Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Факультет информационных технологий и программирования

Кафедра «Компьютерные технологии»

Ю.Ю. Коноплев

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

Вариант № 25

Санкт-Петербург
2009

Оглавление

Введение

В данной лабораторной работе исследуется применение генетических алгоритмов для построения конечных управляющих автоматов. Рассматривается задача об «Умном муравье – 3». С помощью метода рулетки генерируется конечный автомат Мура, который описывает поведение муравья.

Для эмуляции действий муравья студентами кафедры «Компьютерные технологии» СПбГУ ИТМО была создана программа «Виртуальная лаборатория» [1]. Решением поставленной задачи является плагин для данной лаборатории, который описывает представление муравья в виде автомата, а также реализует генетический алгоритм.

1. Постановка задачи

Цель данной лабораторной работы – построить такой автомат Мили, что муравей, поведение которого им описывается, съедает как можно больше еды на поле.

1.1. Задача об «Умном муравье - 3»

Приведем описание задачи об «Умном муравье – 3» [2]. Игра происходит на поверхности тора размером 32 на 32 клетки (рис. 1). В некоторых клетках (обозначены на рис. 1 черным цветом) находится еда. Клетки поля, в которых нет еды, обозначены серым цветом. Для каждой клетки вероятность появления еды в ней равна заранее предопределенной величине μ.

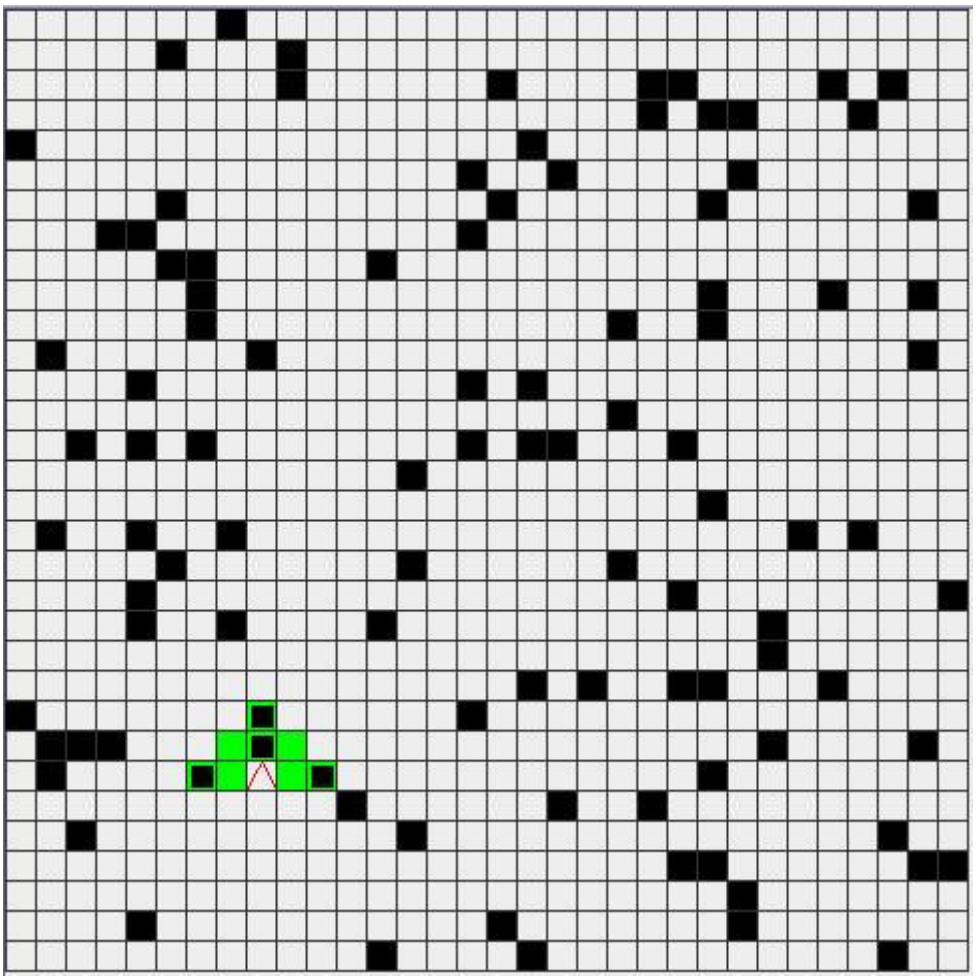

Рис. 1. Игровое поле

В левой верхней клетке в начале игры находится муравей. На рис.1 он обозначен красным треугольником, который также показывает направление его взора. 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, в каких клетках из 8 находящихся рядом с ним и помеченных зеленым цветом находится еда. Таким образом, в каждой клетке для муравья существует $2^8 = 256$ входных воздействий. За один игровой ход муравей может совершить одно из трех действий:

- сделать шаг вперед, съедая при этом еду, если она там находится (M – move);

- повернуть налево (L – left);
- повернуть направо (R – right).

Съеденная муравьем еда не восполняется, муравей жив на протяжении всей игры, еда не является необходимым ресурсом для его жизни. Муравей может ходить по любым клеткам поля. Игра длится 200 ходов, на каждом из которых муравей совершает одно из четырех действий. По окончании игры подсчитывается количество еды, съеденной муравьем. Это значение и есть результат игры.

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

1.2. Автомат Мили

Автомат Мили – это конечный автомат, генерирующий свои выходные воздействия в зависимости от текущего состояния и входного воздействия. Пример диаграммы переходов автомата Мили приведен на рис.2.

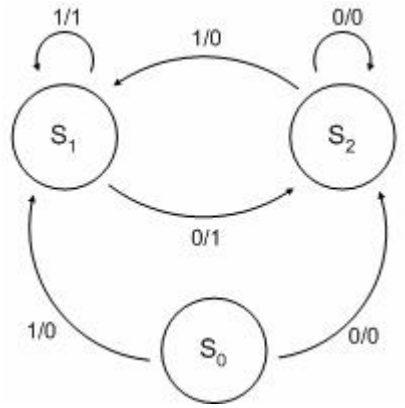

Рис.2

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

2. Генетический алгоритм

Разработанный алгоритм генетического программирования [3] состоит из пяти частей:

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

2.1. Представление особи

Каждая особь представляет собой некоторый конечный автомат Мили с заданным числом состояний, задаваемых деревьями решений [4].

