Санкт-Петербургский государственных университет информационных технологий, механики и оптики

Факультет информационных технологий и программирования
Кафедра компьютерных технологий

Е. А. Мандриков

Совместное использование автоматного программирования и генетических алгоритмов для управления мультиагентными системами.

Магистерская диссертация

Научный руководитель: А. А. Шалыто, д. т. н., профессор

Санкт-Петербург
2010

ОГЛАВЛЕНИЕ

СПИСОК ТЕРМИНОВ

Автоматное программирование – парадигма программирования, при использовании которой программа или её фрагмент осмысливается как модель какого-либо формального автомата.

Искусственный интеллект – точного определения этой науки не существует, но её принято считать разделом информатики, который изучает возможность обеспечения разумных рассуждений и действий с помощью вычислительных систем и иных искусственных устройств. При этом в большинстве случаев заранее неизвестен алгоритм решения задачи.

Эволюционный алгоритм – точного определения этой науки не существует, но её принято считать разделом информатики, который изучает возможность обеспечения разумных рассуждений и действий с помощью вычислительных систем и иных искусственных устройств. При этом в большинстве случаев заранее неизвестен алгоритм решения задачи.

Популяция – совокупность индивидуумов, способная к устойчивому самовоспроизводству, относительно обособленная от других групп, с представителями которых потенциально возможен генетический обмен.

Индивид (особь) – единичный представитель популяции.

Генотип – наследственная информация, закодированная в хромосомах, которая вместе с факторами внешней среды определяет фенотип организма. Генотип не всегда соответствует одному и тому же фенотипу. Некоторые гены проявляются в фенотипе только в определённых условиях.

Ген – определённая часть хромосомы, кодирующая врождённое качество индивида.

Хромосома – структура, содержащая генетический код индивида.

Фенотип – совокупность характеристик, присущих индивиду на

определённой стадии развития. Фенотип формируется на основе генотипа, опосредованного рядом внешнесредовых факторов.

Интеллектуальный агент – в искусственном интеллекте, под данным термином понимаются разумные сущности, наблюдающие за окружающей средой и действующие в ней, при этом их поведение рационально в том смысле, что они способны к пониманию и их действия всегда направлены на достижение какой-либо цели.

Многоагентная система или *мультиагентная система* (*МАС*, англ. *Multi-agent system*) – это система, образованная несколькими взаимодействующими интеллектуальными агентами.

Самоорганизация – процесс упорядочения в системе за счёт внутренних факторов, без внешнего специфического воздействия.

Ряд терминов, используемых в данной работе, не имеет точного и устоявшегося перевода на русский язык в виду малого числа публикаций в нашей стране, посвящённых эволюционным алгоритмам. Поэтому эти термины в данной работе приводятся с оригинальным написанием на английском языке.

Offspring – результат репродукции одного или нескольких индивидов.

ВВЕДЕНИЕ

Для ряда задач поведение системы удобно представлять в виде конечных автоматов. Примерами таких задач могут служить: лексический и синтаксический анализ [1], визуализация алгоритмов [2], автоматизированное управление, сетевое взаимодействие и др. Данный подход к описанию поведения системы используется в парадигме автоматного программирования [3] [4] [5].

В большинстве случаев автоматы проектируются вручную, однако эвристическое построение автоматов часто затруднено или невозможно. На данный момент эволюционные вычисления являются одним из наиболее используемых направлений искусственного интеллекта, который успешно применяется для автоматического создания программ. В частности можно вести речь об автоматическом создании автоматных программ при помощи генетического программирования [6] [7].

В последнее время возрос интерес к построению модульных роботов [8]. Поэтому целесообразно поставить вопрос о генерации автоматных программ для управления такими системами.

По теме магистерской работы опубликовано 14 печатных работ, в том числе в двух журналах, входящих в перечень ведущих рецензируемых научных журналов Российской Федерации. Список публикаций приведён на странице 45.

Работа содержит 50 страниц и состоит из списка терминов, введения, 3 глав и заключения. Список литературы содержит 18 наименований. Работа иллюстрирована 19 рисунками и содержит 2 таблицы.

Глава 1 содержит основные понятия о генетических алгоритмах, производится обзор существующих методов генерации конечных автоматов. Приводятся основные характеристики многоагентных систем, рассматриваются

агенты, чьим поведением управляет автомат.

В главе 2 приведено описание экспериментальных задач, решение которых невозможно без кооперативного поведения агентов. На примере данных задача продемонстрирована применимость автоматного и генетического программирования для управления многоагентными системами. Описан способ представления нескольких автоматов в виде составной хромосомы.

В главе 3 описывается программное средство GAAP (Genetic Algorithms for Automata-based Programming) для генерации автоматов управления с помощью генетического программирования, которое было разработано автором данной работы и использовано для её выполнения.

ГЛАВА 1. АВТОМАТНОЕ И ГЕНЕТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ, МНОГОАГЕНТНЫЕ СИСТЕМЫ

В данной главе приводятся основные понятия о генетическом программировании, производится обзор существующих методов генерации конечных автоматов. Приводятся основные характеристики многоагентных систем, рассматриваются агенты, чьим поведением управляет автомат.

1.1. Генетическое программирование

В данном разделе приводятся основные понятия об эволюционных вычислениях и генетическом программировании.

1.1.1. Эволюционные вычисления

Эволюционные вычисления – это методы оптимизации, базирующиеся на эволюции популяции особей, в процессе которой ведётся поиск максимального значения целевой функции. Эти методы применимы для задач оптимизации плохо определённых функций. Основной идеей эволюционных вычислений является использование принципа естественного отбора, заключающегося в том, что наиболее приспособленные особи дают потомство, формирующее следующее поколение. В среднем, следующее поколение является более приспособленным к окружающей среде, чем предыдущее.

Сегодня эволюционные вычисления применяются в различных сферах науки: начиная с дизайна антенн для спутников и заканчивая расшифровкой генома человека.

Наличие большого числа модификаций эволюционных алгоритмов и параметров их настройки, которые могут динамически изменяться от итерации к итерации, затрудняют решение конкретной задачи. А то, насколько удачным

окажется применение эволюционного алгоритма при решении той или иной задачи, во многом зависит от выбранных параметров. Вообще говоря, строгих правил, универсальных для всех задач, нет и быть не может. Так как в теореме NFL (*No Free Lunch*) [9] утверждается, что для всех алгоритмов, которые определяют экстремум функции качества, производительность их одинакова, если усреднить результаты по всем возможным функциям качества. Практическое значение этой теоремы состоит в том, что не существует панацеи на все случаи жизни, а несомненный успех какого-либо оптимизационного метода в определённой области знаний не гарантирует такого же успеха в другой области. Это означает, что для каждой специфической области необходимо проводить исследования и отыскивать тот оптимизационный метод, который подходит для неё наилучшим образом.

1.1.2. Общая схема работы эволюционных алгоритмов

Функция приспособленности. При использовании эволюционных алгоритмов поиск оптимума ведётся в пространстве гипотез X. Целевая функция $f: X \to R$ называется *функцией приспособленности* или *фитнесс-функцией*. Задача эволюционного алгоритма – максимизация функции приспособленности.

Особью эволюционного алгоритма принято называть некоторую гипотезу $i \in X$. *Поколением* или *популяцией* особей называют множество гипотез P_t, где t – *возраст популяции*, а мощность множества гипотез $|P_t|$ – *размер популяции*. *Возраст особи* $i \in P_t$ – число популяций $P_j : i \in P_t, j \le t$. Значение целевой функции для особи $i \in P_t$ будем обозначать как $f_t(i)$. Закодированный генотип особи принято называть *хромосомой*.

Кроссовер – Как известно в теории эволюции, важную роль играет то, каким образом признаки родителей передаются потомкам. В генетических алгоритмах за передачу признаков родителей потомкам отвечает *оператор*

кроссовера. В литературе по генетическим алгоритмам этот оператор называют также кроссинговер, скрещивание, рекомбинация. В дальнейшем в данной работе будет использоваться термин кроссовер.

Итак, кроссовер – процесс обмена генетическим материалом. Это операция, при которой две хромосомы обмениваются своими частями. Отличительной особенностью генетических алгоритмов от эволюционных является использование оператора кроссовера. Как правило, оператор кроссовера принимает две особи и порождает одну или две.

Мутация – случайное изменение одной или нескольких позиций в хромосоме. Мутации, которые проявляются на уровне фенотипа, могут иметь как пагубные последствия, так и положительные – приводить к появлению у особи новых полезных признаков. Таким образом, мутации являются двигателем естественного отбора, так как являются механизмом поддержания разнообразия особей в популяции.

Генерация случайной особи. Как правило, выделяют ещё один оператор для генетических алгоритмов – оператор генерации случайной особи, который может быть использован при создании начальной популяции, при пополнении популяции случайными особями и в качестве мутации.

Схема эволюционного алгоритма изображена на рис. 1.

