МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

Национальный технический университет
«Харьковский политехнический институт»

В.Д. Дмитриенко, И.П. Хавина

Системы искусственного интеллекта

Лабораторный практикум
для студентов дневной и заочной форм обучения по направлениям
«Компьютерная инженерия» и «Кибербезопасность»,
в том числе для иностранных студентов

Утверждено редакционно-издательским советом НТУ «ХПИ»,
протокол № 3 от 22.12.16 г.

Харьков
НТУ «ХПИ»
2018

УДК 004.891 (072)

Д 53

Рецензенты:
О.И. Филипенко, д-р техн. наук, проф., Харьковский национальный университет радиоэлектроники;
А.А. Серков, д-р техн. наук, проф., Национального технического университета «ХПИ»

Лабораторний практикум включає основні відомості щодо методів та алгоритмів для створення інтелектуальних систем. Містить завдання для лабораторних робіт та приклади їх розв'язання.

Призначено для фахівців в галузі інформатики та штучного інтелекту, студентів і аспірантів вищих навчальних закладів.

Дмитриенко В.Д.

Д 53 Системы искусственного интеллекта: лабораторный практикум. / В.Д. Дмитриенко, И.П. Хавина. – Харьков : «Мадрид», 2018. – 136 с. На русск. яз.

ISBN

Лабораторный практикум включает основные сведения по методам и алгоритмам для создания интеллектуальных систем. Содержит задания для лабораторных работ и примеры их решения.

Предназначено для специалистов в области информатики и искусственного интеллекта, студентов и аспирантов вузов.

Ил. 25. Табл. 15. Библиогр.: 12 назв.

УДК 004.89 (072)

ISBN

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

Национальный технический университет
«Харьковский политехнический институт»

В.Д. Дмитриенко, И.П. Хавина

Системы искусственного интеллекта

Лабораторный практикум
для студентов дневной и заочной форм обучения по направлениям
«Компьютерная инженерия» и «Кибербезопасность»,
в том числе для иностранных студентов

Утверждено редакционно-издательским советом НТУ «ХПИ»,
протокол № 3 от 22.12.16 г.

Харьков
НТУ «ХПИ»
2018

Министерство образования и науки Украины
Национальный технический университет
«Харьковский политехнический институт»

До друку і в світ дозволяю

Проректор НТУ ХПІ проф. Мигущенко Р.П.

В.Д. Дмитриенко, И.П. Хавина

Системы искусственного интеллекта

Лабораторный практикум
для студентов дневной и заочной форм обучения по направлениям
«Компьютерная инженерия» и «Кибербезопасность»,
в том числе для иностранных студентов

Утверждено редакционно-издательским советом НТУ «ХПИ»,
протокол № 3 от 22.12.16 г.

Харьков
НТУ «ХПИ» 2018

Навчальне видання

ДМИТРІЄНКО Валерій Дмитрович
ХАВІНА Інна Петрівна

СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

Лабораторний практикум
для студентів денної та заочної форм навчання за напрямами
«Комп'ютерна інженерія» та «Кібербезпека»,
в тому числі для іноземних студентів

Відповідальний за випуск проф. Семенов С.Г.
Роботу до видання рекомендував проф. Заполовський М. І.

Редактор М. П. Єфремова

План 2017 р., поз. 81
Підп. до друку 18.12.2018. Формат 60x84 1/16. Папір друк. №2. Друк – різографія.
Гарнітура Times New Roman. Ум. друк.арк. .
Наклад 300 прим. Зам. № _____. Ціна договірна

Друкарня НТУ «ХП»
61002, Харків, вул. Кирпичова, 2.

Содержание

Все новые идеи проходят три стадии. На первой идея кажется нелепой, на второй встречает яростное сопротивление, а на третьей воспринимается как нечто очевидное.

Артур Шопенгауэр

Вступление

Учебное пособие посвящено описанию методов и алгоритмов искусственного интеллекта (ИИ), которые показали хорошие результаты при применении в различных областях науки и техники.

Лабораторная работа 1 посвящена распознаванию образов, где показаны методы решения широкого круга задач из этой области, определена структура систем распознавания, введены различные методы определения сходства образов и решающие правила.

Под обучением в системах ИИ понимают способность системы приобретать ранее неизвестные навыки и умения. В лабораторной работе 2 даны некоторые способы построения обучающихся систем.

В лабораторной работе 3 показано построение интеллектуальных систем, которые основаны на знаниях. Даны теоретические основы языка логического программирования Пролог, который является языком исчисления предикатов, с помощью которых описываются факты и правила предметной области. Пролог позволяет использовать переменные и строить программы для автоматического решения логических задач.

Эволюционное моделирование – направление в искусственном интеллекте, в основе которого лежат принципы, заимствованные из эволюционной биологии и популяционной генетики и объединяющие компьютерные методы, – генетические алгоритмы, эволюционное программирование и эволюционные стратегии – для решения задач прогнозирования, оптимизации, апроксимации и др. в различных областях науки и техники.

В качестве объектов эволюции в лабораторной работе 4 выбраны конечные автоматы, на входы которых поступают сигналы из внешней среды, представляющей собой источник последовательных сигналов из конечного алфавита. Необходимо получить автомат, правильно предсказывающий входную последовательность. Знакомство с этой темой позволит решать ряд практических задач по нахождению оптимальных решений или оптимальных состояний в любой области, невзирая на большие временные затраты.

Принятие решений в таких сферах как анализ процессов в макроэкономике, прогнозирование поведения финансовых рынков очень сложно, так как такие объекты являются сложными плохо-обусловленными системами, которые характеризуются: недостаточной априорной информацией; зашумленными или короткими выборками данных; плохо-обусловленными с размытыми характеристиками.

Эти проблемы могут быть решены с помощью метода группового учета аргументов, который находит для каждой выборки модель с помощью перебора возможных моделей-кандидатов и оценивает ее по внешнему точностному критерию на независимой выборке данных. Этому направлению посвящена лабораторная работа 5.

Лабораторная работа 6 посвящена генетическим алгоритмам. Основное назначение генетических алгоритмов заключается в решении поисковых задач, которые характеризуются большой размерностью пространства поиска решения.

Лабораторная работа 7 на примере работы нейронной сети Хебба знакомит с простыми нейронными сетями и методами их обучения.

В лабораторной работе 8 показан агентно-ориентированный подход, основанный на теории коллективного поведения агентов, который применяют для моделирования распределенных динамических систем в режиме реального времени.

Лабораторная работа № 1
МЕТОДЫ И АЛГОРИТМЫ РАСПОЗНАВАНИЯ ОБРАЗОВ

Цель работы: приобретение, закрепление знаний и получение практических навыков работы с простейшими алгоритмами распознавания на основе представления изображений в виде точек в *n*-мерном векторном пространстве.

Распознавание образов является важным разделом искусственного интеллекта. В настоящее время это – сложившиеся научное и практическое направления, связанные с решением широкого круга задач, относящихся к проблемам распознавания [1].

1.1. Образы и распознавание образов

Образы можно разделить на две категории: абстрактные и конкретные. Примером абстрактных образов могут быть идеи и аргументы. Распознавание таких образов относится к концептуальному распознаванию, которое в данном курсе не рассматривается.

Примером конкретного распознавания является распознавание букв, символов, рисунков, биологических изображений, трехмерных физических объектов, речевых сигналов, электрокардиограмм, сейсмических волн. Некоторые из этих образов – пространственные, другие – временные [1].

Последние два десятилетия основной интерес был направлен на два основных типа проблем распознавания:

1) Механизм распознавания, осуществляемый живыми организмами. Психологи, физиологи, биологи и нейрофизиологи приложили много усилий для изучения того, как живые организмы воспринимают объекты. Многое из результатов этих исследований отражено в литературе по бионике и другим близким областям.

2) Развитие теории и практики компьютерного решения конкретных задач.

