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

М. Е. Кевер

Отчет по лабораторной работе
«Применение генетических алгоритмов для решения задачи «Умный муравей-3»

Вариант № 63

2010

Оглавление

1. Введение

Целью лабораторной работы является исследование генетических алгоритмов, используемых для построения конечных автоматов. В качестве примера рассматривается задача «Умный муравей-3». При выполнении лабораторной работы используется программа 3Genetic, написанная студентами кафедры «Компьютерные технологии» СПбГУ ИТМО.

2. Постановка задачи

Действие задачи «Умный муравей-3» происходит на поверхности тора размером 32×32, в некоторых клетках которого расположена еда. В одной из клеток находится муравей, видящий восемь клеток перед собой. За ход муравей может выполнить следующие действия: повернуть налево; повернуть направо; сделать шаг вперед и, если в новой клетке есть еда, съесть ее. Муравей может сделать не более 200 ходов. Цель муравья – съесть как можно больше еды.

Задачей лабораторной работы являлось построение с помощью генетических алгоритмов конечного автомата Мили, решающего задачу об «Умном муравье-3». Для представления автоматов использовались сокращенные таблицы переходов. Генерация поколений осуществлялась при помощи клеточного генетического алгоритма и метода элитизма.

3. Алгоритм

3.1. Представление автоматов

Особью генетического алгоритма для решения данной задачи является конечный автомат Мили – детерминированный конечный автомат, каждому переходу которого сопоставлено некоторое действие (в данной задаче это либо шаг вперед, либо поворот вправо или влево). Входным алфавитом этого автомата являются значения восьми логических переменных, обозначающих наличие или отсутствие еды в каждой из видимых муравьем клеток.

Автомат хранится как множество (массив) состояний с выделенным начальным состоянием. Каждое состояние хранит в себе множество «важных» входных параметров (в виде битовой маски) и таблицы переходов и действий в зависимости от значений этих параметров (в виде массивов).

Число состояний автомата является настраиваемым параметром алгоритма. Все генерируемые алгоритмом автоматы имеют одинаковое число состояний, не изменяющееся во время выполнения алгоритма.

3.2 Генетический алгоритм

Для решения задачи используется клеточный генетический алгоритм. Популяция представляет собой прямоугольную сетку, в каждой ячейке которой находится ровно одна особь. Разработанный алгоритм состоит из следующих операций:

1. Создание начального поколения.
2. Переход к следующему поколению на каждой итерации алгоритма:
   2.1. Скрещивание каждой особи с особями в соседних ячейках.
   2.2. Выбор потомков для каждой ячейки из результатов соответствующих ей скрещиваний.
   2.3. Мутация случайно выбранных особей.

3.2. Создание начального поколения

Вся сетка заполняется автоматами, сгенерированными следующим образом:

1. Создается автомат с заданным в настройках числом состояний.
2. Для каждого состояния:
   2.1. Случайно выбирается подмножество входных параметров, являющихся «важными», равновероятно среди всех возможных подмножеств.
   2.2. Таблица переходов заполняется случайными номерами состояний.
   2.3. Таблица действий заполняется случайными действиями.

3.3. Мутация

После создания нового поколения каждый индивид с вероятностью, заданной в конфигурационном файле, подвергается мутации. Для каждого состояния с соответствующими настраиваемыми вероятностями мутирует множество «важных» входных параметров и таблицы переходов и действий. Наконец, в зависимости от того, какой из двух видов мутации выбран, может произойти изменение начального состояния.

3.4. Скрещивание и отбор потомка

При переходе к следующему поколению каждая особь скрещивается с одним из соседей в таблице клеточного алгоритма. При скрещивании по двум особям строятся две новые. Каждая из особей получается следующим образом:

1. Каждое *i*-ое состояние автомата получается путем скрещивания *i*-ых состояний родительских автоматов.
2. Множество «важных» входных параметров – это параметры, входящие в множества обоих родителей, и выбранные с вероятностью ½ каждый параметры из входящих в множество ровно одного родителя.
3. Для каждого значения входных параметров вычисляется *влияние* каждого из родителей – число параметров, равных 1, содержащихся в его списке «важных» параметров. После этого среди родителей с вероятностями, пропорциональным их *влияниям*, выбирается тот, номер перехода и действие которого будут использованы.
4. Наконец, случайно (равновероятно) выбирается номер начального состояния из номеров начальных состояний родительских автоматов.

После этого из двух полученных автоматов выбирается тот, который имеет большее значение функции приспособленности, и переводится в следующее поколение.