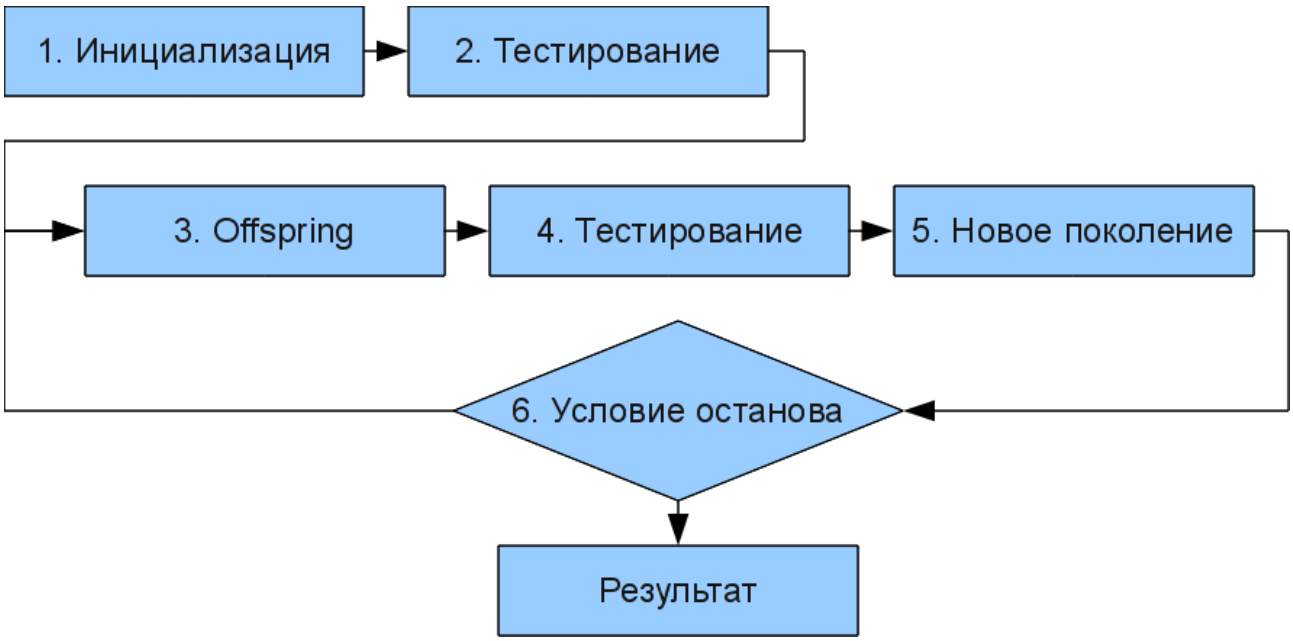

Рис. 1. Общая схема работы эволюционного алгоритма

Таким образом, принцип действия эволюционного алгоритма:

1. Инициализация. Создать случайное множество гипотез $P_t = i_k^t, t = 0$ – *начальная популяция*.
2. Тестирование начальной популяции. Для каждого $i_k^0 \in P_0$ произвести вычисления $f_0(i_k^0)$.
3. Формирование offspring. Путём отбора особей из P_t и применения к ним генетических операторов, сформировать потомство $O = i'_k$.
4. Тестирование. Для каждого $i'_k \in O$ произвести вычисления $f_t(i'_k)$.
5. Формирование следующего поколения. Путём отбора особей из $P_t \cap O$ сформировать $P_{t+1} = i_k^{t+1}$.
6. Повторять шаги 3–5 для $t = 1,2,\ldots$ до тех пор, пока не выполнится условие останова.

Тестирование обычно является наиболее трудоёмкой по времени операцией. Существует несколько ключевых моментов:

1. Для каждого $i \in P_t$ значение $f_t(i)$ может вычисляться независимо от значений $P_t \backslash i$.
2. Допускается, что значения функции приспособленности $f_{t_1}(i)$ и $f_{t_2}(i)$ для особи i, вычисленные для популяций P_{t_1} и P_{t_2} соответственно, могут быть связаны отношением: $f_{t_1}(i) \neq f_{t_2}(i)$.

Математическое обоснование того, почему эволюционный алгоритм находит оптимальное решение, не приведено. Для наиболее простых моделей эволюционных алгоритмов сходимость может быть доказана с помощью теоремы о схемах [10]. В работе [11] показано, что в рамках некоторых условий использование оператора кроссовера позволяет найти оптимум за конечное время, что неверно без его использования.

1.1.3. Стратегии отбора

Существует несколько стратегий при выборе особей для проведения генетических операторов и при формировании нового поколения.

- *Турнирный отбор* может быть описан следующим образом: из популяции P_t выбирается случайным образом k особей и особь с максимальным значением функции приспособленности из выбранных попадает в offspring. Данная операция повторяется $|P_t|$ раз. Число k называется *численностью турнира* (*tournament size*). Она часто равна двум. В этом случае говорят о *двоичном* или *парном турнире* (*binary tournament*). Чем больше численность турнира, тем меньше шансов у особей.
- При *пропорциональном отборе* находят значение $f' = \frac{\sum_{\forall i \in P_t} f_t(i)}{|P_t|}$. Затем для каждой особи вычисляется отношение $\lfloor \frac{f_t(i)}{f'} \rfloor$, которое показывает сколько раз должна быть отобрана данная особь.

- При *отборе усечением* выбираются все особи $i \in P_t$ для которых $f_t(i) \geq f'$, где f' – некоторое пороговое значение.
- При *отбор по рулетке* вероятность выбора особи $i \in P_t$ прямо пропорциональна значению $f_t(i)$.

Следует отметить, что при пропорциональном отборе и отборе по рулетке необходимо дополнительное ограничение на значение функции приспособленности: $\forall i \in P_t : f_t(i) > 0$.

1.1.4. Стратегии формирования нового поколения

Формирование следующей популяции можно производить при помощи стратегий отбора описанных выше. При этом существуют разные подходы:

- В новую популяцию могут попадать только новые особи (принцип замещения родителей).
- Потомки могут сравниваться с родителями и в случае увеличения значения функции приспособленности заменять их.
- Использование *принципа «элитизма»* заключается в том, что в новую популяцию всегда попадает k лучших особей предыдущей популяции. При использовании данного принципа принято говорить, что производится отбор «с элитизмом k». Использование «элитизма» позволяет не потерять хорошую особь, но в то же время, из-за этого алгоритм может «застрять» в локальном экстремуме. Поэтому имеет смысл совместно с «элитизмом» вводить ограничение на максимальное время жизни особи.

1.1.5. Условие останова

Условием останова могут выступать различные условия. Имеет смысл привести наиболее используемые:

- Достижение максимального значения функции пригодности f_{max}. Следует отметить, что для некоторых задач невозможно указать значение $f_{max} = \max_{i \in X} f(i)$, т.к. его вычисление сопоставимо по сложности с поиском точки i_{max}, на котором оно достигается ($f_{max} = f(i_{max})$).
- Достижение максимального возраста популяции t_{max}.
- Прекращение роста максимальной приспособленности в популяции на протяжении некоторого числа последних популяций.
- *Вырождение популяции*: $\sigma < \epsilon$, где σ – стандартное отклонение для значений приспособленности популяции, а ϵ – наперёд заданная константа.

1.1.6. Генетическое программирование

Генетическое программирование является эволюционным алгоритмом, особями которого являются компьютерные программы. За последнее десятилетие в связи с резким увеличением мощности компьютеров возрос интерес к генетическому программированию как к средству автоматизации разработки программного обеспечения. В настоящее время с помощью генетического программирования получен ряд результатов, превосходящих аналогичные результаты людей, например создание сортирующей сети и т.д.

Ключевой идеей генетического программирования является представление хромосомы на достаточно высоком уровне абстракции, позволяющем учитывать специфику и структуру компьютерных программ.

1.2. Генерация управляющих автоматов

В ряде задач удобно представление логики в виде конечного автомата Мили [4] [5], описывающего работу объекта в виде последовательностей состояний, в каждом из которых каким-то образом обрабатываются входные воздействия. На основе исследований [6] [7], предлагается в качестве абстракций для генетического программирования использовать автоматные программы.

1.2.1. Конечные детерминированные автоматы

Конечным детерминированным автоматом A типа Мили называется совокупность шести объектов $(S, S_0, X, Z, \delta, \lambda)$, которые имеют следующий смысл:

- S – конечное множество состояний автомата;
- $S_0 \in S$ – начальное состояние;
- X – конечное множество входных воздействий;
- Z – конечное множество выходных воздействий;
- $\delta : S \times X \to S$ – функция переходов автомата;
- $\lambda : S \times X \to Z$ – функция действий автомата,

где элементы множеств S, X и Z связаны в абстрактном времени $T = 0,1,2,\ldots$ следующими отношениями: $\begin{cases} s(t+1) = \delta(s(t), x(t)); \\ z(t) = \lambda(s(t), x(t)). \end{cases}$

В рамках парадигмы автоматного программирования [3] конечный автомат Мили может быть обобщён за счёт понятия предикатов – отображений вычислительных состояний в логические переменные. Заметим, что они не могут быть вынесены в состояния автомата, так как зависят от текущей ситуации. Переход будет осуществляться в зависимости не только от входных

воздействий и текущего состояния, но и от значений предикатов. Предикаты также называются входными переменными. Также имеет смысл использовать несколько выходных воздействий при переходе, причём в таком случае необходимо задание порядка выполнения выходных воздействий. Таким образом, если множество предикатов обозначить за P, то указанные функции примут следующий вид: $\begin{cases} \delta : S \times X \times 2^P \to S \\ \lambda : S \times X \times 2^P \to Z' \end{cases}$, где $Z' \in Z''$.

Следует отметить, что от предикатов и множественных выходных воздействий можно избавиться, увеличив набор входных и выходных воздействий. Однако такой способ усложняет логику системы для восприятия человеком. Кроме того, при этом множество входных воздействий растёт экспоненциально от числа входных переменных, а также растёт число состояний автомата.

1.2.2. Представление автоматов в виде хромосом

Существуют различные методы представления автоматов в виде хромосом:

- битовые строки;
- строки над числами;
- деревья решений;
- сокращённые таблицы;
- графы.