Это направление, в котором активно работают как инженеры, так

и прикладные математики. Здесь нет единой теории, которая позволила бы решить все виды задач распознавания образов. Большинство методов имеют проблемную ориентацию и примеры решения конкретных задач, связанных с распознаванием изображений и речевых сигналов [1].

В распознавании образов разделяют общую задачу на три фазы: получение данных; предварительная обработка данных; принятие решения о классификации [1].

В фазе получения данных аналоговые данные из физического мира собираются с помощью соответствующих преобразователей и превращаются в цифровой формат для компьютерной обработки. На этом этапе физические переменные превращаются в ряды измеренных данных, которые затем используются как входные на второй фазе (предобработка данных) и образуют ряд классификационных признаков на выходе. Третья фаза – это классификатор, который представляет собой решающую функцию [1].

Существует большое число различных форм представления изображений в распознающих устройствах или программах. Одной из наиболее простых и понятных является форма, использующая представление изображений в виде точек в некотором n-мерном пространстве. Каждая ось такого пространства естественным образом соотносится с одним из n входов или с одним из n рецепторов распознающей системы. Каждый из рецепторов может находиться в одном из m состояний, если они дискретны, или иметь бесконечно большое число состояний, если рецепторы непрерывны. В зависимости от вида используемых рецепторов может порождаться непрерывное, дискретное или непрерывно-дискретное n-мерное пространство.

Как правило, в пространстве изображений вводится метрика – функция, которая каждой упорядоченной паре точек x и y пространства ставит в соответствие действительное число $d(x, y)$. При этом функция $d(x, y)$ обладает следующими свойствами:

1) $d(x, y) \geq 0$, $d(x, y) = 0$ тогда и только тогда, когда $x = y$;
2) $d(x, y) = d(y, x)$;
3) $d(x, y) \leq d(x, z) + d(z, y)$.

Введение метрики $d(x, y)$ в пространстве изображений позволяет говорить о близости или удаленности точек в этом пространстве или о мере сходства или различия анализируемых изображений. Понятие меры сходства изображений широко используется в теории распознавания образов. Однако формализация этого понятия при решении конкретных задач распознавания, как правило, не является тривиальной задачей. Более того, эта задача является одной из основных задач теории распознавания образов. Рассмотрим общие требования к мере сходства изображений.

Пусть задано некоторое конечное множество $S = \{S_1, S_2, \ldots, S_n\}$ входных изображений, каждое из которых является точкой в n-мерном пространстве изображений. Меру сходства изображений можно ввести как функцию двух аргументов $L(S_k, S_i)$, где $S_k, S_i \in S$. Общие требования к этой функции можно свести к следующему:

1) функция $L(S_k, S_i)$ должна обладать свойством симметрии, т.е.

$$L(S_k, S_i) = L(S_i, S_k);$$

2) область значений функции $L(S_k, S_i)$ – множество неотрицательных чисел, т.е.

$$L(S_k, S_i) \geq 0, \quad k, i = 1, 2, \ldots, n;$$

3) мера сходства изображения с самим собой должна принимать экстремальное значение по сравнению с любым другим изображением, т.е. в зависимости от способа введения меры сходства должно выполняться одно из двух соотношений:

$$L(S_k, S_k) = \max_i L(S_k, S_i),$$

$$L(S_k, S_k) = \min_i L(S_k, S_i);$$

4) в случае непрерывного n-мерного пространства и компактных образов функция $L(S_k, S_i)$ должна быть монотонной функцией удаления точек S_k и S_i друг от друга в этом пространстве.

Анализ свойств метрики и меры сходства изображений показыва-

ет, что требования к функции $L(S_k, S_i)$ нетрудно выполнить в метрических пространствах. В частности, если в метрическом пространстве введено расстояние, то оно может быть использовано в виде меры сходства изображений.

1.2. Распознавание по расстояниям в *n*-мерном пространстве

Эталонные изображения $X_1, X_2, \ldots, X_m$ некоторого числа m различных классов изображений или образов в n-мерном пространстве задаются в виде точек $(x_{11}, x_{12}, \ldots, x_{1n})$, $(x_{21}, x_{22}, \ldots, x_{2n})$, …, $(x_{m1}, x_{m2}, \ldots, x_{mn})$. Любое входное изображение S_i также представляется в виде точки $(x_{Si1}, x_{Si2}, \ldots, x_{Sin})$ в этом пространстве. Принадлежность входного изображения S_k к одному из m классов определяется с помощью расстояний между точкой S_i и всеми точками $X_1, X_2, \ldots, X_m$, соответствующими эталонным образам. Расстояние и является мерой сходства входного изображения с эталонами классов или образов. Входное изображение относится к тому образу, расстояние до эталонного изображения которого минимально, т.е. решающим правилом является следующее соотношение

$$S_i \in X_j, \text{ если } L(S_i, X_j) = \min_j (L(S_i, X_j)). \tag{1.1}$$

В теории распознавания образов часто используются расстояния по Евклиду (1.2) и по Минковскому (1.3):

$$L(S_i, X_j) = \sqrt{\sum_{k=1}^{n} (s_{ik} - x_{jk})^2}, \tag{1.2}$$

$$L(S_i, X_j) = \sqrt[\lambda]{\sum_{k=1}^{n} (s_{ik} - x_{jk})^{\lambda}}, \tag{1.3}$$

где λ – целое положительное число, большее двух.

Операции возведения в степень и извлечения корня не всегда удобно использовать при определении расстояний, поскольку они являются нелинейными операциями. Поэтому для определения расстояний в пространстве изображений часто используется и сумма модулей разностей между соответствующими компонентами *n*-мерных векторов:

$$L(S_i, X_j) = \sum_{k=1}^{n} \left| s_{ik} - x_{jk} \right|. \qquad (1.4)$$

В выражения (1.2) – (1.4) разности всех компонент векторов входят с одинаковыми единичными весами. В тех случаях, когда компоненты векторов, соответствующих распознаваемым изображениям, отличаются на порядки, например, одни компоненты векторов измеряются метрами, а другие – сантиметрами или миллиметрами, то при использовании расстояний (1.2) – (1.4) компоненты, имеющие небольшие численные значения, могут практически не влиять на величину расстояний.

В то же время с точки зрения решения реальных задач распознавания именно эти компоненты могут играть определяющую роль. Поэтому для более адекватного учета подобных компонент в выражения (1.2) – (1.4) могут вводиться весовые коэффициенты, учитывающие практическую ценность различных компонент вектора. В этом случае выражения (1.2) – (1.4) преобразуются к виду:

$$L(S_i, X_j) = \sqrt{\sum_{k=1}^{n} \eta_k (s_{ik} - x_{jk})^2}, \qquad (1.5)$$

$$L(S_i, X_j) = \sqrt[\lambda]{\sum_{k=1}^{n} \eta_k (s_{ik} - x_{jk})^{\lambda}}, \qquad (1.6)$$

$$L(S_i, X_j) = \sum_{k=1}^{n} \eta_k \left| s_{ik} - x_{jk} \right|. \qquad (1.7)$$

Предварительное задание весовых коэффициентов η_k в формулах, определяющих расстояния, требует наличия определенной априорной информации и не всегда может быть сделано оптимальным образом. Поэтому особый интерес представляют расстояния, в которых заложена идея выравнивания весов слагаемых от различных компонент, если они существенно отличаются по своим абсолютным значениям. Примером такого расстояния является расстояние по Камберру

$$L(S_i, X_j) = \sum_{k=1}^{n} \frac{|s_{ik} - x_{jk}|}{|s_{ik} + x_{jk}|}. \qquad (1.8)$$

При решении задач распознавания технических сигналов часто возникают ситуации, когда сигналы, отличающиеся только амплитудой или смещением по оси ординат, или небольшими нелинейными искажениями, необходимо относить к одному классу (рис. 1.1, табл. 1.1).