3.5. Функция приспособленности

Проводятся несколько испытаний, в ходе каждого из которых муравей запускается на карте, в каждую клетку которой с заданной вероятностью помещается еда для муравья. Функция приспособленности *Fitness* для одного испытания вычисляется по формуле:

$$Fitness = \left(Apples - \frac{Steps}{200} \right) \cdot \frac{MapSize \cdot FoodP}{TotalFood},$$

где *Apples* – число съеденных яблок, *Steps* – номер хода, на котором муравей съел, *MapSize* – размер карты, *FoodP* – вероятность появления еды, *TotalFood* – количество еды на сгенерированной карте.

Значения по всем испытаниям суммируются и умножаются на $3/CompetitionsCount$, где *CompetitionsCount* – число испытаний. Как число испытаний, так и вероятность появления еды являются настраиваемыми параметрами алгоритма.

Выбор карт для испытаний – очень тонкий момент алгоритма, так как он может производиться принципиально разными способами:

1. Для каждого испытания каждого муравья генерируется своя случайная карта.
2. Все карты для испытаний генерируются изначально и фиксируются.

При выполнении лабораторной работы были реализованы и сравнены оба этих подхода.

4. Результаты работы

При оценке результатов работы алгоритма важен прежде всего критерий, по которому оценивается «качество» получающихся автоматов. Таким критерием может являться не только значение функции приспособленности, используемой при отборе, но и значение любой другой функции. Ввиду того, что существует выбор между принципиально разными оценочными функциями (см. 3.5), имеет смысл провести их сравнение.

Все испытания будут проводиться на картах с вероятностью появления еды $FoodP = 0.05$.

4.1. Отбор и оценка на случайных картах

В этом подходе для испытания каждого муравья будем использовать свой сгенерированный набор случайных карт.

Даже если зафиксировать способы «обучения» и «тестирования» муравья, по-прежнему остается свобода в выборе числа испытаний. На рис. 1 и 2 приведены графики развития популяции для двух различных значений параметра. *Max* соответствует максимуму функции приспособленности для данной популяции, *Mean* – среднему значению.

Результат вполне предсказуем: при небольшом числе испытаний велика вероятность того, что муравью «повезет с картами», и появляются особи с большим значением функции приспособленности, которые объективно могут быть и не лучше особей с меньшими значениями.

Следует отметить, что из-за случайности карт и малого количества еды лучшими оказываются «локально оптимальные» автоматы. Их поведение выглядит очень просто: «иду туда, где есть еда».

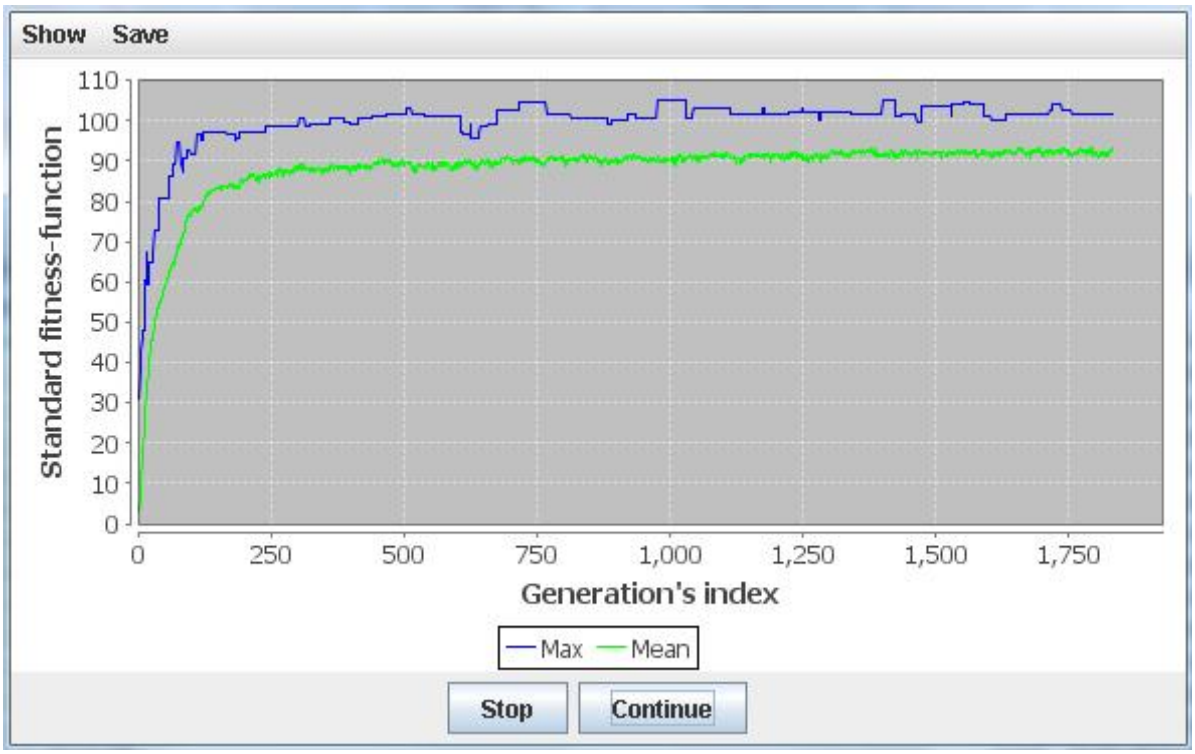

Рис.1. График функции приспособленности при отборе и оценке на случайных картах, число испытаний – 3

