Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Факультет информационных технологий и программирования

Кафедра «Компьютерные технологии»

Т. М. Инюшин

Отчет по лабораторной работе «Использование генетических алгоритмов для построения управляющих автоматов»

Вариант №22

Санкт-Петербург
2009

Введение

Цель лабораторной работы – применение генетических алгоритмов для построения конечных автоматов. В качестве примера рассматривается построение конечного автомата Мура, решающего задачу об «Умном муравье-3». 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.

Постановка задачи

Построить с помощью генетических алгоритмов конечный автомат Мура, решающий задачу об «Умном муравье-3». Использовать представление автоматов с помощью графов переходов и сокращенных таблиц. Способ скрещивания выбрать самостоятельно. Использовать островной генетический алгоритм и метод рулетки для генерации очередного поколения.

Автомат Мура

Задача решается с помощью генетического алгоритма, который строит конечный автомат Мура с заранее заданным числом состояний. Автомат Мура – совокупность пяти объектов:

$$A = (S, X, Y, \delta, \mu),$$

где S – множество состояний, X – множество входных воздействий, Y – множество выходных воздействий, $\delta : S \times X \to S$, – функция переходов, $\mu : S \to Y$ – функция выходов. Часто фиксируют также начальное состояние $s \in S$.

Задача об «Умном муравье-3»

Используется двумерный тор размером 32 на 32 клетки. В каждой клетке поля расположено яблоко с заранее заданной вероятностью μ. Изначально муравей находится в клетке с координатами (0, 0) и смотрит вправо. Задача – за 200 ходов съесть максимальное число яблок. При этом перед собой он видит восемь клеток поля (2^8 = 256 входных воздействий).

Реализация

Виртуальная лаборатория состоит из ядра и подключаемых модулей. Для решения поставленной задачи требуется создать два модуля. Первый модуль – модуль генетического алгоритма, использующий островной генетический алгоритм и метод рулетки для генерации очередного поколения. Второй модуль должен реализовывать «особь» – конечный автомат, решающий задачу об «Умном муравье». Для «особи» необходимо реализовать операции мутации и скрещивания.

Представление автоматов в программе

В программе автомат представлен в виде набора состояний, для каждого из которых хранится таблица соответствий вида:

```
[входное воздействие > (действие + новое состояние)],
```

которая определяет поведение автомата. Также хранится номер начального состояния. Автомат представлен в виде сокращенных таблиц, каждая из которых является полной для одного состояния. Число элементов таблицы ограничивается некоторой константой, задаваемой в конфигурационном файле.

Метод скрещивания

Оператор скрещивания получает на вход две особи и выдает также две особи. При этом используются сокращенные таблицы первого и второго родителей. Строки этих таблиц, в которых «действия» совпадают, случайным образом распределяются между детьми. Остальные части таблиц дописываются уже к определенному ребенку, но не случайным образом.

Метод мутации

При мутации с некоторой вероятностью стартовым состоянием становится случайное состояние автомата. Затем изменяется либо выходное воздействие, связанное с этим состоянием, либо переход по какому-либо входному воздействию.

Модель генетического алгоритма

Островная модель – это модель параллельного генетического алгоритма. Она заключается в следующем: пусть имеется n процессов и k особей. Разобьем их на n подпопуляций по k/n особей. Каждая их них будет развиваться отдельно с помощью некого генетического алгоритма. Таким образом, можно сказать, что особи расселены по n изолированным островам.

Изредка (например, каждые p поколений) процессы (или острова) будут обмениваться несколькими хорошими особями (генетическим материалом). Это называется миграцией. Так как населенность островов обычно невелика, подпопуляции будут склонны к преждевременной сходимости. Поэтому важно

правильно установить частоту миграции. Миграция слишком большого числа особей приведет к смешению всех подпопуляций, и тогда островная модель будет несильно отличаться от обычного генетического алгоритма (с одним островом). Если же миграция будет слишком редкой, то она не сможет предотвратить преждевременного схождения подпопуляций. На рис. 1 представлена схема работы островного генетического алгоритма.