Полное описание приведённых методов выходит за рамки данной работы, поэтому опущено.

1.2.3. Некоторые задачи генерации автоматов

Примерами задач генерации управляющих автоматов являются:

- Задача об умной муравье.
- Задача ориентирования на местности.
- Распознавание регулярных языков.
- Обобщённая задача о муравье.
- Повторяющаяся дилемма заключённого.
- Задача построения управляющего автомата для разливочной линии.
- Задача о «флибах».
- Построение системы управления беспилотным летательным аппаратом.

1.3. Многоагентные системы и автоматы

В данном разделе приводятся основные характеристики многоагентных систем и рассматриваются агенты, чьим поведением управляет автомат.

1.3.1. Многоагентные системы

Многоагентная система (МАС, англ. *Multi-agent system*) – это система, образованная несколькими взаимодействующими интеллектуальными агентами.

Интеллектуальный агент – в искусственном интеллекте, под данным термином понимаются разумные сущности, наблюдающие за окружающей средой и действующие в ней, при этом их поведение рационально в том смысле, что они способны к пониманию и их действия всегда направлены на достижение какой-либо цели. На рис. 1 приведена схема такого агента.

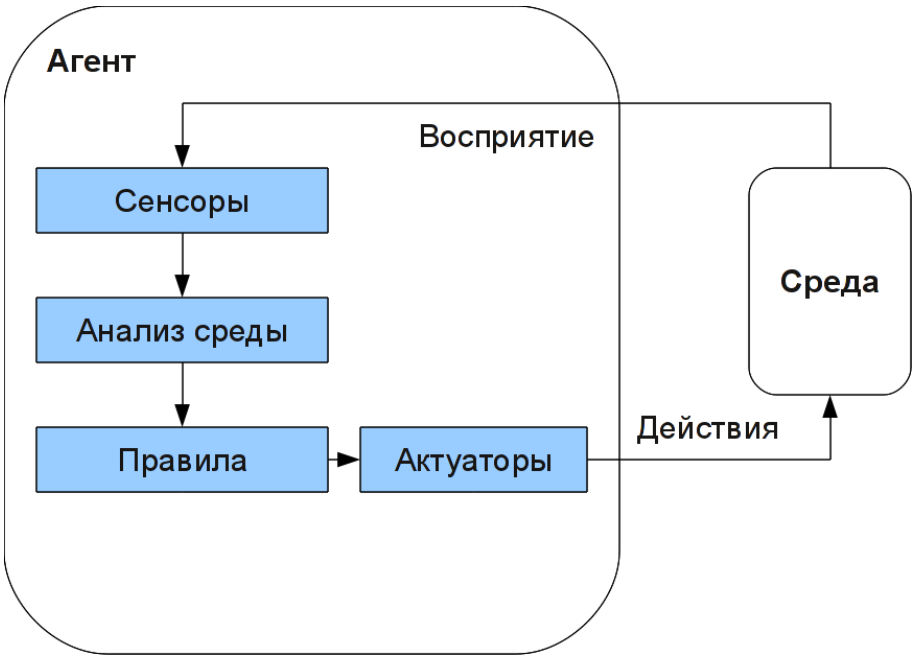

Рис. 2. Агент

В многоагентной системе (МАС) агенты имеют несколько важных характеристик [12]:

- **Автономность**: агенты, хотя бы частично, независимы.
- **Ограниченность представления**: ни у одного из агентов нет представления о всей системе, или система слишком сложна, чтобы знание о ней имело практическое применение для агента.
- **Децентрализация**: нет агентов, управляющих всей системой [13].

В многоагентных системах может проявляться самоорганизация и сложное (коллективное) поведение даже если стратегия поведения каждого агента достаточно проста. Это лежит в основе муравьиных алгоритмов. *Самоорганизация* – процесс упорядочения в системе за счёт внутренних факторов, без внешнего специфического воздействия.

Главное достоинство МАС – это гибкость. Многоагентная система может быть дополнена и модифицирована без переписывания значительной части программы. Также эти системы обладают способностью к самовосстановлению

и обладают устойчивостью к сбоям, благодаря достаточному запасу компонентов и самоорганизации.

Агенты могут быть сконструированы по-разному. Различия могут заключаться в оборудовании и программном обеспечении. Часто таких агентов называют *гетерогенными* в противоположность *гомогенным*, которые сконструированы идентичным образом и имеют изначально одинаковые возможности.

Окружение (среда) агентов может быть статическим (не зависящим от времени) или динамическим. Из-за простоты обработки и более строгого математического описания большинство методов искусственного интеллекта в одноагентном случае были разработаны для статического окружения. В многоагентных системах само присутствие нескольких агентов делает окружение динамическим.

Информация, которой обладает система агентов, является обычно распределённой: агенты могут следить за окружающей средой из разных положений, получать информацию в различные моменты времени или интерпретировать эту информацию по-разному. Таким образом, состояние окружающей среды является частично обозримым для каждого агента. Так как рассматриваемые агенты гомогенные, то и восприятие информации у них одинаковое.

В дальнейшем мы будем рассматривать только вычислительных агентов, т.е. тех, что созданы для решения специальных задач и приводятся в исполнение на вычислительных устройствах.

1.3.2. Автоматный агент

В работе [7] рассматривается совместное применение генетического программирования, конечных автоматов и искусственных нейронных сетей для

построения системы управления беспилотным летательным аппаратом. Летательный аппарат фактически является гомогенным агентом, который может быть изображён схематически как показано на рис. 3. И рассматривается соревнование (игра) двух групп таких агентов.

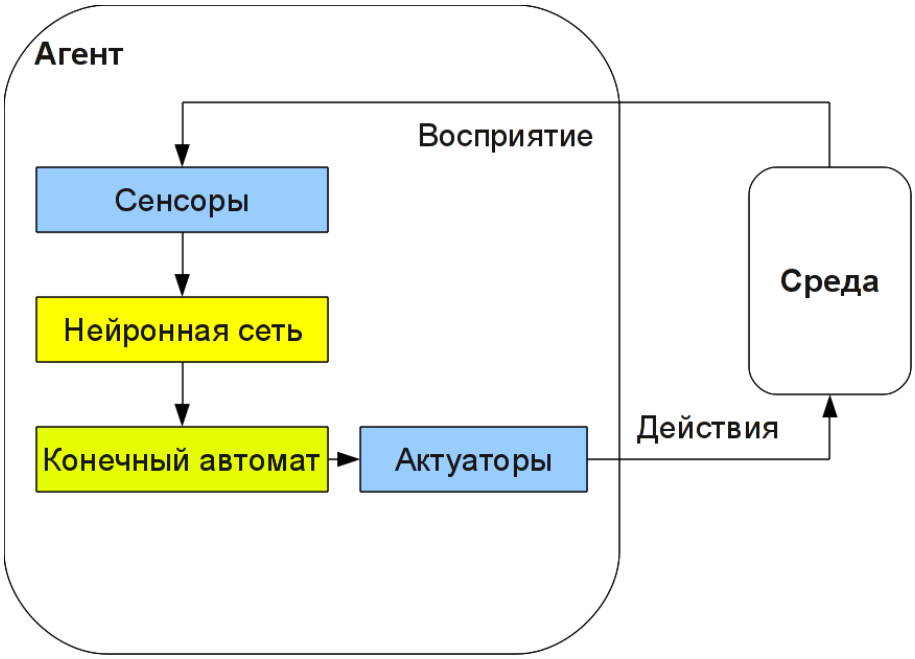

Рис. 3. Автоматный агент

Использование нейронной сети (или другого преобразователя вещественных данных о среде в дискретные) позволяет не уменьшая общности упростить рассмотрение окружающей среды.

В рассматриваемой модели каждый участник игры воспринимает результат поведения остальных участников только как реакцию на его поведение некоторой более или менее сложно организованной внешней среды. Никакой информацией не только о поведении, но даже о наличии других участников (агентов) не располагает. Данный пример показывает, что в ряде ситуаций в дополнительной информации нет никакой необходимости, так как и без неё автоматы способны добиваться целесообразного и даже оптимального поведения. Вместе с тем возникает ряд не очень приятных характеристик поведения – рост сложности процедуры принятия решений (глубина памяти

автоматов), весьма быстрый рост времени достижения оптимального поведения и т.п. И вообще термин «коллективное поведение» мало подходит к описываемой ситуации – можно сказать, что речь идёт о некоторой модели совокупного поведения. Когда речь идёт о «коллективе», обычно подразумевается некоторая структура отношений, наличие обмена информацией, организация взаимодействия между членами коллектива.

При попытках построить модели поведения со взаимодействием следует постоянно помнить, что только достаточно простые модели, зависящие от небольшого числа параметров, позволяют разобраться в эффектах, возникающих в этих моделях и моделируемых ими ситуациях [14].

Простейшим типом взаимодействия двух автоматов можно считать: однородное взаимодействие с ограниченным числом соседей [15]. При взаимодействии с ограниченным числом соседей для каждого члена коллектива указывается его окрестность – список автоматов, называемых соседями данного автомата, с которыми он может осуществлять взаимодействие. В дальнейшем будем рассматривать одностороннее взаимодействие – автомат воспринимает информацию от своих соседей. Однородность ограниченного взаимодействия заключается в том, что размеры окрестности для всех автоматов одинаковы. Таким образом, однородное взаимодействие задаётся ориентированным графом отношений.

Далее в данной работе в качестве агентов будем рассматривать конечные автоматы, которые взаимодействуют по состоянию, а синхронизация осуществляется тактовым генератором.