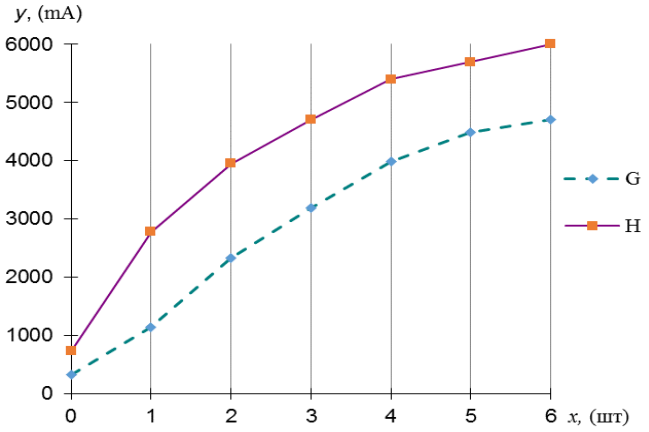

Рис. 1.1 Экспериментальные данные

Таблица 1.1 – Координаты кривых G и H

x	0	1	2	3	4	5	6
$y_i(H)$	731	2781	3955	4699	5399	5699	5990
$y_i(G)$	334	1140	2335	3186	3986	4486	4700

В этом случае может использоваться расстояние по Кендалу:

$$L(G^i, H^j) = 1 - \frac{2}{n(n-1)} \sum_{q=1}^{n-1} \sum_{\substack{k=2, \\ k>q}}^{n} \Delta_{qk}^i \Delta_{qk}^j, \qquad (1.9)$$

где

$$\Delta_{qk}^i = \begin{cases} 1, \text{ если } y_{iq} > y_{ik}, \\ -1, \text{ если } y_{iq} < y_{ik}, \quad q < k, \\ 0, \text{ если } y_{iq} = y_{ik}. \end{cases}$$

$$\Delta_{qk}^j = \begin{cases} 1, \text{ если } y_{jq} > y_{jk}, \\ -1, \text{ если } y_{jq} < y_{jk}, \quad q < k, \\ 0, \text{ если } y_{jq} = y_{jk}. \end{cases}$$

Если компоненты обоих векторов G^i и H^j упорядочены однотипно, то $\Delta_{qk}^i = \Delta_{qk}^j$, $q = \overline{1,(n-1)}$, $k = \overline{2,n}$, $q < k$, и результат суммирования в выражении (1.9) равен половине числа размещений из n по два: $A_n^2 / 2 = n(n-1)/2$. Отсюда следует, что выражение (1.9) при однотипном упорядочении векторов G^i и H^j принимает минимальное значение, равное нулю:

$$L(G^i, H^j) = 1 - \frac{2}{n(n-1)} \frac{n(n-1)}{2} = 1 - 1 = 0.$$

Если $\Delta_{qk}^i \neq \Delta_{qk}^j$ ни при одном значении индексов q и k, то любое произведение $\Delta_{qk}^i \Delta_{qk}^j = -1$, $q = \overline{1,(n-1)}$, $k = \overline{2,n}$, $q < k$, и, следовательно, в этом случае $L(G^i, H^j) = 1 - (-1) = 2$. Таким образом, рассто-

яние по Кендалу может принимать значения из интервала [0, 2]. Расстояние по Кендалу – это и расстояние для оценки близости в некотором смысле двух функций, заданных в n точках.

Пример вычисления расстояния по Кендалу. Пусть даны амплитуды сигналов G и H (рис 1.1) в виде массивов (табл. 1.1).

Шаг 0. Инициализируем $L_{кен} = 0$, $n = 7$.

Шаг 1. Организовать внешний цикл по q от 1 до $n - 1 = 6$, $q = 1$.

Шаг 2. Организовать внутренний цикл по k от 2 до $k = 7$.

$k = 2$.

Определяем значение Δ_{qk}^{G}

$$\Delta_{qk}^{G} = \begin{cases} 1, \text{ если } y_q > y_k, \\ -1, \text{ если } y_q < y_k, \\ 0, \text{ если } y_q = y_k. \end{cases}$$

Для кривой G $y_1(0) = 334$, $y_2(1) = 1140$, $y_1 < y_2$, значит, при $q = 1$ и $k = 2$, $\Delta_{12}^{G} = -1$.

Аналогично для кривой H определяем значение Δ_{12}^{H}:

$y_1(0) = 731$, $y_2(1) = 2781$, $y_1 < y_2$, $\Delta_{12}^{H} = -1$.

Шаг 3. Вычисляем на этом шаге текущее значение $L_{кен}$

$$L_{кен} = 1 - (\Delta_{12}^{G} \cdot \Delta_{12}^{H}) = 1 - (-1)(-1) = 0.$$

Шаг 4. Меняем индекс k внутреннего цикла до $k = 7$.

Повторяем действия шагов 2 и 3.

Шаг 5. Меняем индекс q внешнего цикла до $q - 1 = 6$.

Повторяем действия шагов 2 и 4.

Шаг 6. Останов.

В результате вычислений получили $L_{кен} = 0$.

Для оценки близости двух функций $F(x)$ и $G(x)$, заданных векторами своих значений в n точках, часто применяется и расстояние по Чебышеву:

$$L(G(y_i), H(y_i)) = \max_i \left| G(y_i) - H(y_i) \right|. \qquad (1.10)$$

В зависимости от требований в задаче распознавания применяют различные виды функций для определения меры схоства объектов.

1.3. Распознавание по углу между векторами

Мера близости между двумя векторами в *n*-мерном векторном пространстве может быть задана в виде угла. Если задано входное изображение $S_i = (s_{i1}, s_{i2}, \ldots, s_{in})$ и векторы эталонных изображений $X_1 = (x_{11}, x_{12}, \ldots, x_{1n})$, $X_2 = (x_{21}, x_{22}, \ldots, x_{2n})$, …, $X_m = (x_{m1}, x_{m2}, \ldots, x_{mn})$, то мера сходства между входным и эталонными изображениями определяется выражением

$$L(S_i, X_j) = \arccos\left(\frac{s_{i1}x_{j1} + s_{i2}x_{j2} + \ldots + s_{in}x_{jn}}{\sqrt{s_{i1}^2 + s_{i2}^2 + \ldots + s_{in}^2} \cdot \sqrt{x_{j1}^2 + x_{j2}^2 + \ldots + x_{jn}^2}} \right),$$

или

$$L(S_i, X_j) = \arccos\left(\sum_{k=1}^{n} s_{ik}x_{jk} \Big/ \left|S_i\right| \cdot \left|X_j\right| \right), \qquad (1.11)$$

где $\left|S_i\right|$, $\left|X_j\right|$ – соответственно длины векторов S_i и X_j.

Принадлежность входного изображения S_i к одному из *m* образов определяется с помощью решающего правила

$$S_i \in X_j, \text{ если } \quad L(S_i, X_j) = \min_j L(S_i, X_j). \qquad (1.12)$$

При этом в решающем правиле и далее по тексту для обозначения *j*-го образа и эталонного изображения *j*-го образа применяется одно и тоже обозначение X_j ($j = \overline{1, m}$).

1.4. Распознавание изображений по скалярному произведению

Мера близости изображений по углу между векторами (1.11) основана на скалярном произведении векторов:

$$< S_i,\ X_j > \ = |S_i||X_j|\cos\alpha = \sum_{k=1}^{n} s_{ik} x_{jk}, \qquad (1.13)$$

где α – угол между векторами S_i и X_j.

