Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

А. К. Заикин

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

Вариант № 34

Санкт-Петербург
2009

Оглавление

Введение

Цель лабораторной работы – изучение генетических алгоритмов для построения систем конечных автоматов. Рассматривается построение системы из двух вложенных автоматов Мили, решающей задачу об «Умном муравье - 3».

Для выполнения задания необходимо создать для виртуальной лаборатории модуль, который бы решал поставленную задачу и удовлетворял требованиям, описанным в работе [1].

1. Постановка задачи

Построить с помощью генетических алгоритмов систему из двух вложенных автоматов Мили (глава 1.1), решающую задачу об «Умном муравье – 3» (глава 1.2). Использовать представление автоматов с помощью битовых строк. Способ скрещивания выбрать самостоятельно. Использовать островной генетический алгоритм и элитизм для генерации очередного поколения.

1.1. Автомат Мили

Автомат Мили – конечный автомат, генерирующий свои выходные действия в зависимости от текущего состояния и входного сигнала (воздействия).

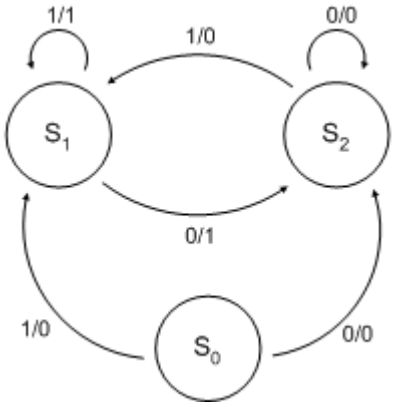

Рис. 1. Пример диаграммы переходов автомата Мили из трех состояний

Как видно из приведённой диаграммы, над каждой дугой расположена пара значений: входное и выходное воздействия, причём последнее зависит не только от состояния, в котором находится автомат, но и от значения входного воздействия [2].

1.2. Задача об «Умном муравье - 3»

В задаче об «Умном муравье - 3» рассматривается поле, располагающееся на поверхности тора и имеющее размер 32 на 32 клетки. Каждая клетка поля с некоторой, заранее определенной, вероятностью содержит яблоко.

Муравей видит восемь клеток перед собой (рис. 2) и может выполнять одно из следующих действий:

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

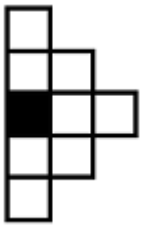

Рис. 2. Видимая область муравья

Максимальное число шагов – 200. Цель задачи – создать муравья, который управляется системой автоматов с фиксированным числом состояний и съедает максимальное число яблок.

2. Реализация

В виртуальной лаборатории уже создан модуль генетического алгоритма, использующий островной генетический алгоритм и элитизм для генерации очередного поколения. Поэтому необходимо создать только «особь» – систему из двух вложенных конечных автоматов, решающую задачу об «Умном муравье - 3». Для нее необходимо реализовать операции мутации и скрещивания.

2.1. Описание используемого представления автоматов

Каждый автомат представляется в виде битовой строки, имеющей следующую структуру для автомата из *N* состояний (рис. 3).

$$[\text{состояние } 1][\text{состояние } 2][\text{состояние } 3]\dots[\text{состояние } N]$$

Рис. 3. Представление автомата в виде одномерного массива

Внутри каждого состояния закодированы переходы в порядке входного воздействия (рис. 4).

$$\underbrace{[[\text{переход } 0][\text{переход } 1]\dots[\text{переход } 255]]}_{\text{состояние 1}}\underbrace{[[\text{переход } 0][\text{переход } 1]\dots[\text{переход } 255]]}_{\text{состояние 2}}\dots$$

Рис. 4. Представление состояний в виде одномерного массива

Под входным воздействием понимается состояние поля, которое видит муравей. Поскольку он видит восемь клеток перед собой, то возможно 256 вариантов расположения яблок на этих клетках. Запись `[переход T]` означает, что это переход при входном воздействии с номером `T`. Приведем алгоритм определения номера входного воздействия по состоянию видимой области поля (листинг 1).

Листинг 1. Сопоставление видимой области и номера входного воздействия

```
int power = 1;
int index = 0;
for (int i = 0; i < visible.length; i++) {
     if (visible[i]) {
       index += power;
     }
     power *= 2;
}
```

Видимые клетки поля представлены в виде одномерного битового массива `visible[8].` Если в клетке есть яблоко, то значение соответствующего элемента массива `visible` равно `true(1)`, в противном случае – `false(0)`. Битовый массив `visible[8]` используется в качестве двоичного представления номера входного воздействия и переводится в десятичную систему исчисления с помощью алгоритм (листинг 1). После выполнения цикла `for` значение переменной `index` будет равно номеру входного воздействия.

Каждый переход кодируется десятью битами (рис. 5).

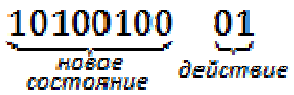

Рис. 5. Представление перехода в виде битовой строки

Первые восемь бит являются двоичным представлением номера нового состояния (от 0 до 255). Состояния конечного автомата нумеруются, начиная с единицы. Нулевое значение, а так же значение большее, чем число состояний, означает, что перехода для данного входящего воздействия нет.

Кодирование последних двух бит приведено в таблице.

Таблица. Кодирование двух последних бит

Код действия	Расшифровка
00	Ничего не делать
01	Поворот влево
10	Шаг вперед
11	Поворот вправо

2.2. *Описание метода скрещивания*

Для скрещивания двух автоматов применяется классический оператор одноточечного кроссовера (*1-point crossover*): для родительских хромосом случайным образом выбирается точка раздела, и они обмениваются отсеченными частями [3, 4]. Полученные две строки являются потомками (рис. 6).

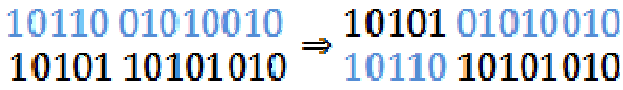

Рис. 6. Одноточечный кроссовер

2.3. *Описание метода мутации*

При мутации в каждом переходе номер нового состояния и действие изменяется на произвольное с вероятностью один процент [3, 4].

2.4. *Описание метода генерации очередного поколения*

Для генерации очередного поколения используется островной генетический алгоритм и элитизм.

Островная модель (*island model*) – модель параллельного генетического алгоритма. Она заключается в следующем: имеется 6000 особей. Разобьем их на 30 подпопуляций по 200 особей. Каждая их них будет развиваться отдельно с помощью некого генетического алгоритма. Изредка (например, каждые пять поколений) острова будут обмениваться несколькими хорошими особями (миграция) [3, 4].

Элитизм — стратегия формирования нового поколения, которая заключается в следующем: в новое поколение обязательно включается заданное количество (например, один процент от общего количества особей) лучших особей предыдущего поколения [3, 4].

2.5. *Описание способа вычисления функции приспособленности*

Функция приспособленности *f* вычисляется по следующей формуле:

$$f = \frac{\sum_{i=1}^{k} a_i}{k},$$

где k – число запусков муравья, a_i – число съеденных яблок на i-ой попытке.

3. Результаты работы модуля

3.1. *Графики функции приспособленности*

На рис. 7 приведен график максимального значения функции приспособленности.