Рис.2. График функции приспособленности при отборе и оценке на случайных картах, число испытаний – 10

4.2. Отбор и оценка на фиксированном наборе карт

Зафиксируем набор из нескольких заранее сгенерированных карт и будем испытывать всех муравьев на них (рис. 3, 4).

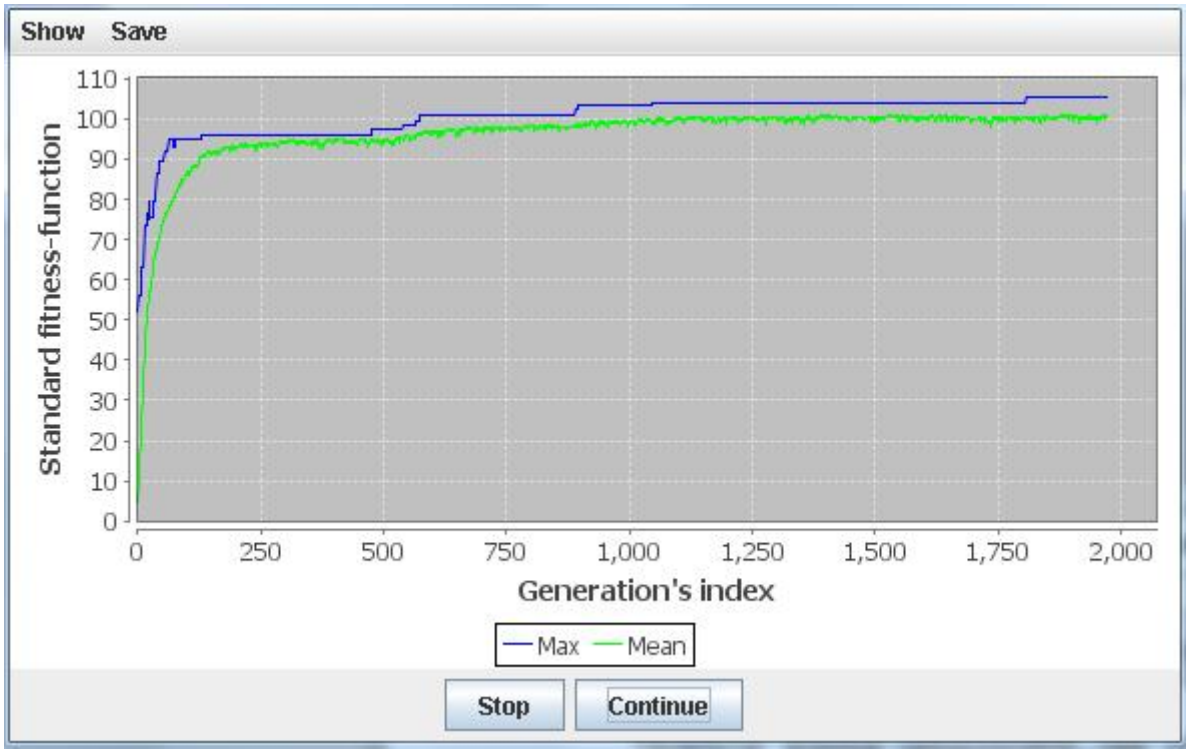

Рис.3. График функции приспособленности при отборе и оценке на фиксированном наборе карт, число испытаний – 3

