Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

А.И. Калиниченко

Отчет по лабораторной работе

«Использование генетических алгоритмов для построения управляющих автоматов»

Вариант №18

Санкт-Петербург
2009

Оглавление

Введение

Цель лабораторной работы – изучение 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, которая позволяет реализовать генетические алгоритмы и особи для них в виде плагинов.

1. Постановка задачи

Построить с помощью генетических алгоритмов автомат Мура, решающий задачу об "Умном муравье - 3" [2]. Использовать представление автоматов с помощью битовых строк. Способ скрещивания выбрать самостоятельно. Использовать клеточный генетический алгоритм и метод рулетки для генерации очередного поколения.

1.1. 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 перед собой восемь клеток (рис. 1) 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.

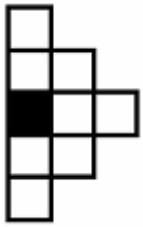

Рис. 1. Видимая область муравья

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

1.2. Автомат Мура

В рассматриваемом варианте автомат для управления муравьем является автоматом Мура – совокупностью пяти объектов $A = \{S, X, Y, \delta, \mu\}$, где S – множество вершин, X – множество входных воздействий, Y – множество выходных воздействий, δ – отображение $S \times X \to S$, μ – отображение $S \to Y$. Пример автомата Мура изображен на рис. 2.

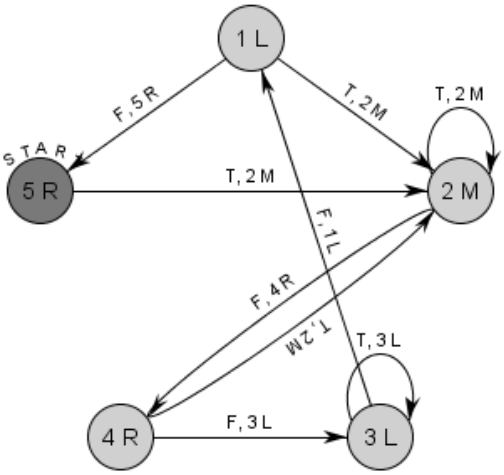

Рис. 2. Автомат Мура

2. Генетический алгоритм

Виртуальная лаборатория состоит из ядра и подключаемых модулей. Для решения поставленной задачи требуется создать два модуля. Первый – для реализации клеточного генетического алгоритма и метода рулетки для генерации очередного поколения. Второй – для реализации «особи» – конечного автомата Мура, решающего задачу об «Умном муравье - 3». Для «особи» необходимо реализовать операции мутации и скрещивания.

2.1. Используемое представление автоматов

Каждый автомат представляется в виде битовой строки. Для автомата из N состояний она имеет структуру, показанную на 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 закодировано действие, которое выполняет муравей при попадании в это состояние, и переходы в порядке входных воздействий (рис. 4).

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

Рис. 4. Представление состояний в виде одномерного массива

Действие кодируется двумя битами:

- 00 – ничего не делать;
- 01 – поворот налево;
- 10 – шаг вперед;
- 11 – поворот направо.

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 и соответственно 256 входных воздействий.

Каждый переход кодируется восемью битами (рис. 5).

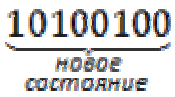

Рис. 5. Представление перехода в виде битовой строки

Эти восемь бит являются двоичным представлением номера нового состояния (от 0 до 255). Состояния конечного автомата нумеруются, начиная с единицы. Нулевое значение, а также значение большее, чем число состояние автомата, означает, что перехода нет.

2.2. Метод скрещивания

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

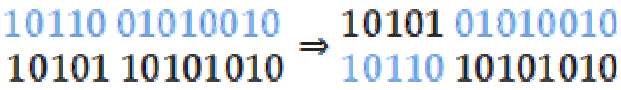

Рис. 6. Одноточечный кроссовер

2.3. Метод мутации

При мутации в каждом переходе номер нового состояния и действие меняются на произвольные с вероятностью один процент.

2.4. Метод генерации очередного поколения

Для генерации очередного поколения используется клеточный генетический алгоритм (*Cellular Genetic Algorithms*) и метод рулетки [3,4].

Cellular Genetic Algorithms – модель параллельных генетических алгоритмов. Пусть дано 2500 процессов, расположенных на сетке размером 50×50 ячеек, замкнутой, как показано на рис. 7 (левая сторона замыкается с правой, а верхняя с нижней. В результате получается тор).

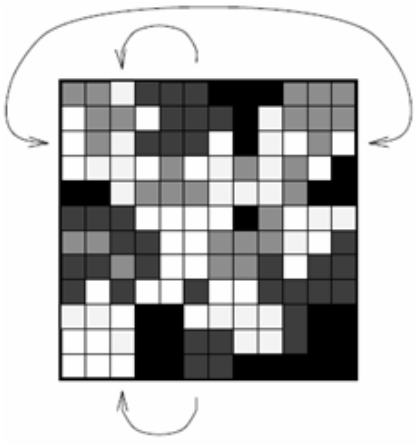

Рис. 7. Модель клеточного генетического алгоритма

Каждый процесс может взаимодействовать только с четырьмя своими соседями (сверху, снизу, слева, справа). В каждой ячейке находится ровно одна особь. Каждый процесс будет выбирать лучшую особь среди своих соседей, скрещивать с ней особь из своей ячейки и одного полученного ребенка помещать в свою ячейку вместо родителя.

Метод рулетки – способ отбора новых особей. Пусть особи располагаются на колесе рулетки, так что размер сектора каждой особи пропорционален ее приспособленности. Изначально промежуточная популяция пуста. Запуская рулетку N раз, выберем требуемое число особей для записи в промежуточную популяцию. Ни одна выбранная особь не удаляется с рулетки. Промежуточная популяция – это набор особей, которые получили право размножаться. После отбора особи промежуточной популяции случайным образом разбиваются на пары. Каждая из них с вероятностью один процент скрещивается методом описанным выше.

