Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики
Факультет информационных технологий и программирования
Кафедра «Компьютерные Технологии»

А.Ю. Рост

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

Вариант №11

Санкт-Петербург
2011

Оглавление

Введение

В лабораторной работе предлагалось рассмотреть применение деревьев решений для кодирования автомата, решающего задачу об умном муравье-3. Автомат строился с помощью генетического алгоритма, причем было рассмотрено несколько стратегий отбора.

1. Постановка задачи

Цель лабораторной работы — исследовать эффективность использования «упрощающих» операторов над деревьями решений (с некоторой вероятностью) после применения операторов кроссовера или мутации. Предлагалось рассмотреть вероятности 0.0, 1.0 и несколько промежуточных.

1.1. Задача об умном муравье-3

Дано поле размером 32×32, расположенное на поверхности тора. В каждой клетке поля с некоторой вероятностью находится яблоко. Муравей начинает движение из некоторой фиксированной клетки. За один ход можно выполнить следующие действия:

- повернуть налево;
- повернуть направо;
- сделать шаг вперед (если в клетке есть яблоко, то оно считается съеденным);
- ничего не делать.

Всего делается 200 ходов. Муравей видит перед собой 8 клеток (рис. 1).

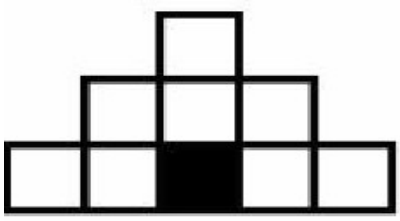

Рис.1 — Область, видимая муравьем

Требуется построить автомат с заданным числом состояний, который управляет муравьем так, что муравей ест как можно больше яблок.

2. Реализация

Исходный программный код написан на языке java. Для построения графиков использовалась библиотека jfreechart (http://www.jfree.org/jfreechart/). Для получения случайных чисел использовался алгоритм, известный как вихрь Мерсенна. Реализация этого алгоритма была взята с сайта http://cs.gmu.edu/~sean/research/mersenne/MersenneTwister.java.

2.1. Представление автомата деревом решений

Дерево решений (рис. 2) является способом задания булевой функции, зависящей от дискретного числа булевых переменных. Внутренние вершины дерева маркированы символами переменных, дуги — значениями этих переменных, а в листьях содержатся значения функции.

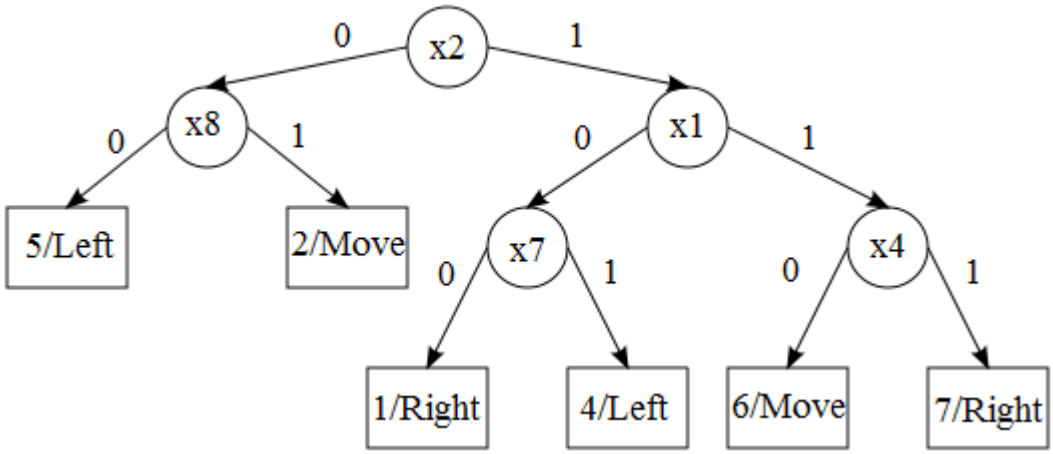

Рис. 2 — Дерево решений

Для задания автомата необходимо задать функции переходов с помощью таких деревьев, то есть надо каждому состоянию автомата сопоставить функции переходов и выходов. Каждая из таких функций может быть выражена с помощью дерева решений. В этих деревьях переменными являются входные переменные автомата, а множество значений — пары вида (Состояние, Действие).

