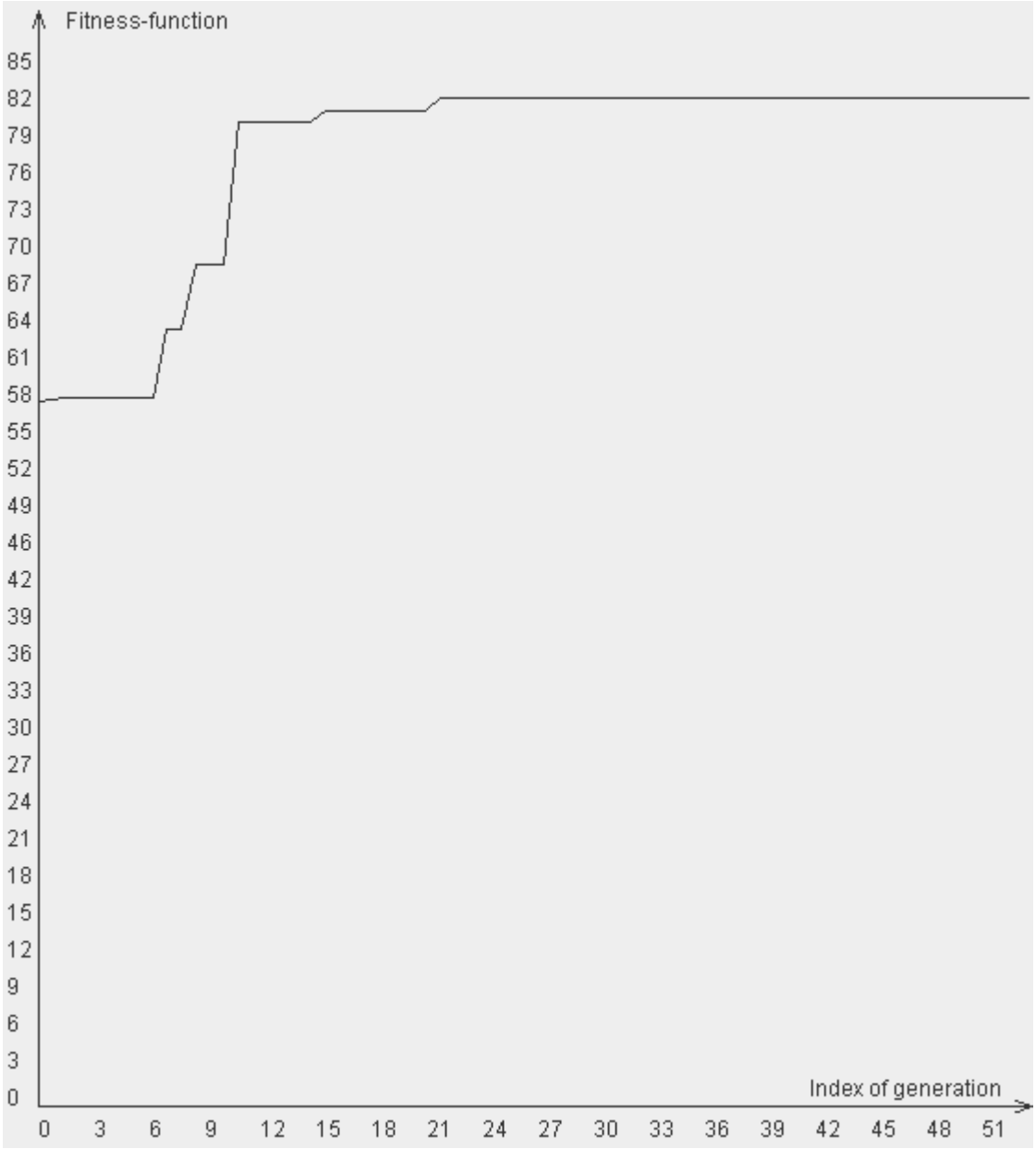

Рис. 5

Из графика видно, что выбранный генетический алгоритм быстро сходится, так как уже после 20 поколений максимальное значение функции приспособленности перестает улучшаться.

3.3. График среднего значения функции приспособленности

На рис. 6 приведен график зависимости среднего значения функции приспособленности среди особей поколения.