В программе автомат представлен массивом состояний `State`, каждый из которых хранит 256 переходов в другие состояния представленные в виде соответствия **{**входной сигнал**;** реакция (`L,R,M`)**;** номер состояние в которое перейти**}**. Также хранится номер стартового состояния.

2.2. Создание начального поколения

Начальное поколение состоит из фиксированного числа особей (по умолчанию – 200) случайно сгенерированных автоматов Мили с заранее определенным числом состояний.

2.3. Мутация

При мутации стартовым состоянием становится случайное из автомата, затем случайно выбирается один из двух равновероятных вариантов:

- изменение действий, совершаемого муравьем, который находится в новом стартовом состоянии – случайно и равновероятно выбирается выходной сигнал из множества Y.
- изменение состояния, в которое ведет переход – случайно и равновероятно выбирается переход. После этого состояние, в которое ведет переход, заменяется на случайно выбранное состояние.

Также в генетическом алгоритме предусмотрена операция «большой мутации», которая осуществляет генерацию полностью новых случайных автоматов. Она может быть использована, когда в популяции поведение муравьев почти идентично.

2.4. Скрещивание (Кроссовер)

Операция скрещивания получает на вход две особи и выдает также две особи. Процесс скрещивания происходит следующим образом. Создаются два потомка, эквивалентные родителям. Далее для каждого состояния потомка с вероятностью 50% производится обмен входными и выходными воздействиями. Таким образом, состояния родителей перемешиваются в следующем поколении.

2.5. Отбор особей для формирования следующего поколения

Генерация очередного поколения осуществляется с помощью метода «рулетки»: всем автоматам ставятся в соответствие отрезки длины, равной значению фитнес-функции, затем выбирается случайная точка, которая будет принадлежать одному из отрезков, соответствующий автомат добавляется в следующее поколение. Выбор осуществляется

столько раз, сколько особей должно быть в поколении. Затем производится скрещивание пар автоматов, и, наконец, мутация каждого из получившихся детей.

2.6. Вычисление функции приспособленности (фитнес-функции)

Функция приспособленности вычисляется следующим образом: проводится эксперимент с участием муравья с поведением, заданным текущим автоматом Мили. С помощью лаборатории эмулируются его перемещения в зависимости от расположения еды: он проходит по полю несколько раз, каждый раз подсчитывается число съеденных яблок. Полученные результаты усредняются (сумма делится на число попыток) и выдаются в качестве ответа.

2.7. Настраиваемые параметры алгоритма

Следующие параметры алгоритма генетического программирования могут быть изменены:

- размер поколения;
- число состояний;
- число экспериментов над каждым муравьем;
- вероятность мутации;
- вероятность появления еды в каждой клетке.

3. РЕЗУЛЬТАТЫ

Вычислительные эксперименты проводились с различными значениями параметров алгоритма. Далее приводится описание графа переходов для автомата Мили с пятью состояниями (первое является начальным), значение фитнесс-функции для лучшего муравья полученного в 67513 поколении равно 52. Для каждого состояния показано выходное действие, затем таблица переходов по всем 256 входным воздействиям.

Лучшие особи представлены на рис.3. Слева значение фитнесс-функции для лучшей особи, полученной для поколения с номером находящимся справа.

Fitness	Generation
23	7
27	9
29	16
32	18
34	28
38	31
39	70
40	82
41	195
42	971
43	1694
44	3708
45	6288
46	9513
47	11136
49	19441
50	41075
51	51790
52	67513

Рис.3

Вероятность появления еды для каждой клетки равна 0.1 – математическое ожидание количества еды на поле равно 102.4, вероятность мутации особи после скрещивания равна 0.02.

Пять состояний (начальное = 1); fitness-функция=52.0

> Состояние № 1

3 M, 1 L, 4 M, 5 M, 3 R, 3 L, 4 L, 5 M, 1 L, 3 M, 2 R, 2 L, 5 L, 2 R, 3 R, 5 L, 5 L, 5 M, 5 R, 5 M, 4 L, 3 R, 4 R, 1 L, 3 R, 3 M, 1 L, 4 M, 4 M, 3 L, 2 M, 5 M, 4 M, 5 R, 2 R, 4 M, 5 R, 4 L, 3 R, 5 R, 2 R, 2 R, 4 L, 1 M, 4 L, 5 R, 1 M, 2 L, 5 R, 4 L, 2 L, 4 L, 3 M, 2 R, 4 R, 5 R, 4 M, 4 M, 1 M, 4 M, 4 L, 5 L, 4 R, 3 M, 2 R, 2 L, 4 R, 5 R, 2 R, 4 M, 3 L, 3 M, 2 M, 3 M, 5 L, 1 M, 1 M, 1 R, 1 R, 4 R, 1 L, 4 R, 5 L, 5 L, 4 L, 5 R, 3 L, 5 L, 4 M, 4 R, 1 L, 5 L, 2 R, 4 R, 1 M, 4 M, 2 R, 5 R, 1 R, 3 R, 4 M, 2 L, 5 R, 3 L, 1 M, 5 M, 2 L, 2 L, 1 L, 3 M, 1 M, 5 M, 5 R, 3 R, 4 R, 2 R, 2 L, 3 M, 5 L, 2 R, 5 R, 1 M, 4 L, 4 R, 4 L, 3 L, 3 R, 3 L, 3 M, 5 M, 1 M, 1 R, 1 M, 3 R, 4 R, 3 M, 2 L, 2 R, 3 R, 5 M, 4 L, 1 R, 3 R, 2 R, 1 R, 5 L, 5 R, 5 R, 2 L, 4 R, 2 L, 5 L, 4 L, 3 L, 1 R, 5 M, 3 M, 4 L, 4 R, 4 M, 4 L, 5 L, 2 L, 3 M, 2 R, 2 M, 5 L, 3 M, 5 R, 5 M, 5 M, 1 L, 2 L, 4 M, 2 R, 5 L, 4 L, 5 M, 5 M, 5 L, 2 L, 1 L, 2 R, 2 M, 5 R, 4 M, 5 L, 5 R, 3 L, 2 M, 5 M, 5 M, 4 R, 1 R, 3 R, 4 L, 1 M, 2 L, 4 L, 1 R, 2 R, 2 R, 1 M, 4 R, 2 L, 1 M, 1 R, 5 R, 1 R, 4 L, 1 M, 2 L, 4 R, 3 L, 2 L, 4 L, 5 L, 1 M, 4 L, 4 L, 4 R, 4 R, 5 L, 1 R, 5 R, 3 R, 3 L, 5 R, 2 M, 3 M, 4 R, 3 L, 2 R, 5 L, 2 M, 4 M, 1 L, 3 M, 5 M, 4 R, 5 M, 2 M, 3 M, 3 L, 2 M, 1 M, 2 L, 3 L, 5 R, 2 M, 1 M, 1 M, 5 M, 3 M, 2 L, 2 R