Выводы по главе 1

1. Рассмотрены различные эволюционные алгоритмы.
2. Производится обзор существующих методов генерации конечных автоматов.
3. Приведены основные характеристики многоагентных систем.
4. Рассмотрены агенты, чьим поведением управляет автомат. Но поведение данных агентов нельзя назвать коллективным.

ГЛАВА 2. ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ УПРАВЛЕНИЯ МНОГОАГЕНТНЫМИ СИСТЕМАМИ

В этой главе приведено описание экспериментальных задач, решение которых невозможно без кооперативного поведения агентов.

2.1. Описание экспериментальных задач

В данном разделе приводится описание двух экспериментальных задач.

2.1.1. Построение гомогенных агентов

Рассмотрим многоагентную среду, все агенты которой – гомогенные, т.е. обладают одинаковым восприятием среды и одинаковым поведением. Таким образом все агенты могут быть описаны одним и тем же конечным автоматом.

Модель рассматриваемых агентов приведена на рис. 4. В качестве входных воздействий для одного автомата будем рассматривать номера состояний четырёх его соседей, либо -1, если соседа нет. Таким образом диапазон значений для каждого из входных воздействий – от -1 до числа состояний в автоматах. Возможны пять действий:

- Шаг влево (L)
- Шаг вправо (R)
- Шаг вверх (U)
- Шаг вниз (D)
- Ничего не делать (N)

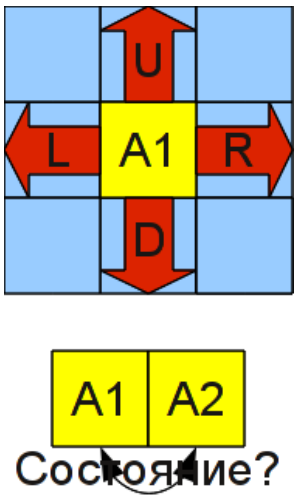

Рис. 4. Модель рассматриваемых агентов

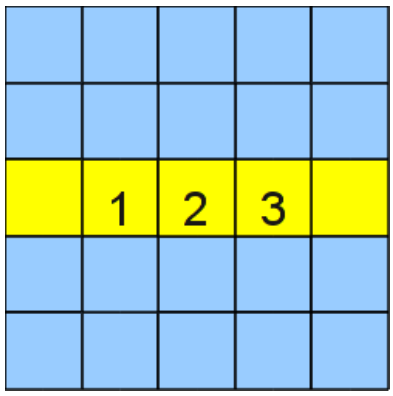

Рис. 5. Начальное положение агентов

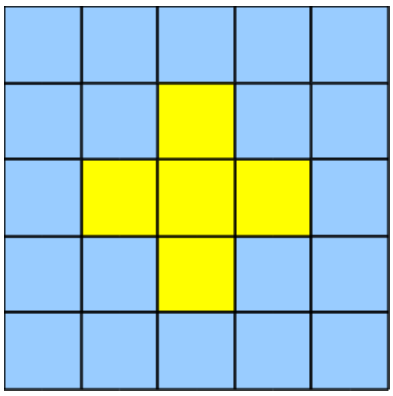

Рис. 6. Требуемое положение агентов

Начальное положение пяти агентов на поле размером пять на пять клеток изображено на рис. 5, требуемое положение агентов после некоторого времени их работы изображено на рис. 6. Причём все автоматы должны придти в одно конечное состояние. Следует отметить, что для некоторых из агентов (на рис. 5 с номерами 1, 2 и 3) их начальное расположение неразличимо, таким образом агенты не смогут достичь требуемого положения без обмена информацией об окружающей среде.

Также можно рассмотреть случай, в котором агентов значительно больше – рис. 7 и 8. В данном случае должен наблюдаться рост числа состояний автоматов, т.к. число неразличимых положений больше.

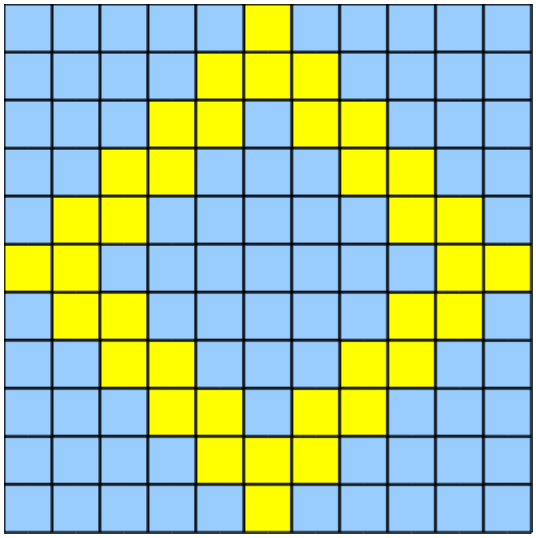

Рис. 7. Начальное положение агентов во втором случае

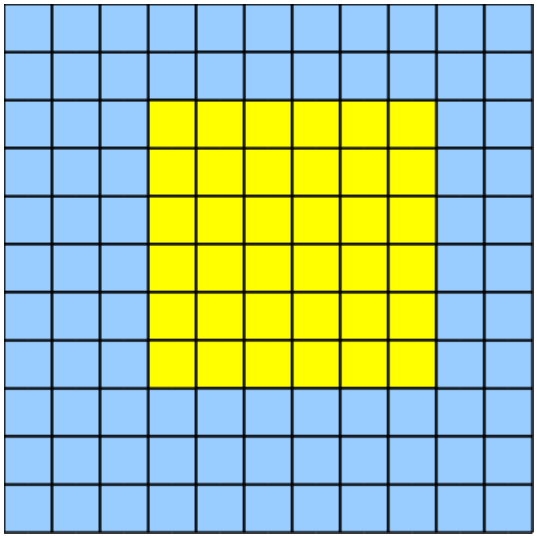

Рис. 8. Требуемое положение агентов во втором случае

2.1.2. Построение гетерогенных агентов

Более сложной является задача автоматического построения гетерогенных агентов. В данном случае различные агенты должны выполнять различные функции и соответственно обладать различным поведением.

Рассмотрим многоагентную среду, в которой агенты такие же как и в предыдущей задаче (рис. 4), но должны выполнять различные функции. Причём вся система должна описывать поведение змейки (рис. 9), т.е. агент отвечающий за голову змейки (H) должен стремиться к поеданию яблок (A), а агенты отвечающие за хвост (T) к поддержанию целостности цепочки.

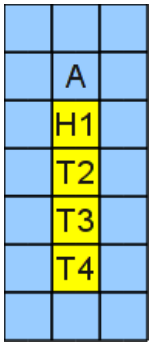

Рис. 9. Начальное положение змейки

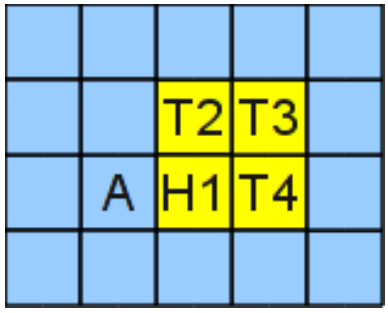

Рис. 10. Сложное положение змейки

Как и в предыдущей задаче – особое внимание следует уделить случаю, когда положение агентов неразличимо (на рис. 9 с номерами 2 и 3, на рис. 10 – 1, 2, 3 и 4), если нет обмена информацией между ними. Поэтому как и в предыдущей задаче в качестве входных воздействий для одного автомата будем рассматривать номера состояний четырёх его соседей; -1, если соседа нет и ноль в случае яблока.

2.2. Генетический алгоритм

В данном разделе приводится описание генетических алгоритмов и полученных с их помощью решений для описанных ранее экспериментальных задач.

2.2.1. Генетический алгоритм для построения гомогенных автоматных агентов

Для построения гомогенных автоматных агентов воспользуемся представлением автоматов в виде строк над числами, т.е. хромосома – это фактически таблицы переходов автомата. Такого представления вполне достаточно для решения поставленной задачи, т.к. число входных воздействий и их диапазон невелики.

Для решения поставленной задачи была выбрана следующая функция приспособленности:

$$F(A)=\frac{100}{1+S}+\frac{50}{1+C}+\frac{40}{1+M}+\frac{10}{1+N}$$, где

- S – сумма расстояний между конечным положением и требуемым,
- C – штраф за отсутствие соседей в процессе работы,
- N – штраф за неправильное конечное состояние,
- M – штраф за отсутствие действий.

Параметры генетического алгоритма:

- элитизм – 20 %;
- процент мутаций – 30%;
- процент кроссоверов – 70%;
- размер популяции – 200 особей, размер offspring – 400 особей;
- максимальное число состояний – 5.

На рис. 11 приведён график роста фитнесс-функции для одного из запусков генетического алгоритма.