Некоторые системы распознавания используют скалярное произведение в качестве меры сходства изображений в n-мерном векторном пространстве

$$L(S_i, X_j) = \sum_{k=1}^{n} s_{ik} x_{jk}. \qquad (1.14)$$

В этом случае принадлежность входного изображения S_i к какому-либо образу определяется с помощью решающего правила

$$S_i \in X_j, \text{ если } \ L(S_i, X_j) = \max_j L(S_i, X_j). \qquad (1.15)$$

1.5. Распознавание изображений по принадлежности к заданной области пространства

При этом способе распознавания все пространство изображений V разбивается на непересекающиеся области V_1, V_2, …, V_m, V_{m+1}, где V_1, V_2, …, V_m – области, содержащие изображения только одного соответствующего образа X_1, X_2, …, X_m; V_{m+1} – область, не содержащая изображений, относящихся к указанным образам. В этом случае принадлежность входного изображения $S_i = (s_{i1}, s_{i2}, \ldots, s_{in})$ к некоторому j-му образу $(j = \overline{1, m})$ определяется решающим правилом

$$S_i \in X_j, \text{ если } (s_{i1}, s_{i2}, \ldots, s_{in}) \in V_j. \qquad (1.16)$$

Если области V_j $(j = \overline{1,m})$ заданы в евклидовом пространстве в виде шаров с центрами в точках $(x^*_{j1}, x^*_{j2}, \ldots, x^*_{jn})$ и радиусами R_j, то решающее правило (1.16) принимает вид

$$S_i \in X_j, \text{ если } L(S_i, X_j) = \sqrt{\sum_{k=1}^{n} (x^*_{jk} - s_{ik})^2} \le R_j. \qquad (1.17)$$

Для конструирования областей в пространстве изображений могут использоваться любые меры сходства, например, расстояния с весовыми коэффициентами (1.18) – (1.20), расстояние по Камберра (1.21) и т.д.:

$$L(S_i, X_j) = \sqrt{\sum_{k=1}^{n} \eta_k (s_{ik} - x_{jk})^2}, \qquad (1.18)$$

$$L(S_i, X_j) = \sqrt[\lambda]{\sum_{k=1}^{n} \eta_k (s_{ik} - x_{jk})^{\lambda}}, \qquad (1.19)$$

$$L(S_i, X_j) = \sum_{k=1}^{n} \eta_k \left| s_{ik} - x_{jk} \right|, \qquad (1.20)$$

$$L(S_i, X_j) = \sum_{k=1}^{n} \frac{\left| s_{ik} - x_{jk} \right|}{\left| s_{ik} + x_{jk} \right|}, \qquad (1.21)$$

где η_k $(k = \overline{1,n})$ – весовые коэффициенты; λ – целое положительное число, большее двух.

Решающее правило (1.16) для расстояний (1.18)–(1.20), в которых используются весовые коэффициенты, требует использования не шаров, а эллипсоидов в n-мерном пространстве с центрами в точках $(x^*_{j1}, x^*_{j2}, \ldots, x^*_{jn})$ и длинами полуосей $l_{j1}, l_{j2}, \ldots, l_{jn}$, где

$j = \overline{1, m}$, m – число непересекающихся областей $V_1, V_2, ..., V_m$, содержащих изображения, относящиеся к указанным m образам.

При использовании для распознавания угла между векторами непересекающиеся области V_j $(j = \overline{1, m})$ задаются в виде конусов, а решающее правило имеет вид

$$S_i \in X_j, \text{ если } L(S_i, X_j) =$$

$$= \phi_{ij} = \arccos\left(\frac{s_{i1}x_{j1} + s_{i2}x_{j2} + \ldots + s_{in}x_{jn}}{\sqrt{s_{i1}^2 + s_{i2}^2 + \ldots + s_{in}^2} \cdot \sqrt{x_{j1}^2 + x_{j2}^2 + \ldots + x_{jn}^2}} \right) \le \phi_{j\max},$$

где ϕ_{ij} – угол между предъявленным изображением S_i и эталонным изображением X_j; $\phi_{j\max}$ – предельно допустимый угол для j-го образа между эталонным и распознаваемым изображениями.

1.6. Распознавание объектов по качественным признакам

В большинстве случаев образы и отдельные изображения характеризуются с помощью количественных характеристик: геометрических размеров, веса, площади, объема и т. д. В этих случаях количественные изменения характеристик конкретного изображения обычно не сразу ведут к изменению образа, к которому относится распознаваемое изображение. Только достигнув определенных для каждого образа границ, количественные изменения вызывают качественный скачок – переход к другому образу.

Образы и конкретные изображения могут характеризоваться не только количественными, но и качественными характеристиками (свойствами, признаками, атрибутами). Эти признаки не могут быть описаны (или обычно не описываются) количественно, например, цвет, вкус, ощущение, запах. Образы либо обладают какими-то качественными характеристиками, либо не обладают.

Между качественными и количественными характеристиками об-

разов есть существенное различие, однако это различие во многих случаях нельзя абсолютизировать, поскольку каждому качественному атрибуту присущи и определенные интервалы изменения количественных характеристик, за пределами которых меняется и качественный атрибут. Например, определенному цвету изображения соответствует конкретный диапазон длин электромагнитных волн, за пределами которого цвет изменится.

Существуют различные подходы к распознаванию изображений с качественными характеристиками. В данной лабораторной работе рассмотрим один из них, основанный на двоичном кодировании наличия или отсутствия какого-либо качественного признака. В рассматриваемом подходе изображение X_k некоторого образа с качественными характеристиками представляется в виде двоичного вектора $X_k = (x_{k1}, x_{k2}, \ldots, x_{kj}, \ldots, x_{kn})$, где n – размерность пространства признаков.

Если изображение X_k обладает j-м признаком, то $x_{kj} = 1$, а если нет, то $x_{kj} = 0$, т. е. здесь отождествляется объект и двоичный вектор, его описывающий.

Рассмотрим в качестве примера четыре объекта (вишня, апельсин, яблоко, дыня), каждый из которых имеет три признака: цвет, наличие косточки или семечек. В табл. 1.2 приведены числовые значения признаков для рассматриваемого примера после их двоичного кодирования.

Таблица 1.2 – Наличие и кодирование свойств объектов

	Вектор признаков	Желтый цвет	Оранжевый цвет	Красный цвет	Есть косточка	Есть семечки
Вишня	X_1	$x_{11} = 0$	$x_{12} = 0$	$x_{13} = 1$	$x_{14} = 1$	$x_{15} = 0$
Апельсин	X_2	$x_{21} = 0$	$x_{22} = 1$	$x_{23} = 0$	$x_{24} = 0$	$x_{25} = 1$
Яблоко	X_3	$x_{31} = 1$	$x_{32} = 0$	$x_{33} = 1$	$x_{34} = 0$	$x_{35} = 1$
Дыня	X_4	$x_{41} = 1$	$x_{42} = 0$	$x_{43} = 0$	$x_{44} = 0$	$x_{45} = 1$

Наиболее простой метод решения задач распознавания объектов с качественными характеристиками после двоичного кодирования атрибутов – свести решение исходной задачи к решению задачи распознавания объектов с количественными характеристиками в n-мерном

векторном пространстве.

Для этого необходимо для каждого качественного признака ввести в *n*-мерном векторном пространстве ось. Если для рассматриваемого объекта признак существует, то на оси откладывается единица, если нет – то нуль. В результате получается многомерное двоичное пространство признаков, где можно использовать различные расстояния, применяемые для распознавания объектов с количественными характеристиками.

В рассматриваемом примере в результате введения количественных характеристик вместо качественных признаков (табл. 1.2) получается пятимерное двоичное пространство, где можно применять расстояния по Евклиду (1.2), по Минковскому (1.3), расстояние, использующее сумму модулей разностей между соответствующими компонентами *n*-мерных векторов (1.4), это, соответственно, формулы:

$$L_1(S_i, X_j) = \sqrt{\sum_{k=1}^{n} (s_{ik} - x_{jk})^2},$$

$$L_2(S_i, X_j) = \sqrt[\lambda]{\sum_{k=1}^{n} (s_{ik} - x_{jk})^{\lambda}},$$

$$L_3(S_i, X_j) = \sum_{k=1}^{n} \left| s_{ik} - x_{jk} \right|,$$

где $L_p(S_i, X_j)$, $p = \overline{1,3}$ – соответствующее расстояние между входным изображением $S_i = (s_{i1}, \ldots, s_{in})$ и эталонным изображением $X_j = (x_{j1}, \ldots, x_{jn})$ j-го образа; λ – целое положительное число, большее двух. Расстояния (1.2) – (1.4) могут использоваться также и с весовыми коэффициентами.