> Состояние № 2

1 L, 4 M, 4 L, 2 M, 3 R, 2 R, 2 M, 1 L, 2 L, 4 L, 1 L, 1 M, 5 M, 4 M, 5 L, 1 M, 1 L, 2 M, 2 M, 2 M, 4 M, 2 M, 4 R, 2 M, 2 M, 4 R, 1 R, 1 M, 3 L, 4 L, 3 R, 1 L, 5 L, 1 L, 3 M, 4 M, 4 M, 5 L, 4 L, 3 R, 2 L, 5 L, 1 R, 3 L, 3 L, 4 R, 3 L, 2 R, 2 M, 4 R, 1 M, 5 R, 3 L, 1 R, 5 L, 5 M, 5 R, 5 M, 3 L, 5 L, 3 L, 2 R, 5 R, 5 R, 4 M, 2 M, 2 L, 1 R, 1 R, 4 R, 4 M, 1 R, 1 R, 1 R, 2 L, 1 R, 1 R, 2 M, 1 L, 5 M, 4 R, 3 L, 4 L, 2 L, 5 M, 4 R, 4 R, 5 L, 4 M, 4 L, 3 L, 3 L, 4 M, 5 R, 2 M, 2 M, 3 M, 3 M, 2 L, 2 L, 5 L, 4 L, 5 M, 3 M, 2 R, 3 L, 2 M, 1 L, 5 L, 5 M, 4 L, 5 L, 1 L, 1 L, 5 L, 5 M, 1 R, 2 L, 4 M, 5 R, 3 M, 1 L, 1 M, 4 L, 3 R, 5 M, 5 L, 2 M, 3 M, 3 L, 5 M, 5 R, 1 R, 2 M, 3 R, 4 L, 2 M, 2 M, 2 R, 4 M, 5 M, 5 L, 5 L, 2 M, 2 M, 4 M, 5 M, 1 M, 1 R, 2 M, 2 M, 2 M, 3 L, 2 R, 1 L, 4 M, 5 R, 4 L, 5 L, 4 M, 4 R, 4 L, 1 R, 3 L, 2 L, 5 R, 3 M, 4 R, 3 M, 4 M, 3 M, 3 M, 5 M, 2 R, 4 L, 3 L, 3 L, 2 L, 2 L, 3 R, 4 L, 5 L, 5 R, 3 R, 1 L, 3 R, 2 M, 1 M, 4 M, 4 L, 5 L, 1 M, 4 L, 2 L, 1 M, 4 M, 2 R, 5 M, 5 L, 5 M, 3 M, 3 L, 1 L, 4 L, 2 R, 2 R, 1 R, 1 M, 1 R, 2 M, 2 M, 1 M, 4 L, 4 M, 5 L, 3 M, 5 M, 1 R, 3 M, 1 M, 4 R, 2 L, 1 R, 4 M, 2 M, 5 R, 2 R, 5 L, 2 L, 2 R, 5 L, 5 M, 1 L, 3 M, 4 L, 4 L, 1 L, 1 L, 1 R, 5 R, 4 R, 4 L, 5 R, 4 M, 4 R, 3 M, 2 L, 1 R, 3 L, 2 L, 2 R, 1 M, 3 L, 1 L, 4 R, 3 R

> Состояние № 3

3 M, 5 M, 4 M, 4 M, 3 M, 1 R, 2 R, 2 M, 5 M, 1 M, 4 L, 5 M, 5 M, 2 M, 2 R, 1 L, 2 R, 5 R, 3 L, 2 R, 3 L, 5 M, 5 M, 4 M, 1 R, 2 R, 1 R, 4 M, 5 L, 4 R, 2 L, 5 M, 3 L, 4 M, 3 R, 2 M, 4 M, 2 L, 2 L, 5 R, 3 R, 4 L, 3 M, 2 M, 1 L, 4 L, 5 M, 2 R, 2 M, 1 L, 5 M, 2 L, 2 R, 4 R, 1 L, 3 R, 4 L, 1 R, 5 L, 2 M, 5 M, 1 R, 3 M, 5 L, 4 M, 1 L, 5 R, 5 L, 3 L, 3 M, 5 M, 5 R, 4 R, 1 M, 4 M, 1 M, 2 L, 1 M, 1 M, 3 R, 4 M, 2 M, 5 L, 4 L, 4 M, 5 L, 5 R, 2 M, 3 M, 5 R, 5 R, 3 M, 5 M, 5 M, 3 R, 2 M, 3 R, 5 L, 4 R, 1 M, 2 M, 3 M, 2 L, 5 R, 1 M, 2 L, 2 M, 4 R, 1 M, 1 R, 2 L, 2 M, 1 L, 2 R, 2 L, 4 R, 1 R, 3 M, 2 L, 2 L, 3 M, 3 L, 4 M, 2 R, 3 L, 2 L, 3 M, 2 M, 2 M, 5 M, 2 L, 4 M, 5 R, 3 M, 5 M, 3 L, 2 M, 2 L, 4 R, 5 M, 5 R, 3 M, 4 R, 1 M, 4 M, 2 L, 1 L, 5 L, 1 M, 2 R, 5 R, 4 R, 4 L, 2 R, 5 L, 1 L, 2 L, 1 M, 5 R, 3 R, 5 L, 1 M, 5 M, 1 M, 2 L, 3 L, 3 M, 2 M, 4 M, 1 L, 5 R, 1 L, 5 R, 4 M, 1 R, 1 R, 4 R, 2 L, 4 R, 1 M, 2 L, 2 M, 3 L, 5 M, 1 L, 3 L, 4 M, 4 M, 4 R, 3 R, 1 R, 4 R, 2 L, 3 M, 5 M, 1 M, 3 M, 4 M, 4 L, 2 M, 5 M, 1 L, 1 R, 1 M, 5 M, 5 M, 2 M, 5 R, 3 R, 5 L, 3 L, 4 R, 5 R, 2 M, 4 L, 3 M, 3 M, 3 R, 3 L, 2 M, 4 L, 5 R, 1 L, 3 M, 1 M, 2 L, 1 M, 1 M, 5 R, 4 L, 3 L, 3 M, 1 L, 3 L, 3 M, 4 L, 4 R, 2 M, 5 M, 3 M, 1 R, 2 R, 4 L, 2 R, 3 M, 3 M, 1 L, 5 L, 1 R, 3 L, 4 R, 4 M, 5 R, 1 R, 2 M, 4 R