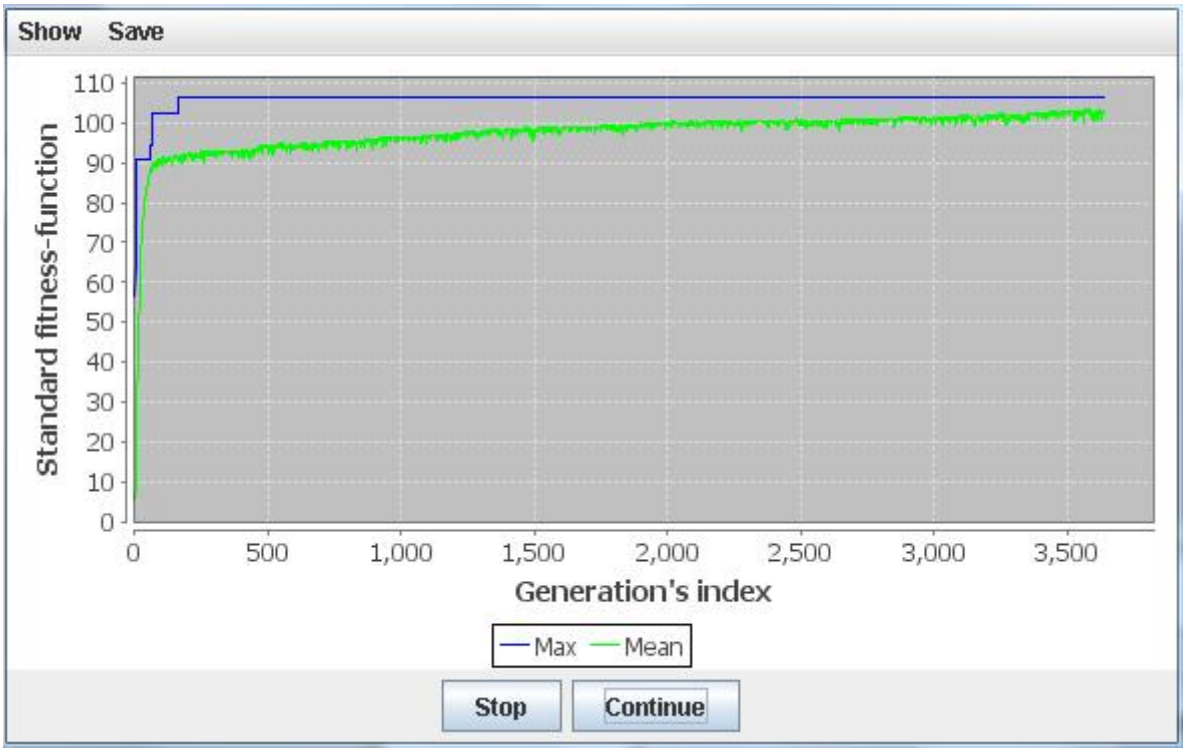

Рис.4. График функции приспособленности при отборе и оценке на фиксированном наборе карт, число испытаний – 1

При таком подходе муравьи «обучаются» под конкретные карты и процесс постепенно сходится к максимуму. Автоматы, получаемые таким образом, уже не получаются столь «интуитивно понятными», как при случайных картах.

4.3. Отбор на случайных картах, оценка на фиксированных картах

Функцию приспособленности будем вычислять на случайно генерируемых картах, при этом параллельно оценивая получающиеся автоматы на фиксированном наборе из 10 карт.

На всех последующих графиках *Max* и *Max2* соответствуют максимумам функции приспособленности и оценочной функции соответственно, *Mean* и *Mean2* – средним значениям.