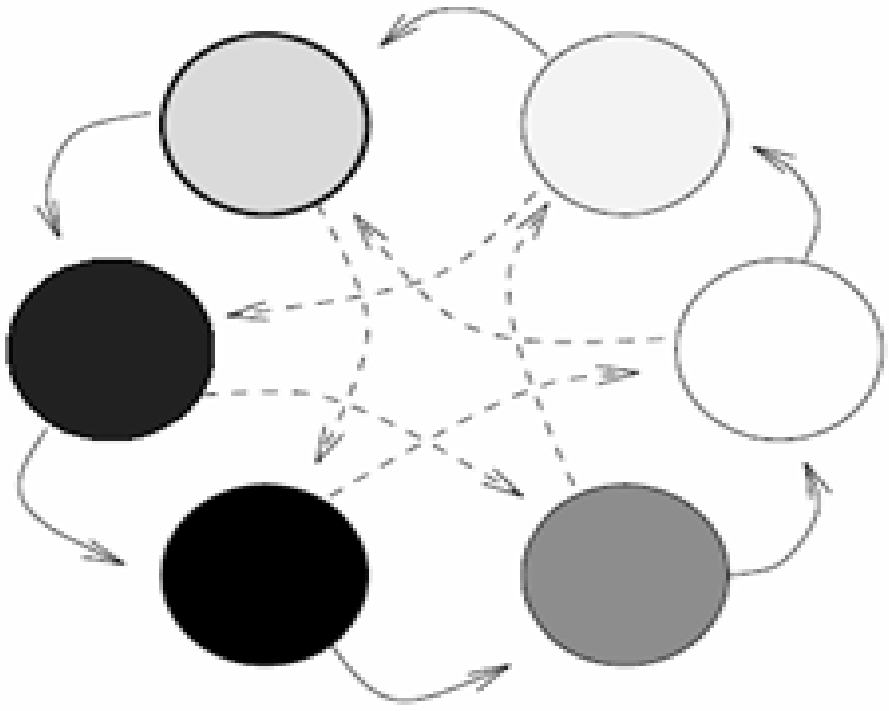

Рис. 1. Схема островного генетического алгоритма

Метод генерации очередного поколения

Начальное поколение состоит из фиксированного числа случайно сгенерированных автоматов. Все автоматы в поколении имеют одинаковое наперед заданное число состояний.

Для генерации очередного поколения используется островной генетический алгоритм и метод рулетки.

Шаг алгоритма состоит из трех стадий: генерация промежуточной популяции путем отбора текущего поколения, скрещивание особей промежуточной популяции путем скрещивания (кроссовера). Это приводит к формированию нового поколения, далее происходит мутация нового поколения.

Промежуточная популяция – это набор особей, которые получили право размножаться. Приспособленные особи могут быть записаны в эти популяции несколько раз. «Плохие» особи с большой вероятностью туда не попадут.

В генетическом алгоритме с одним островом вероятность каждой особи попасть в промежуточную популяцию пропорциональна ее приспособленности – работает пропорциональный отбор. В данной работе отбор реализован следующим образом: пусть особи располагаются на колесе рулетки, так что размер сектора каждой особи пропорционален ее приспособленности. В начале промежуточная популяция пуста. `N` раз запуская рулетку, выберем требуемое число особей для записи в промежуточную популяцию. Ни одна выбранная особь не удаляется с рулетки.

Рулеточный алгоритм подставляется в качестве базового генетического алгоритма для островной модели. В результате получается модель, которая используется в данном модуле.

Способ вычисления функции приспособленности

Функция приспособленности вычисляется следующим образом: полученный автомат используется для управления муравьем, подсчитывается число съеденных яблок и выдается в качестве ответа.

Результаты работы

В результате работы рассматриваемого генетического алгоритма был получен автомат, который решает задачу об «Умном муравье-3». Автомат содержит пять состояний, значение функции приспособленности равно 58.

График максимального значения функции приспособленности

На рис. 2 приведен график зависимости максимального значения функции приспособленности среди особей поколения.