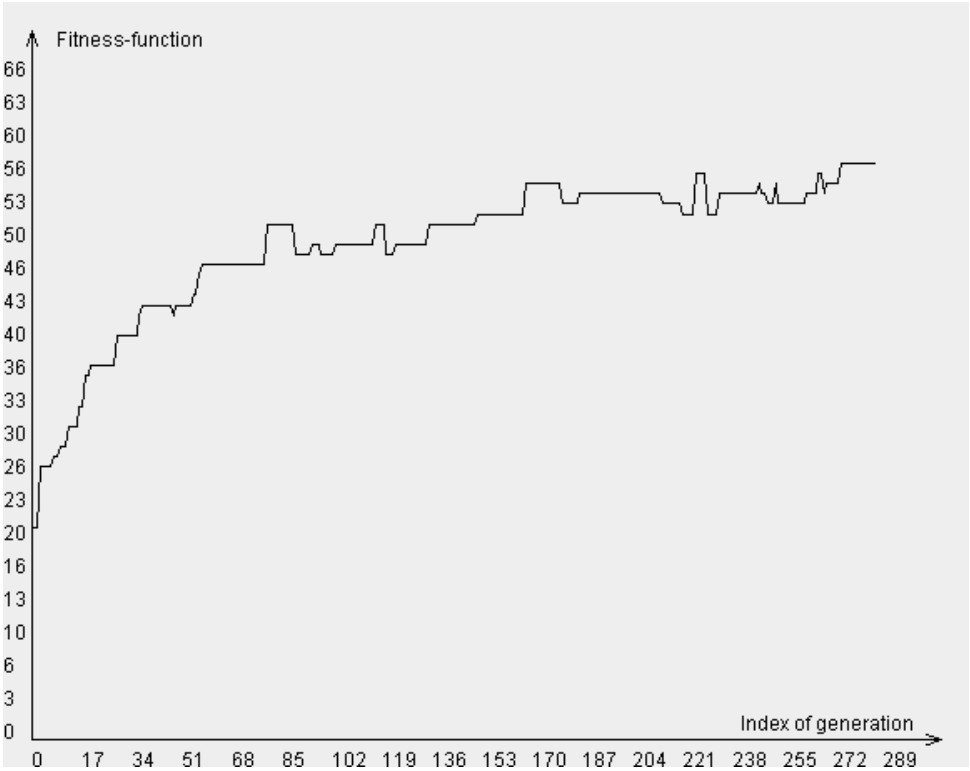

Рис. 7. График максимального значения функции приспособленности

На рис. 8 приведен график среднего значения функции приспособленности.

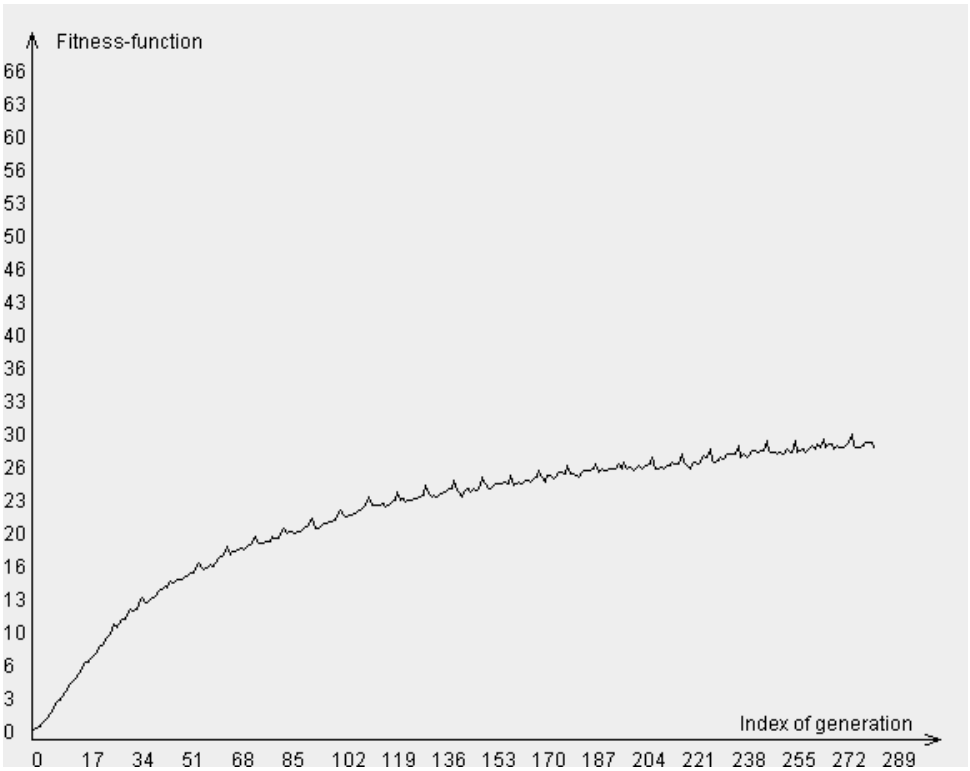

Рис. 8. График среднего значения функции приспособленности

3.2. Таблица переходов полученного автомата

Полученная система двух конечных автоматов представлена ниже в виде таблицы переходов (листинг 2). По горизонтали расположены сначала состояния внешнего автомата, а потом внутреннего (вложенного). По вертикали расположены входные воздействия.

Переход записан в следующей форме: (<новое состояние>, <действие>). Запись --- означает, что перехода нет.

Действие может принимать одно из следующих значений:

1. L – поворот налево;
2. M – шаг вперед;
3. R – поворот направо.

Внешний автомат:

1. Начальное состояние – 1;
2. Число состояний – 5.

Внутренний автомат:

1. Начальное состояние – 0;
2. Число состояний – 3.

Число съедаемых яблок – 57

Листинг 2. Таблица переходов полученной системы автоматов

	0	1	2	3	4	0	1	2
0	(3, R)	(1, M)	(1, R)	(3, M)	(1, L)	(0, M)	(1, M)	(2, R)
1	(1, M)	(4, M)	(1, M)	(1, M)	(1, M)	(2, L)	(1, R)	(0, R)
2	---	(4, M)	(3, M)	(0, M)	---	(0, M)	(2, R)	(0, M)
3	(0, R)	(2, M)	(4, R)	(2, M)	(4, M)	(1, M)	(2, M)	(2, R)
4	(1, R)	(1, R)	(2, L)	(3, R)	(4, R)	(2, L)	(2, L)	(2, L)
5	(4, L)	(2, M)	(0, M)	(1, M)	(2, M)	(2, L)	(2, R)	(1, R)
6	(4, R)	---	(3, R)	(3, L)	(4, R)	(2, M)	(0, M)	(2, M)
7	(2, M)	(0, R)	(1, R)	(4, R)	(2, M)	(2, M)	(0, M)	(0, L)
8	(4, L)	---	(2, R)	(4, L)	(0, R)	(0, R)	(2, M)	(2, R)
9	(4, R)	(1, R)	---	(1, M)	(3, M)	(2, R)	(2, M)	(1, R)
10	(1, L)	(0, R)	(2, M)	---	(0, L)	(0, R)	(1, M)	(0, R)
11	(0, M)	(4, R)	(3, R)	(2, M)	(2, R)	(0, M)	(2, R)	(0, L)
12	(1, M)	(3, L)	(3, M)	(4, R)	(3, R)	(1, L)	(0, L)	(1, R)
13	(3, R)	(0, L)	---	(3, R)	(1, M)	(0, M)	(1, R)	(2, R)
14	(2, M)	(0, R)	---	(4, L)	(2, M)	(0, M)	(2, L)	(1, R)
15	---	---	(1, L)	(0, M)	(0, L)	(0, M)	(2, L)	(1, R)
16	---	(1, M)	(4, L)	---	(3, L)	(0, L)	(0, M)	(0, L)
17	(2, L)	(2, R)	(4, L)	(4, M)	(3, L)	(0, L)	(1, M)	(1, R)
18	(0, R)	(3, L)	(1, M)	(0, L)	(2, M)	(1, R)	(1, R)	(0, M)
19	(1, M)	---	(2, R)	(0, R)	(4, L)	(1, L)	(1, L)	(2, L)
20	(2, R)	(0, R)	(3, M)	---	(2, L)	(0, L)	(1, L)	(0, R)
21	(2, R)	(2, M)	(2, M)	(1, M)	---	(2, M)	(1, L)	(2, R)