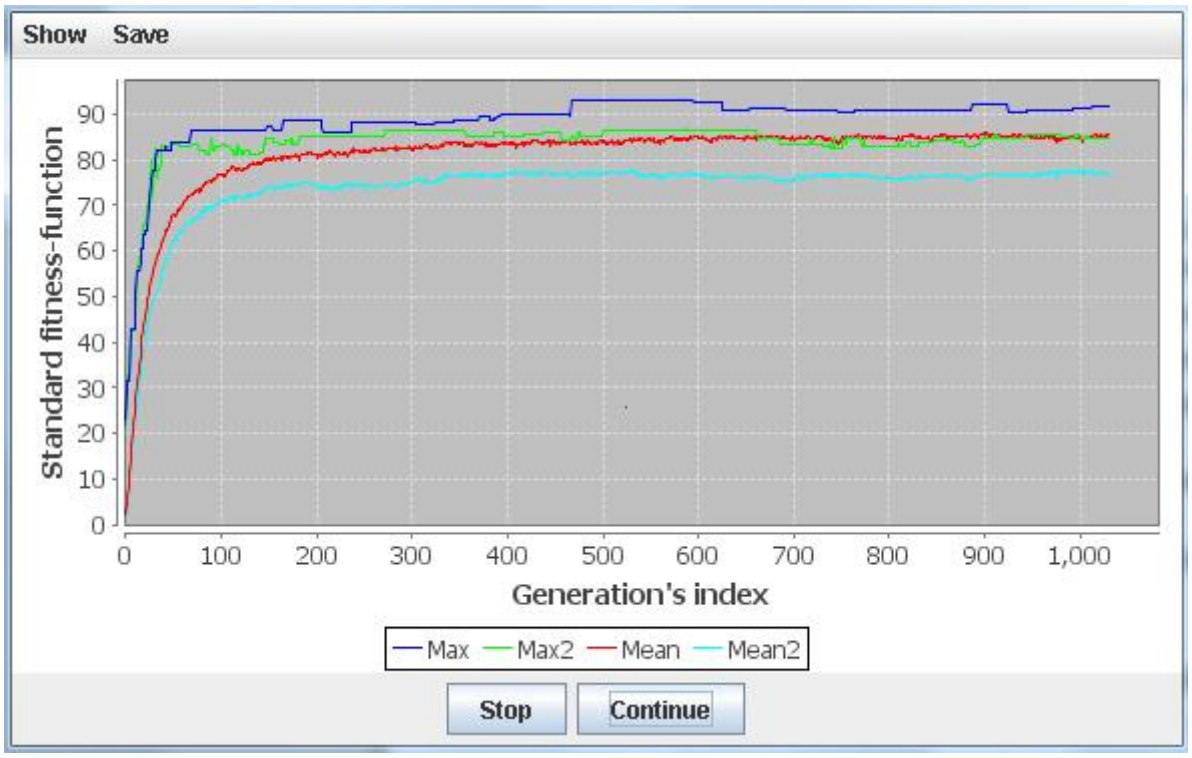

Рис.5. График функции приспособленности при отборе на случайных картах и оценке на фиксированном наборе карт, число испытаний – 10

Как видно из графика на рис. 5, муравьи, обученные на случайных картах, хорошо ведут себя и на фиксированных (но хуже, чем обученные специально для этих карт).

4.4. Отбор на фиксированных картах, оценка на случайных картах

При небольшом числе испытаний получаются муравьи, обученные только для этих карт и значительно хуже ведущие себя в общем случае. При увеличении числа испытаний разница уменьшается, но тенденция сохраняется (рис. 6, 7).

4.5. Отбор и оценка на разных фиксированных наборах карт

Для описанного случая результаты очень схожи с полученными в п. 4.4. (рис. 8, 9).

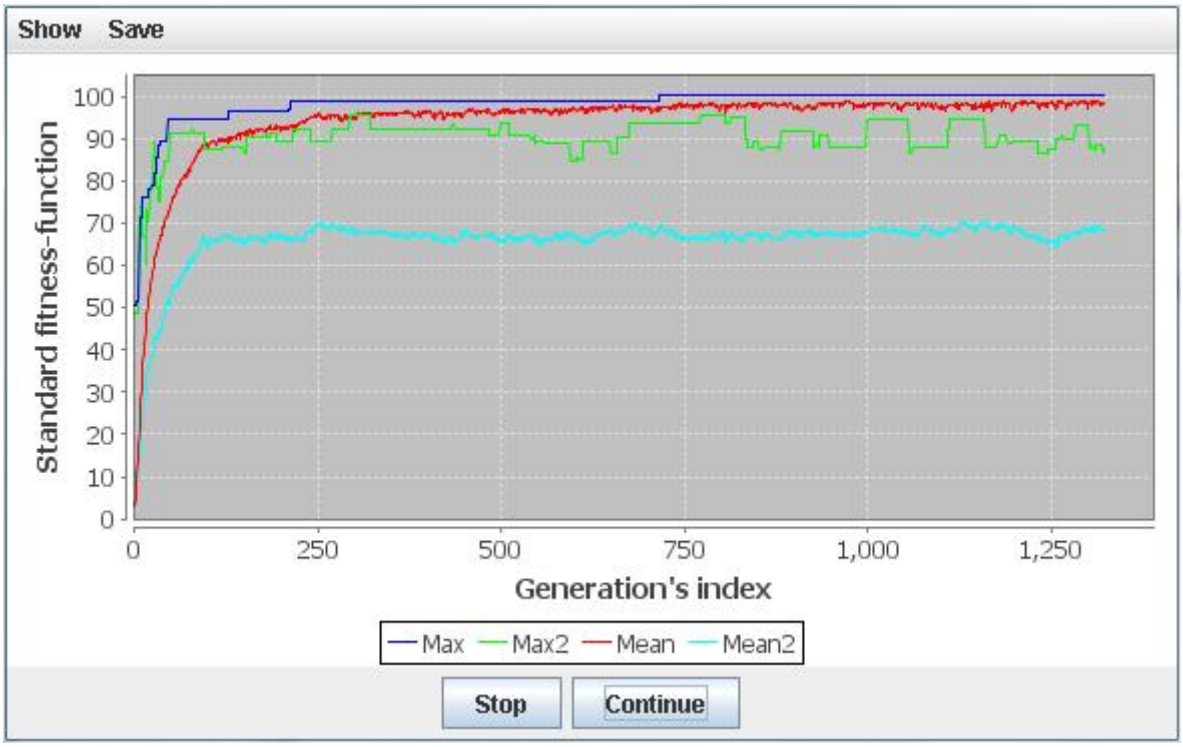

Рис.6. График функции приспособленности при отборе на фиксированных картах и оценке на случайных картах, число испытаний – 3