При двоичном кодировании качественных признаков может применяться и расстояние по Хеммингу, которое вводится для любых двоичных векторов. Расстояние по Хеммингу между двумя двоичными векторами равно числу несовпадающих компонент векторов.

Если вектора имеют все одинаковые компоненты, то расстояние между ними равно нулю, если вектора не имеют совпадающих компо-

нент, то расстояние равно размерности векторов.

Более тонкая классификация объектов с качественными признаками получается при введении для каждой пары объектов X_j, X_i, для которых введено двоичное кодирование качественных признаков, переменных, характеризующих их общность или различие с помощью табл. 1.3.

Таблица 1.3 – Взаимосвязь признаков

X_j	X_i	
	1	0
1	a	h
0	g	b

Переменная a в табл. 1.3 предназначена для подсчета числа общих признаков объектов X_j, и X_i. Она может быть вычислена с помощью соотношения

$$a = \sum_{k=1}^{n} x_{jk} x_{ik},$$

где x_{jk}, x_{ik} – двоичные компоненты векторов, описывающих объекты X_j и X_i.

С помощью переменной b подсчитывается число случаев, когда объекты X_j и X_i не обладают одним и тем же признаком,

$$b = \sum_{k=1}^{n} (1 - x_{jk})(1 - x_{ik}).$$

Переменные g и h предназначены соответственно для подсчета числа признаков, присутствующих у объекта X_i и отсутствующих у объекта X_j, и, присутствующих у объекта X_j и отсутствующими у объекта X_i,

$$g = \sum_{k=1}^{n} x_{ik} (1 - x_{jk}), \qquad h = \sum_{k=1}^{n} (1 - x_{ik}) x_{jk}.$$

Из анализа переменных a, b, g, h следует, что, чем больше сходство между объектами X_j и X_i, тем больше должна быть переменная a, т.е. мера близости объектов или функция сходства должна быть возрастающей функцией от a, функция сходства должна быть симметричной относительно переменных g и h. Относительно переменной b однозначный вывод сделать не удается, поскольку, с одной стороны, отсутствие одинаковых признаков у объектов может свидетельствовать об их сходстве, однако, с другой стороны, если у объектов общим является только отсутствие одинаковых признаков, то они не могут относиться к одному классу.

Наиболее часто применяются следующие функции сходства.

Функция сходства Рассела и Рао

$$S_1(X_i, X_j) = \frac{a}{a+b+g+h} = \frac{a}{n}. \qquad (1.22)$$

Функция сходства Жокара и Нидмена

$$S_2(X_i, X_j) = \frac{a}{n-b}. \qquad (1.23)$$

Функция сходства Дайса

$$S_3(X_i, X_j) = \frac{a}{2a+g+h}. \qquad (1.24)$$

Функция сходства Сокаля и Снифа

$$S_4(X_i, X_j) = \frac{a}{a+2(g+h)}. \qquad (1.25)$$

Функция сходства Сокаля и Мишнера

$$S_5(X_i, X_j) = \frac{a+b}{n}. \qquad (1.26)$$

Функция сходства Кульжинского

$$S_6(X_i, X_j) = \frac{a}{g+h}. \quad (1.27)$$

Функция сходства Юла

$$S_7(X_i, X_j) = \frac{ab-gh}{ab+gh}. \quad (1.28)$$

Предлагаем самостоятельно проанализировать области существования функций.

Индивидуальные задания

Общее задание

Разработайте алгоритм и программу, моделирующую распознавание различных типов объектов в *n*-мерном векторном пространстве, согласно номеров индивидуальных формул по табл. 1.4.

Таблица 1.4 − Варианты индивидульных заданий

Номер формулы	Последняя цифра номера в журнале									
	1	2	3	4	5	6	7	8	9	0
1.2	+		+		+		+		+	+
1.3	+	+		+	+	+		+	+	
1.4		+	+	+		+	+	+		+
1.5	+		+		+		+		+	
1.7		+		+		+		+		+
1.8	+		+		+		+		+	
1.9		+		+		+		+		+
1.10		+		+		+		+		+
1.11	+		+		+		+		+	
1.22–1.25	+		+		+		+		+	
1.25–1.28		+		+		+		+		+

В качестве исходных данных используйте буквы из своего имени или фамилии (не менее 4). Задайтесь размерностью *n*-мерного вектор-

ного пространства и числом m эталонных объектов образов (n и m не менее 5). Задайтесь объектами и определите их принадлежность к тому или иному образу.

Содержание отчета

1. Тема лабораторного занятия.
2. Индивидуальное задание.
3. Описание применяемых методов решения.
4. Текст основной части программы.
5. Результаты выполнения индивидуального задания.
6. Выводы по работе.

Контрольные вопросы

1. Какова структура систем распознавания образов?
2. Какие есть фазы распознавания образов?
3. Как представляют образ в n-мерном пространстве при распознавании?
4. Что такое решающие правила?
5. Какие есть меры подобия между образами в n-мерном пространстве?
6. Предложите свою уникальную функцию сходства для объектов с качественными характеристиками.
7. Используя формулу суммы модулей разностей между соответствующими компонентами n-мерных векторов: определить меру сходства объектов 1 и 2 с объектом 3:

1-й (1 1 0 0 1 1 0 0 1 1),
2-й (0 0 1 1 1 1 1 0 0 0),
3-й (0 0 0 0 1 1 0 0 1 1).

Какой объект 1 или 2 больше похож на 3-й?

8. Предложите для одной из функций сходства (1.2) − (1.10) примеры распознавания, в одном из которых функция сходства должна принимать минимальное значение, а в другом − максимальное.

Лабораторная работа № 2
ОБУЧАЮЩИЕСЯ СИСТЕМЫ

Цель работы: приобретение и закрепление знаний и получение практических навыков по созданию простейших самообучающихся систем.

2.1. Типы обучения

Под обучением в системах ИИ, как в психологии, так и в обыденной жизни, понимают способность системы (или программы) приобретать ранее неизвестные навыки и умения. Идеи возможности обучения и самообучения ЭВМ возникли с первых дней их существования. Казалось, что благодаря возможности быстрого и надежного запоминания больших объемов различных сведений, самостоятельного обучения и обучения с помощью экспертов, вычислительные машины смогут быстро превзойти человека в любой области. Однако десятилетия труда кибернетиков, психологов и программистов с момента появления первых ЭВМ показали, что за внешней простотой идей обучения скрываются чрезвычайно сложные проблемы, и в настоящее время вычислительным машинам еще очень далеко в этой области до человека [2].

Многолетний труд тысяч специалистов в области обучения искусственных систем все же не пропал даром и налицо имеется определенный прогресс в этом направлении. Прежде всего, более глубоко изучены процессы обучения человека и установлено, что существуют различные уровни обучения, связанные между собой иерархической структурой (рис. 2.1).

Уровень 1 представляет собой простое занесение в память компьютера того, что он должен "знать". К первому уровню относятся подавляющее большинство обычных компьютерных программ современных ЭВМ.

Программа или робот при выполнении конкретных условий, за-

данных программистом, осуществляет то, что однозначно определено и может меняться только путем перепрограммирования.

Таковы, например, действия промышленных роботов первого поколения, движения которых однозначно сформированы движениями квалифицированного рабочего, выполнявшего ту или иную технологическую операцию и одновременно программировавшего будущие действия робота, запускаемого определенным кодом и способного только слепо повторять движения человека, независимо от состояний внешней среды и объекта воздействия [2].