> Состояние № 4

3 M, 3 L, 5 L, 3 R, 3 M, 3 R, 2 L, 5 M, 3 L, 2 L, 5 L, 2 R, 2 R, 4 R, 5 R, 3 R, 1 M, 2 M, 3 L, 5 L, 1 M, 5 L, 1 M, 4 M, 2 L, 3 M, 3 R, 2 R, 3 L, 2 M, 3 L, 3 L, 2 R, 2 R, 2 L, 4 M, 3 R, 3 R, 4 L, 1 L, 4 R, 2 R, 4 L, 1 R, 3 L, 2 R, 5 R, 4 R, 2 L, 3 L, 5 M, 1 M, 3 M, 5 R, 2 R, 5 M, 1 M, 3 L, 4 L, 5 M, 3 L, 4 L, 1 M, 1 R, 1 M, 3 L, 3 L, 2 L, 4 M, 2 L, 4 M, 5 L, 2 M, 4 R, 3 L, 5 M, 5 R, 1 L, 3 R, 5 L, 3 L, 1 L, 5 M, 5 L, 2 R, 2 L, 2 R, 1 L, 2 L, 2 R, 4 M, 5 M, 2 L, 1 R, 1 M, 2 M, 5 L, 2 M, 4 L, 2 R, 1 R, 1 M, 4 M, 3 M, 3 R, 5 L, 4 M, 1 R, 1 M, 3 L, 1 M, 1 M, 3 R, 1 M, 5 L, 4 R, 5 L, 2 L, 2 M, 1 R, 2 M, 2 M, 3 L, 3 M, 4 L, 1 M, 3 R, 3 L, 3 M, 1 R, 1 M, 1 M, 4 M, 2 R, 4 M, 5 R, 4 M, 5 M, 5 M, 5 L, 4 L, 1 M, 4 R, 2 R, 1 R, 5 L, 4 M, 2 L, 2 L, 1 M, 2 M, 4 M, 3 R, 4 M, 2 M, 1 M, 1 R, 5 L, 1 R, 2 R, 4 M, 2 L, 1 R, 4 L, 4 M, 1 L, 1 R, 2 R, 4 R, 1 R, 5 M, 2 L, 1 L, 4 L, 5 R, 3 L, 4 M, 1 M, 4 R, 4 L, 2 L, 1 L, 4 L, 2 R, 1 R, 5 M, 1 R, 1 M, 2 L, 2 R, 1 R, 1 L, 3 L, 3 R, 5 M, 3 L, 2 L, 3 R, 2 M, 5 R, 4 R, 1 M, 3 L, 4 L, 1 R, 4 M, 1 M, 2 R, 4 L, 3 L, 5 M, 3 M, 4 R, 1 R, 5 L, 1 L, 5 L, 1 L, 2 R, 4 L, 3 R, 2 M, 4 R, 1 M, 5 L, 2 M,

3 M, 3 R, 4 R, 1 R, 1 M, 1 L, 2 M, 2 L, 5 R, 3 R, 5 L, 1 L, 3 M, 2 M, 5 L, 5 M, 2 L, 3 R, 4 R, 4 L, 4 M, 1 R, 1 L, 3 L, 1 R, 4 L, 5 L, 2 R, 5 L

> Состояние № 5

1 M, 4 R, 3 L, 5 R, 3 M, 2 M, 3 L, 2 L, 4 L, 2 M, 4 L, 1 L, 3 M, 3 R, 2 M, 2 L, 1 M, 4 L, 2 M, 3 R, 5 M, 4 L, 3 M, 5 L, 5 L, 5 M, 3 L, 2 R, 4 L, 1 R, 5 L, 3 R, 1 R, 3 R, 4 L, 4 L, 4 L, 2 L, 2 L, 3 L, 4 L, 1 L, 5 R, 5 L, 2 R, 5 M, 4 L, 4 M, 3 R, 5 M, 3 L, 2 L, 2 M, 3 M, 1 M, 4 R, 5 R, 2 M, 2 M, 2 R, 2 L, 1 M, 3 M, 5 L, 1 M, 3 R, 2 M, 5 L, 2 R, 1 R, 3 R, 2 R, 5 L, 2 L, 5 R, 3 R, 4 R, 3 M, 3 L, 3 L, 3 M, 1 M, 5 M, 5 L, 3 L, 1 M, 4 R, 2 L, 1 L, 1 R, 3 R, 2 M, 4 M, 3 R, 4 M, 1 L, 2 R, 4 M, 1 M, 1 L, 2 R, 3 L, 3 M, 3 M, 4 R, 4 R, 3 R, 2 L, 3 R, 5 M, 3 L, 1 L, 3 L, 1 R, 4 M, 3 L, 4 R, 4 L, 1 R, 1 M, 4 R, 4 L, 1 L, 5 R, 4 R, 3 L, 5 R, 5 L, 5 M, 4 R, 4 L, 2 M, 1 M, 2 M, 3 L, 5 L, 4 R, 2 R, 2 L, 3 L, 3 M, 2 R, 1 R, 3 M, 3 M, 2 L, 4 M, 2 L, 5 R, 5 L, 2 R, 4 M, 3 L, 2 L, 4 L, 4 M, 4 L, 2 L, 2 L, 5 R, 4 M, 3 L, 3 L, 5 L, 3 M, 3 M, 4 R, 4 M, 3 R, 1 R, 1 R, 1 R, 4 L, 2 L, 3 M, 3 L, 4 R, 3 L, 2 R, 5 M, 2 R, 5 L, 5 L, 5 M, 5 L, 3 L, 4 R, 2 R, 1 M, 3 L, 3 M, 1 M, 4 M, 1 M, 1 L, 1 R, 2 L, 5 R, 3 M, 4 M, 3 M, 5 L, 5 R, 4 L, 2 M, 4 M, 4 L, 1 L, 5 R, 5 L, 4 M, 4 L, 4 L, 2 M, 5 M, 3 M, 3 M, 5 R, 1 R, 1 R, 5 L, 4 R, 1 L, 2 R, 2 L, 3 R, 4 R, 2 M, 3 L, 2 M, 1 M, 5 L, 3 R, 5 M, 3 M, 5 R, 2 L, 4 M, 1 L, 5 L, 1 L, 1 M, 3 R, 5 L, 3 R, 1 L, 4 R, 1 M, 4 R, 3 R, 1 L, 4 R, 2 R, 2 R, 1 M, 2 R