2.2. Генетический алгоритм

На рис. 3 изображена общая схема генетического алгоритма:

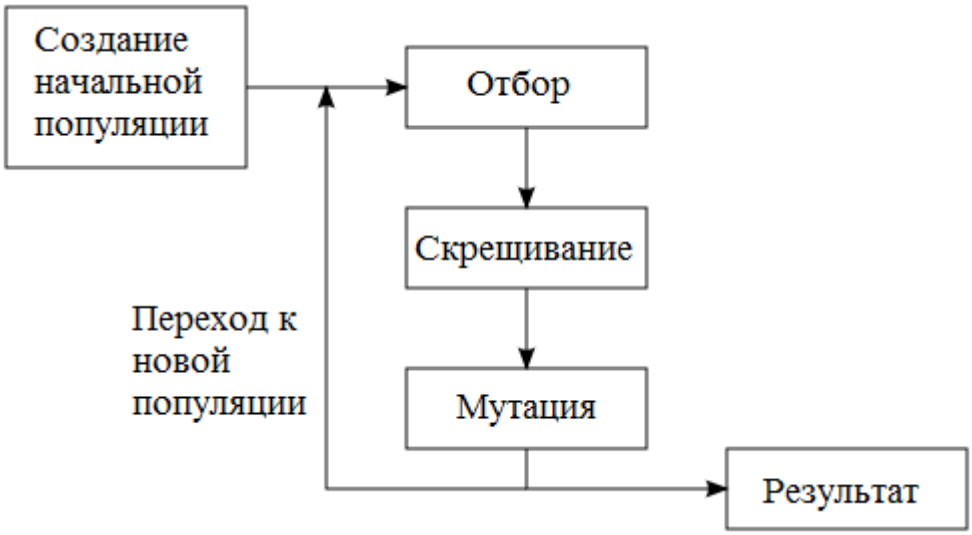

Рис. 3 — Схема генетического алгоритма

Начальная популяция формируется случайных образом. Следует заметить, что каждая особь является решением поставленной задачи.

В лабораторной работе рассматривались следующие стратегии отбора:

1. *Ранговый отбор* (*rank selection*) — для каждой особи ее вероятность попасть в промежуточную популяцию пропорциональна ее порядковому номеру в отсортированной по возрастанию приспособленности популяции. Такой вид отбора не зависит от средней приспособленности популяции.
2. *Турнирный отбор* (*tournament selection*) осуществляется следующим образом: из популяции случайным образом выбирается *t* особей, и лучшая из них помещается в промежу-

точную популяцию. Этот процесс повторяется N раз, пока промежуточная популяция не будет заполнена. В лабораторной работе рассматривался вариант $t = 2$.

2.3. Генетические операции над деревьями решений

Определим следующие операции над деревьями решений:

- Случайное порождение дерева. При этом случайным образом выбирается является ли вершина листом, то есть либо случайным образом выбирается переменная, либо — значение функции. Если была выбрана переменная, то рекурсивно генерируются дети вершины. В лабораторной работе вершина становилась листом с вероятностью 15%. Кроме того, исходно была ограничена высота дерева, а именно высота каждого дерева не превосходит удвоенного числа переменных.
- Мутация дерева решений (рис. 4). Выбирается случайный узел в поддереве. После этого поддерево, соответствующее выбранной вершине, заменяется на случайно порожденное.

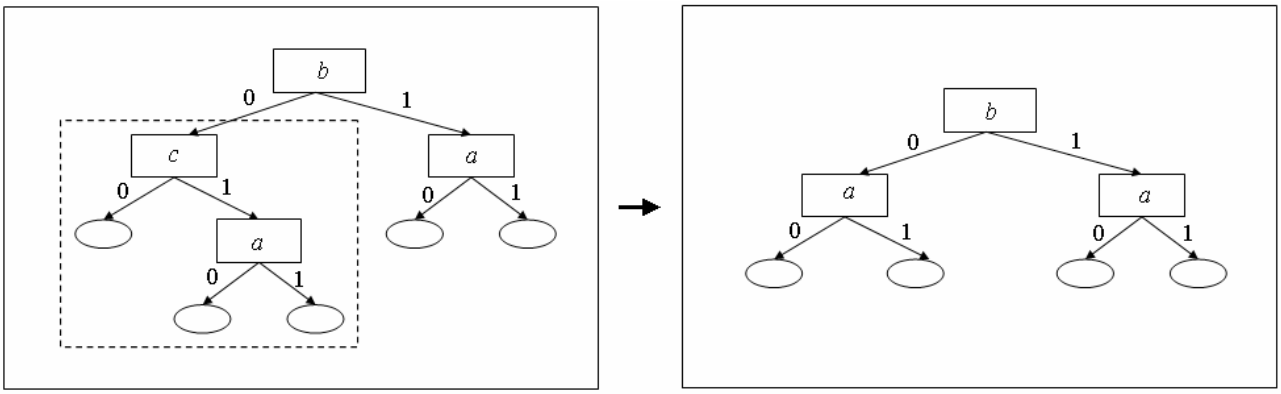

Рис. 4 — Мутация дерева решений

- Скрещивание деревьев решений (рис. 5). В каждом из скрещиваемых деревьев выбирается по случайной вершине. После этого поддерево, соответствующее выбранной вершине в первом дереве, заменяется на поддерево, соответствующее выбранной вершине во втором дереве.