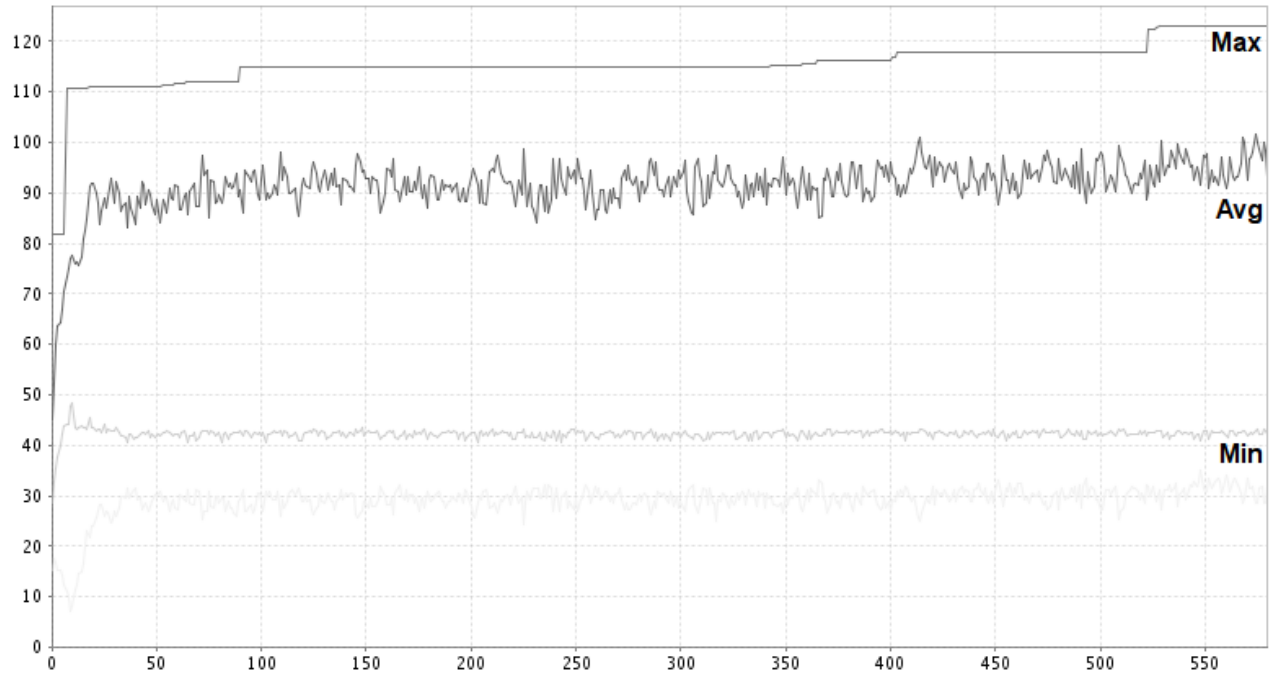

Рис. 11. График роста фитнесс-функции

В табл. 1 Приведена таблица переходов для автомата построенного при помощи генетического программирования.

Таблица 1. Таблица переходов автомата (неиспользуемые переходы опущены)

Исходное состояние	**L**	**R**	**U**	**D**	**Действие**	**Конечное состояние**
0	0				N	1
0		0			N	1
0			2		R	3
0				2	L	3
0		1			U	0
0	1				D	0
0	3	3	3	3	N	3
1		0			R	2
1	0				L	2
2		0			N	3

На рис. 12 показаны положения агентов в разные моменты времени. В начальный момент времени ($t=1$) происходит только обмен информацией об окружающей среде. В остальные моменты времени ($t=2...3$) обмен информацией сопровождается изменением окружающей среды, т.к. агенты меняют своё положение.

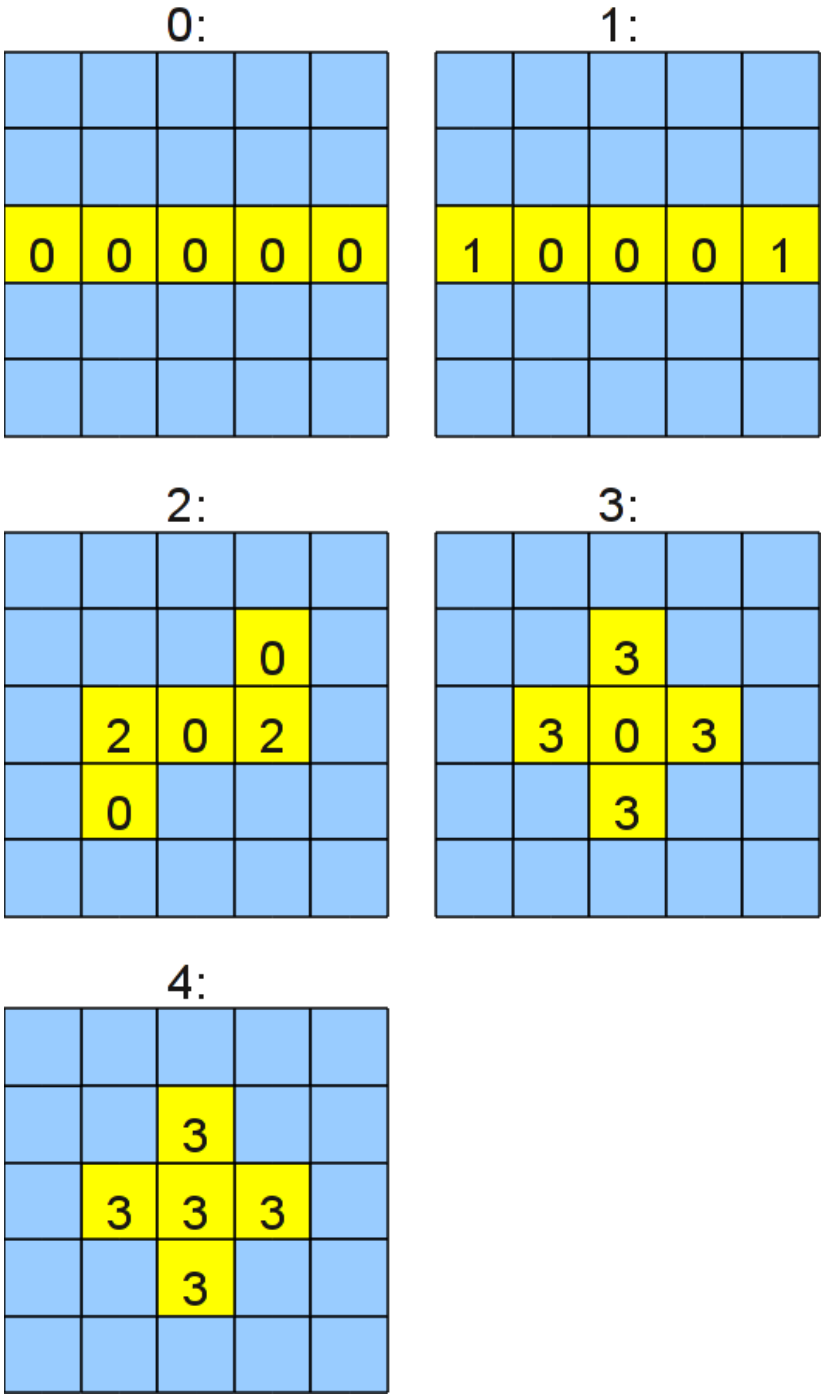

Рис. 12. Пример поведения

2.2.2. Генетический алгоритм для построения гетерогенных автоматных агентов

Так как поведение агентов должно быть различно, то и автоматы, описывающие это поведение, должны быть различны. Поэтому введём понятие составной хромосомы и определим для неё генетические операторы – скрещивание и мутацию. Данная хромосома будет содержать внутри себя хромосомы двух и более автоматов, которые в свою очередь могут использовать уже известные методы кодирования – строки над числами, сокращённые таблицы, деревья решений и т.д. Причём для вложенных хромосом операторы скрещивания и мутации уже определены.

Чтобы скрестить две составные хромосомы необходимо последовательно скрестить все их компоненты. В результате чего получим две новые составные хромосомы. Такой кроссовер изображён схематически на рис. 13.

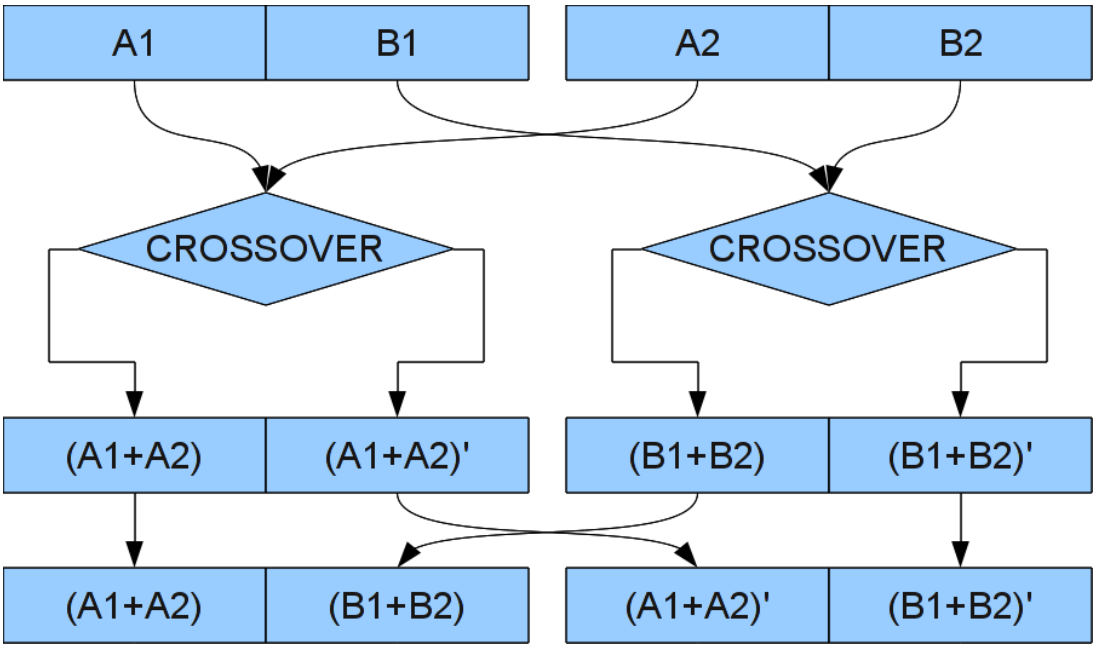

Рис. 13. Кроссовер для составной хромосомы

Мутация для составных хромосом осуществляется схожим образом – необходимо последовательно мутировать все компоненты, входящие в состав хромосомы. Такая мутация изображена схематически на рис. 14.