3.1. График максимального значения фитнес-функции

На рис.4 представлен график зависимости максимального значения фитнес-функции от номера поколения.

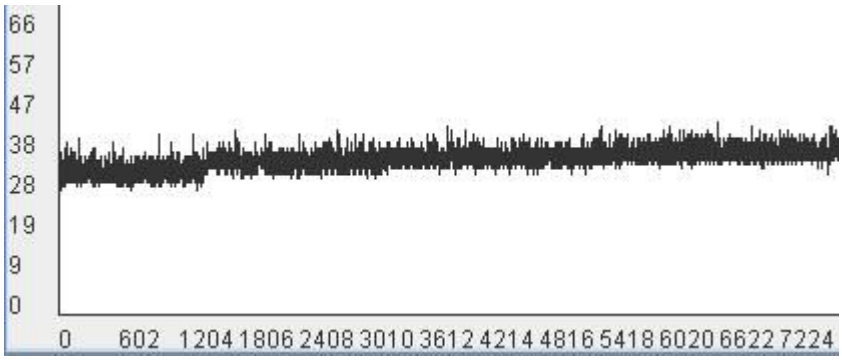

Рис. 4

На рисунке видно, что существует много локальных максимумов. Это свидетельствует о том, что метод рулетки выбирает средние особи. При этом самые лучшие могут не попасть в следующее поколение.

3.2. График среднего значения фитнес-функции

На рис.5 представлен график зависимости усредненного значения фитнес-функции от номера поколения.

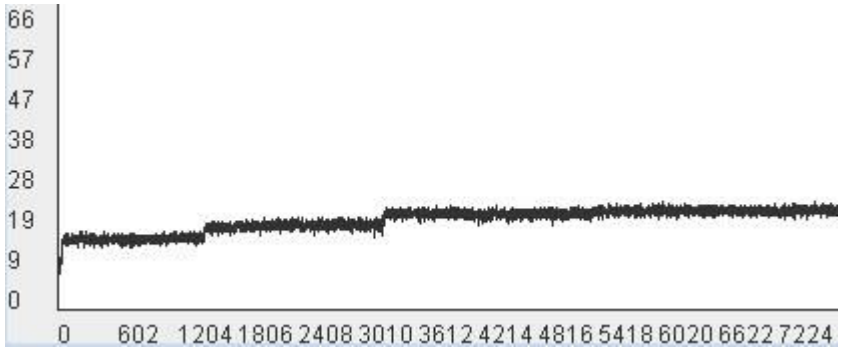

Рис. 5

Заключение

Использование метода рулетки плохо подходит для выделения наилучших особей при скрещивании. Такое поведение можно объяснить тем, что значения фитнес-функции для особей не сильно друг от друга отличаются. Поэтому даже у самой приспособленной особи есть неплохой шанс не попасть в следующее поколение. Островной алгоритм позволяет получить большие значения функции приспособленности для такого же представления особей за гораздо меньшее число поколений.

Использование автоматов Мили для представления муравья в задаче об «Умном муравье – 3» оправдано, так как позволяет найти муравья, съедающего половину еды на поле. При этом еда расположена случайным образом.

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» и ее модификациях
http://is.ifmo.ru/works/_ant.pdf
3. *Яминов Б.* Генетические алгоритмы
http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005
4. *Данилов В.Р., Шалыто А.А.* Метод генетического программирования для генерации автоматов, представленных деревьями решений
http://is.ifmo.ru/download/2008-03-07-danilov.pdf

Приложение. Исходный код плагина для «Виртуальной лаборатории» на языке *Java*.

Представление особи

Файл AntToTask25Mover.java

```
    package individual.smartant;
    package individual.anttotask25;

    import task.ant.extended.ExtendedAntTask;
    import task.ant.extended.ExtendedAnt;
    import task.ant.extended.Ant;

    /**
     * @author Yuri Konoplev
     */
    public class AntToTask25Mover implements ExtendedAntTask.Mover {

        /**
         * pointer to AntToTask25 which is used in move procedure
         */
        public final AntToTask25 automaton;

        private final double mu;
        private AntToTask25.State cs;
        private ExtendedAnt ant;


        /**
         * @see task.ant.extended.ExtendedAntTask.Mover#move()
         */
        public boolean move() {
            int cf = this.automaton.getCf();
            this.cs = this.automaton.move(this.cs, this.ant);
            return cf != this.automaton.getCf();
        }

        /**
         * @see
task.ant.extended.ExtendedAntTask.Mover#restart(task.ant.extended.Ant)
         */
        public void restart(Ant a) {
            this.ant = (ExtendedAnt) a;
        }

        /**
         * @param automaton pointer to AntToTask25 which is used in move
procedure
         * @param mu probability of food in the current cell
         */
```

```
        public AntToTask25Mover(AntToTask25 automaton, double mu) {
            this.mu = mu;
            this.automaton = automaton;
            this.cs = automaton.state[automaton.initialState];
        }

        /**
         * @return probability of food in the current cell
         * @see task.ant.extended.ExtendedAntTask.Mover#getMu()
         */
        public double getMu() {
            return this.mu;
        }
    }
```
---end----------