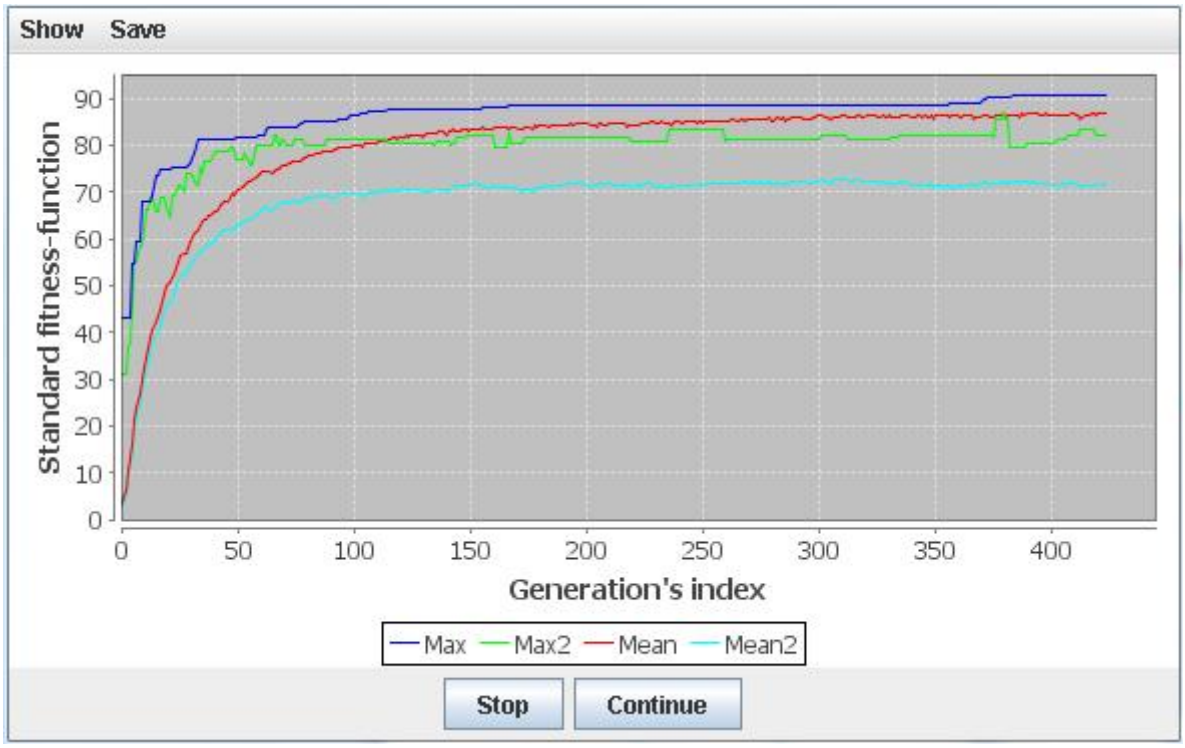

Рис.7. График функции приспособленности при отборе на фиксированных картах и оценке на случайных картах, число испытаний – 20