22	---	(1, R)	(0, M)	(0, L)	(1, L)	(2, R)	(0, L)	(2, R)
23	(3, R)	(4, L)	---	---	(4, L)	(1, R)	(0, L)	(2, M)
24	(0, M)	(2, R)	(1, M)	(2, M)	(4, R)	(1, M)	(0, L)	(0, L)
25	(1, R)	(2, R)	(3, L)	(0, M)	---	(2, M)	(2, L)	(2, R)
26	(0, M)	(4, M)	(1, R)	(4, M)	(4, R)	(2, L)	(1, R)	(2, R)
27	---	(1, M)	(2, L)	(4, L)	(3, L)	(0, L)	(0, L)	(0, L)
28	---	---	(0, L)	(1, L)	(0, L)	(2, L)	(1, R)	(1, R)
29	(1, M)	(1, R)	(0, M)	(1, L)	---	(2, R)	(0, M)	(1, L)
30	---	(0, R)	(3, R)	(0, L)	(0, M)	(0, L)	(2, L)	(0, M)
31	(2, M)	(1, M)	(4, M)	(3, M)	(3, M)	(2, R)	(1, L)	(0, L)
32	(3, L)	(1, L)	(1, R)	(1, L)	(3, L)	(0, L)	(2, M)	(2, R)
33	(3, M)	(4, R)	(4, R)	(0, M)	(1, M)	(2, R)	(1, R)	(1, L)
34	(4, R)	(1, L)	(3, L)	(4, M)	(3, L)	(0, M)	(2, L)	(1, L)
35	(0, L)	(1, L)	(2, M)	(1, L)	(3, R)	(1, M)	(0, R)	(2, M)
36	(2, R)	(4, L)	(1, R)	---	(1, M)	(2, R)	(2, M)	(2, R)
37	---	(4, R)	(0, L)	(1, L)	(1, M)	(2, L)	(2, R)	(2, M)
38	(0, L)	---	---	(1, L)	---	(2, M)	(0, R)	(0, M)
39	---	(2, L)	(3, L)	(4, L)	(2, R)	(1, L)	(2, M)	(2, M)
40	(1, R)	(0, L)	---	---	---	(1, M)	(1, M)	(0, L)
41	(2, R)	(1, L)	(2, M)	(1, M)	(4, R)	(2, L)	(2, M)	(1, M)
42	(3, R)	(0, M)	---	(1, R)	(1, R)	(1, M)	(0, M)	(1, M)
43	(4, M)	(1, L)	(3, M)	(3, L)	(3, L)	(0, M)	(1, R)	(2, M)
44	---	(1, M)	(1, L)	(0, L)	---	(1, M)	(2, L)	(0, M)
45	(2, R)	(1, R)	(0, R)	(4, L)	(4, L)	(1, R)	(1, L)	(2, R)
46	(2, M)	(2, M)	(2, R)	(4, R)	(2, M)	(1, L)	(2, R)	(2, R)
47	(3, L)	(4, R)	(4, M)	(3, R)	(1, R)	(0, L)	(1, R)	(0, R)
48	(0, R)	(3, L)	(1, L)	(3, L)	---	(2, M)	(0, M)	(2, L)
49	---	(4, L)	(0, L)	(2, L)	(2, L)	(0, L)	(0, R)	(2, L)
50	(1, R)	(4, L)	(4, R)	(0, M)	---	(2, L)	(2, M)	(0, M)
51	---	(4, L)	(2, M)	(0, M)	(0, R)	(2, M)	(0, R)	(1, L)
52	---	(4, M)	---	(1, M)	---	(2, R)	(0, M)	(2, L)
53	---	(4, L)	(2, R)	(3, R)	---	(1, M)	(2, R)	(1, R)
54	(2, L)	(2, L)	(0, M)	(1, L)	(3, R)	(1, R)	(1, R)	(0, M)
55	(0, R)	---	(2, L)	---	(2, M)	(1, R)	(1, R)	(0, L)
56	---	(0, M)	(0, R)	(1, L)	(3, M)	(0, M)	(2, R)	(1, M)
57	(1, L)	(3, M)	(0, M)	(3, R)	(4, L)	(1, R)	(0, R)	(2, L)
58	(0, M)	(4, L)	(4, M)	(2, R)	(1, M)	(2, L)	(2, R)	(0, M)
59	(0, L)	(3, M)	(3, M)	(0, M)	(0, L)	(0, M)	(1, M)	(0, M)
60	(1, L)	(0, R)	(4, R)	(0, R)	(4, R)	(0, M)	(2, L)	(0, L)
61	(4, R)	(3, R)	(3, M)	(4, M)	(4, L)	(2, M)	(0, R)	(1, L)
62	(1, M)	(0, M)	(3, R)	(3, R)	(2, L)	(1, M)	(1, R)	(2, L)
63	---	(3, L)	(0, R)	(3, M)	---	(1, M)	(0, M)	(2, R)
64	(2, M)	---	(3, R)	(3, M)	(2, R)	(2, R)	(0, M)	(2, M)
65	(4, L)	---	(0, R)	(3, M)	---	(2, M)	(2, M)	(1, M)
66	(2, L)	(2, R)	(2, R)	(3, M)	(2, L)	(2, L)	(2, R)	(0, L)
67	(0, R)	---	---	(4, M)	(2, L)	(1, M)	(2, M)	(0, R)
68	---	(4, R)	(0, R)	(0, R)	---	(2, R)	(1, R)	(2, R)
69	(4, M)	(3, M)	(0, R)	---	(4, L)	(0, L)	(0, R)	(1, M)