2.5. Способ вычисления функции приспособленности

Функция приспособленности вычисляется по следующей формуле:

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

где $\boldsymbol{k}$ –число запусков муравья, $\boldsymbol{a_i}$ – число съеденных яблок на i-ой попытке.

3. Построенный автомат

3.1. Графики функции приспособленности

На рис. 8 приведен график максимального значения функции приспособленности.

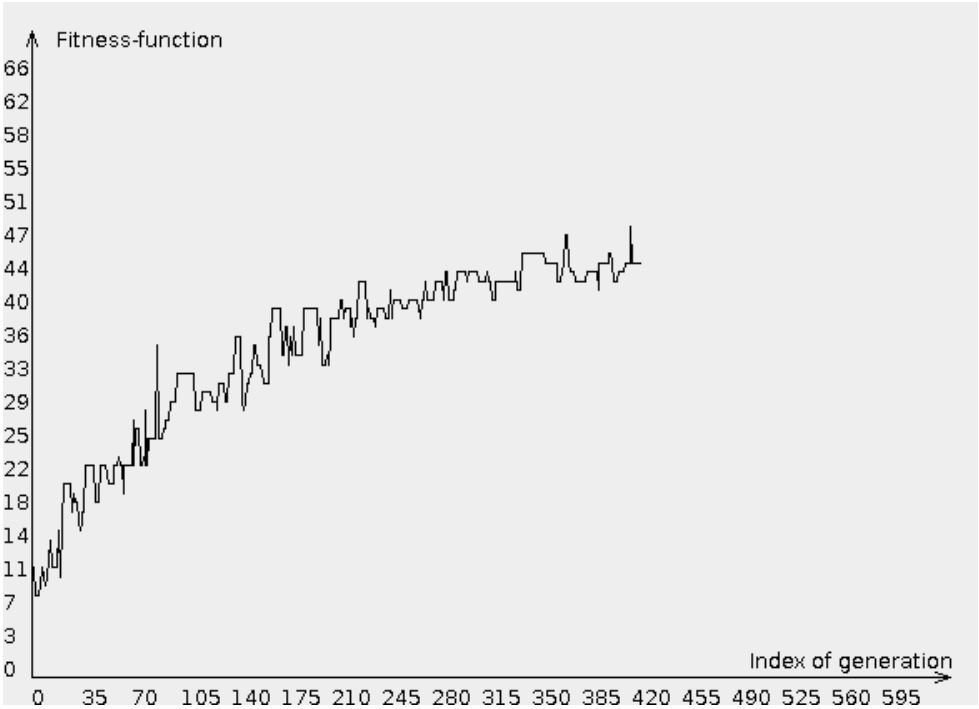

Рис. 8. График максимального значения функции приспособленности

На рис. 9 приведен график среднего значения функции приспособленности.

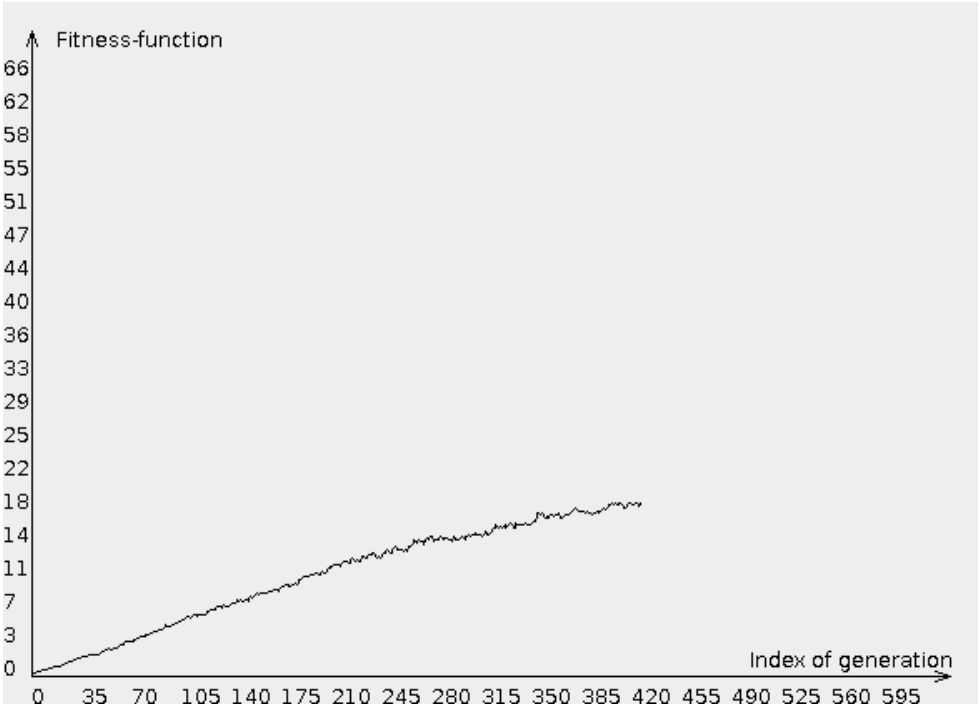

Рис. 9. График среднего значения функции приспособленности

3.2. Таблица переходов полученного автомата

Полученный конечный автомат представлен в виде таблицы переходов (листинг 1). По горизонтали расположены состояния автомата. По вертикали расположены входные воздействия.

Переход записан в виде номера нового состояния. Запись «---» означает, что перехода нет.

Действие может принимать одно из следующих значений:

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

Для каждого состояния действие указано в шапке таблицы.

Автомат:

1. Начальное состояние – 4.
2. Число состояний – 5.

Число съедаемых яблок – 49.