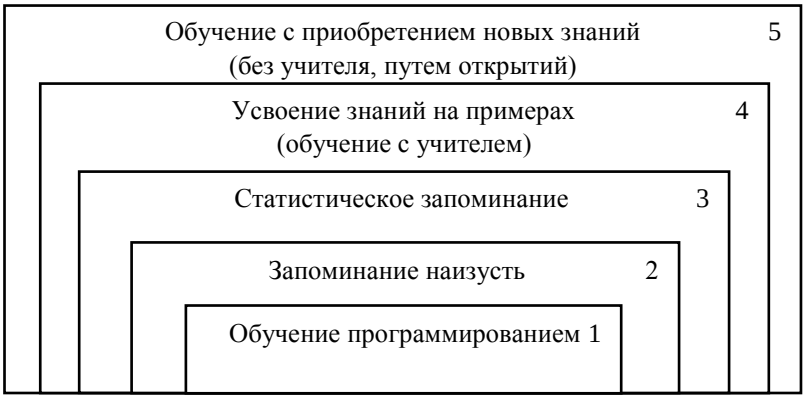

Рис. 2.1 Уровни обучения в системах ИИ

Уровень 2, или запоминание наизусть предполагает запоминание ситуаций и соответствующих им действий. Примерами интеллектуальных программ с запоминанием наизусть могут служить программы, написанные инженером фирмы IBM А. Сэмюэлем для игры в шашки на доске 8×8 в период с 1956 по 1967 гг. Как известно, шашки ходят по диагоналям вперед на соседнее свободное поле, а бьют – вперед и назад, перескакивая через шашку противника и становясь на соседнее за этой шашкой свободное поле. Дамки, в отличие от шашек, могут ходить вперед и назад на любое из свободных полей по диагонали и бить шашку противника независимо от числа свободных до нее полей, но если имеется хотя бы одно свободное поле за этой шашкой. Шашки могут превращаться в дамки, дойдя до последней горизонтали поля.

Первая программа, написанная А. Сэмюэлем в 1956 г. содержала около 180 тыс. позиций, заимствованных из лучших книг по игре в шашки. Поскольку даже для компьютеров 21 века это достаточно много, то А. Сэмюэль с целью ускорения доступа упорядочил позиции вначале по числу шашек и дамок у обоих партнеров, а затем – по частоте их появления в играх программы. При приближении памяти к переполнению использовалась процедура исключения позиций, мало встречавшихся (или совсем не возникавших) в партиях программы за какой-то последний период времени [2].

Уровень 3 – статистическое запоминание. Оно является усовершенствованным запоминанием наизусть, так как в этом случае хранится информация, полученная на основе анализа не отдельных, а большого числа случаев. Вернемся к программам игры в шашки А. Сэмюэля. Хотя 180 тысяч позиций – это и много, однако, это на порядки меньше, чем может быть при игре. Поэтому, если позиции в памяти не было, то ход машина рассчитывала с помощью полиномиальной функции оценивания

$$F = \sum_{i=1}^{38} a_i P_i, \qquad (2.1)$$

где a_i – параметры; P_i – характеристики, среди которых можно отметить контроль центра, материальный выигрыш, число угроз, число продвинутых шашек и т.д.

В первых программах расчет функции оценивания (2.1) выполнялся для глубины в три полухода (варианты, в которых на последнем полуходе была подстановка под удар своей шашки или взятие шашки противника, продолжались до более спокойных позиций и достигали порой до двадцати полуходов).

Функция F могла использоваться и в тех позициях, где лучшие ходы были уже известны. В идеальном случае функция (2.1) должна была давать те же лучшие ходы, что записаны в памяти машины, однако это случалось далеко не всегда, и возникала необходимость корректировать параметры a_i. Коррекция выполнялась следующим образом.

Программа рассчитывала по F число L лучших и число X худших ходов по сравнению с хранящимися в памяти, а затем вычисляла коэффициент

$$K = \frac{(X - L)}{(X + L)}.$$

Если $L = 0$ и, следовательно, $K = 1$, то наблюдается полное соответствие между оценками позиций по выражению (2.1) и теми оценками, которые заложены в память машины из лучших книг. Естественно, что функция F в этом случае идеальна, и менять ее не нужно. Если $X = 0$, то $K = -1$, и наблюдается полное расхождение между оценками из книг и с помощью функции (2.1). Понятно, что нельзя одним коэффициентом K оценивать все 38 параметров a_i, поэтому в программах А. Сэмюэля каждый параметр оценивался своим коэффициентом

$$K_i = \frac{(X_i - L_i)}{(X_i + L_i)},$$

где L_i и X_i – соответственно число ходов, превосходящих лучший ход из книг и уступающих ему по величине одночлена a_iP_i.

Программа, обучаясь по лучшим партиям шашечных мастеров, могла найти оптимальные параметры оценочной функции. Для определения хороших наборов коэффициентов требовалось порядка двадцати партий, при этом для полиномиального оценивания достигался 32 %-й выбор наилучшего хода. Такой относительно низкий процент лучших ходов связан с простым видом оценивающего выражения (2.1).

Уровень 4 – усвоение знаний на примерах с помощью учителя. Приведенный пример получения коэффициентов оценочной функции (2.1) при игре в шашки можно рассматривать и как обучение с учителем, в результате которого происходит статистическое запоминание. В обучении с учителем есть одна опасность: если учитель плох, то и обучающаяся программа или система будет соответствующего качества. Это обнаружил и А. Сэмюэль – если программа обучалась в игре со слабыми партнерами, то качество ее игры резко падало.

Уровень 5 – обучение без учителя. Этот вид обучения также можно иллюстрировать примером из шашек. Программа может играть сама против себя, пользуясь разными вариантами функции оценивания F и извлекая знания самостоятельно. Пусть A – лучшая версия программы, оценочную функцию которой обозначим F_a. Пусть эта программа играет против программы B с оценочной функцией F_b. Если выигрывает программа A, то слегка меняются параметры функции F_b и играется следующая партия. Если выигрывает программа B, то программа A получает штрафное очко. После определенной суммы штрафных очков производится взаимный обмен функциями F_a и F_b и существенно меняются параметры F_b неудачно игравшей программы.

Иногда третий, четвертый и пятый уровни обучения объединяют под общим названием обучения *с обратной связью*.

В общем виде система ИИ работает с объектами, которые характеризуются *именем* и совокупностью *атрибутов*. Атрибуты, в свою очередь, содержат некоторую *информацию*, представленную как в числовой, так и в логической форме. Кроме этого, атрибуты имеют свой тип, который задается *индексом* (i).

2.2. Структура обучающихся систем

Система может работать в трех режимах (рис. 2.2): обучение с учителем; самообучение и распознавание. В начале работы задается априорная информация о распознаваемых объектах системы (вводятся имена и атрибуты объектов). После этого вводятся атрибуты неизвестного объекта.

В *режиме распознавания* система соотносит введенные атрибуты с атрибутами известных объектов и определяет, к какому из них можно отнести неизвестный объект.

В *режиме обучения с учителем* учитель дает указания системе, к какому из объектов (1 или 2) отнести введенные атрибуты (т.е. сам распознает объект).