70	(0, M)	(0, R)	(3, R)	(3, M)	---	(0, L)	(2, L)	(0, L)
71	(2, R)	(1, R)	(3, L)	---	(0, L)	(2, L)	(1, L)	(1, L)
72	(0, L)	(3, M)	(2, M)	(4, M)	(2, R)	(1, R)	(2, R)	(0, M)
73	(2, R)	(2, L)	---	(2, L)	---	(2, M)	(2, L)	(0, R)
74	(3, M)	(3, L)	(4, M)	(2, M)	(2, M)	(1, R)	(2, R)	(0, L)
75	(2, M)	(1, L)	(1, M)	(0, L)	(2, M)	(1, R)	(1, M)	(2, R)
76	(2, M)	(4, M)	---	---	(0, M)	(2, M)	(1, M)	(2, R)
77	(1, R)	(4, L)	---	(3, L)	(2, R)	(1, R)	(2, L)	(2, R)
78	(3, R)	(1, L)	(3, L)	(3, R)	(2, M)	(2, M)	(1, M)	(1, M)
79	(4, R)	(0, L)	(4, L)	(0, L)	---	(0, R)	(2, L)	(2, L)
80	(3, M)	---	---	---	(2, R)	(0, L)	(0, M)	(0, L)
81	---	(3, R)	(1, L)	(3, L)	(1, M)	(1, M)	(1, R)	(1, L)
82	---	(2, M)	(4, L)	(3, M)	(1, R)	(0, L)	(1, R)	(0, L)
83	(2, R)	(0, R)	(1, R)	(1, R)	(2, L)	(2, L)	(0, R)	(0, L)
84	(3, M)	(2, L)	(2, R)	(2, M)	(1, R)	(2, M)	(1, R)	(0, M)
85	(4, M)	(2, L)	---	(4, L)	(4, R)	(1, M)	(2, L)	(1, M)
86	(2, R)	(4, L)	(3, M)	(4, R)	(4, M)	(0, M)	(0, M)	(2, L)
87	(1, L)	---	(1, M)	(0, M)	(1, R)	(2, M)	(2, R)	(2, L)
88	(1, M)	---	(1, R)	(2, M)	---	(2, M)	(0, L)	(0, R)
89	---	(4, L)	(0, M)	---	(3, M)	(2, L)	(1, M)	(1, R)
90	(2, R)	(2, M)	(1, L)	(4, M)	(2, L)	(2, M)	(0, L)	(2, M)
91	(3, M)	(4, R)	---	---	---	(2, L)	(2, M)	(2, M)
92	---	---	(0, L)	(4, R)	(1, R)	(1, M)	(2, L)	(2, R)
93	(3, M)	(0, L)	(4, R)	---	(3, R)	(0, R)	(0, L)	(0, L)
94	(0, L)	(4, M)	(1, R)	(1, R)	(1, R)	(1, R)	(1, R)	(1, M)
95	---	(1, L)	(4, R)	(1, R)	(1, L)	(1, M)	(1, R)	(2, L)
96	(4, M)	(1, M)	(1, L)	(4, L)	---	(0, L)	(0, M)	(0, R)
97	(1, R)	(3, R)	(4, L)	(3, R)	(3, M)	(2, M)	(0, M)	(0, R)
98	(0, M)	(4, L)	---	(4, R)	(0, M)	(0, L)	(2, L)	(2, R)
99	(1, R)	---	(3, M)	(2, M)	(4, L)	(1, M)	(0, M)	(0, R)
100	(3, M)	(3, M)	(2, L)	(2, L)	(3, R)	(2, R)	(0, R)	(2, L)
101	(3, L)	(1, L)	(1, R)	(1, L)	(4, R)	(1, L)	(2, L)	(0, L)
102	(4, R)	(2, L)	---	(4, M)	(4, M)	(1, R)	(0, R)	(1, L)
103	---	(0, L)	---	(1, R)	(3, M)	(2, R)	(0, L)	(0, R)
104	(2, M)	(1, L)	(1, M)	---	(0, L)	(2, R)	(1, M)	(0, L)
105	(0, M)	(0, L)	---	(3, R)	(1, L)	(1, R)	(2, R)	(2, L)
106	(2, L)	(3, R)	(0, L)	(2, L)	(3, L)	(0, M)	(0, L)	(2, R)
107	(2, L)	(0, L)	(0, L)	(4, M)	(3, M)	(0, L)	(2, R)	(2, L)
108	(1, M)	---	(3, L)	(3, L)	(2, M)	(2, M)	(0, L)	(0, L)
109	(1, M)	(0, M)	(0, M)	(1, M)	(3, R)	(2, L)	(2, M)	(0, M)
110	(1, M)	---	(0, M)	(0, L)	(0, L)	(0, R)	(1, M)	(0, M)
111	(2, M)	(3, R)	(2, R)	(3, R)	(0, M)	(0, M)	(2, L)	(0, R)
112	(0, M)	(4, L)	(1, L)	(1, M)	(2, M)	(0, M)	(2, R)	(1, M)
113	---	(1, L)	(0, L)	(3, L)	(3, L)	(1, L)	(2, R)	(1, M)
114	(3, M)	(3, M)	(3, M)	(4, L)	(2, R)	(1, M)	(1, M)	(0, M)
115	(0, L)	---	(4, R)	(4, L)	---	(1, M)	(0, L)	(0, M)
116	(2, M)	(0, L)	(4, L)	(4, R)	(0, M)	(2, M)	(1, L)	(0, R)
117	(2, R)	(1, R)	(1, R)	(4, M)	(4, R)	(2, L)	(1, M)	(1, R)

118	---	(3, M)	---	(0, M)	(2, R)	(2, L)	(1, M)	(2, L)
119	---	(2, M)	---	(1, R)	(1, M)	(0, L)	(2, L)	(1, M)
120	(1, R)	---	(2, L)	(2, R)	(0, R)	(0, L)	(2, L)	(1, M)
121	(1, R)	(0, R)	(1, R)	(4, R)	(1, L)	(1, L)	(0, R)	(1, L)
122	(4, R)	---	---	(3, R)	(1, R)	(2, M)	(1, M)	(2, M)
123	---	(1, M)	(3, R)	(4, M)	(0, L)	(2, R)	(1, L)	(2, L)
124	(3, R)	(4, M)	(1, M)	(4, R)	(4, L)	(2, M)	(0, R)	(2, M)
125	(3, R)	(1, M)	(1, L)	(3, L)	(1, L)	(1, L)	(0, M)	(0, M)
126	(3, L)	(0, R)	(0, L)	(1, L)	---	(0, L)	(1, L)	(0, L)
127	(4, L)	(0, M)	(4, R)	---	(3, L)	(1, R)	(2, L)	(1, R)
128	(1, L)	(4, M)	(0, L)	(1, M)	(3, M)	(0, L)	(2, M)	(2, R)
129	(0, M)	(0, R)	(0, M)	(0, R)	(4, M)	(2, M)	(0, M)	(0, L)
130	(2, M)	---	---	(3, M)	(4, L)	(1, L)	(2, M)	(2, L)
131	(4, L)	(2, M)	(3, L)	(4, R)	---	(1, L)	(1, R)	(1, M)
132	(1, R)	(3, M)	(1, M)	---	(4, L)	(0, R)	(0, L)	(1, L)
133	(1, M)	(1, M)	(4, M)	(1, R)	---	(2, L)	(0, R)	(1, L)
134	(4, R)	(0, M)	(0, M)	---	(3, M)	(1, M)	(1, M)	(0, R)
135	(4, R)	(0, L)	(0, L)	(1, L)	(2, R)	(2, R)	(0, L)	(2, L)
136	(1, M)	(3, M)	(2, L)	(2, M)	(3, R)	(0, R)	(2, M)	(1, M)
137	(2, M)	(1, R)	(2, L)	(2, R)	(4, L)	(1, R)	(2, M)	(2, R)
138	---	(4, R)	(3, M)	(0, M)	---	(1, R)	(2, R)	(1, L)
139	(2, M)	(4, M)	(4, R)	(4, R)	(2, L)	(0, L)	(0, M)	(0, L)
140	(4, L)	---	(4, L)	(0, R)	(2, L)	(0, M)	(1, L)	(2, M)
141	(0, R)	(0, M)	(0, M)	(1, L)	(3, L)	(0, L)	(0, M)	(0, M)
142	(4, L)	(2, L)	(2, L)	---	(2, M)	(0, R)	(2, R)	(1, M)
143	(2, M)	(0, L)	(2, M)	(2, L)	(3, M)	(0, M)	(0, R)	(0, L)
144	(1, L)	---	(1, L)	(3, L)	---	(2, M)	(1, L)	(1, L)
145	(0, M)	(0, L)	(3, R)	(1, L)	(1, R)	(1, L)	(0, R)	(2, M)
146	(3, M)	---	(4, L)	(3, L)	(4, L)	(1, M)	(2, L)	(0, R)
147	(4, M)	(2, M)	(4, R)	(3, R)	---	(1, M)	(1, L)	(2, R)
148	(4, L)	(4, R)	(0, M)	---	(3, L)	(2, R)	(0, M)	(2, R)
149	(1, M)	(2, L)	(2, M)	---	(0, M)	(0, M)	(0, L)	(0, R)
150	(0, R)	(1, M)	(1, L)	(3, L)	(3, L)	(0, L)	(0, L)	(2, R)
151	---	(4, M)	(2, M)	(2, L)	(4, M)	(1, R)	(0, R)	(1, L)
152	---	(0, L)	(3, R)	(1, M)	(2, M)	(1, M)	(1, M)	(1, R)
153	---	(0, R)	(4, R)	(4, M)	(4, R)	(0, R)	(1, L)	(0, L)
154	(3, L)	(3, R)	(3, M)	(4, L)	(4, L)	(1, R)	(1, L)	(0, L)
155	(4, L)	(2, L)	---	(0, L)	(3, L)	(1, M)	(2, M)	(2, R)
156	(3, M)	(2, L)	(0, M)	(4, M)	(4, R)	(1, R)	(1, L)	(1, R)
157	(2, R)	(4, L)	---	(4, L)	---	(0, R)	(2, L)	(1, R)
158	---	(1, R)	---	---	(4, L)	(2, L)	(0, L)	(0, L)
159	(1, R)	(2, M)	---	(2, R)	(1, M)	(2, R)	(0, L)	(0, M)
160	(4, M)	(3, R)	(4, M)	(4, M)	(2, M)	(2, L)	(2, R)	(0, M)
161	(3, L)	(4, M)	(2, R)	(4, L)	(4, M)	(0, R)	(1, L)	(2, L)
162	(3, M)	(1, M)	---	(0, L)	(3, R)	(2, L)	(2, L)	(1, L)
163	---	(0, M)	(4, R)	---	(0, M)	(2, R)	(2, L)	(0, M)
164	(1, R)	(2, L)	---	---	(1, M)	(0, L)	(1, R)	(0, M)
165	(2, R)	(2, L)	(3, L)	(4, M)	(3, M)	(0, R)	(1, M)	(1, L)