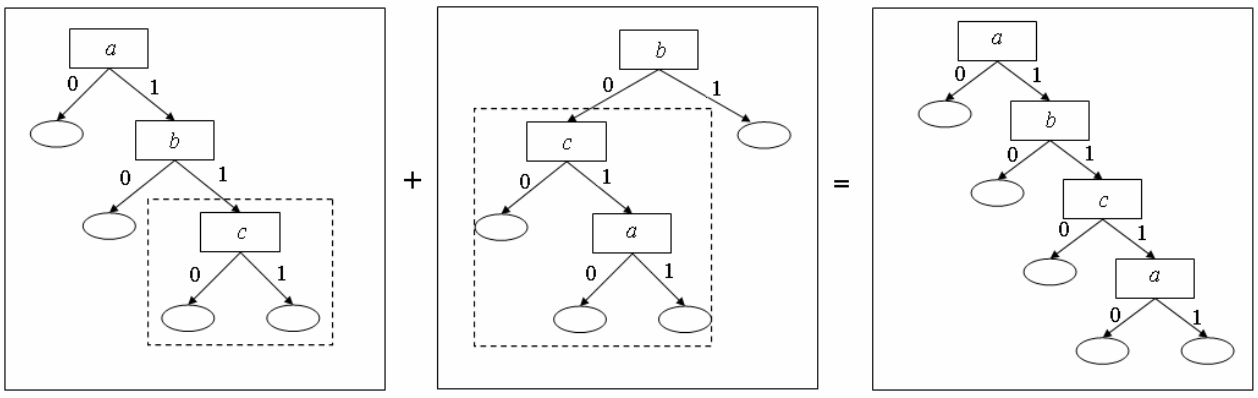

Рис. 5 — Скрещивание деревьев решений

Стоит отметить, что после выполнения этих операций могут порождаться деревья, у которых некоторые метки повторяются на пути от корня до листа. Таким образом, необходимо ввести операцию подрезания дерева — удаления недостижимых ветвей. Эта операция выполняется с некоторой вероятностью после операций скрещивания и мутации.

3. Результаты работы

Алгоритм тестировался при следующих параметрах:

1. вероятность нахождения еды в ячейке — 5%;
2. вероятность мутации — 0.7%;
3. 50 особей в каждом поколении;
4. функция приспособленности для заданного поля рассчитывалась по формуле: *Fitness = Apples + (200 – Step)/200,* где *Apples* – число яблок, съедаемых муравьем за 200 шагов, *Step* – номер шага, на котором муравей съедает последнее яблоко. Поскольку вычисления производились на 30 полях, то бралось среднее значение функции приспособленности.

На рис. 6 приводится график, получаемый в ходе выполнения работы.

Рис. 6 — График максимальной и средней высоты дерева

Производились вычисления следующих параметров:

- максимальная высота дерева в поколении;
- средняя по поколению высота деревьев, достижимых из стартового состояния.

При рассмотренных стратегиях отбора алгоритм показал в целом одинаковое поведение. Если выставлять достаточно маленькую вероятность подрезания дерева, то со временем появлялись деревья, которые вырастали довольно высоко. Средняя высота по поколению оставалась в допустимых пределах. На рис. 7, 8 показано, как выглядит такое поведение.