Листинг 1. Таблица переходов полученного автомата

	(0, L)	*(1, M)*	*(2, M)*	*(3, L)*	*(4, M)*
0	*(0)*	*(2)*	*(2)*	*(2)*	*(1)*
1	*(2)*	*(1)*	*(2)*	*(2)*	*(2)*
2	*(2)*	*(4)*	*(2)*	*(1)*	*(4)*
3	*(4)*	*(3)*	*(4)*	*(4)*	*(1)*
4	*(0)*	*(0)*	*(0)*	*(2)*	*(0)*
5	*(4)*	*(4)*	*(1)*	*(3)*	*(2)*
6	*(4)*	*(4)*	*(2)*	*(1)*	*(4)*
7	*(1)*	*(4)*	*(0)*	*(3)*	*(2)*
8	*(3)*	*(3)*	*(0)*	*(2)*	*---*
9	*(2)*	*---*	*(0)*	*(0)*	*(1)*
10	*---*	*(3)*	*(2)*	*(2)*	*(2)*
11	*(1)*	*(4)*	*(1)*	*(1)*	*(4)*
12	*(2)*	*(1)*	*(0)*	*(0)*	*(1)*
13	*(1)*	*(4)*	*(1)*	*(0)*	*(4)*
14	*(2)*	*---*	*(2)*	*(3)*	*---*
15	*(3)*	*(0)*	*(4)*	*---*	*(0)*
16	*(3)*	*(3)*	*(3)*	*(0)*	*(2)*
17	*(4)*	*(1)*	*(0)*	*(4)*	*(0)*
18	*(1)*	*(0)*	*(1)*	*(3)*	*(4)*
19	*(4)*	*(2)*	*(4)*	*---*	*(0)*
20	*---*	*(4)*	*(0)*	*(3)*	*(3)*
21	*(3)*	*(1)*	*(4)*	*(4)*	*(0)*
22	*(4)*	*(4)*	*(0)*	*(1)*	*(2)*
23	*---*	*(2)*	*(1)*	*(2)*	*(4)*
24	*(3)*	*(2)*	*(4)*	*(2)*	*(4)*
25	*(1)*	*---*	*(3)*	*(2)*	*(2)*
26	*(3)*	*(4)*	*(1)*	*---*	*(2)*
27	*(3)*	*(4)*	*(4)*	*(1)*	*(4)*
28	*(3)*	*---*	*(3)*	*(4)*	*(3)*

29	*(2)*	*---*	*(1)*	*(0)*	*---*
30	*(4)*	*(2)*	*(0)*	*(0)*	*(4)*
31	*(3)*	*(0)*	*(2)*	*(2)*	*(1)*
32	*(0)*	*(0)*	*(0)*	*(3)*	*---*
33	*(3)*	*(2)*	*(2)*	*(2)*	*(2)*
34	*(1)*	*(2)*	*(1)*	*(3)*	*---*
35	*(3)*	*(4)*	*(2)*	*(1)*	*(2)*
36	*(3)*	*---*	*(2)*	*(4)*	*(2)*
37	*(0)*	*---*	*---*	*(2)*	*(3)*
38	*(4)*	*(2)*	*(0)*	*(4)*	*(3)*
39	*(3)*	*(2)*	*(0)*	*(3)*	*(3)*
40	*(3)*	*(1)*	*(0)*	*(2)*	*---*
41	*(0)*	*(4)*	*(0)*	*---*	*(2)*
42	*(1)*	*(0)*	*(4)*	*---*	*(1)*
43	*(3)*	*(4)*	*(1)*	*(4)*	*---*
44	*(0)*	*---*	*---*	*(2)*	*(1)*
45	*---*	*(0)*	*(2)*	*(1)*	*(3)*
46	*(0)*	*(3)*	*(0)*	*(3)*	*(3)*
47	*(1)*	*(1)*	*(0)*	*(3)*	*(3)*
48	*(3)*	*(1)*	*(4)*	*(4)*	*(0)*
49	*(2)*	*---*	*(2)*	*---*	*(3)*
50	*(4)*	*(1)*	*---*	*(2)*	*(3)*
51	*(4)*	*(2)*	*(2)*	*---*	*(1)*
52	*---*	*(3)*	*(3)*	*(2)*	*---*
53	*(3)*	*(1)*	*(1)*	*(3)*	*(1)*
54	*---*	*(3)*	*(4)*	*(4)*	*(0)*
55	*---*	*(1)*	*(2)*	*(3)*	*(0)*
56	*---*	*(3)*	*(2)*	*(0)*	*---*
57	*(2)*	*(4)*	*(4)*	*(0)*	*(1)*
58	*(4)*	*(2)*	*(1)*	*(4)*	*(4)*
59	*(4)*	*(4)*	*(2)*	*(1)*	*(1)*
60	*(1)*	*(4)*	*(2)*	*(2)*	*(3)*
61	*(0)*	*(3)*	*(4)*	*---*	*(3)*
62	*(0)*	*(3)*	*(3)*	*(0)*	*(0)*
63	*(1)*	*(2)*	*(4)*	*(4)*	*(2)*
64	*(0)*	*(2)*	*(4)*	*(0)*	*---*
65	*(1)*	*(2)*	*(2)*	*(4)*	*(2)*
66	*(4)*	*(1)*	*(1)*	*(2)*	*---*
67	*---*	*(4)*	*(2)*	*---*	*(0)*
68	*(3)*	*(0)*	*(1)*	*(3)*	*(1)*
69	*(0)*	*(4)*	*(3)*	*(0)*	*(3)*
70	*(1)*	*(3)*	*(4)*	*(2)*	*(4)*
71	*(0)*	*(0)*	*(0)*	*(1)*	*(0)*
72	*(1)*	*(2)*	*(2)*	*(4)*	*(1)*
73	*(3)*	*(3)*	*(4)*	*(0)*	*(3)*
74	*(0)*	*(1)*	*(0)*	*(4)*	*(4)*
75	*(4)*	*---*	*(0)*	*(1)*	*(3)*
76	*(0)*	*(1)*	*(0)*	*(2)*	*(3)*
77	*---*	*(0)*	*(0)*	*(1)*	*---*
78	*(3)*	*(2)*	*(4)*	*(2)*	*(4)*
79	*(2)*	*---*	*(4)*	*(1)*	*(3)*
80	*(4)*	*(0)*	*(4)*	*(4)*	*(0)*
81	*(4)*	*(3)*	*(4)*	*(3)*	*(4)*
82	*(1)*	*(2)*	*(0)*	*---*	*(2)*
83	*(2)*	*(3)*	*(4)*	*(4)*	*---*