166	---	(2, L)	(3, R)	(1, M)	(1, M)	(0, M)	(1, L)	(1, L)
167	(2, M)	(2, M)	(1, R)	(0, M)	(2, M)	(2, R)	(1, R)	(0, R)
168	(3, R)	(2, M)	(1, L)	---	---	(0, M)	(0, R)	(2, M)
169	---	(0, M)	---	(2, R)	(1, L)	(1, R)	(0, M)	(1, R)
170	(2, M)	(1, L)	(0, R)	(4, L)	(1, L)	(1, M)	(2, M)	(1, L)
171	(0, R)	(3, M)	---	---	(4, M)	(0, R)	(1, M)	(2, R)
172	---	(0, M)	(3, L)	(3, M)	---	(2, L)	(2, R)	(0, R)
173	(2, L)	(1, L)	---	(4, L)	(3, M)	(0, M)	(0, M)	(0, L)
174	(0, R)	(0, R)	(1, L)	(4, M)	(0, R)	(1, L)	(2, R)	(1, R)
175	(0, L)	(1, R)	(0, M)	(2, M)	(4, L)	(0, M)	(2, M)	(2, L)
176	---	---	---	(3, M)	(3, L)	(2, M)	(0, R)	(2, R)
177	(1, M)	(3, M)	(1, M)	(0, R)	(3, L)	(2, M)	(2, L)	(0, R)
178	---	(0, R)	---	---	(0, M)	(0, R)	(2, R)	(2, R)
179	(0, L)	(0, R)	(1, M)	(3, R)	(0, R)	(0, R)	(2, R)	(1, L)
180	(4, M)	(4, L)	---	(3, R)	(3, M)	(0, L)	(2, M)	(2, R)
181	(1, L)	(1, M)	(0, R)	---	(2, M)	(0, M)	(1, M)	(2, R)
182	(3, M)	(2, M)	---	---	(2, M)	(2, L)	(2, M)	(1, R)
183	(3, L)	(3, R)	---	(4, R)	(4, L)	(1, M)	(0, L)	(1, L)
184	---	(0, R)	(1, L)	(2, R)	(0, L)	(1, L)	(1, L)	(0, L)
185	(4, R)	(1, L)	---	(4, L)	(2, M)	(2, R)	(0, R)	(1, R)
186	(3, R)	(3, L)	(0, L)	(4, R)	(1, L)	(2, M)	(1, L)	(0, L)
187	---	(1, M)	(4, M)	(4, M)	(1, M)	(2, M)	(0, M)	(2, L)
188	(3, L)	(3, L)	(1, R)	---	(1, L)	(2, L)	(0, L)	(0, R)
189	(2, R)	(1, R)	(1, L)	(3, L)	---	(1, R)	(1, L)	(1, L)
190	---	---	(3, R)	(2, R)	(0, M)	(2, R)	(1, L)	(0, L)
191	(2, R)	(1, M)	(2, L)	(3, M)	---	(2, L)	(1, M)	(0, L)
192	---	---	(1, M)	(3, L)	---	(0, R)	(1, M)	(1, R)
193	(1, L)	---	(3, L)	(1, M)	---	(0, L)	(2, M)	(2, M)
194	(0, M)	(0, M)	(4, R)	(1, M)	(2, L)	(0, L)	(0, M)	(1, L)
195	(2, R)	(4, L)	(1, L)	(3, R)	(2, M)	(2, R)	(0, M)	(1, M)
196	(3, M)	(1, M)	(0, R)	(0, L)	(0, R)	(0, L)	(2, L)	(1, R)
197	(3, M)	(4, R)	(1, M)	(4, R)	(2, M)	(1, L)	(0, L)	(2, R)
198	(2, L)	(4, L)	(2, M)	(1, R)	(4, L)	(1, L)	(1, M)	(2, M)
199	(0, L)	(3, L)	(1, L)	---	(3, R)	(1, L)	(0, R)	(0, R)
200	(4, L)	(4, M)	---	(3, L)	---	(1, L)	(2, R)	(2, R)
201	(0, L)	(1, M)	(3, M)	(0, L)	---	(0, M)	(1, M)	(0, M)
202	---	(1, L)	(4, M)	(3, L)	(0, L)	(0, L)	(0, R)	(0, L)
203	(0, M)	---	(2, M)	(1, R)	---	(1, M)	(2, L)	(0, M)
204	(4, L)	---	(4, M)	(1, M)	(4, M)	(0, M)	(0, M)	(1, M)
205	(3, M)	(1, L)	(2, M)	(3, L)	(4, L)	(1, M)	(1, L)	(0, L)
206	(0, R)	(1, L)	(4, L)	(3, M)	(0, M)	(2, M)	(2, L)	(2, R)
207	(3, M)	(3, L)	(2, L)	(0, R)	(0, L)	(2, M)	(1, R)	(1, R)
208	(4, L)	(1, M)	(3, M)	(3, M)	(0, L)	(1, M)	(2, L)	(1, L)
209	(3, M)	---	(0, L)	(0, L)	---	(0, R)	(2, M)	(0, M)
210	---	(0, R)	(0, M)	---	---	(0, R)	(2, M)	(0, R)
211	(0, M)	---	(3, R)	(3, L)	(4, M)	(1, R)	(1, R)	(1, M)
212	---	---	(1, R)	(0, R)	(2, L)	(2, L)	(2, R)	(2, R)
213	(2, M)	(0, M)	---	(1, M)	---	(2, M)	(0, M)	(1, L)