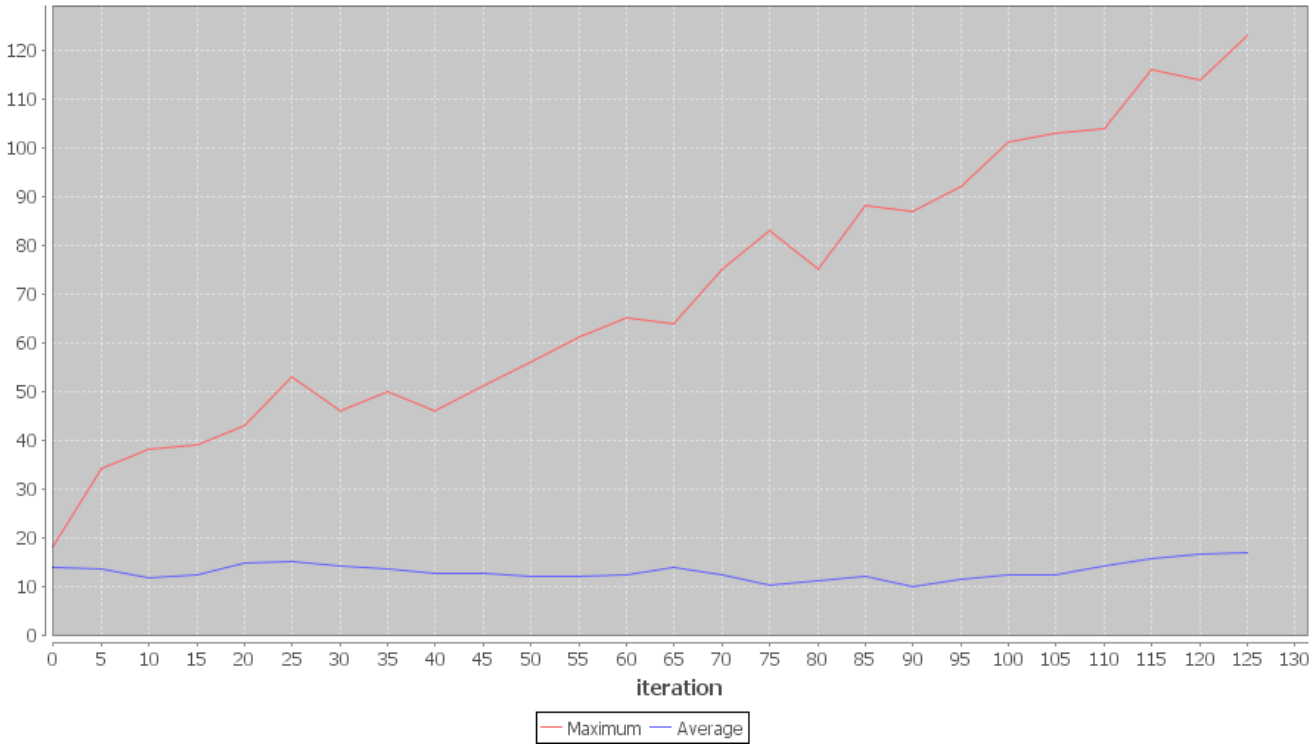

Рис. 7 — Ранговый отбор. Вероятность подрезания дерева 1%

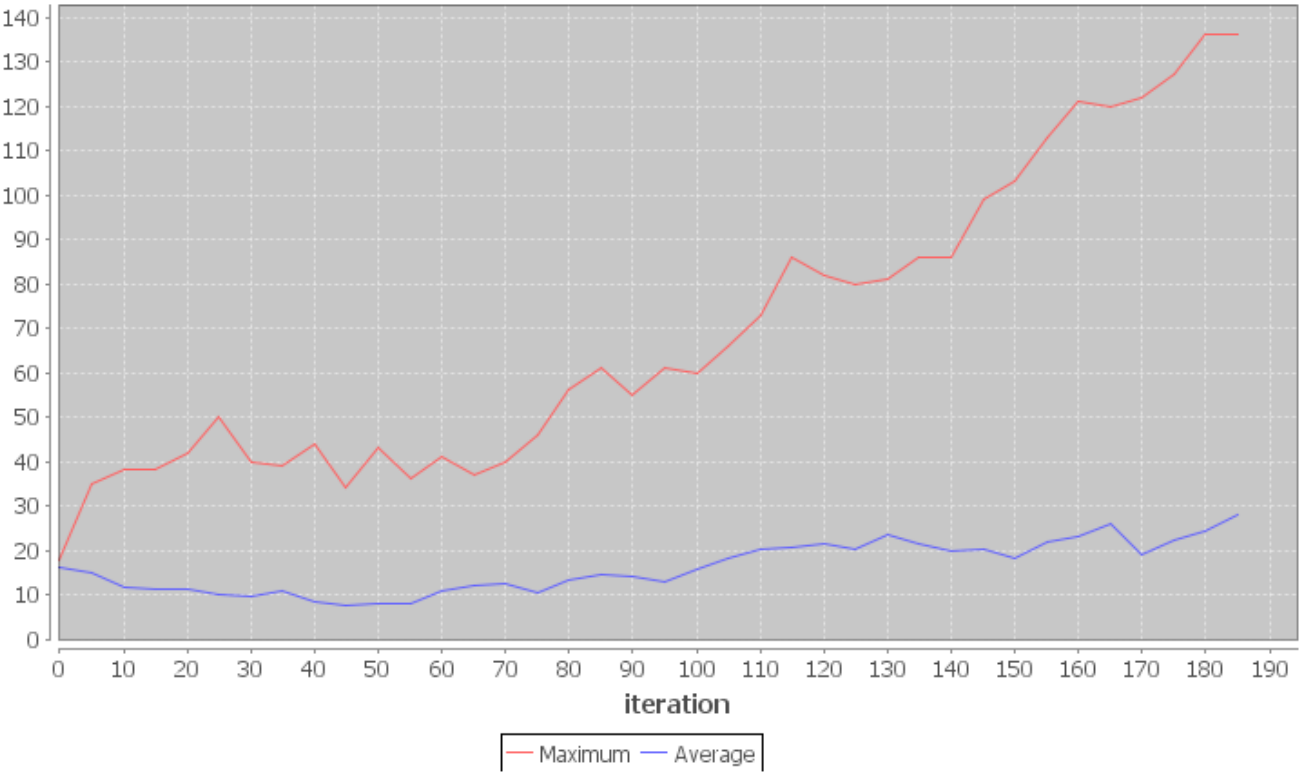

Рис. 8 — Турнирный отбор. Вероятность подрезания дерева 1%

В ходе выполнения работы измерения высоты производились раз в пять популяций. Функция подсчета рекурсивно запускалась от корня дерева. При достаточно малых значениях вероятности применения упрощающих операторов время работы программы увеличивалось, иногда в 3-4 раза. Было проведено множество запусков алгоритма и в результате оказалось, что время затрачиваемое на выполнение коррелирует с максимальной высотой дерева в популяции. После более тщательного изучения проблемы выяснилось, что типичным является следующий

сценарий развития автомата:

1. В первую очередь разрастается одно из состояний автомата. Высота остальных колеблется в допустимых границах.
2. Увеличивается число задействованных состояний.