84	*(4)*	*---*	*(4)*	*(4)*	*(4)*
85	*(3)*	*(3)*	*(1)*	*(1)*	*---*
86	*(1)*	*(2)*	*(1)*	*(2)*	*(1)*
87	*(4)*	*(4)*	*(4)*	*(1)*	*(4)*
88	*(1)*	*(3)*	*(2)*	*(2)*	*(3)*
89	*---*	*---*	*(0)*	*(0)*	*(3)*
90	*---*	*(4)*	*(3)*	*---*	*(1)*
91	*(4)*	*---*	*(1)*	*(1)*	*---*
92	*---*	*(4)*	*(1)*	*(4)*	*(4)*
93	*---*	*(1)*	*---*	*(3)*	*(0)*
94	*(1)*	*(0)*	*---*	*(0)*	*(4)*
95	*(4)*	*(4)*	*(0)*	*(1)*	*(2)*
96	*(4)*	*(1)*	*(3)*	*(2)*	*(2)*
97	*---*	*(2)*	*(3)*	*(2)*	*(0)*
98	*---*	*(4)*	*(0)*	*(2)*	*---*
99	*(1)*	*(2)*	*(4)*	*(4)*	*---*
100	*(0)*	*(4)*	*(3)*	*(0)*	*(2)*
101	*(3)*	*(4)*	*(0)*	*(2)*	*(2)*
102	*(4)*	*(4)*	*---*	*(2)*	*(2)*
103	*(3)*	*(2)*	*(4)*	*---*	*(3)*
104	*(0)*	*(1)*	*(1)*	*(4)*	*(3)*
105	*(1)*	*(0)*	*(2)*	*---*	*(4)*
106	*---*	*(2)*	*(1)*	*(3)*	*(1)*
107	*(1)*	*(0)*	*(0)*	*(0)*	*---*
108	*(2)*	*(0)*	*(1)*	*(3)*	*(2)*
109	*(4)*	*(1)*	*(0)*	*(2)*	*---*
110	*---*	*(1)*	*(4)*	*(3)*	*(1)*
111	*(2)*	*(3)*	*(0)*	*(3)*	*(2)*
112	*---*	*(1)*	*---*	*(4)*	*(1)*
113	*---*	*(1)*	*(1)*	*(4)*	*(4)*
114	*(3)*	*(2)*	*---*	*(1)*	*(4)*
115	*(0)*	*(3)*	*(3)*	*(1)*	*---*
116	*(2)*	*(1)*	*(1)*	*(3)*	*(4)*
117	*(1)*	*(1)*	*(4)*	*(0)*	*(2)*
118	*---*	*(4)*	*(4)*	*(0)*	*(0)*
119	*---*	*---*	*(1)*	*(4)*	*(3)*
120	*(2)*	*(3)*	*(0)*	*(2)*	*(0)*
121	*(0)*	*(2)*	*(0)*	*(0)*	*(1)*
122	*---*	*(4)*	*(0)*	*(3)*	*---*
123	*(0)*	*---*	*(1)*	*(4)*	*(4)*
124	*(0)*	*---*	*(2)*	*(2)*	*(2)*
125	*(4)*	*(0)*	*---*	*(3)*	*---*
126	*(4)*	*(2)*	*---*	*(4)*	*(3)*
127	*(0)*	*---*	*(1)*	*(0)*	*(3)*
128	*(4)*	*(2)*	*(1)*	*(2)*	*(1)*
129	*(3)*	*(3)*	*(1)*	*(1)*	*(0)*
130	*(3)*	*(2)*	*(4)*	*(2)*	*---*
131	*(2)*	*(0)*	*(3)*	*(3)*	*(2)*
132	*(3)*	*---*	*(4)*	*(4)*	*(2)*
133	*(1)*	*---*	*(0)*	*(4)*	*(1)*
134	*(0)*	*(4)*	*(3)*	*(1)*	*(1)*
135	*(0)*	*(2)*	*(2)*	*(3)*	*---*
136	*---*	*(1)*	*(1)*	*(1)*	*(4)*
137	*(4)*	*(3)*	*(1)*	*---*	*(3)*
138	*(0)*	*(2)*	*(4)*	*(3)*	*(3)*