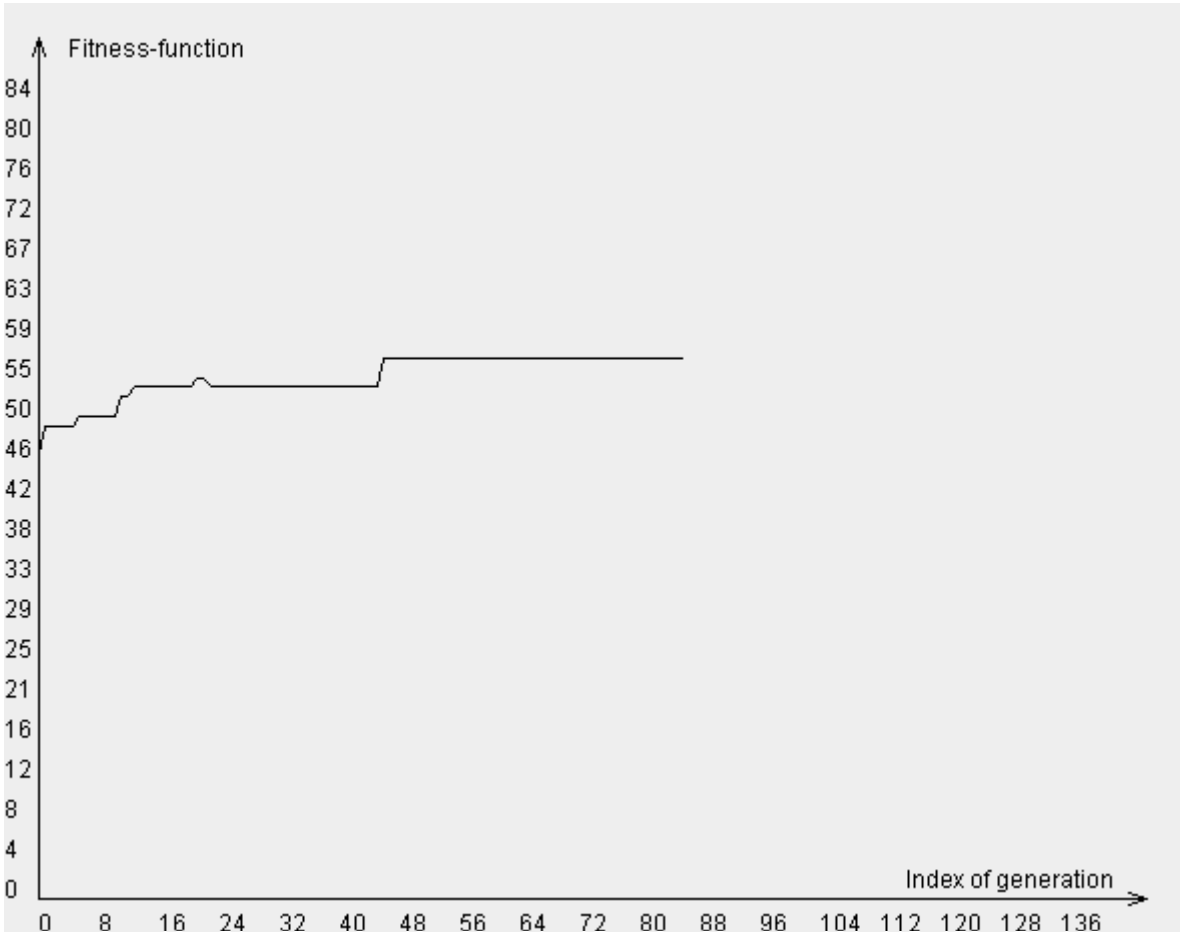

Рис. 2. График зависимости максимального значения функции приспособленности среди особей поколения

График среднего значения функции приспособленности

На рис. 3 приведен график зависимости максимального значения функции приспособленности среди особей поколения.