В силу этого, средняя по поколению средняя высота дерева в автомате не является показательной величиной. Решено было переключится на рассматривание средней по поколению максимальной высоты дерева в автомате. После этого был получен график, приведенный на рис. 9.

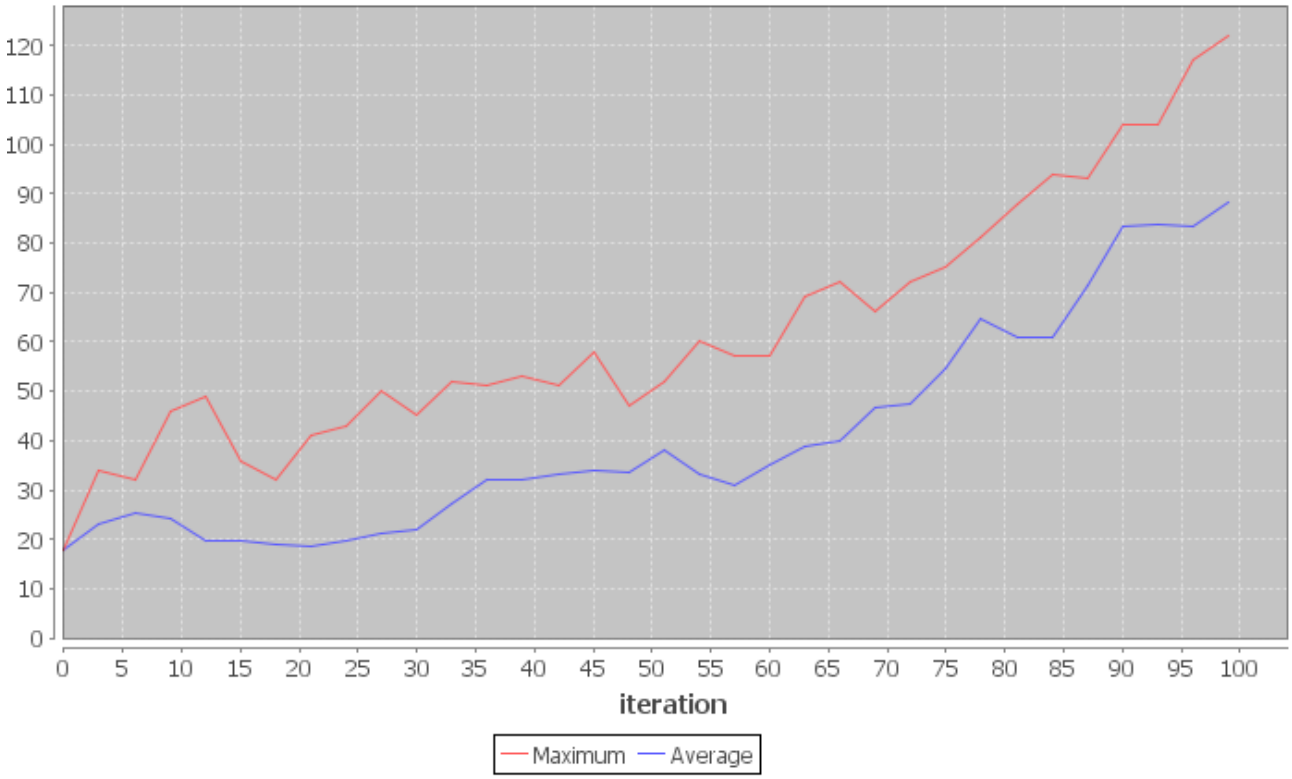

Рис. 9 — График с новой усредненной величиной

В этом случае нагляднее показана тенденция разрастания деревьев. Если ограничить высоту и рассмотреть, сколько итераций необходимо алгоритму, чтобы при фиксированной вероятности использования упрощающих операторов средняя по поколению максимальная высота дерева в автомате превысила установленный порог, то можно получить представление о скорости разрастания деревьев решений. Табл. 1 получается, если в качестве максимальной высоты взять 60 и усреднить результаты.

Таблица 1 — Усредненные результаты эксперимента

	Rank selection	Tournament selection
90%	3000	3000
50%	3000	3000
25%	3000	3000
10%	875,91	685,92
5%	310,74	310,54
1%	207,84	193,02

Исследуем влияние упрощающих операторов на эффективность алгоритма. График функции приспособленности (рис. 10) ведет себя немонотонно. Это связано с тем, что каждое новое поколение обучалось на новом наборе случайных карт. Кроме того, в алгоритме отсутствовал элитизм среди особей.