139	*(3)*	*(3)*	*(2)*	*(0)*	*(2)*
140	*(0)*	*(1)*	*(4)*	*(0)*	*(3)*
141	*(0)*	*(2)*	*(0)*	*(3)*	*(0)*
142	*(4)*	*(4)*	*(4)*	*(0)*	*(0)*
143	*---*	*(2)*	*(3)*	*(3)*	*(2)*
144	*(3)*	*(3)*	*(2)*	*(2)*	*(4)*
145	*(4)*	*(3)*	*(1)*	*(4)*	*---*
146	*(0)*	*(3)*	*(2)*	*(2)*	*(2)*
147	*(0)*	*(1)*	*(0)*	*(4)*	*(1)*
148	*---*	*(0)*	*(0)*	*---*	*---*
149	*(2)*	*(2)*	*(1)*	*(3)*	*(0)*
150	*(1)*	*(4)*	*(4)*	*---*	*---*
151	*(0)*	*---*	*(0)*	*(0)*	*---*
152	*(1)*	*---*	*(2)*	*(0)*	*(1)*
153	*(0)*	*(3)*	*(4)*	*(3)*	*(3)*
154	*(2)*	*---*	*(2)*	*(3)*	*(3)*
155	*(0)*	*(4)*	*(1)*	*---*	*---*
156	*(0)*	*(0)*	*(4)*	*(0)*	*(3)*
157	*(3)*	*(2)*	*(3)*	*(2)*	*(0)*
158	*(0)*	*(4)*	*(2)*	*(4)*	*(1)*
159	*(1)*	*---*	*(0)*	*(2)*	*(1)*
160	*---*	*(0)*	*(3)*	*(1)*	*---*
161	*(0)*	*(0)*	*(4)*	*(4)*	*(2)*
162	*(4)*	*(0)*	*(0)*	*(4)*	*(1)*
163	*(2)*	*(0)*	*(3)*	*(4)*	*(3)*
164	*(1)*	*(1)*	*(2)*	*---*	*---*
165	*(2)*	*(4)*	*(3)*	*(2)*	*(2)*
166	*(3)*	*(2)*	*(4)*	*(1)*	*(3)*
167	*(0)*	*(4)*	*---*	*(4)*	*(4)*
168	*(3)*	*(1)*	*---*	*(1)*	*(2)*
169	*---*	*---*	*(2)*	*(3)*	*(3)*
170	*(1)*	*(1)*	*(4)*	*(3)*	*---*
171	*(0)*	*(1)*	*(4)*	*(0)*	*(2)*
172	*(0)*	*(2)*	*---*	*(2)*	*(1)*
173	*(3)*	*---*	*(4)*	*(0)*	*(0)*
174	*(0)*	*(2)*	*(1)*	*(3)*	*(3)*
175	*(2)*	*(2)*	*(0)*	*---*	*---*
176	*(3)*	*---*	*(2)*	*---*	*(0)*
177	*(0)*	*(2)*	*(4)*	*(4)*	*(1)*
178	*(0)*	*(2)*	*(0)*	*(3)*	*(2)*
179	*---*	*(3)*	*(3)*	*(3)*	*(4)*
180	*(2)*	*(3)*	*(4)*	*(1)*	*(4)*
181	*(3)*	*(2)*	*(1)*	*(3)*	*(3)*
182	*(2)*	*(0)*	*(0)*	*(1)*	*(1)*
183	*(2)*	*(4)*	*(1)*	*(1)*	*(2)*
184	*(2)*	*(1)*	*---*	*(0)*	*---*
185	*(2)*	*(1)*	*---*	*(3)*	*(2)*
186	*(2)*	*(0)*	*(2)*	*(2)*	*(1)*
187	*---*	*(3)*	*(1)*	*(0)*	*(1)*
188	*(4)*	*(0)*	*(2)*	*(4)*	*(0)*
189	*---*	*(2)*	*(1)*	*(0)*	*(1)*
190	*(0)*	*(3)*	*---*	*(0)*	*---*
191	*(3)*	*(2)*	*(2)*	*(4)*	*(0)*
192	*(4)*	*(0)*	*(1)*	*(0)*	*---*
193	*(2)*	*(0)*	*(4)*	*(0)*	*(4)*

194	*(4)*	*(3)*	*(3)*	*(1)*	*(4)*
195	*---*	*(4)*	*(4)*	*(2)*	*(0)*
196	*(2)*	*(2)*	*(0)*	*(4)*	*(2)*
197	*---*	*(2)*	*(1)*	*(4)*	*---*
198	*(0)*	*(1)*	*(1)*	*(0)*	*(1)*
199	*(3)*	*---*	*(2)*	*(0)*	*(4)*
200	*(4)*	*(4)*	*(2)*	*(2)*	*(4)*
201	*(1)*	*(3)*	*(2)*	*(0)*	*(1)*
202	*(2)*	*(3)*	*(4)*	*---*	*---*
203	*(2)*	*(1)*	*(0)*	*(0)*	*---*
204	*(1)*	*(0)*	*---*	*(4)*	*---*
205	*(1)*	*(3)*	*(3)*	*---*	*(4)*
206	*(2)*	*(0)*	*---*	*(4)*	*(3)*
207	*(4)*	*---*	*---*	*(0)*	*---*
208	*(1)*	*(0)*	*(1)*	*(0)*	*(1)*
209	*(1)*	*(2)*	*(0)*	*(1)*	*(3)*
210	*(2)*	*(1)*	*(3)*	*(3)*	*(1)*
211	*(3)*	*(2)*	*---*	*(0)*	*(2)*
212	*(4)*	*(1)*	*(2)*	*(2)*	*(1)*
213	*---*	*(0)*	*(1)*	*---*	*(1)*
214	*(4)*	*(1)*	*---*	*(1)*	*(2)*
215	*(1)*	*(4)*	*(2)*	*---*	*(3)*
216	*(1)*	*---*	*(1)*	*(0)*	*---*
217	*(4)*	*(3)*	*---*	*(3)*	*(2)*
218	*(1)*	*(4)*	*(3)*	*---*	*(0)*
219	*(1)*	*---*	*---*	*(4)*	*(3)*
220	*(4)*	*(2)*	*(0)*	*---*	*(4)*
221	*(3)*	*(1)*	*(2)*	*(0)*	*(0)*
222	*(2)*	*(0)*	*(0)*	*(4)*	*(0)*
223	*---*	*(3)*	*(2)*	*(1)*	*(2)*
224	*(0)*	*(1)*	*(4)*	*(2)*	*(3)*
225	*---*	*(3)*	*---*	*(2)*	*---*
226	*(3)*	*(0)*	*---*	*(3)*	*(0)*
227	*---*	*(2)*	*(3)*	*---*	*(2)*
228	*(3)*	*(2)*	*(1)*	*(3)*	*(1)*
229	*(2)*	*(2)*	*(4)*	*(2)*	*(4)*
230	*(4)*	*(1)*	*(4)*	*(1)*	*(4)*
231	*(1)*	*(1)*	*(2)*	*(0)*	*(3)*
232	*(2)*	*(2)*	*(1)*	*(1)*	*(1)*
233	*(4)*	*(2)*	*(2)*	*---*	*---*
234	*---*	*(3)*	*(0)*	*(1)*	*(1)*
235	*(3)*	*(0)*	*(3)*	*(0)*	*(2)*
236	*(0)*	*(1)*	*(4)*	*(0)*	*(1)*
237	*(3)*	*(1)*	*(0)*	*(1)*	*(3)*
238	*(2)*	*(2)*	*(3)*	*(0)*	*(4)*
239	*(3)*	*(0)*	*(1)*	*(4)*	*(4)*
240	*(4)*	*(3)*	*(3)*	*(2)*	*(2)*
241	*(1)*	*---*	*(3)*	*(3)*	*(4)*
242	*(1)*	*(4)*	*(2)*	*(2)*	*(0)*
243	*(1)*	*(0)*	*(0)*	*---*	*(0)*
244	*---*	*(0)*	*(3)*	*(3)*	*(2)*
245	*---*	*(2)*	*---*	*(0)*	*(0)*
246	*(3)*	*(1)*	*(4)*	*---*	*(0)*
247	*(3)*	*(4)*	*(2)*	*(0)*	*(1)*
248	*(2)*	*(4)*	*(4)*	*(2)*	*(4)*