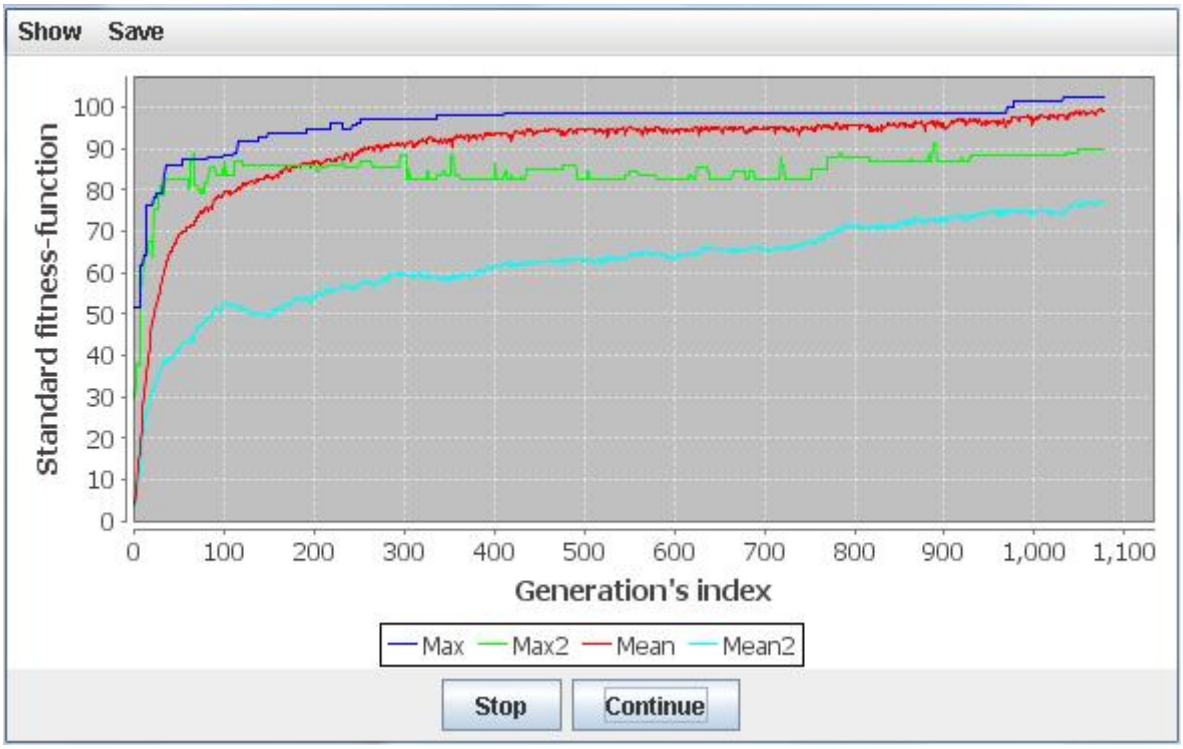

Рис.8. График функции приспособленности при отборе и оценке на разных фиксированных наборах карт, число испытаний – 3

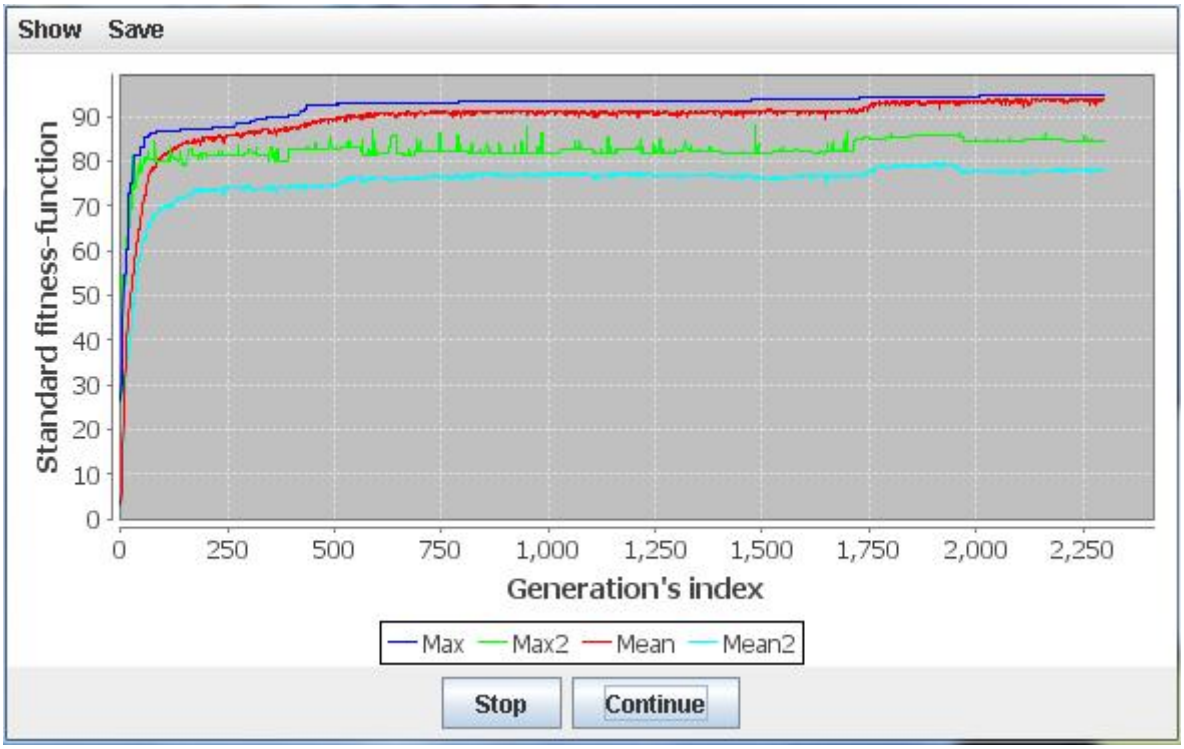

Рис.9. График функции приспособленности при отборе и оценке на разных фиксированных наборах карт, число испытаний – 10

4.6. Итог

В зависимости от выбранной функции приспособленности получаются автоматы с различным поведением, оптимально ведущие себя при поставленных перед ними условиях. Следовательно, нельзя говорить об «идеальном» решении данной задачи. В качестве примера получающегося муравья в приложении приведен автомат с пятью состояниями, имеющий значение функции приспособленности при испытании на 10 случайных картах равное 103.45.