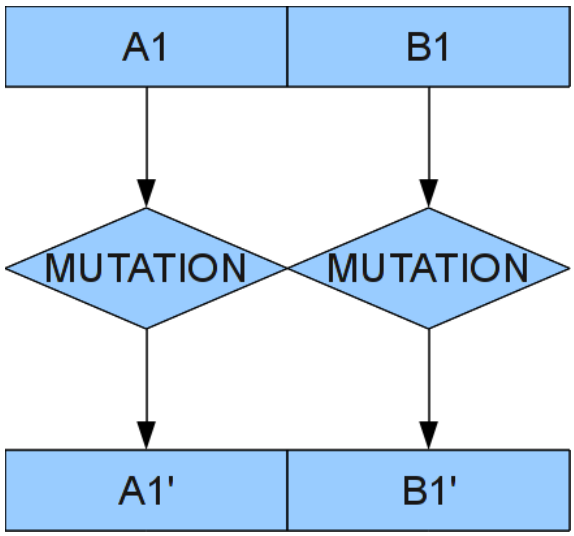

Рис. 14. Мутация для составной хромосомы

Данное представление позволяет с помощью генетического программирования получать несколько автоматных агентов с различными функциями (поведением).

Для решения поставленной задачи рассмотрим два типа агентов – агент отвечающий за голову змейки и агенты отвечающие за её хвост, таким образом составная хромосома будет состоять из двух компонентов.

Была выбрана следующая функция приспособленности:

$$F(A)=\frac{1000}{1+E}+\frac{100}{1+T}+\frac{10}{1+M}$$, где

- E – число не съеденных яблок,
- T – штраф за неправильное поведение хвоста,
- M – штраф за отсутствие действий.

Параметры генетического алгоритма:

- элитизм – 10 %;
- процент мутаций – 40%;
- процент кроссоверов – 60%;
- размер популяции – 200 особей, размер offspring – 400 особей;
- максимальное число состояний – 8.

На рис. 15 приведён график роста фитнесс-функции для одного из запусков генетического алгоритма.

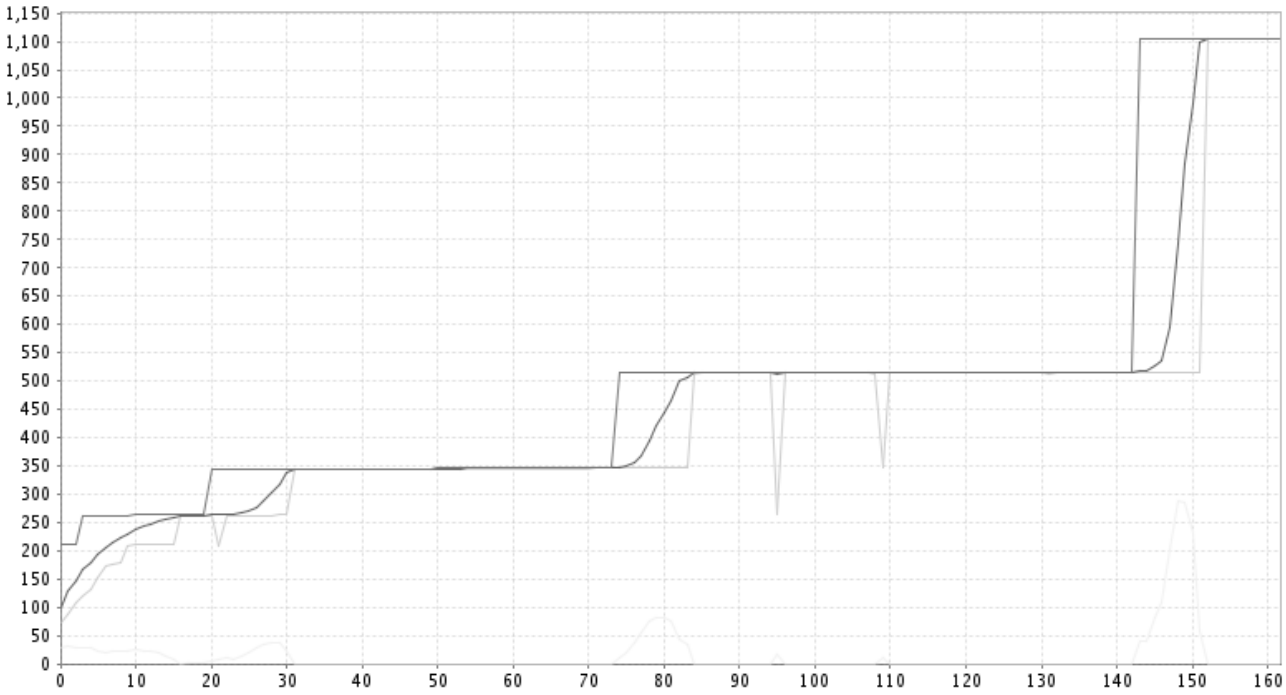

Рис. 15. График роста фитнесс-функции

На рис.16 и 17 схематически изображены автоматы, построенные при помощи генетического программирования для управления головой и хвостом змейки соответственно.

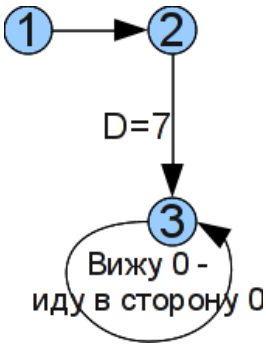

Рис. 16. Автомат управления головой змейки (неиспользуемые переходы и состояния опущены)

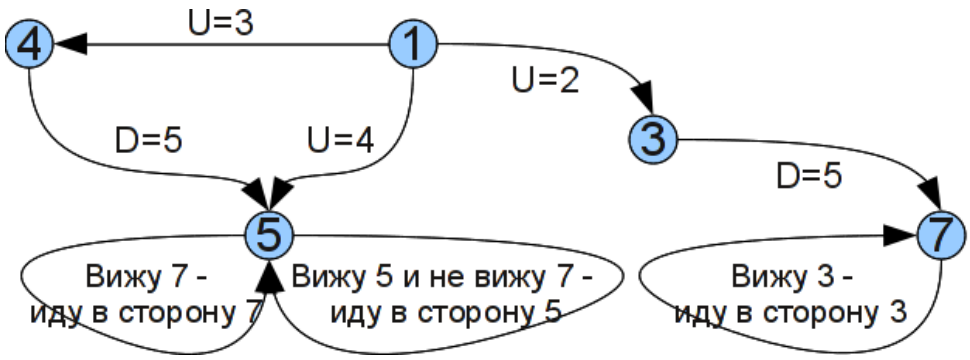

Рис. 17. Автомат управления хвостом змейки (неиспользуемые переходы и состояния опущены)

На рис. 18 показано поведение змейки в начальные моменты времени, когда происходит лишь обмен информацией. Фактически получается, что этот процесс можно охарактеризовать фразой - «на первый, второй рассчитайся». После чего змейка может передвигаться по игровому полю поедая яблоки и при этом не разрываться.

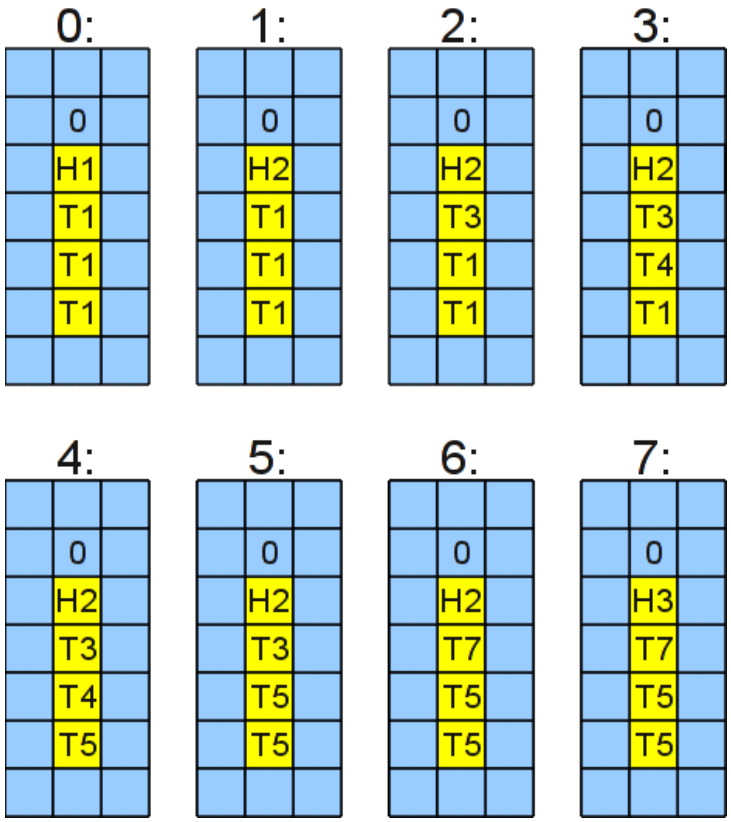

Рис. 18. Поведение полученной змейки

Выводы по главе 2

1. Приведено описание экспериментальных задач, решение которых невозможно без обмена информацией между агентами.
2. На примере экспериментальных задач продемонстрирована применимость автоматного и генетического программирования для построения систем управления многоагентными системами двух типов – с гомогенными и гетерогенными агентами.
3. Описан способ представления нескольких автоматов в виде составной хромосомы.