249	*---*	*(1)*	*(4)*	*(2)*	*(2)*
250	*(0)*	*(4)*	*(2)*	*(3)*	*(4)*
251	*(2)*	*(0)*	*(2)*	*(2)*	*(4)*
252	*(0)*	*(1)*	*(0)*	*(4)*	*(0)*
253	*(4)*	*(1)*	*(2)*	*(4)*	*(2)*
254	*(1)*	*(3)*	*---*	*(2)*	*(0)*
255	*---*	*(0)*	*(1)*	*(0)*	*(1)*

Заключение

В статье [5] рассмотрено построение автомата Мура для задачи «Умный муравей». В данной работе рассматривается построение автомата Мура для 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. *Бедный Ю.Д., Шалыто А.А.* Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». http://is.ifmo.ru/works/_ant.pdf
3. http://is.ifmo.ru/genalg/
4. *Яминов Б.* 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

Модуль «особи»:

1. `Ant18FactoryLoader.java` – класс, реализующий загрузку модуля в виртуальную лабораторию (листинг 2).
2. `Ant18.java` – класс, реализующий работу с автоматом Мура (листинг 3).
3. `Ant18Mover.java` – класс, реализующий интерфейс работы с управляющим объектом (листинг 4).
4. `BitArray.java` – класс, реализующий работу с представлением автомата в виде битовой строки (листинг 5).
5. `Ant18Factory.java` – класс, реализующий создание произвольной особи (листинг 6).

Модуль генетического алгоритма:

1. `CellularGA.java` – класс, реализующий клеточный генетический алгоритм (листинг 7).
2. `CellularGALoader.java` – класс, реализующий загрузку модуля в виртуальную лабораторию (листинг 8).

Конфигурационные файлы

Для сборки модулей виртуальной лаборатории был создан ant-скрипт, который находится в файле `build.xml` (листинг 9). Файл `build.properties` содержит параметры для этого скрипта (листинг 10).

Для того чтобы визуализатор стал доступен пользователю, необходимо изменение файла конфигурации модуля визуализатора [3]. Необходимо в параметре `filter` дописать название индивидуума «`task18`». Визуализатор эмулирует поведение одной из особей, полученных в ходе работы генетического алгоритма.

Модуль «особи»:

1. `automaton.conf` – конфигурация задания (листинг 11).
    - `external.count.states` – число состояний автомата.
    - `count.attempts` – число запусков муравья.
    - `mu` – вероятность расположения еды в каждой клетке.
2. `plugin.properties` – конфигурация модуля (листинг 12).

Модуль генетического алгоритма:

1. `CellularGA.conf` – конфигурация клеточного генетического алгоритма (листинг 13).
    - `gridHeight` – число клеток по вертикали.
    - `gridWidth` – число клеток по горизонтали.
    - `probabilityMutation` – вероятность мутации.
2. `plugin.properties` – конфигурация модуля (листинг 14).

Листинг 2. Файл Ant18FactoryLoader.java

```
package individual.task18;

import individual.task18.factory.Ant18Factory;
import java.util.jar.JarFile;
import laboratory.util.AbstractLoader;

/**
 *
 * @author Александр
 */
public class Ant18FactoryLoader extends AbstractLoader<Ant18Factory> {

    public Ant18FactoryLoader(JarFile jarFile) {
        super(jarFile, "automaton.conf");
    }

    public Ant18Factory load(Object... args) {
        Ant18Factory.attempts = properties.getInt("count.attempts");
        return new
Ant18Factory(properties.getInt("external.count.states"),
properties.getDouble("mu"));
    }
}
```

Листинг 3. Файл Ant18.java

```
package individual.task18;

import individual.task18.factory.Ant18Factory;
import individual.task18.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 Ant18 implements Visualizable {

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

    /**
     * Число состояний автомата.
     */
    public final int externalStatesNumber;

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

    /**
     * Номер текущего состояния автомата.
     */
    public int currentExternalStateNumber;

    /**
     * Число входных воздействий.
     */
    public final int ct = 256;

    /**
     * Конструктор.
     * @param externalStatesNumber чило состояний внешнего автомата
     * @param externalInitialState начальное состояние внешнего
автомата
     * @param mu
     */
    public Ant18(int externalStatesNumber, int externalInitialState,
double mu) {
        external = new BitArray(externalStatesNumber, ct);
```

```
        this.externalStatesNumber = externalStatesNumber;
        this.externalInitialState = externalInitialState;
        currentExternalStateNumber = externalInitialState;
        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) {
        } else {
            action = external.getAction(nextState);
            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 Ant18Mover(this, mu)};
    }

    public double fitness() {
        if (fitness != Double.NEGATIVE_INFINITY) {
            return fitness;
        }
        fitness = 0;
        for (int j = 0; j < Ant18Factory.attempts; j++) {
            ExtendedAnt ant = new ExtendedAnt(mu);
            for (int i = 0; i < Ant.NUMBER_STEPS; i++) {
                if (move(ant)) {
```

```
                    fitness++;
                }
            }
        }
        fitness /= (double) Ant18Factory.attempts;
        return fitness;
    }

    public Individual mutate(Random r) {
        Ant18 ant = new Ant18(externalStatesNumber,
r.nextInt(externalStatesNumber), 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);
                }
            }
        }

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

        return ant;
    }

    public Individual[] crossover(Individual p, Random r) {
        Ant18[] sons = new Ant18[2];
        Ant18 parent = (Ant18) p;

        sons[0] = new Ant18(externalStatesNumber,
r.nextInt(externalStatesNumber), mu);
        sons[1] = new Ant18(externalStatesNumber,
r.nextInt(externalStatesNumber), mu);

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

            if (sons[0].external.getAction(externalState) != null &&
                    sons[1].external.getAction(externalState) != 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\t";
        for (int i = 0; i < externalStatesNumber; i++) {
            res += "(" + i+", " + external.getAction(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+")\t";
                else
                    res += "---\t";
            }
            res += "\t";            res += "\n";
        }
        return res;
    }

}
```