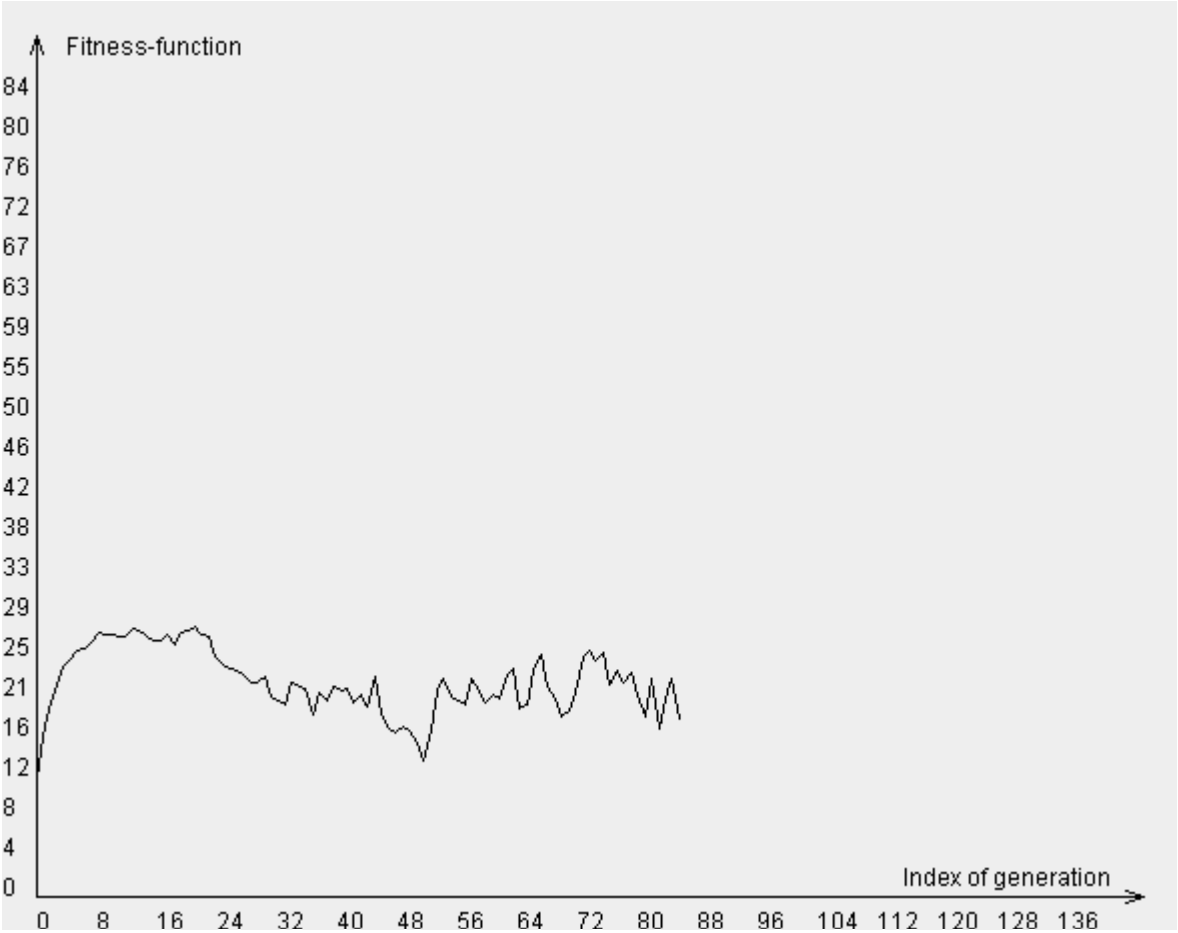

Рис. 3. График зависимости среднего значения функции приспособленности среди особей поколения