ГЛАВА 3. РЕАЛИЗАЦИЯ

В данной главе описывается программное средство GAAP (Genetic Algorithms for Automata-based Programming) для генерации автоматов управления с помощью генетического программирования, которое было разработано автором данной работы и использовано для её выполнения.

3.1. Программное средство GAAP

Программное средство GAAP предназначено для написания генетических алгоритмов и исследования их применительно к различным задачам (не обязательно к задачам генерации конечных автоматов). Данное программное средство было разработано автором совместно с В. А. Кулевым и применено в работах [16] [17] [18].

Программное средство разработано на языке *Java* с использованием автоматизированного средства сборки *Maven*[1] и техники *TDD*[2]. Программное средство обладает следующими достоинствами:

- программные интерфейсы (*API*[3]) просты для использования;
- программное средство может быть легко использовано в составе других приложений, написанных на языке *Java*;
- программное средство содержит реализацию различных эволюционных алгоритмов (в том числе реализацию островной модели) для решения исследуемых задач;
- содержит реализации различных представлений автоматов в виде хромосом;

1 **Apache Maven** – система автоматической сборки проектов (http://maven.apache.org/)

2 **TDD (Test-Driven Development)** – разработка через тестирование

3 **API (Application Programming Interface)** – набор готовых классов, функций, структур и констант, предоставляемых приложением для использования во внешних программных продуктах.

- содержит реализации генетических операторов;
- содержит различные методы отбора особей;
- содержит набор компонентов для реализации распределённых вычислений функции приспособленности;
- содержит компоненты позволяющие наглядно продемонстрировать работу генетического алгоритма и работу автоматизированного объекта.

При выполнении данной работы в программное средство была добавлена возможность моделирования многоагентных систем и сред.

Программное средство GAAP включает в себя семь модулей, описание которых приведено в табл. 2.

Таблица 2. Описание модулей инструментального средства GAAP

Модуль	Описание
ga-core	Набор базовых интерфейсов и реализации различных эволюционных алгоритмов.
ga-gui	Набор компонентов, позволяющих наглядно продемонстрировать работу генетического алгоритма в виде графиков и таблиц.
ga-func	Бинарное и вещественное кодирование для представления особей. Генетические операторы для задач оптимизации функций многих переменных.
ga-grid	Компоненты для реализации распределённых вычислений функции приспособленности. Распределённая модель генетического алгоритма.
fsm-core	Различные представления конечных автоматов в виде хромосом и генетические операторы. Реализация модели «конечный автомат + объект управления = автоматизированный объект», позволяющая конечному автомату воздействовать на объект управления.
fsm-gui	Набор компонентов, позволяющих наглядно продемонстрировать процесс запуска автоматизированного объекта и состояние объекта управления.
examples	Реализации объектов управления для тестовых задач.

3.2. Использование программных интерфейсов GAAP

Эволюционный алгоритм задаётся реализацией следующих базовых интерфейсов системы. Диаграмма классов и интерфейсов изображена на рис. 19.

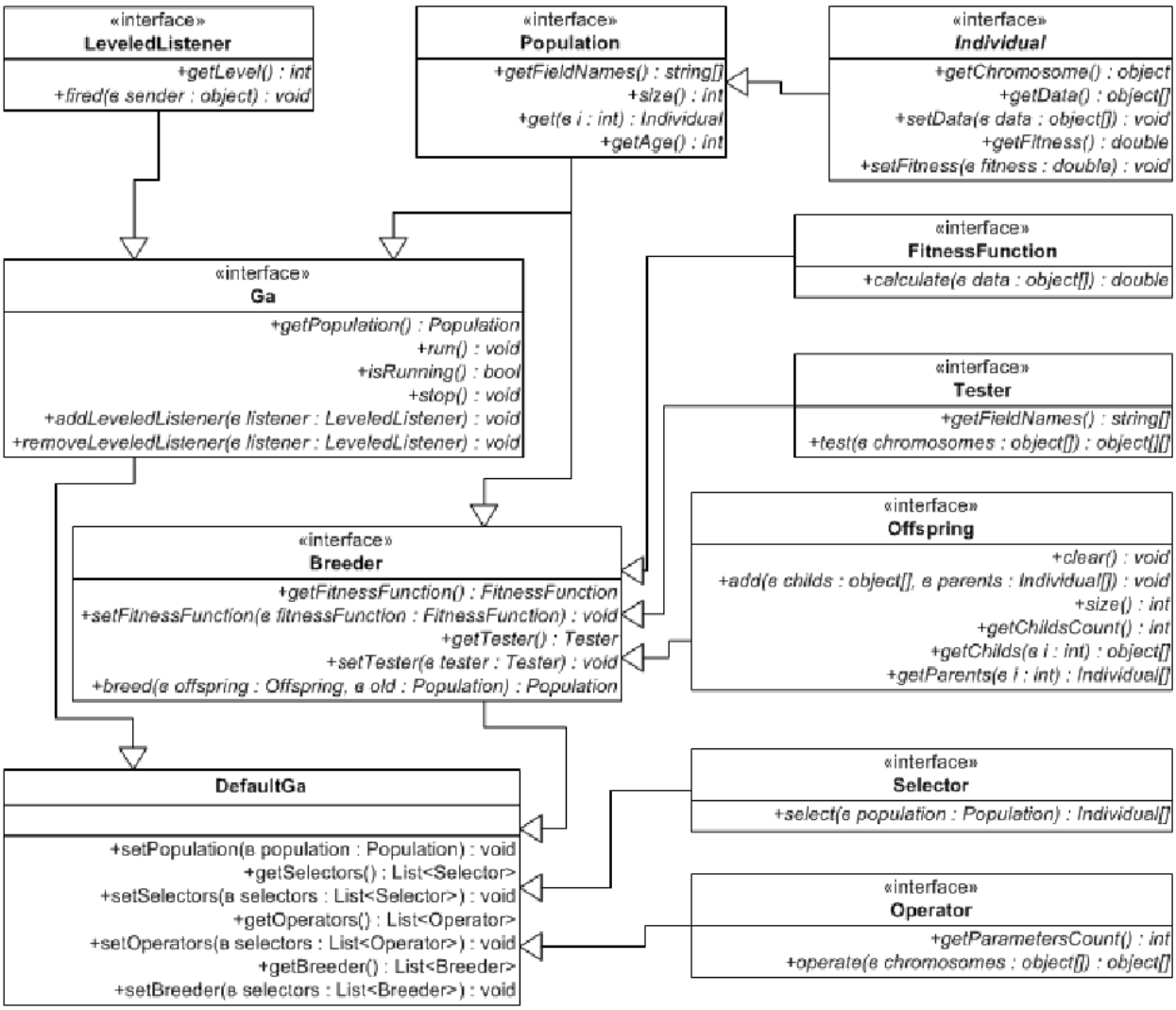

Рис. 19. Диаграмма классов и интерфейсов инструментального средства

Здесь:

- Ga – интерфейс для генетических алгоритмов;
- Operator – интерфейс для генетических операторов, таких как мутация, скрещивание и т.д;
- Selector – интерфейс для алгоритмов отбора особей из популяции;
- Population – интерфейс для популяций генетического алгоритма;
- Individual – интерфейс особей генетического алгоритма;
- Tester – интерфейс для вычислителей функции приспособленности;

- FitnessFunction – интерфейс для функций приспособленности;
- Breeder – интерфейс для алгоритмов построения популяций.

Инструментальное средство содержит стандартные реализации данных интерфейсов, имена классов со стандартными реализациями обычно имеют префикс Default. Также инструментальное средство включает в себя набор различных алгоритмов отбора особей из популяции, например отбор по рулетке, отбор усечением и т.д. Инструментальное средство содержит реализации следующих представлений автоматов в виде хромосом:

- полными таблицами переходов – org.gaap.fms.impl.simple;
- сокращёнными таблицами переходов – org.gaap.fsm.impl.compact;
- деревьями решений – org.gaap.fsm.impl.tree;
- сотавными хромосомами (разработано в ходе выполнения данной работы) – org.gaap.fsm.impl.composite

3.3. Установка программного средства

Последняя версия исходных кодов системы доступна по следующему адресу: http://svn.godin.net.ru/trunk/

Для сборки системы требуется *Subversion*[4] и *Maven* версии 2.0.9. Ниже приведён список команд для полной сборки проекта из исходных кодов:

1. svn checkout http://godin.net.ru/svn/trunk/ gaap
2. cd gaap
3. mvn clean install

4 **Apache Subversion** – централизованная система управления версиями (http://subversion.apache.org/)

3.4. Использование программного средства

Для реализации собственного генетического алгоритма необходимо выполнить следующие шаги:

1. Реализовать класс, описывающий хромосому – представление сущности в виде особи генетического алгоритма.
2. Реализовать классы, реализующие интерфейс Operator и описывающие генетические операции над классом-хромосомой. Для упрощения реализции можно воспользоваться абстрактными классами – CrossoverAdapter и MutationAdapter.
3. Реализовать интерфейс Tester для оценки фенотипа хромосомы.
4. Реализовать интерфейс FitnessFunction для вычисления функции-приспособленности по данным фенотипа.