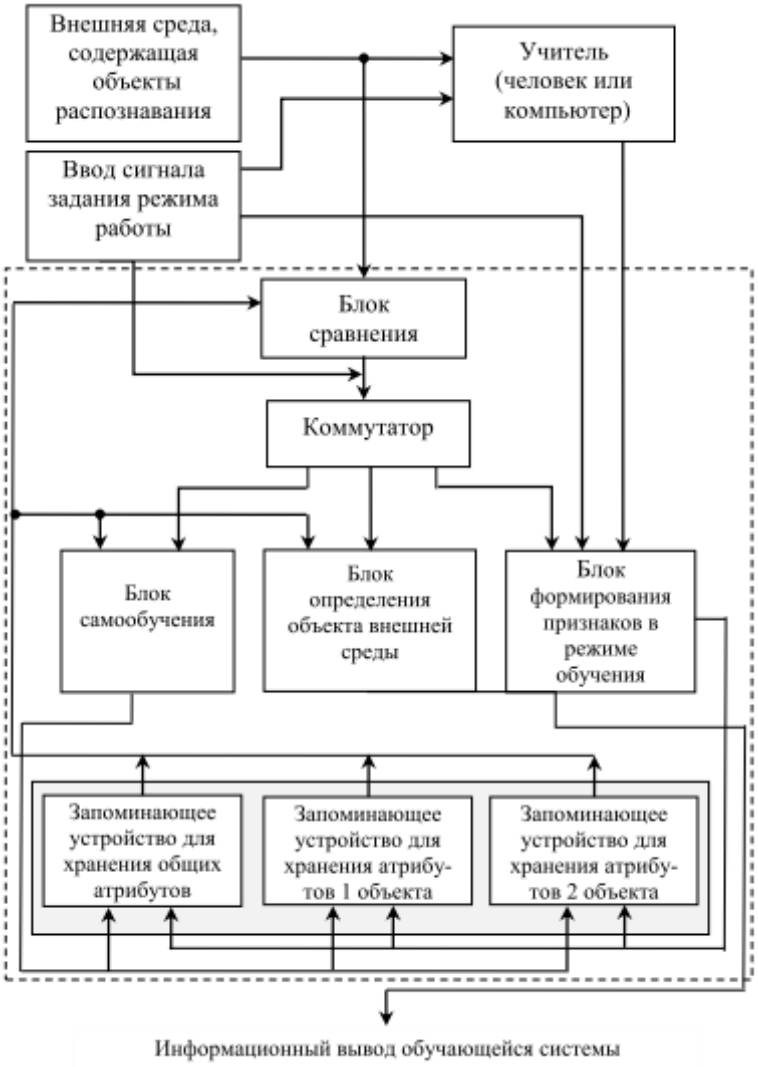

Рис. 2.2 Структура обучающейся системы

В *режиме самообучения* система сама соотносит введенные атрибуты с одним из объектов.

После опознания объекта в режиме обучения система производит *модификацию атрибутов* этого объекта в зависимости от индекса атрибута. Индекс определяет способ обработки атрибута в режиме обучения. Возможны следующие способы обработки:

1) запоминание атрибутов в объекте. (Атрибут, как он есть в чистом виде, запоминается в списке атрибутов объекта);

2) запоминание атрибута с выделением списка общих атрибутов. (Кроме запоминания в списке атрибутов объекта, проверяется наличие такого же атрибута у других объектов, и если он есть, то он выносится в список общих атрибутов);

3) запоминание атрибутов и статистическая обработка (модификация числовой характеристики атрибута);

4) запоминание атрибутов, статистическая обработка и выделение списка общих атрибутов.

Для атрибутов вводится также и способ статистической обработки (*St*). Она применяется для тех атрибутов, которые имеют числовую характеристику, для логических атрибутов отмечается, что статистическая обработка отсутствует $i = 0$).

2.3. Пример работы обучающейся системы

Рассмотрим обучающуюся программу, которая должна распознавать два объекта: PC конфигурации 1 на базе процессора Intel и PC конфигурации 2 на основе процессора AMD.

Некоторые общие свойства и свойства, присущие только каждому из этих двух объектов, приведены на рис. 2.3.

Часть общих свойств (стоимость, объем винчестера и оперативного ЗУ), хотя и присущи обоим объектам, но существенно отличаются своими числовыми характеристиками, по величине которых легко распознать предъявляемый объект.

Рис. 2.3 Свойства распознаваемых объектов

В режиме обучения числовые значения этих свойств могут изменяться (например, могут предъявляться ПК с разными объемами ОЗУ и винчестера, разной стоимостью), что вызывает необходимость статистической обработки входной информации. Рассмотрим несколько простейших способов такой обработки.

Предположим, что статистическая обработка *первого вида* предполагает вычисление среднего арифметического значения числовой характеристики атрибута при его многократном предъявлении в режиме обучения. Как известно, среднее арифметическое S некоторых n величин a_1, a_2, ..., a_n определяется выражением

$$S = (a_1 + a_2 + \ldots a_n)/n.$$

Поэтому, если вычисляется среднее арифметическое i-го атрибута a_i после его предъявления в $(n + 1)$-й раз, то из предыдущего соотношения несложно получить

$$S_{ai}(n+1) = (nS_{ai}(n) + a_i(n+1))/(n+1),$$

где $S_{ai}(k),\ k = n, n+1$ – среднее арифметическое *i*-го атрибута после его *k*-го предъявления; $a_i(n+1)$ – численное значение *i*-го атрибута в момент его $(n+1)$-го предъявления.

Статистическая обработка *второго вида* предполагает вычисление среднего S_{qai} с весом $q = (q_1, q_2, \ldots, q_n),\ q_i > 0,\ \sum q_i = 1$ по множеству численных значений $a_i(1), a_i(2), \ldots, a_i(n)$ атрибута a_i при его многократном появлении в режиме обучения

$$S_{qai} = \sum_{j=1}^{n} q_j a_i(j).$$

Полагая, что последние предъявления наиболее информативны, например, в силу того, что атрибут может меняться в процессе обучения, зададим, что при (*n* + 1)-м предъявлении атрибута вес q_{n+i} определяется выражением

$$q_{n+1} = \begin{cases} 1, \text{ при } n = 1, \\ 1/n, \text{ при } n = 2, 3, \ldots, m, \\ 1/(m+1), \text{ при } n \geq m+1, \end{cases}$$

а веса q_i, q_2, ..., q_n равны между собой и их сумма определяется с помощью предыдущего соотношения выражением

$$\sum_{j=1}^{n} q_j = 1 - q_{n+1}.$$

В этом случае среднее с весом *i*-го атрибута S_{qai} при его (*n* + 1)-м предъявлении может быть определено выражением

$$S_{qai}(n+1) = (1 - q_{n+1}) S_{qai}(n) + q_{n+1} a_i(n+1),$$

где $S_{qai}(k)$, $k = n$, $n + 1$ – среднее с весом *i*-го атрибута после его *k*-го

предъявления; q_{n+1} – вес i-го атрибута при его $(n + 1)$-м предъявлении.

Статистическая обработка *третьего вида* предусматривает в режиме обучения вычисление среднего геометрического S_{qai} множества численных значений i-го атрибута при его n-кратном появлении

$$S_{qai}(n) = (a_i(1)a_i(2)\dots a_i(n))^{1/n}.$$

Если известно среднее геометрическое при n-кратном появлении атрибута, то при $(n + 1)$-м его появлении, используя предыдущее выражение, можно вычислить и среднее геометрическое $S_{qai}(n+1)$

$$S_{qai}(n+1) = ((S_{qai}(n))^n a_i(n+1))^{1/n+1},\ n \geq 1.$$

При статистической обработке *четвертого вида* определяется интервал $[a_{imin}(m),\ a_{imax}(m)]$, в котором должны находиться числовые характеристики i-го атрибута после его m предъявлений. При первом появлении i-го атрибута $a_i(1)$ задают $a_{imin}(1) = a_{imax}(1) = a_i(1)$. При следующем появлении атрибута, когда выполняется неравенство $a_i(k) \neq a_i(1),\ k \geq 2$, определяют граничные точки интервала соотношениями:

$$a_{i\min}(k) = a_i(1),\ a_{i\max}(k) = a_i(k), \text{ если } a_i(k) > a_i(1),$$
$$a_{i\min}(k) = a_i(k),\ a_{i\max}(k) = a_i(1), \text{ если } a_i(k) < a_i(1).$$

При последующих появлениях i-го атрибута граничные точки интервала задаются соотношениями:

$$a_{i\min}(n+1) = a_{i\min}(n),\ a_{i\max}(n+1) = a_{i\max}(n),$$
$$\text{если } a_{i\min}(n) \leq a_i(n+1) \leq a_{i\max}(n);$$

$$a_{i\min}(n+1) = a_i(n+1),\ a_{i\max}(n+1) = a_{i\max}(n),$$
$$\text{если } a_i(n+1) < a_{i\min}(n);$$

$$a_{i\min}(n+1) = a_{i\min}(n),\ a_{i\max}(n+1) = a_i(n+1),$$
$$\text{если } a_i(n+1) > a_{i\max}(n).$$

Будем рассматривать обучение системы (программы) на примере двух конкретных объектов, и закладывать в ее основу общие принципы распознавания двух любых объектов, а также предполагать возможность распространения этих принципов на распознавание трех и большего числа объектов.