При всех построениях были использованы следующие параметры:

– размеры клеточной популяции (size.rows × size.columns) = 10 × 10;

– вероятность мутации (probability.mutation) = 0,02;

– вероятность мутации начального состояния (probability.mutation.initial.state) = 0,3;

– вероятность мутации состояния (table.mutations.probability) = 0,2;

– вероятность мутации маски для сокращенной таблицы (table.mutations.probability) = 0,2;

– число изменяемых при мутации переходов (corrupted.transitions.count) = 10;

– число состояний (states.count) = 5.

5. Заключение

Результаты лабораторной работы показали, что с помощью генетических алгоритмов можно сгенерировать автомат Мили с пятью состояниями, который эффективно решает задачу об «Умном муравье-3».

Источники

1. Инструкция по работе с виртуальной лабораторией 3Genetic. http://is.ifmo.ru/genalg/labs_2010-2011/interface_manual.pdf
2. *Mitchell M.* An Introduction to Genetic Algorithms, MIT Press, 1998.
3. *Яминов Б.* Генетические алгоритмы. http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005

Приложение. Построенный автомат

INDIVIDUAL

Fitness = 103.45

States = 5

Initial state = 1

State 0:

mask = 11000111

end state = [1, 2, 1, 2, 3, 3, 3, 3, 1, 1, 1, 1, 1, 3, 1, 2, 1, 2, 1, 2, 3, 3, 3, 3, 1, 1, 1, 1, 1, 2, 1, 2]

actions = [R, M, R, M, R, R, R, R, R, R, R, R, R, R, R, M, R, M, R, M, R, R, R,R, R, R, R, R, R, M, R, M]

State 1:

mask = 11111111

end state = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 3, 1, 3, 1, 1, 1, 2, 1, 3, 3, 3, 3, 2, 1, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 2, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 3, 1, 3, 1, 2, 1, 2, 1, 3, 3, 3, 3, 2, 1, 2, 2, 3, 2, 2, 2, 0, 2, 2, 0, 2, 4, 2, 2, 4, 4, 2, 0, 1,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 1, 3, 1, 2, 3, 2, 1, 3, 3, 3, 3, 2, 1, 3, 2, 3, 2, 3, 2, 2, 0, 0, 0, 4, 4, 2, 2, 2, 2, 4, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 1, 3, 1, 2, 3, 2, 1, 3, 3, 3, 3, 2, 1, 3, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2, 2, 2, 4, 4, 4, 2, 4]

actions = [M, M, M, M, R, R, R, M, R, M, R, M, R, R, R, R, L, L, L, M, L, L, L,L, L, L, L, L, M, M, R, M, L, M, L, M, R, R, L, R, L, L, L, L, R, R, R, R, L, L, L, L, L, L, L, L, L, L, L, L, L, L, R, L, M, M, M, M, R, R, R, M, R, M, R, M, R, R, R, R, L, L, L, M, L, R, L, L, L, L, L, L, M, L, R, M, L, M, L, M, L, R, L,R, L, L, L, L, R, M, R, R, L, L, L, L, R, L, L, R, L, L, L, L, L, L, R, R, M, M, M, M, R, R, R, R, R, M, R, R, R, R, R, R, L, L, L, L, L, L, L, L, L, L, L, L, M, M, M, M, L, M, L, M, L, L, L, R, L, L, L, L, L, L, R, L, L, L, L, L, L, L, R, R,R, L, L, L, L, R, M, L, L, M, M, M, M, R, M, R, R, R, M, R, M, R, R, R, R, L, L, L, L, L, L, M, L, L, L, L, M, L, M, M, L, M, L, M, L, L, L, R, L, L, L, L, L, L, L, R, L, L, L, L, L, L, L, R, L, R, L, L, L, L, L, R, L]

State 2:

mask = 01001111

end state = [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

actions = [M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M,M, M, M, M, M, M, M, M, M]

State 3:

mask = 01011001

end state = [2, 1, 2, 1, 0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1]

actions = [M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M]

State 4:

mask = 11001110

end state = [1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1]

actions = [M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M]