Листинг 4. Файл Ant18Mover.java

```
package individual.task18;

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

/**
 *
 * @author Александр
 */
public class Ant18Mover implements ExtendedAntTask.Mover {

    private final double mu;
    public final Ant18 automaton;
    private ExtendedAnt ant;

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

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

    public Ant18Mover(Ant18 automaton, double mu) {
        this.mu = mu;
        this.automaton = automaton;
    }

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

Листинг 5. Файл BitArray.java

```
package individual.task18.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 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 число 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 * ct +
ACTION_LENGTH)];
    }

    /**
     * Установка действия.
     * @param stateNumber номер состояния
     * @param action новое действие
     */
```

```
    public void setAction(int stateNumber, Ant.Action action) {
        if (stateNumber > 254) {
            throw new RuntimeException("State number out of range!");
        }
        int begin = stateNumber * ((NUMBER_LENGTH ) * ct +
ACTION_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) {
        if (nextState >= countStates || stateNumber >= countStates) {
            throw new RuntimeException("setNextState!");
        }
        nextState++;
        int begin = stateNumber * (NUMBER_LENGTH * ct + ACTION_LENGTH)
+ NUMBER_LENGTH * index + ACTION_LENGTH;
        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
     * @return действие при переходе
     */
    public Ant.Action getAction(int stateNumber) {
        if (stateNumber >= countStates) {
        }
        int begin = stateNumber * (NUMBER_LENGTH * ct + ACTION_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;
            }
        }
    }

    /**
     * Возвращает 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 * ct + ACTION_LENGTH)
+ NUMBER_LENGTH * index + ACTION_LENGTH;
        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;
    }
}
```

Листинг 6. Файл Ant18Factory.java

```
package individual.task18.factory;

import individual.task18.Ant18;
import java.util.Random;
import laboratory.common.ga.IndividualFactory;
import task.ant.extended.Ant;

/**
 *
 * @author Александр
 */
public class Ant18Factory implements IndividualFactory {

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

    public Ant18Factory(int externalStatesNumber, double mu) {
        this.externalStatesNumber = externalStatesNumber;
        this.mu = mu;
    }

    public Ant18 randomIndividual() {
        Ant18 ant = new Ant18(externalStatesNumber,
r.nextInt(externalStatesNumber), mu);

        for (int i = 0; i < externalStatesNumber; i++) {
                switch (r.nextInt(3)) {
                    case 0:
                        ant.external.setAction(i, Ant.Action.L);
                        break;
                    case 1:
                        ant.external.setAction(i, Ant.Action.M);
                        break;
                    case 2:
                        ant.external.setAction(i, Ant.Action.R);
                        break;
                }
            for (int j = 0; j < 256; j++) {
                ant.external.setNextState(i, j,
r.nextInt(externalStatesNumber + 1) - 1);
            }
        }
/*        for(boolean ind : ant.external.array) {
        System.out.print((ind)?1:0);
        //System.out.print(" ");
        }
        System.out.println();*/

        return ant;
    }
}
```

Листинг 7. Файл CellularGA.java

```
package ga.cellular;

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

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

public class CellularGA implements GA {

    private Individual[][] myGeneration;
    private double[][] myIndivFitness;

    private int myWidth;
    private int myHeight;



    private final double probabilityMutation;

    private final IndividualFactory factory;

    private final Random r;

    List<Individual> newGeneration;


     private Individual getBestNeighbour(int i, int j)
     {
        List<Individual> toChoose = new ArrayList<Individual>();
        double[] fit = new double[4];
        int[] p = new int[4];
        if (i > 0) {
            toChoose.add(myGeneration[i - 1][j]);
            fit[toChoose.size() - 1] = myIndivFitness[i - 1][j];
          }
          if (i < myHeight - 1)
          {
            toChoose.add(myGeneration[i + 1][j]);
            fit[toChoose.size() - 1] = myIndivFitness[i + 1][j];
          }
          if (j > 0) {
            toChoose.add(myGeneration[i][j - 1]);
            fit[toChoose.size() - 1] = myIndivFitness[i][j - 1];
          }
          if (j < myWidth - 1)
          {
            toChoose.add(myGeneration[i][j + 1]);
            fit[toChoose.size() - 1] = myIndivFitness[i][j + 1];
```

```
            }
        for (int f1 = 0; f1 < toChoose.size(); f1++) {
            p[f1] = 0;
            for (int f2 = 0; f2 < toChoose.size(); f2++) {
                if (fit[f1] >= fit[f2]) {
                    p[f1]++;
                }
            }
        }
        int best = 0;
        for (int f1 = 0; f1 < toChoose.size(); f1++) {
            p[f1] = r.nextInt(p[f1]);
            if (p[f1] > p[best]) {
                best = f1;
            }
        }
           return toChoose.get(best);
     }

    public void nextGeneration() {
/*        int size = generation.size();
        List<Individual> newGeneration = new
ArrayList<Individual>(size);
        newGeneration.addAll(generation.subList(0, sizeElite));
        for (int i = 0; i < (size - sizeElite) / 2; i++) {
            Individual a1, a2;
            a1 = generation.get(r.nextInt(size));
            a2 = generation.get(r.nextInt(size));
            Individual p = (a1.compareTo(a2) < 0) ? a1 : a2;
            a1 = generation.get(r.nextInt(size));
            a2 = generation.get(r.nextInt(size));
            Individual[] s = p.crossover((a1.compareTo(a2) < 0) ? a1 :
a2, r);
            newGeneration.add(s[0]);
            newGeneration.add(s[1]);
        }
        if (newGeneration.size() < size) {

newGeneration.add(generation.get(r.nextInt(size)).mutate(r));
        }
        for (int i = 0; i < size; i++) {
            if (r.nextDouble() < probabilityMutation) {
                newGeneration.set(i, newGeneration.get(i).mutate(r));
            }
        }
        generation = newGeneration;
        Collections.sort(generation);
        */
     for (int i = 0; i < myHeight; i++)
          {
               for (int j = 0; j < myWidth; j++)
               {
                    Individual best = getBestNeighbour(i, j);
                    Individual[] s =
myGeneration[i][j].crossover(best, r);
                    double r0 = s[0].fitness();
```