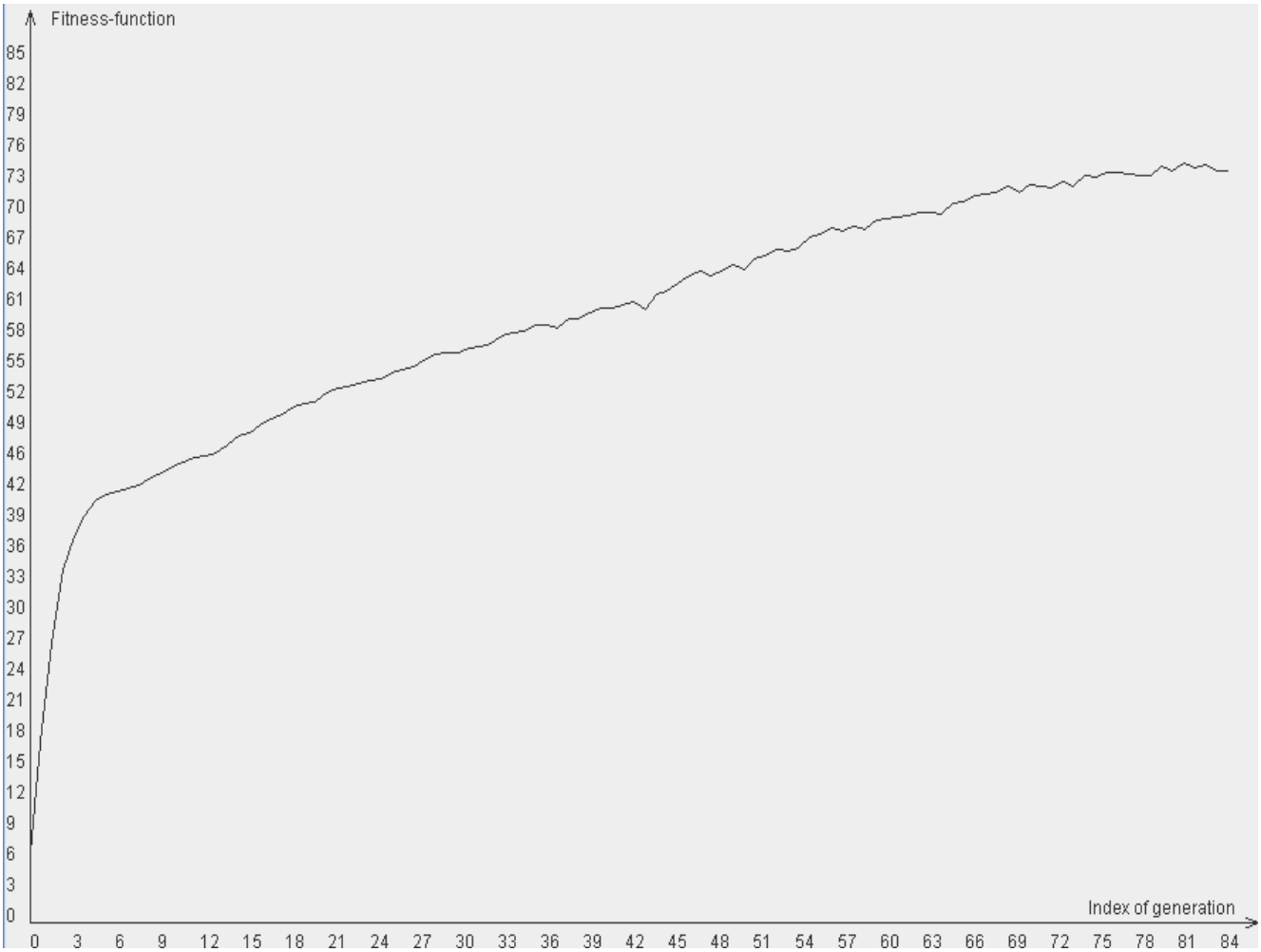

Рис. 6

Из графика следует, что средняя приспособленность особей в каждом поколении стремится к максимальному значению функции приспособленности.

Заключение

Результаты лабораторной работы показали, что используемые методы достаточно эффективны для решения рассматриваемой задачи. При этом, предлагаемый алгоритм генерирует автомат с пятью состояниями и функцией приспособленности такими же, как и при полном переборе. Заметим, что известен автомат с пятью состояниями, построенный для этой же задачи вручную, который управляет муравьем, съедающим 81 яблоко [2].

Источники

1. Инструкция по созданию plugin'ов к виртуальной лаборатории. http://svn2.assembla.com/svn/not_instrumental_tool/docs/pdf/interface_manual.pdf

2. *Бедный Ю.Д., Шалыто А.А.* Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». http://is.ifmo.ru/works/_ant.pdf

3. *Яминов Б.* Генетические алгоритмы. http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005

Приложение. Конфигурационные файлы

Для сборки модулей виртуальной лаборатории был создан Ant-скрипт, который находится в файле `build.xml`. Файл `build.properties` содержит параметры для этого скрипта.

Модуль в «особи»:

`automaton.conf` – конфигурация задания;
`statesNumber` – число состояний автомата;
`plugin.properties` – конфигурация модуля.

Модуль генетического алгоритма:

`cellularGA.conf` – конфигурация клеточного генетического алгоритма;
`gridWidth` – ширина тора;
`gridHeight` – высота тора;
`probabilityMutation` – вероятность мутации;
`plugin.properties` – конфигурация модуля.