214	(2, R)	(3, M)	---	(3, M)	(2, R)	(0, M)	(2, R)	(1, R)
215	(2, M)	(3, L)	(3, R)	(0, M)	(2, M)	(0, R)	(1, L)	(0, R)
216	---	(3, M)	(4, M)	---	(0, M)	(0, R)	(2, L)	(0, R)
217	(3, R)	---	---	---	---	(1, R)	(0, R)	(0, L)
218	(3, M)	(0, L)	(4, L)	(3, R)	(1, M)	(2, M)	(2, R)	(0, L)
219	---	(1, L)	(1, L)	(3, M)	(3, M)	(1, M)	(2, R)	(0, R)
220	(4, R)	(2, R)	---	(4, R)	(3, R)	(2, M)	(2, M)	(2, R)
221	(0, L)	---	(1, M)	---	(3, R)	(2, R)	(2, M)	(0, M)
222	(4, L)	(3, R)	(1, R)	(1, R)	(2, R)	(2, M)	(0, R)	(0, L)
223	(1, L)	(3, L)	(3, R)	(1, R)	(2, R)	(0, M)	(1, L)	(2, M)
224	(3, R)	(1, M)	(2, M)	(0, M)	(1, M)	(0, R)	(0, R)	(2, M)
225	(3, R)	(4, L)	---	(1, L)	(3, R)	(0, L)	(1, L)	(2, M)
226	(3, M)	(3, R)	(2, M)	(2, L)	---	(1, L)	(1, M)	(0, L)
227	(0, L)	(3, R)	(2, L)	(3, R)	(4, L)	(0, R)	(0, L)	(1, M)
228	(4, M)	(2, L)	(1, R)	---	(3, L)	(1, M)	(0, R)	(2, M)
229	(1, R)	---	(1, L)	(2, M)	(4, L)	(0, M)	(1, M)	(0, L)
230	(3, M)	---	(1, L)	---	(1, L)	(2, M)	(1, M)	(1, R)
231	(0, L)	(2, M)	(0, L)	(1, L)	(0, L)	(2, R)	(0, L)	(2, R)
232	---	(1, R)	---	(0, R)	(4, R)	(2, L)	(0, L)	(0, R)
233	(1, L)	(4, R)	(0, M)	(2, R)	(4, L)	(1, L)	(0, R)	(2, R)
234	(2, R)	(4, M)	(4, M)	(1, R)	(4, R)	(1, R)	(2, L)	(0, M)
235	(0, L)	---	(0, M)	(1, M)	(3, L)	(1, L)	(2, R)	(2, R)
236	(0, R)	(4, M)	(4, R)	(0, L)	(2, M)	(2, M)	(2, R)	(0, L)
237	(2, R)	(1, R)	(2, R)	---	(0, M)	(0, M)	(2, R)	(2, M)
238	(4, L)	(0, M)	---	(0, R)	(0, L)	(0, L)	(0, R)	(0, R)
239	(1, M)	(3, R)	(2, M)	(3, L)	(4, M)	(2, R)	(0, R)	(0, M)
240	(1, L)	---	(3, L)	(0, R)	(3, L)	(2, R)	(2, M)	(1, R)
241	(2, M)	(4, R)	(1, R)	(2, M)	(3, R)	(2, R)	(2, M)	(1, L)
242	(4, L)	(4, M)	(4, L)	(2, M)	---	(1, L)	(1, L)	(0, R)
243	(0, R)	(2, M)	---	(0, M)	(0, M)	(2, L)	(2, L)	(0, L)
244	(0, L)	(4, L)	(2, M)	---	(1, M)	(1, M)	(0, L)	(1, L)
245	---	---	(0, R)	(2, M)	(0, M)	(0, R)	(0, L)	(1, R)
246	(1, R)	(2, L)	(4, L)	---	(3, M)	(0, L)	(0, M)	(0, L)
247	(3, M)	(2, L)	---	---	(1, M)	(1, R)	(0, R)	(0, L)
248	(4, R)	(2, R)	(4, L)	(2, R)	(4, L)	(0, M)	(1, R)	(1, M)
249	(0, M)	(2, L)	(4, M)	---	(4, L)	(2, R)	(2, R)	(2, R)
250	(4, L)	(2, M)	(4, M)	---	---	(0, M)	(1, M)	(2, L)
251	(2, L)	(4, M)	(3, R)	---	(3, L)	(1, L)	(2, M)	(1, L)
252	(3, M)	(3, R)	(2, R)	(4, M)	(1, M)	(1, R)	(0, R)	(1, R)
253	---	(4, M)	(3, M)	(3, R)	(0, M)	(1, M)	(1, R)	(0, R)
254	(3, L)	(1, M)	(1, M)	(0, L)	(4, R)	(0, L)	(0, L)	(0, M)
255	(3, L)	(1, M)	(0, M)	(3, R)	(3, L)	(0, R)	(1, R)	(2, R)

Заключение

В статье [5] рассмотрено построение системы автоматов Мили для решения задачи «Умный муравей». В данной работе рассматривается построение системы для решения задачи «Умный муравей - 3», что раньше нигде не встречалось.

Источники

1. Инструкция по созданию plugin’ов к виртуальной лаборатории. http://svn2.assembla.com/svn/not_instrumental_tool/docs/pdf/interface_manual.pdf
2. Классификация абстрактных автоматов. http://ru.wikipedia.org/wiki/классификация_абстрактных_автоматов
3. http://is.ifmo.ru/genalg/
4. Генетические алгоритмы. http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005
5. *Давыдов А. А., Соколов Д. О., Царев Ф. Н., Шалыто А. А.* Применение островного генетического алгоритма для построения автоматов Мура и систем взаимодействующих автоматов Мили на примере задачи об умном муравье / Сборник докладов XI международной конференции по мягким вычислениям и измерениям (SCM'2008). СПб.: СПбГЭТУ. 2008, с. 266-270. http://is.ifmo.ru/genalg/_scm2008_sokolov.pdf