Таблица переходов полученного автомата

State count: 5 Initial state: 3 fitness: 58.0

STATE (0)	STATE (1)	STATE (2)	STATE (3)	STATE (4)
135(M, 3)	233(R, 0)	139(M, 0)	90(M, 2)	93(M, 4)
182(M, 3)	183(M, 4)	137(R, 2)	40(M, 0)	92(L, 2)
87(R, 1)	180(L, 3)	88(M, 3)	181(L, 3)	186(L, 3)
134(L, 1)	37(M, 2)	226(M, 2)	180(R, 4)	45(R, 4)
228(R, 3)	222(M, 3)	37(R, 3)	83(M, 0)	40(M, 3)
226(M, 2)	127(M, 4)	35(M, 2)	80(M, 2)	228(L, 1)
129(M, 3)	78(R, 4)	33(M, 1)	221(R, 0)	224(R, 0)
221(M, 1)	124(L, 2)	174(L, 3)	173(L, 0)	34(M, 3)
32(L, 2)	218(R, 4)	32(L, 2)	74(R, 2)	33(R, 1)
220(L, 3)	28(M, 4)	220(M, 0)	120(M, 3)	80(R, 2)
78(R, 2)	74(L, 0)	218(L, 0)	166(L, 0)	124(L, 1)
29(L, 2)	121(L, 2)	75(M, 1)	71(R, 2)	123(M, 0)
169(L, 1)	73(L, 0)	214(L, 3)	164(M, 2)	23(R, 3)
214(L, 2)	165(L, 4)	24(M, 0)	163(R, 3)	22(R, 4)
213(L, 2)	210(R, 4)	118(M, 4)	210(L, 3)	161(L, 2)

```
118(R, 2)    209(M, 1)    162(R, 4)    68(L, 4)     158(M, 3)
163(R, 0)    115(M, 4)    67(R, 2)     18(L, 3)     110(M, 0)
162(R, 3)    68(R, 4)     255(R, 3)    253(L, 3)    63(M, 3)
20(L, 2)     206(R, 2)    161(M, 0)    252(M, 0)    109(L, 2)
160(L, 3)    205(L, 0)    158(R, 3)    16(L, 4)     202(R, 0)
110(R, 4)    16(M, 3)     157(L, 0)    63(M, 3)     154(R, 2)
16(R, 4)     62(L, 4)     16(L, 0)     155(M, 2)    106(L, 2)
203(L, 0)    248(M, 3)    110(R, 2)    152(M, 2)    57(R, 1)
15(R, 4)     107(R, 1)    156(L, 4)    104(M, 3)    9(R, 1)
12(M, 1)     245(L, 0)    61(M, 4)     245(L, 2)    102(M, 0)
103(L, 2)    149(R, 0)    104(R, 2)    242(M, 1)    8(R, 1)
242(L, 3)    148(L, 2)    242(M, 1)    100(L, 2)    53(L, 4)
53(M, 4)     146(R, 4)    240(L, 0)    96(L, 3)     52(R, 3)
4(R, 1)      98(M, 1)     144(L, 1)    143(R, 4)    237(R, 2)
145(L, 3)    49(R, 3)     190(M, 3)    48(R, 0)     48(M, 4)
190(M, 4)    48(L, 4)     94(R, 3)     141(M, 3)    188(L, 3)
141(R, 4)    235(R, 0)    235(R, 2)    0(M, 1)      235(L, 3)
```

Ячейки имеют вид ***входное_воздействие (действие, номер_нового_состояния)***. Следует понимать, что входное воздействие – восьмибитное число, характеризующее расположение еды в области видимости муравья.

Конфигурационные файлы

Для сборки модулей виртуальной лаборатории был использован `ant-скрипт`, который находится в файле `build.xml,` и пакет `NetBeans 6.5.`

Модуль «особи»:

`automaton.conf` — конфигурация задания:

- `Count.states` — число состояний автомата;
- `Count.attempts` — число попыток муравья;
- `Mu` — вероятность того, что еда есть в некоторой клетке;
- `Count.significantpredicates` — максимальное число значимых входных данных для автомата.

Настройки модуля генетического алгоритма совпадают с таковыми для стандартного островного алгоритма, реализованного в виртуальной лаборатории.

Заключение

Результаты лабораторной работы показали, что используемые методы достаточно эффективны для построения автомата Мура, заданного сокращенными таблицами, который решает задачу об «Умном муравье-3». Кроме того, как показывают графики функции приспособленности, достаточно неплохие результаты получены уже на первых поколениях особей, что говорит о эффективности представления автомата и островной модели генетического алгоритма. К недостаткам описанного варианта работы можно отнести большое время генерации очередного поколения, но это компенсируется тем, что в результате получен автомат, который как раз наиболее быстро и эффективно выполняет задачу об «Умном муравье-3».

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