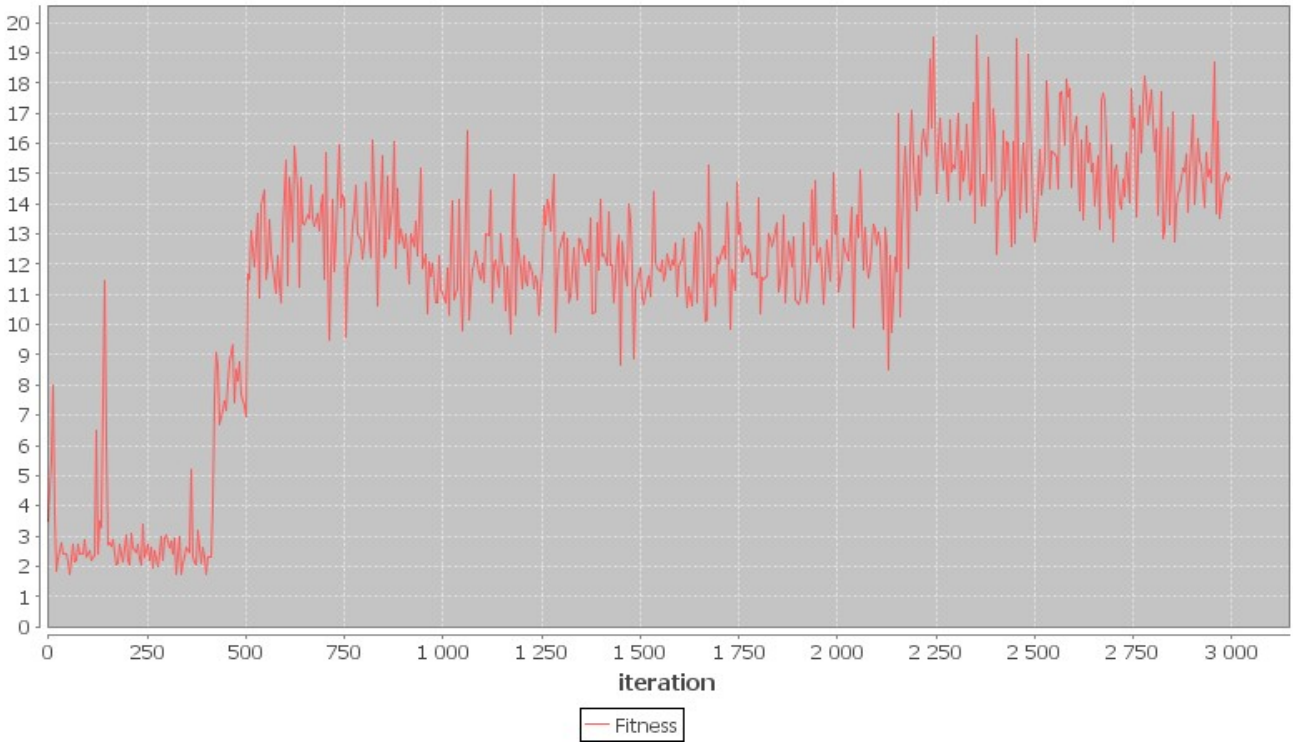

Рис. 10 — График функции приспособленности

Исправим эти недочеты следующим образом:

- Зафиксируем 50 полей. Для увеличения количества съеденных яблок на произвольном поле будем генерировать карты с различными вероятностями нахождения еды в ячейке.
- Добавим элитизм. Пусть 10% особей переходят в следующее поколение без изменений.
- В качестве функции приспособленности возьмем процент съеденных яблок.

График функции приспособленности, получаемый после внесения указанных выше изменений, представлен на рис. 11.