Приложение

Исходные тексты программ

1. `Ant34.java` – класс, реализующий работу с системой автоматов Мили.

```
package individual.task34;

import individual.task34.factory.Ant34Factory;
import individual.task34.tools.BitArray;
import java.util.Random;
import laboratory.common.Visualizable;
import laboratory.common.ga.Individual;
import task.ant.extended.Ant;
import task.ant.extended.ExtendedAnt;

/**
 *
 * @author Александр
 */
public class Ant34 implements Visualizable {

    public final double mu;
    private double fitness = Double.NEGATIVE_INFINITY;
    /**
     * Представление внешнего автомата.
     */
    public final BitArray external;

    /**
     * Представление внутреннего автомата.
     */
    public final BitArray internal;

    /**
     * Количество состояний внешнего автомата.
     */
    public final int externalStatesNumber;

    /**
     * Начальное состояние внешнего автомата
     */
    public final int externalInitialState;

    /**
     * Количество состояний внутреннего автомата.
     */
```

```
    public final int internalStatesNumber;

    /**
     * Начальное состояние внешнего автомата.
     */
    public final int internalInitialState;

    /**
     * Номер текущего состояния внешнего автомата.
     */
    public int currentExternalStateNumber;
    /**
     * Номер текущего состояния внутреннего автомата.
     */
    public int currentInternalStateNumber;
    /**
     * Число входных воздействий.
     */
    public final int ct = 256;

    /**
     * Конструктор.
     * @param externalStatesNumber количество состояний внешнего
автомата
     * @param externalInitialState начальное состояние внешнего
автомата
     * @param internalStatesNumber количество состояний внутреннего
автомата
     * @param internalInitialState начальное состояние внутреннего
автомата
     * @param mu
     */
    public Ant34(int externalStatesNumber, int externalInitialState,
int internalStatesNumber, int internalInitialState, double mu) {
        external = new BitArray(externalStatesNumber, ct);
        this.externalStatesNumber = externalStatesNumber;
        this.externalInitialState = externalInitialState;
        currentExternalStateNumber = externalInitialState;
        internal = new BitArray(internalStatesNumber, ct);
        this.internalStatesNumber = internalStatesNumber;
        this.internalInitialState = internalInitialState;
        currentInternalStateNumber = internalInitialState;
        this.mu = mu;
    }

    /**
     * Метод, выполняющий шаг.
     * @param ant муравей
```

```
     * @return true, если муравей съел яблоко, инача возвращает false
     */
    public boolean move(ExtendedAnt ant) {
        boolean[] visible = ant.F();
        int power = 1;
        int index = 0;
        for (int i = 0; i < visible.length; i++) {
            if (visible[i]) {
                index += power;
            }
            power *= 2;
        }
        int nextState =
external.getNextState(currentExternalStateNumber, index);
        Ant.Action action;
        if (nextState == -1) {
            action = internal.getAction(currentInternalStateNumber,
index);
            nextState =
internal.getNextState(currentInternalStateNumber, index);
            currentInternalStateNumber = nextState;
        } else {
            action = external.getAction(currentExternalStateNumber,
index);
            currentExternalStateNumber = nextState;
        }
        if (action == Ant.Action.L) {
            ant.L();
        } else if (action == Ant.Action.R) {
            ant.R();
        } else if (action == Ant.Action.M) {
            ant.M();
            if (visible[0]) {
                return true;
            }
        }
        return false;
    }

    public Object[] getAttributes() {
        return new Object[]{new Ant34Mover(this, mu)};
    }

    public double fitness() {
        if (fitness != Double.NEGATIVE_INFINITY) {
            return fitness;
        }
        fitness = 0;
```

```
        for (int j = 0; j < Ant34Factory.attempts; j++) {
            ExtendedAnt ant = new ExtendedAnt(mu);
            for (int i = 0; i < Ant.NUMBER_STEPS; i++) {
                if (move(ant)) {
                    fitness++;
                }
            }
        }
        fitness /= (double) Ant34Factory.attempts;
        return fitness;
    }

    public Individual mutate(Random r) {
        Ant34 ant = new Ant34(externalStatesNumber,
r.nextInt(externalStatesNumber), internalStatesNumber,
r.nextInt(internalStatesNumber), mu);

        System.arraycopy(external.array, 0, ant.external.array, 0,
external.array.length);
        for (int i = 2; i < externalStatesNumber; i++) {
            for (int j = 0; j < ct; j++) {
                if (r.nextDouble() < 0.1) {
                    ant.external.setNextState(i, j,
r.nextInt(externalStatesNumber + 1) - 1);
                }
                if (r.nextDouble() < 0.1) {
                    switch (r.nextInt(3)) {
                        case 0:
                            ant.external.setAction(i, j,
Ant.Action.L);
                            break;
                        case 1:
                            ant.external.setAction(i, j,
Ant.Action.M);
                            break;
                        case 2:
                            ant.external.setAction(i, j,
Ant.Action.R);
                            break;
                    }
                }
            }
        }

        System.arraycopy(internal.array, 0, ant.internal.array, 0,
internal.array.length);
        for (int i = 2; i < internalStatesNumber; i++) {
            for (int j = 0; j < ct; j++) {
```

```
                if (r.nextDouble() < 0.1) {
                    ant.internal.setNextState(i, j,
r.nextInt(internalStatesNumber));
                }
                if (r.nextDouble() < 0.1) {
                    switch (r.nextInt(3)) {
                        case 0:
                            ant.internal.setAction(i, j,
Ant.Action.L);
                            break;
                        case 1:
                            ant.internal.setAction(i, j,
Ant.Action.M);
                            break;
                        case 2:
                            ant.internal.setAction(i, j,
Ant.Action.R);
                            break;
                    }
                }
            }
        }
        return ant;
    }

    public Individual[] crossover(Individual p, Random r) {
        Ant34[] sons = new Ant34[2];
        Ant34 parent = (Ant34) p;

        sons[0] = new Ant34(externalStatesNumber,
r.nextInt(internalStatesNumber), internalStatesNumber,
r.nextInt(internalStatesNumber), mu);
        sons[1] = new Ant34(externalStatesNumber,
r.nextInt(internalStatesNumber), internalStatesNumber,
r.nextInt(internalStatesNumber), mu);

        while (true) {
            int externalNumber = r.nextInt(external.array.length);
            int externalState = externalNumber / 2560, externalIndex =
(externalNumber % 2560) / 10;

            int internalNumber = r.nextInt(internal.array.length);
            int internalState = internalNumber / 2560, internalIndex =
(internalNumber % 2560) / 10;

            System.arraycopy(external.array, 0,
sons[0].external.array, 0, externalNumber);
```

```
            System.arraycopy(parent.external.array, 0,
sons[1].external.array, 0, externalNumber);

            System.arraycopy(external.array, externalNumber,
sons[1].external.array, externalNumber, external.array.length -
externalNumber);
            System.arraycopy(parent.external.array, externalNumber,
sons[0].external.array, externalNumber, external.array.length -
externalNumber);


            System.arraycopy(internal.array, 0,
sons[0].internal.array, 0, internalNumber);
            System.arraycopy(parent.internal.array, 0,
sons[1].internal.array, 0, internalNumber);

            System.arraycopy(internal.array, internalNumber,
sons[1].internal.array, internalNumber, internal.array.length -
internalNumber);
            System.arraycopy(parent.internal.array, internalNumber,
sons[0].internal.array, internalNumber, internal.array.length -
internalNumber);

            if (sons[0].external.getAction(externalState,
externalIndex) != null &&
                    sons[1].external.getAction(externalState,
externalIndex) != null &&
                    sons[0].internal.getNextState(internalState,
internalIndex) != -1 &&
                    sons[0].internal.getAction(internalState,
internalIndex) != null &&
                    sons[1].internal.getNextState(internalState,
internalIndex) != -1 &&
                    sons[1].internal.getAction(internalState,
internalIndex) != null) {
                 break;
            }

        }
        return sons;
    }

    public int compareTo(Individual o) {
        if (fitness() > o.fitness()) {
            return -1;
        } else {
            return 1;
        }
```

```
    }

    public String toString() {
        String res="External:";
        res += "\n Start: "+externalInitialState;
        res += "\n Internal:";
        res += "\n Start: "+internalInitialState;
        res += "\n\t";
        for (int i = 0; i < externalStatesNumber; i++) {
            res += i+"\t";
        }
        res += "\t";
        for (int i = 0; i < internalStatesNumber; i++) {
            res += i+"\t";
        }
        res += "\n";

        for (int j = 0; j < ct; j++) {
            res += j+"\t";
            for (int i = 0; i < externalStatesNumber; i++) {
                int nextState = external.getNextState(i, j);
                if (nextState != -1)
                    res += ("+nextState+", "+external.getAction(i,
j)+")\t";
                else
                    res += "---\t";
            }
            res += "\t";
            for (int i = 0; i < internalStatesNumber; i++) {
                res += ("+internal.getNextState(i, j)+",
"+internal.getAction(i, j)+")\t";
            }
            res += "\n";
        }
        return res;
    }
}
```

2. `Ant34FactoryLoader.java` — класс, реализующий загрузку модуля в виртуальную лабораторию.

```
package individual.task34;

import individual.task34.factory.Ant34Factory;
import java.util.jar.JarFile;
import laboratory.util.AbstractLoader;

/**
 *
```

```
 * @author Александр
 */
public class Ant34FactoryLoader extends AbstractLoader<Ant34Factory> {

    public Ant34FactoryLoader(JarFile jarFile) {
        super(jarFile, "automaton.conf");
    }

    public Ant34Factory load(Object... args) {
        Ant34Factory.attempts = properties.getInt("count.attempts");
        return new
Ant34Factory(properties.getInt("external.count.states"),
properties.getInt("internal.count.states"),
properties.getDouble("mu"));
    }
}
```