Файл AntToTask25Factory.java

```
    package individual.anttotask25;

    import laboratory.common.ga.IndividualFactory;
    import task.ant.extended.Ant;

    import java.util.Random;

    /**
     * @author Yuri Konoplev
     */
    public class AntToTask25Factory implements IndividualFactory {

        private final Random r = new Random();
        public static int attempts;
        private final double mu;
        private final int countStates;

        /**
         * @see laboratory.common.ga.IndividualFactory#randomIndividual()
         */
        public AntToTask25 randomIndividual() {
            AntToTask25 ant = new AntToTask25(this.countStates,
this.r.nextInt(this.countStates), this.mu);
            for (int i = 0; i < this.countStates; i++) {
                Ant.Action[] action = new Ant.Action[256];
                byte[] endState = new byte[256];
                for (int k = 0; k < 256; k++) {
                    switch (this.r.nextInt(3)) {

                        case 0:
                            action[k] = Ant.Action.L;
                            break;
                        case 1:
                            action[k] = Ant.Action.R;
                            break;
                        default:
```

```
                            action[k] = Ant.Action.M;
                            break;
                        }
                    }
                    for (int j = 0; j < 256; j++) {
                        endState[j] = (byte) this.r.nextInt(this.countStates);
                    }
                    ant.state[i] = new AntToTask25.State(action, endState);
                }
                return ant;
            }

            /**
             * @param countStates amount of states, which is written in properties
file
             * @param mu Probability of food in the current cell
             */
            public AntToTask25Factory(int countStates, double mu) {
                this.countStates = countStates;
                this.mu = mu;
            }
        }
```
---end----------

Файл AntToTask25.java

```
    package individual.anttotask25;

    import laboratory.common.ga.Individual;
    import laboratory.common.Visualizable;

    import java.util.Random;

    import task.ant.extended.Ant;
    import task.ant.extended.ExtendedAnt;

    /**
     * @author Yuri Konoplev
     */
    public class AntToTask25 implements Visualizable {

        /**
         * Probability of changing initial state during mutation
         */
        public static double pmis = 0.66;
        /**
         * Probability of food in the current cell
         */
        public final double mu;
        private double fitness = Double.NEGATIVE_INFINITY;
        /**
         * Initial state of Moore automaton
         */
        public final int initialState;
```

```
        /**
         * Array with states
         */
        public final State[] state;
        /**
         * Number of different incoming signals
         */
        public final int MaxSignals = 256;
        private int cf = 0;

        /**
         * Class which represents state in Moore automaton.
         * Every state produces outgoing signal (L, R or M) and
         * give next state for every incoming signal
         */
       // this State meanes State with array of Edges from It
        public static class State {

            /**
             * Array of states which can be reached from current state
             */
            public final byte[] endState;
            /**
             * Outgoing signal (L, R or M)
             */
            //MILI addon
            public final Ant.Action[] action;

            /**
             * @param action Outgoing signal (L, R or M)
             * @param endState Array of states which can be reached from
current state
             */
            public State(Ant.Action[] action, byte[] endState) {
                this.endState = endState;
                this.action = action;
            }
        }

        /**
         * @param cs amount of states in Moore automaton
         * @param is initial state
         * @param mu Probability of food in the current cell
         */
        public AntToTask25(int cs, int is, double mu) {
            this.state = new State[cs];
            this.initialState = is;
            this.mu = mu;
        }

        /**
         * @param s current state
         * @param ant Ant which can see 8 cells around itself, it gave us
information about them
         * @return next state, which will be obtained after proceeding
information
         */
```

```
public State move(State s, ExtendedAnt ant) {
    boolean[] variables = ant.F();

    int index = 0;
    /**
                *   do from signals one byte
                5
                4 7
                > 0 1
                2 6
                3
                example: if everwhere is food the output if 11111111
    */
    for (int i = 0; i < 8; i++) {
        if (variables[i]) {
            index = (index << 1);
            index++;
        } else {
            index = (index << 1);
        }
    }

    Ant.Action a = s.action[index]; // s.action[act_it];
    if ((a == Ant.Action.M) && variables[0]) {
        this.cf++;
    }

    if (a == Ant.Action.L) {
        ant.L();
    } else if (a == Ant.Action.R) {
        ant.R();
    } else if (a == Ant.Action.M) {
        ant.M();
    }
    return this.state[s.endState[index]];
}

/**
 * @return amount of food which was eaten
 */
public int getCf() {
    return this.cf;
}

/**
 * Fitness function value
 * @see laboratory.common.ga.Individual#fitness()
 */
@Override
public double fitness() {
    if (this.fitness == Double.NEGATIVE_INFINITY) {
        this.fitness = 0;
        for (int j = 0; j < AntToTask25Factory.attempts; j++) {
            State s = this.state[this.initialState];
            ExtendedAnt ant = new ExtendedAnt(this.mu);
            for (int i = 0; i < Ant.NUMBER_STEPS; i++) {
                s = move(s, ant);
```

```
                    }
                    this.fitness += this.cf;
                    this.cf = 0;
                }
                fitness /= AntToTask25Factory.attempts;
            }
            return this.fitness;
        }

        /**
         * @see laboratory.common.ga.Individual#mutate(java.util.Random)
         */
        @Override
        public AntToTask25 mutate(Random r) {
            int cs = this.state.length;
            AntToTask25 ant = new AntToTask25(
                    cs,
                    (r.nextDouble() < pmis) ? r.nextInt(cs) :
this.initialState,
                    this.mu);
            System.arraycopy(this.state, 0, ant.state, 0, cs);
            int is = r.nextInt(cs);
            State s = this.state[is];
            if (r.nextBoolean()) {
                Ant.Action[] action = new Ant.Action[MaxSignals];
                for (int i = 0; i<MaxSignals; i++) {
                    switch (r.nextInt(3)) {
                        case 0:
                            action[i] = Ant.Action.L;
                            break;
                        case 1:
                            action[i] = Ant.Action.R;
                            break;
                        default:
                            action[i] = Ant.Action.M;
                            break;
                    }
                }
                ant.state[is] = new State(action, this.state[is].endState);
            } else {
                byte[] endState = new byte[this.MaxSignals];
                System.arraycopy(s.endState, 0, endState, 0, this.MaxSignals);
                endState[r.nextInt(this.MaxSignals)] = (byte) r.nextInt(cs);
                ant.state[is] = new State(this.state[is].action, endState);
            }
            return ant;
        }

        /**
         * @see
laboratory.common.ga.Individual#crossover(laboratory.common.ga.Individual,
java.util.Random)
         */
        @Override
        public AntToTask25[] crossover(Individual p, Random r) {
            AntToTask25[] parent = new AntToTask25[2];
            parent[0] = this;
```

```
            parent[1] = (AntToTask25) p;
            AntToTask25[] child = new AntToTask25[2];
            int ri = r.nextInt(2);
            int cs = this.state.length;
            for (int i = 0; i < 2; i++) {
                child[Math.abs(ri - i)] = new AntToTask25(cs,
parent[i].initialState, this.mu);
            }
            for (int j = 0; j < cs; j++) {
                ri = r.nextInt(2);
                for (int i = 0; i < 2; i++) {
                    child[Math.abs(ri - i)].state[j] = parent[i].state[j];
                }
            }
            return child;
        }

        /**
         * @see java.lang.Comparable#compareTo(java.lang.Object)
         */
        @Override
        public int compareTo(Individual individual) {
            return Double.compare(individual.fitness(), fitness());
        }

        /**
         * @see java.lang.Object#toString()
         */
        @Override
        public String toString() {
            String res = this.state.length + " states (initial=" +
(this.initialState + 1) +
                    "), fitness=" + fitness() + "\n";
                  int num_st = 1;
            for (State s : this.state) {
                  res += "State number=" + num_st + "\n";
                  num_st++;

                for (int i = 0; i < this.MaxSignals; i++) {
                    res += s.endState[i] + 1;

                        Ant.Action a = s.action[i];
                          if (a == Ant.Action.L) {
                                    res += " L";
                            } else if (a == Ant.Action.R) {
                                    res += " R";
                            } else if (a == Ant.Action.M) {
                                    res += " M";
                            }

                    if (i < this.MaxSignals - 1) {
                        res += ", ";
                    }
                    if ((i + 1) % 64 == 0) {
                        res += "\n";
                    }
                }
```

```
                res += "\n";
            }
            return res;
        }

        /**
         * @see laboratory.common.Visualizable#getAttributes()
         */
        @Override
        public Object[] getAttributes() {
            return new Object[]{new AntToTask25Mover(this, this.mu)};
        }
    }
```
---end----------