```
                    double r1 = s[1].fitness();
                int f0 = 0;
                int f1 = 0;
                int f2 = 0;
                if (r0 > r1) {
                    f0++;
                }
                else {
                    f1++;
                }
                if (r0 > myIndivFitness[i][j]) {
                    f0++;
                }
                else {
                    f2++;
                }
                if (r1 > myIndivFitness[i][j]) {
                    f1++;
                }
                else {
                    f2++;
                }
                int p0 = r.nextInt((int) f2 + 1);
                int p1 = r.nextInt((int)f0 + 1);
                int p2 = r.nextInt((int)f1 + 1);
                if (p1 > p0 && p1 > p2) {
                    newGeneration.add(s[0]);
                }
                else {
                    if (p2 > p0) {
                        newGeneration.add(s[1]);
                    }
                    else {
                        newGeneration.add(myGeneration[i][j]);
                    }
                }
                }
            }
            int cur = 0;
            for (int i = 0; i < myHeight; i++) {
                for (int j = 0; j < myWidth; j++) {
                if (r.nextDouble() < probabilityMutation) {
                    myGeneration[i][j] =
newGeneration.get(cur++).mutate(r);
                }
                else {
                    myGeneration[i][j] = newGeneration.get(cur++);
                }

                myIndivFitness[i][j] = myGeneration[i][j].fitness();
                }
            }
        newGeneration.clear();
    }

    public List<Individual> getGeneration() {
```

```
        List<Individual> gen = new ArrayList<Individual>();
        for (int i = 0; i < myHeight; i++) {
           for (int j = 0; j < myWidth; j++) {
                gen.add(myGeneration[i][j]);
           }
        }
        Collections.sort(gen);
        return gen;
    }

    public CellularGA(int boardWidth, int boardHeight, double
probabilityMutation, IndividualFactory factory) {
        this.myWidth = boardWidth;
        this.myHeight = boardHeight;
      this.myGeneration = new Individual[myHeight][myWidth];
      this.myIndivFitness = new double[myHeight][myWidth];
        this.probabilityMutation = probabilityMutation;
        for (int i = 0; i < myHeight; i++) {
           for (int j = 0; j < myWidth; j++) {
                myGeneration[i][j] = factory.randomIndividual();
                myIndivFitness[i][j] = myGeneration[i][j].fitness();
           }
        }
        newGeneration = new ArrayList<Individual>();

        this.factory = factory;
        r = new Random();
    }

    public void bigMutation() {
        for (int i = 0; i < myHeight; i++) {
           for (int j = 0; j < myWidth; j++) {
                myGeneration[i][j] = factory.randomIndividual();
            }
        }
    }

    public Individual getBest() {
           int ib = 0;
           int jb = 0;
           double bFitness = myGeneration[ib][jb].fitness();
      for (int i = 0; i < myHeight; i++) {
           for (int j = 0; j < myWidth; j++) {
                if (myIndivFitness[i][j] > bFitness) {
                           ib = i;
                           jb = j;
                           bFitness = myIndivFitness[i][j];
                     }
           }
      }
      return myGeneration[ib][jb];
    }
}
```

Листинг 8. Файл CellularGALoader.java

```
package ga.cellular;

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 Andrey Davydov
 */
public class CellularGALoader implements Loader<GA> {

    private final Parser properties;

    public CellularGALoader(JarFile file) {
        Properties in = new Properties();
        try {
            JarEntry ent = new JarEntry("cellularGA.conf");
            in.load(file.getInputStream(ent));
        } catch (IOException e) {
            e.printStackTrace();
        }
        properties = new Parser(in);
    }

    public GA load(Object... args) {
        return new CellularGA(properties.getInt("gridHeight"),
properties.getInt("gridWidth"),
                properties.getDouble("probabilityMutation"),
                (IndividualFactory) args[0]);
    }

    public Properties getProperties() {
        return properties.getProperties();
    }
}
```

Листинг 9. Файл build.xml

```
<?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="${util}/extended-ant.jar"/>
    <property name="simpletask.build" value="${util}/simple-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="${ga.src}/cellular"/>
            <param name="plugin.classpath" value="${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="${individual.src}/task18"/>
            <param name="plugin.classpath" value="${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}/cellular"/>
        </antcall>
    </target>
```

```
    <target name="clean.individual">
        <antcall target="clean.plugin">
            <param name="plugin" value="${individual.src}/task18"/>
        </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>
```

Листинг 10. Файл build.properties

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

ga.dir=gas
individual.dir=individuals

ga.src=ga
individual.src=individual
```

Листинг 11. Файл automaton.conf

```
external.count.states=5
internal.count.states=0
count.attempts=1
mu=0.1
```

Листинг 12. Файл plugin.properties

```
jar=task18.jar
main.class=individual.task18.Ant18FactoryLoader
comment=Реализация задания, описанного в варианте 18.
extension.name=Task18
resources=conf
filter=Extended ant
graphic.settings=70 0
```

Листинг 13. Файл CellularGA.conf

```
gridHeight=40
gridWidth=40
probabilityMutation=0.1
```

Листинг 14. Файл plugin.properties

```
jar=cellular.jar
main.class=ga.cellular.CellularGALoader
comment=Cellular genetic algorithm.
extension.name=cellular

resources=conf
src=src
```