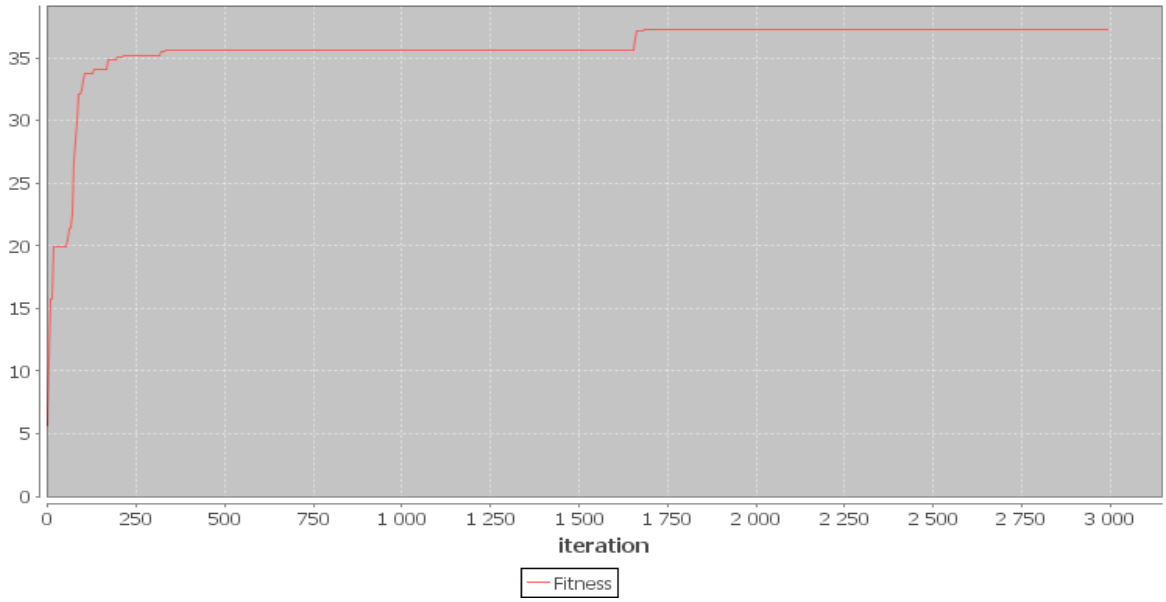

Рис. 11 — Новый график функции приспособленности.

Проведем аналогичные измерения, чтобы установить как изменение в алгоритме повлияло на скорость разрастания деревьев (табл. 2).

Таблица 2 — Новые результаты эксперимента

	Rank selection	Tournament selection
90%	3000	3000
50%	3000	3000
25%	3000	3000
10%	824.3	955.5
5%	478.1	345.2
1%	211.5	184.9

Можно рассматривать влияние упрощающих операторов на скорость роста функции приспособленности. Для этого проведем множество экспериментов с одинаковыми начальными условиями, но разными вероятностями применения упрощающих операторов. В результате получаем графики (рис. 12, 13), которые ведут себя схожим образом. Исходя из этого, можно сделать вывод, что использование упрощающих операторов не влияет на скорость роста функции приспособленности.