Файл FactoryLoader.java

```
    package individual.anttotask25;

    import laboratory.util.AbstractLoader;
    import java.util.jar.JarFile;

    public class FactoryLoader extends AbstractLoader<AntToTask25Factory> {

        /**
         * @see laboratory.common.Loader#load(java.lang.Object[])
         */
        public AntToTask25Factory load(Object... args) {
            AntToTask25Factory.attempts = properties.getInt("count.attempts");
            return new
AntToTask25Factory(this.properties.getInt("count.states"),
            this.properties.getDouble("mu"));
        }

        /**
         * @param file jar-file with this project which contains necessary
properties
         */
        public FactoryLoader(JarFile file) {
            super(file, "automaton.conf");
        }
    }
```

Генетический алгоритм

Файл RouletteGALoader.java

```
    package ga.roulette;

    import laboratory.common.Loader;
    import laboratory.common.ga.GA;
    import laboratory.common.ga.IndividualFactory;
```

```
import laboratory.util.Parser;

import java.io.IOException;
import java.util.Properties;
import java.util.jar.JarEntry;
import java.util.jar.JarFile;

/**
 * @author Nikolay Makarov
 */
public class RouletteGALoader implements Loader<GA> {

    private final Parser properties;
    /**
     * @see laboratory.common.Loader#load(java.lang.Object[])
     */
    public GA load(Object... args) {
        return new RouletteGA(this.properties.getInt("sizeGeneration"),
                this.properties.getDouble("probabilityMutation"),
                (IndividualFactory) args[0]);
    }

    /**
     * Creates Parser object which will be used for reading properties
     * @param file jar-file with this project
     */
    public RouletteGALoader(JarFile file) {
        Properties in = new Properties();
        try {
            JarEntry ent = new JarEntry("rouletteGA.conf");
            in.load(file.getInputStream(ent));
        } catch (IOException e) {
            e.printStackTrace();
        }
        this.properties = new Parser(in);
    }

    /**
     * @see laboratory.common.Loader#getProperties()
     */
    public Properties getProperties() {
        return this.properties.getProperties();
    }
}
```
---end----------

Файл RouletteGA.java

```
package ga.roulette;

import laboratory.common.ga.GA;
import laboratory.common.ga.Individual;
import laboratory.common.ga.IndividualFactory;
```

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Random;

/**
 * @author Nikolay Makarov
 */
public class RouletteGA implements GA {

    private List<Individual> generation;

    private final double probabilityMutation;

    private final IndividualFactory factory;

    private final Random r;

    /**
     * @see laboratory.common.ga.GA#nextGeneration()
     */
    public void nextGeneration() {
        int size = this.generation.size();
        // Sum of individuals' fitness values
        double totalFitness = 0.0;
        for (Individual individual : this.generation) {
        totalFitness += individual.fitness();
        }
        List<Individual> newGeneration = new ArrayList<Individual> (size);
        for (int i = 0; i < size; i++) {
        // random position on roulette
        double val = this.r.nextDouble() * totalFitness;
        // individual which corresponds to this position
        double sum = 0.0;
        for (int j = 0; j < size; j++) {
              double fval = this.generation.get(j).fitness();
              if (sum + fval > val) {
                    newGeneration.add(this.generation.get(j));
                    break;
              }
              sum += fval;
        }
        }
        /**
         * We can crossover neighbour individuals,
         * because we chose them at random positions earlier
         */
        for (int i = 0; i < size; i += 2) {
        Individual father = newGeneration.get(i);
        Individual mother = newGeneration.get(i + 1);
            Individual[] s = father.crossover(mother, this.r);
            newGeneration.set(i, s[0]);
            newGeneration.set(i + 1, s[1]);
        }
        if (size % 2 == 1) { // If amount of individuals is odd, we mutate
last
```