После чего генетический алгоритм может быть запущен. Приведём пример кода запускающего генетический алгоритм:

Листинг 1. Пример кода для запуска генетический алгоритм

```
DefaultGa ga = new DefaultGa();

List<Selector> selectors = new ArrayList<Selector>();
selectors.add(new RouletteSelector(0.7));
selectors.add(new RouletteSelector(0.3));
ga.setSelectors(selectors);

List<Operator> operators = new ArrayList<Operator>();
operators.add(new ExampleCrossover()));
operators.add(new ExampleMutation()));
ga.setOperators(operators);

DefaultBreeder breeder = new DefaultBreeder(new ExampleTester(), 0.9);
breeder.setFitnessFunction(new ExampleFitnessFunction());
ga.setPopulation(breeder.breed(new Offspring(start)));
ga.setBreeder(breeder);

new DefaultBatch().setGa(ga).run();
```

Так как GAAP включает в себя некоторые компоненты из *Spring Framework*[5], то можно достичь аналогичного результата используя XML конфигурацию:

5 **Spring Framerowk** – http://www.springframework.org/

Листинг 2. Пример XML конфигурации для запуска генетического алгоритма

```
<beans>
 <bean id="tester" class="ExampleTester"/>
 <bean id="breeder" class="org.gaap.ga.impl.DefaultBreeder">
  <constructor-arg index="0" ref="tester"/>
  <constructor-arg index="1" value="0.9"/>
  <property name="fitnessFunction"><bean class="ExampleFitnessFunction"/></property>
 </bean>
 <bean id="ga" class="org.gaap.ga.impl.DefaultGa">
  <property name="selectors">
   <list>
    <bean class="org.gaap.ga.impl.selectors.RouletteSelector">
     <constructor-arg value="0.7"/>
    </bean>
    <bean class="org.gaap.ga.impl.selectors.RouletteSelector">
     <constructor-arg value="0.3"/>
    </bean>
   </list>
  </property>
  <property name="operators">
   <list><bean class="ExampleCrossover"/><bean class="ExampleMutation"/></list>
  </property>
  <property name="breeder" ref="breeder"/>
  <property name="population">
   <bean factory-bean="breeder" factory-method="breed">
    <constructor-arg type="org.gaap.ga.Offspring">
     <bean class="org.gaap.ga.Offspring"><constructor-arg value=”start”/></bean>
    </constructor-arg>
    <constructor-arg type="org.gaap.ga.Population"><null/></constructor-arg>
   </bean>
  </property>
 </bean>
</beans>
```

Выводы по главе 3

1. Описано программное средство GAAP.
2. Рассмотрено использование программных интерфейсов GAAP.
3. При выполнении данной работы в программное средство была добавлена возможность моделирования многоагентных систем, основанных на автоматных-агентах.

ЗАКЛЮЧЕНИЕ

Основные результаты работы состоят в следующем:

1. Предложен подход к программированию многоагентных систем двух типов (гетерогенных и гомогенных) на основе автоматного и генетического программирования.
2. В разработанное автором программное средство GAAP добавлена возможность моделирования автоматных-агентов для многоагентных систем двух типов – гомогенных и гетерогенных.
3. Предложенный подход и его реализация апробированы на двух экспериментальных задачах.
4. На основе анализа полученных экспериментальных данных сделан вывод о применимости предложенного подхода.

ПУБЛИКАЦИИ

1. НИОКР в рамках программы «УМНИК 2009»: Разработка инструментального средства для генерации конечных автоматов с использованием генетических алгоритмов.
2. Государственный контракт в рамках Федеральной целевой программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2012 годы»: Технология генетического программирования для генерации автоматов управления системами со сложным поведением.
3. *Е. А. Мандриков, В.А. Кулев, А.А. Шалыто* Применение генетических алгоритмов для создания управляющих автоматов в задаче о “флибах” // Информационные технологии. 2008, №1, стр. 42-45,89.
4. *Е.А. Мандриков, Ю.К. Чеботарева, А.А. Шалыто* Автоматное программирование и параллельные вычисления. / “Известия ВУЗов. Приборостроение”. 2009, стр. 66-73.
5. *Е.А. Мандриков, В.А. Кулев, А.А. Шалыто* Построение автоматов с помощью генетических алгоритмов для решения задачи о “флибах” / Сборник докладов X Международной конференции по мягким вычислениям и измерениям (SCM’2007). СПб.: СПбГЭТУ “ЛЭТИ”. 2007, Том 1, стр. 293-296.
6. *Е.А. Мандриков, В.А. Кулев, Ю.Д. Бедный, В.Р. Данилов* Разработка методов построения автоматов с помощью генетических алгоритмов / Сборник докладов IV Всероссийской межвузовской конференции молодых ученых (КМУ’2007). СПб.: СПбГУ ИТМО. 2007.
7. *Е.А. Мандриков, В.А. Кулев* Разработка инструментального средства для генерации конечных автоматов с использованием генетических

алгоритмов // Научно-технический вестник СПбГУ ИТМО. Выпуск 53. Автоматное программирование, с. 100-103

8. *Е.А. Мандриков, В.А. Кулев* Программный комплекс для разработки и анализа различных генетических алгоритмов / Сборник докладов V Всероссийской межвузовской конференции молодых ученых (КМУ'2008). СПб.: СПбГУ ИТМО. 2008.

9. *Е.А. Мандриков, В.А. Кулев* Разработка инструментального средства для генерации конечных автоматов с использованием генетических алгоритмов / Сборник докладов Всероссийской научной конференции студентов и аспирантов "Молодые исследователи регионам". ГОУ ВПО "ВГТУ". 2008.

10. *Е.А. Мандриков, В.А. Кулев* Применение распределённых вычислений для автоматической генерации конечных автоматов с использованием генетических алгоритмов / Сборник докладов XI Международной конференции по мягким вычислениям и измерениям (SCM'2008). СПб.: СПбГЭТУ "ЛЭТИ". 2008, с. 255-260.

11. *E.A. Mandrikov, V.A. Kulev* Development of Software System for State Machine Generation Using Genetic Algorithm / Proceedings of the Second Spring Young Researchers Colloquium on Software Engineering. SPb.: SPbSU. 2008. V. 1, pp. 59-60. (PDF)

12. *Е.А. Мандриков* Композиция методов генерации конечных автоматов на основе генетических алгоритмов. СПб.: СПбГУ ИТМО. 2008.

13. *E.A. Mandrikov, V.A. Kulev* Applying Automata-Based Programming to Creating Business Processes Management Systems / Proceedings of the Third Spring Young Researchers Colloquium on Software Engineering. SPb.: SPbSU. 2009. pp. 114-115. (PDF)

14. *Е.А. Мандриков, Ю.К. Чеботарева* Генерация числовых

последовательностей, описывающих движение человекоподобного робота, при помощи генетических алгоритмов / Научные доклады научно-практической конференции студентов, аспирантов, молодых ученых и специалистов “Интегрированные модели, мягкие вычисления, вероятностные системы и комплексы программ в искусственном интеллекте”. М.: Физматлит. 2009. Т.2, с. 181-188.

СПИСОК ЛИТЕРАТУРЫ

1. *A.V.Aho, J.D.Ullman* Theory of Parsing, Translation and Compiling, 1973.
2. *Г.А.Корнеев* Автоматизация построения визуализаторов алгоритмов дискретной математики на основе автоматного подхода, 2006.
3. *А.А.Шалыто* Алгоритмизация и программирование для систем логического управления и "реактивных систем", 2001. http://is.ifmo.ru/download/arew.pdf
4. *Н.И.Поликарпова, А.А.Шалыто* Автоматное программирование. Учебно-методическое пособие, 2007. http://is.ifmo.ru/books/_umk.pdf
5. *А.А.Шалыто* SWITCH-технология. Алгоритмизация и программирование задач логического управления, 1998. http://is.ifmo.ru/books/switch/1
6. Технология генетического программирования для генерации автоматов управления системами со сложным поведениемы, 2007. http://is.ifmo.ru/genalg/_2007_02_report-genetic.pdf
7. *Ф.Н.Царев* Совместное применение генетического программирования, конечных автоматов и искусственных нейронных сетей для построения системы управления беспилотным летательным аппаратом, 2008.
8. *Yim, Shen, Salemi, Rus, Moll, Lipson, Klavins, Chirikjian* Modular Self-Reconfigurable Robot Systems: Challenges and Opportunities for the Future, 2007.
9. *D.H.Wolpert, W.G.Macready* No Free Lunch Theorems for Optimization, 1997.
10. *Holland J. H.* Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, 1975.
11. *MacKay D. J. C.* Information Theory, Inference, and Learning Algorithms, 2003. http://labs.google.com/papers/mapreduce-osdi04.pdf
12. *Michael Wooldridge* An Introduction to MultiAgent Systems, 2002.
13. *Liviu Panait, Sean Luke* Cooperative Multi-Agent Learning: The State of the Art, 2005.
14. *В.И.Варшавский, Д.А.Поспелов* Оркестр играет без дирижёра, 1984.

15. *В.И.Варшавский* Коллективное поведение автоматов, 1973.

16. *В.В.Каширин* Метод модификации оценочных функций для оптимизации работы генетических алгоритмов, генерирующих конечные автоматы, 2008. http://is.ifmo.ru/papers/kashirin/

17. *Ю.К.Чеботарева* Применение генетических алгоритмов для генерации числовых последовательностей, описывающих движение, на примере шага вперёд человекоподобного робота, 2009.

18. *В.А.Кулев* Применение генетических алгоритмов для построения конечных автоматов при решении задач ориентирования на местности, 2008.

СПИСОК РИСУНКОВ И ТАБЛИЦ