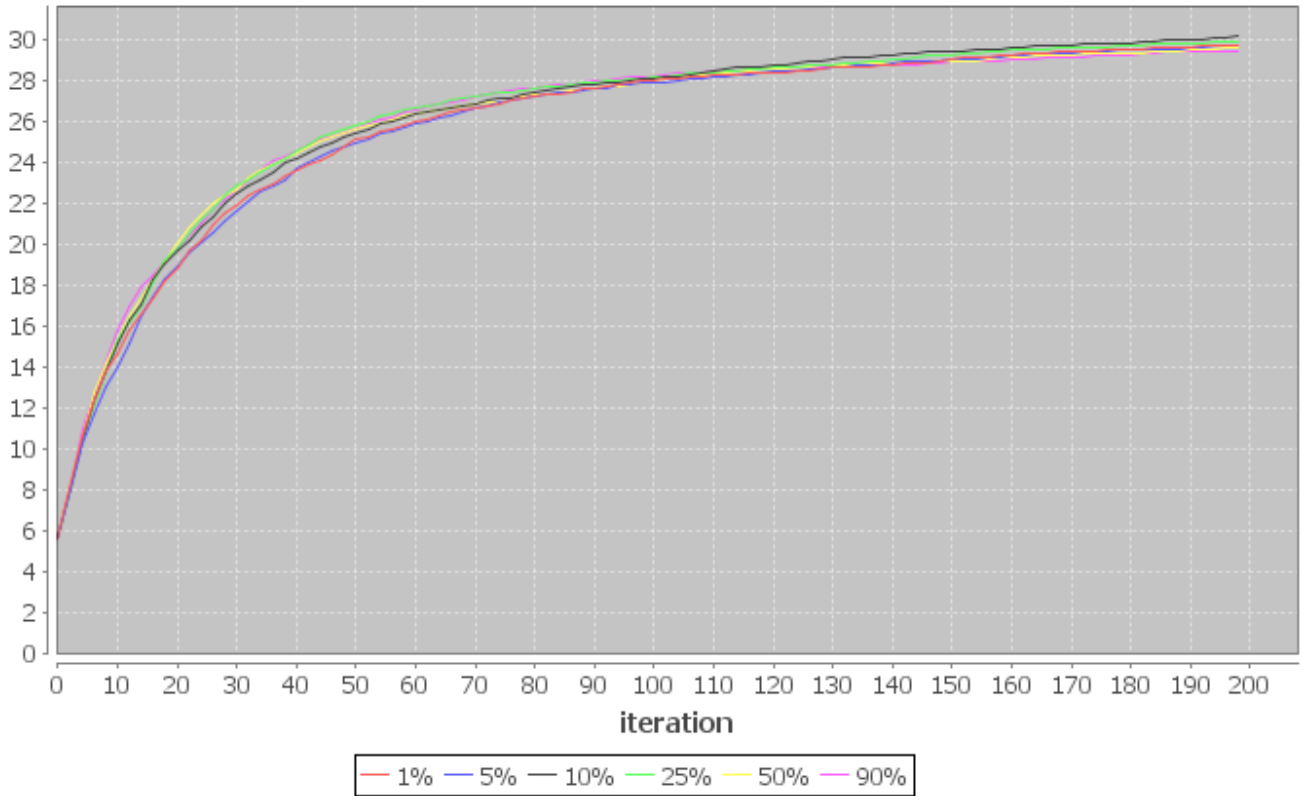

Рис. 12 — Усредненные графики функции приспособленности при ранговом отборе

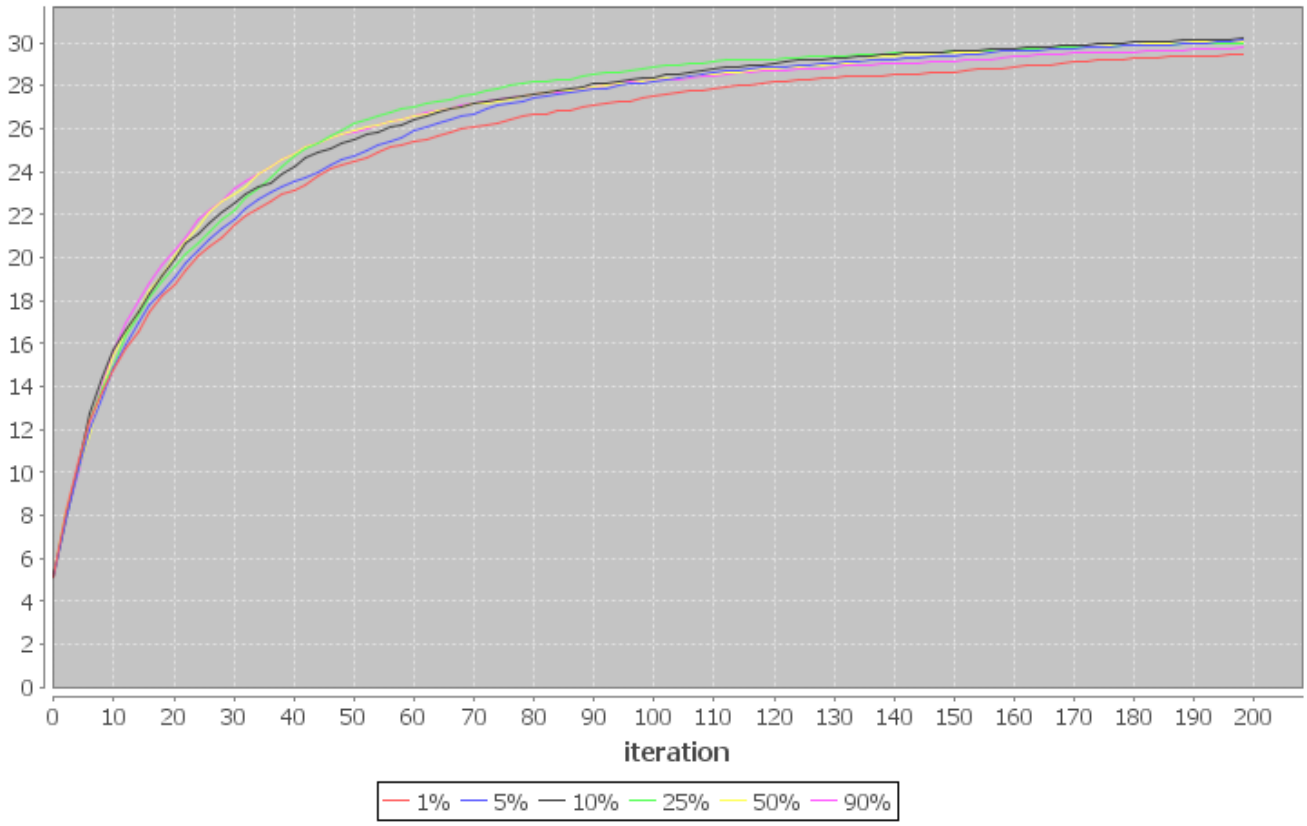

Рис. 13 — Усредненные графики функции приспособленности при турнирном отборе

Заключение

В работе было рассмотрено использование упрощающих операторов для деревьев решений для двух стратегий отбора. При небольшой вероятности подрезания деревьев наблюдалось сильное разрастание некоторых представителей, что негативно сказывалось на времени работы программы. Поскольку использование упрощающих операторов не сказывается на эффективности алгоритма, рекомендуется оставлять вероятность подрезания деревьев не менее 25%.

Источники

1. *Поликарпова Н. И., Шалыто А. А.* Автоматное программирование. СПб.: Питер, 2009.
2. *Данилов В.Р., Шалыто А.А.* Метод представления функций переходов деревьями решений для генерации автоматов с помощью генетического программирования http://is.ifmo.ru/works/_2009_08_01_danilov.pdf