```
                newGeneration.set(size - 1, newGeneration.get(size -
1).mutate(this.r));
            }
            // We mutate all of them
            for (int i = 0; i < size; i++) {
                if (this.r.nextDouble() < this.probabilityMutation) {
                    newGeneration.set(i, newGeneration.get(i).mutate(this.r));
                }
            }
            this.generation = newGeneration;
            Collections.sort(this.generation);
        }

        /**
         * @see laboratory.common.ga.GA#getGeneration()
         */
        public List<Individual> getGeneration() {
            return this.generation;
        }

        /**
         * @param sizeGeneration constant generation size
         * @param probabilityMutation probability of mutation after crossover
         * @param factory pointer to the individual factory
         */
        public RouletteGA(int sizeGeneration, double probabilityMutation,
IndividualFactory factory) {
            this.probabilityMutation = probabilityMutation;
            // Creating new generation
            this.generation = new ArrayList<Individual>(sizeGeneration);
            for (int i = 0; i < sizeGeneration; i++) {
                this.generation.add(factory.randomIndividual());
            }
            Collections.sort(this.generation);
            this.factory = factory;
            this.r = new Random();
        }

        /**
         * Creates new generation
         * @see laboratory.common.ga.GA#bigMutation()
         */
        public void bigMutation() {
            for (int i = 0; i < this.generation.size(); i++) {
                this.generation.set(i, this.factory.randomIndividual());
            }
            Collections.sort(this.generation);
        }

        /**
         * @return best individual in current generation
         * @see laboratory.common.ga.GA#getBest()
         */
        public Individual getBest() {
            return this.generation.get(0);
        }
    }
```

Скрипт для сборки плагина к «Виртуальной лаборатории»

Файл build.properties

```
deploy=deploy
debug.val=true
util=util

ga.dir=gas
individual.dir=individuals

ga.src=ga
individual.src=individual

project.name=Virtual Laboratory
main.class=laboratory.core.Main
```

---end---

Файл build.xml (для компиляции необходимы файлы util.jar, common.jar и extended-ant.jar, которые представляют «Виртуальную лабораторию»)

```
<?xml version="1.0"?>

<project name="visualizer" default="build">

    <property file="build.properties"/>

    <target name="build" depends="build.main, build.GA, build.individual"
/>

    <property name="common.build" value="${util}/common.jar"/>
    <property name="util.build" value="${util}/util.jar"/>
    <property name="exttask.build" value="${util}/extended-ant.jar"/>

    <target name="build.main">
        <mkdir dir="${deploy}"/>
    </target>

    <target name="build.GA">
        <mkdir dir="${deploy}/${ga.dir}"/>
        <antcall target="build.plugin">
            <param name="plugin" value="${ga.src}/roulette"/>
            <param name="plugin.classpath" value="${common.build};
${util.build}"/>
            <param name="plugin.dir" value="${ga.dir}"/>
        </antcall>
    </target>

    <target name="build.individual">
```

```
            <mkdir dir="${deploy}/${individual.dir}"/>
            <antcall target="build.plugin">
                <param name="plugin" value="${individual.src}/AntToTask25"/>
                <param name="plugin.classpath" value="${common.build};
${util.build}; ${exttask.build}"/>
                <param name="plugin.dir" value="${individual.dir}"/>
            </antcall>
        </target>

        <target name="build.plugin">
            <echo message="${plugin}"/>
            <echo message="${plugin.classpath}"/>
            <fail unless="plugin"/>
            <property file="${plugin}/plugin.properties" prefix="plugin"/>
            <fail unless="plugin.main.class"/>
            <fail unless="plugin.extension.name"/>
            <fail unless="plugin.comment"/>
            <condition property="plugin.src" value="src">
                <not>
                    <isset property="plugin.src/"/>
                </not>
            </condition>
            <condition property="plugin.filter" value="">
                <not>
                    <isset property="plugin.filter"/>
                </not>
            </condition>
            <condition property="plugin.graphic.settings" value="">
                <not>
                    <isset property="plugin.graphic.settings"/>
                </not>
            </condition>
            <echo message="${plugin.filter}"/>
            <echo message="${plugin.graphic.settings}"/>
            <condition property="plugin.build" value="classes">
                <not>
                    <isset property="plugin.build"/>
                </not>
            </condition>
            <condition property="plugin.jar"
value="${plugin.extension.name}.jar">
                <not>
                    <isset property="plugin.jar"/>
                </not>
            </condition>
            <mkdir dir="${plugin}/${plugin.build}"/>
            <javac
                    srcdir="${plugin}/${plugin.src}"
                    destdir="${plugin}/${plugin.build}"
                    classpath="${plugin.classpath}"
                    />
            <jar destfile="${deploy}/${plugin.dir}/${plugin.jar}">
                <fileset dir="${plugin}/${plugin.build}"/>
                <fileset dir="${plugin}/${plugin.resources}"/>
                <manifest>
                    <attribute name="Main-Class" value="${plugin.main.class}"/>
```

```
                    <attribute name="Extension-Name"
value="${plugin.extension.name}"/>
                    <attribute name="Comment" value="${plugin.comment}"/>
                    <attribute name="Filter" value="${plugin.filter}"/>
                    <attribute name="GraphicSettings"
value="${plugin.graphic.settings}"/>
                </manifest>
            </jar>
        </target>

        <target name="clean" depends="clean.ga, clean.individual">
            <delete quiet="true" includeEmptyDirs="true">
                <fileset dir="${build}"/>
                <fileset dir="${deploy}"/>
            </delete>
        </target>

        <target name="clean.ga">
            <antcall target="clean.plugin">
                <param name="plugin" value="${ga.src}/roulette"/>
            </antcall>
        </target>

        <target name="clean.individual">
            <antcall target="clean.plugin">
                <param name="plugin" value="${individual.src}/AntToTask25"/>
            </antcall>
        </target>

        <target name="clean.plugin">
            <fail unless="plugin"/>
            <echo message="${plugin}"/>
            <property file="${plugin}/plugin.properties" prefix="plugin"/>
            <condition property="plugin.build" value="classes">
                <not>
                    <isset property="plugin.build"/>
                </not>
            </condition>
            <delete quiet="true" includeEmptyDirs="true">
                <fileset dir="${plugin}/${plugin.build}"/>
            </delete>
        </target>

    </project>
---end----------------------------------------------------------------------------------------------------------------
```