3. Ant34Mover.java — класс, реализующий интерфейс работы с управляющим объектом (муравьем).

```
package individual.task34;

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

/**
 *
 * @author Александр
 */
public class Ant34Mover implements ExtendedAntTask.Mover {

    private final double mu;
    public final Ant34 automaton;
    private ExtendedAnt ant;

    public boolean move() {
        return automaton.move(ant);
    }

    public void restart(Ant ant) {
        this.ant = (ExtendedAnt) ant;
    }

    public Ant34Mover(Ant34 automaton, double mu) {
        this.mu = mu;
        this.automaton = automaton;
    }
```

```
    public double getMu() {
        return mu;
    }
}
```

4. BitArray.java — класс, реализующий работу с представлением автомата в виде битовой строки.

```
package individual.task34.tools;

import task.ant.extended.Ant;

/**
 * Класс, представляющий автомат в виде битовой строки.
 * @author Александр
 */
public class BitArray {

    /**
     *  Номер следующего состояния записывается в 8 бит
     *  (ноль означает, что ссылки нет)
     */
    public static int NUMBER_LENGTH = 8;
    /**
     *  Действие при переходе кодируется 2 битами.
     */
    public static int ACTION_LENGTH = 2;
    /**
     *  Число "входящих воздествий"
     */
    private final int ct;
    /**
     *  Число состояний
     */
    private final int countStates;
    /**
     *  Битовый массив - представление автомата
     */
    public final boolean[] array;

    /**
     * Конструктор
     * @param countStates количество состояний автомата
     * @param ct число "входных воздействий"
     */
    public BitArray(int countStates, int ct) {
        if (countStates > 254) {
```

```
            throw new RuntimeException("Maximum number of states -
254!");
        }
        this.ct = ct;
        this.countStates = countStates;
        array = new boolean[countStates * (NUMBER_LENGTH +
ACTION_LENGTH) * ct];
    }

    /**
     * Установка действия при переходе.
     * @param stateNumber номер состояния
     * @param index номер входного воздействия
     * @param action новое действие
     */
    public void setAction(int stateNumber, int index, Ant.Action
action) {
        if (stateNumber > 254) {
            throw new RuntimeException("State number out of range!");
        }
        int begin = stateNumber * (NUMBER_LENGTH + ACTION_LENGTH) * ct
+ (NUMBER_LENGTH + ACTION_LENGTH) * index + NUMBER_LENGTH;
        if (action == Ant.Action.L) {
            array[begin] = false;
            array[begin + 1] = true;
        } else if (action == Ant.Action.M) {
            array[begin] = true;
            array[begin + 1] = false;
        } else if (action == Ant.Action.R) {
            array[begin] = true;
            array[begin + 1] = true;
        } else {
            array[begin] = false;
            array[begin + 1] = false;
        }
    }

    /**
     * Установка конечного состояния при переходе
     * @param stateNumber номер состония
     * @param index номер входного воздействия
     * @param nextState новый номер конечного состония
     */
    public void setNextState(int stateNumber, int index, int
nextState) {
        nextState++;
        int begin = stateNumber * (NUMBER_LENGTH + ACTION_LENGTH) * ct
+ (NUMBER_LENGTH + ACTION_LENGTH) * index;
```

```
        for (int i = 0; i < NUMBER_LENGTH; i++) {
            if (nextState % 2 == 1) {
                array[begin + i] = true;
            } else {
                array[begin + i] = false;
            }
            nextState /= 2;
        }
    }

    /**
     * Метод возвращает действие при переходе
     * @param stateNumber номер состояния
     * @param index номер входного воздействия
     * @return действие при переходе
     */
    public Ant.Action getAction(int stateNumber, int index) {
        int begin = stateNumber * (NUMBER_LENGTH + ACTION_LENGTH) * ct
+ (NUMBER_LENGTH + ACTION_LENGTH) * index + NUMBER_LENGTH;
        if (array[begin] == true) {
            if (array[begin + 1] == true) {
                return Ant.Action.R;
            } else {
                return Ant.Action.M;
            }
        } else {
            if (array[begin + 1] == true) {
                return Ant.Action.L;
            } else {
                return null;
            }
        }
    }

    /**
     * Возврщает номер конечного состония при переходе
     * @param stateNumber номер состояния
     * @param index номер входного воздействия
     * @return номер конечного состояния
     */
    public int getNextState(int stateNumber, int index) {
        int nextState = 0;
        int power = 1;
        int begin = stateNumber * (NUMBER_LENGTH + ACTION_LENGTH) * ct
+ (NUMBER_LENGTH + ACTION_LENGTH) * index;
        for (int i = 0; i < NUMBER_LENGTH; i++) {
            if (array[begin + i]) {
                nextState += power;
```

```
            }
            power *= 2;
        }
        if (nextState == 0 || nextState > countStates) {
            return -1;
        }
        nextState--;
        return nextState;
    }
}
```

5. Ant34Factory.java – класс, реализующий создание произвольной особи.

```
package individual.task34.factory;

import individual.task34.Ant34;
import java.util.Random;
import laboratory.common.ga.IndividualFactory;
import task.ant.extended.Ant;

/**
 *
 * @author Александр
 */
public class Ant34Factory implements IndividualFactory {

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

    public Ant34Factory(int externalStatesNumber, int
internalStateNumber, double mu) {
        this.externalStatesNumber = externalStatesNumber;
        this.internalStatesNumber = internalStateNumber;
        this.mu = mu;
    }

    public Ant34 randomIndividual() {
        Ant34 ant = new Ant34(externalStatesNumber,
r.nextInt(externalStatesNumber), internalStatesNumber,
r.nextInt(internalStatesNumber), mu);

        for (int i = 0; i < externalStatesNumber; i++) {
            for (int j = 0; j < 256; j++) {
                switch (r.nextInt(3)) {
                    case 0:
                        ant.external.setAction(i, j, Ant.Action.L);
```

```
                        break;
                    case 1:
                        ant.external.setAction(i, j, Ant.Action.M);
                        break;
                    case 2:
                        ant.external.setAction(i, j, Ant.Action.R);
                        break;
                }
                ant.external.setNextState(i,                          j,
r.nextInt(externalStatesNumber + 1) - 1);
            }
        }
        for (int i = 0; i < internalStatesNumber; i++) {
            for (int j = 0; j < 256; j++) {
                switch (r.nextInt(3)) {
                    case 0:
                        ant.internal.setAction(i, j, Ant.Action.L);
                        break;
                    case 1:
                        ant.internal.setAction(i, j, Ant.Action.M);
                        break;
                    case 2:
                        ant.internal.setAction(i, j, Ant.Action.R);
                        break;
                }
                ant.internal.setNextState(i,                          j,
r.nextInt(internalStatesNumber));
            }
        }
        return ant;
    }
}
```

Конфигурационные файлы

1. `build.xml` – скрипт для сборки модуля.
2. `build.properties` – параметры сборки модуля.
3. `automaton.conf` – файл конфигурации задания.

   Параметры:

   - `external.count.states` – количество состояний внешнего автомат;
   - `internal.count.states` – количество состояний внутреннего автомата;
   - `count.attempts` – количество запусков муравья;
   - `mu` – вероятность расположения еды в каждой клетке.

4. `plugin.properties` – файл конфигурации модуля.

   Параметры:

   - `jar` – имя jar-файла;
   - `main.class` – класс, реализующий загрузку модуля в виртуальную лабораторию;
   - `comment` – комментарий к модулю;
   - `extension.name` – название модуля (так же необходимо для корректной работы эмулятора);
   - `resources` – относительный путь к папке где лежит файл конфигурации задания;
   - `filter` – определение типа решаемой задачи («Умный муравей» или «Умный муравей - 3»);
   - `graphic.settings` – графические параметры модуля.