Для обеспечения работы программ с произвольной парой объектов необходимо, чтобы она в режиме обучения, прежде всего, запрашивала имена распознаваемых объектов:

```
Имя 1-го объекта? PC1
Имя 2-го объекта? PC2
```

Затем должен быть запрос об общем характере обработки атрибутов объектов. Предположим, что в программе возможны три способа обработки входной информации:

- ✓ запоминание атрибутов объектов;
- ✓ запоминание атрибутов объектов с выделением списка общих атрибутов;
- ✓ запоминание атрибутов объектов с выделением списка общих атрибутов и статистической обработкой числовой информации атрибутов.

В этом случае на запрос программы возможны три разных ответа:

1. Запоминание.
2. Запоминание и выделение общих атрибутов.
3. Запоминание и выделение общих атрибутов, статистическая обработка.

При наличии в программе режима записи априорной информации делается запрос об использовании этого режима:

Запись априорной информации? Да/Нет.

Если ответ "Да", то программа приступает к вводу атрибутов первого объекта. В связи с тем, что числовые значения атрибутов могут подвергаться различным видам статистической обработки, то, кроме имен атрибутов и их числовых характеристик, необходимо вводить еще и числовое значение переменной, обозначенной через *i*, указыва-

ющее вид необходимой обработки поступающих данных (соответственно $i = 1, 2, 3$ или 4) или отсутствие такой обработки ($i = 0$):

```
Введите атрибуты первого объекта
Введите атрибут 1?  i = 0, Встроен. граф. адаптер
Введите атрибут 2? i = 1, стоимость, 500 $
Введите атрибут 3? i = 4, объем винчестера, 1000 Гб
Введите атрибут 4? i = 0, процессор Intel
Введите атрибут 5? Entrer
```

Затем следует ввод атрибутов второго объекта и атрибутов общего списка:

```
Введите атрибуты второго объекта
 Введите атрибут 1? i = 0, процессор AMD
 Введите атрибут 2? i = 1, стоимость, 600 $
 Введите атрибут 3? i = 4, объем винчестера, 1500 Гб
 Введите атрибут 4?  Entrer
Введите атрибуты общего списка
Введите атрибут 1? i = 0, WiFi
Введите атрибут 2? i = 0, системный блок
Введите атрибут 3? Entrer
```

После этого на экран выводятся списки атрибутов обоих объектов:

Список 1 (PC 1)	Список 2 (PC 2)	Список общих свойств
Встроен. граф. адаптер стоимость 500 $ объем винчестера 1000 Гб	процессор AMD стоимость 600 $ объем винчестера 1500 Гб	WiFi системный блок

Затем программа задает вопрос о дальнейшем режиме работы.

```
Режим работы?
1 - запись априорной информации
2 - режим обучения с учителем
3 - режим распознавания
4 - режим распознавания с самообучением
5 - конец работы с программой
```

Пусть выбран режим работы – обучение с учителем 2.

После этого ответа программа запросит ввод атрибутов любого объекта:

```
Введите атрибуты любого объекта
Введите атрибут 1? i = 1, стоимость, 530 $
Введите атрибут 2? i = 0, Wi-Fi
Введите атрибут 3? i = 0, порт ввода-вывода
Введите атрибут 4? i = 4, объем винчестера, 750 Гб
Введите атрибут 5? Entrer
```

Получив атрибуты неизвестного объекта, программа вычисляет оценочные функции F_1 и F_2 полиномиального вида для каждого из объектов:

$$F_1 = \sum_{j=1}^{k} P_j^1 , \qquad F_2 = \sum_{j=1}^{k} P_j^2 ,$$

$$P_j^q = \begin{cases} 1, \text{ если i} = 0 \text{ и } j\text{-й атрибут имеется во введенном списке} \\ \quad \text{и в памяти атрубутов } q\text{-го объекта, или,} \\ \quad \text{если } i = 1, 2, 3 \text{ и } \left|a_j - S_j^{iq}\right| \le \left|a_j - S_j^{ik}\right|, \ q, k = 1, 2, \ q \ne k, \text{ или,} \\ \quad \text{если } i = 4 \text{ и } a_{j\min}^q \le a_j \le a_{j\max}^q , \\ 0, \text{ если } i = 0 \text{ и } j\text{-й атрибут отсутствует} \\ \quad \text{в памяти атрубутов } q\text{-го объекта, или,} \\ \quad \text{если } i = 1, 2, 3 \text{ и } \left|a_j - S_j^{iq}\right| > \left|a_j - S_j^{ik}\right|, \ q, k = 1, 2, \ q \ne k, \text{ или,} \\ \quad \text{если } i = 4 \text{ и } a_{j\min}^q > a_j \text{ или } a_j > a_{j\max}^q , \end{cases}$$

где k – число атрибутов, введенных программой на предыдущем этапе работы; a_i – численное значение j-го атрибута во введенном списке; S_j^{iq} – среднее значение j-го атрибута в памяти атрибутов сравниваемого q-го (q = 1, 2) объекта при i-м способе статистической обработки входных данных; $a_{j\min}^q, a_{j\max}^q$ – граничные точки допустимого интервала изменений j-го атрибута для q-го объекта при четвертом способе статистической обработки числовых данных.

После вычисления функций F_1 и F_2 производится логический вывод:

- ✓ предъявлен 1-й объект, если $F_1 > F_2$;
- ✓ предъявлен 2-й объект, если $F_1 < F_2$;
- ✓ предъявлен k-й (случайно выбранный) объект, если $F_1 = F_2$.

В рассматриваемом примере в соответствии с последними выражениями имеем:

$$F_1 = \sum_{j=1}^{4} P_j^1 = 1+1+0+1 = 3; \quad F_2 = \sum_{j=1}^{4} P_j^2 = 0+1+0+0 = 1.$$

Отсюда следует, что в этом случае выполняется условие соотношения, что предъявлен первый объект, поэтому программа запрашивает учителя:

Это PC1? Да

Затем программа обрабатывает списки атрибутов с учетом новой информации и выводит их на экран:

Список 1 (PC1)	Список 2 (PC2)	Список 3 (общий)
Встроен. граф. адаптер стоимость 515 $ объем винч. [750,1000] порт ввода-вывода	процессор AMD стоимость 600 $ объем винчестера 1500 Гб	Wi-Fi системный блок

Далее задается вопрос о дальнейшем режиме работы:

Режим обучения продолжается? Да
Введите атрибуты любого объекта
Введите атрибут 1? *i* = 0, Wi-Fi
Введите атрибут 2? *i* = 0, системный блок
Введите атрибут 3? *i* = 0, порт ввода-вывода
Введите атрибут 4? *i* = 0, процессор AMD
Введите атрибут 5? *i* = 4, объем ОЗУ 4 Гб
Введите атрибут 6? Entrer

В рассматриваемом случае оценочные функции F_1 и F_2 совпадают по величине, поэтому программа должна сделать случайный выбор объекта. Пусть в данном примере это будет первый объект. После этого программа запрашивает учителя:

Это PC1? Нет

Вслед за этим программа с учетом новой информации корректирует списки атрибутов и выводит их на экран:

Список 1 (PC1)	Список 2 (PC2)	Список 3 (общий)
Встроен. граф. адаптер стоимость 515 $ объем винчестера [750,1000]	процессор AMD стоимость 600 $ объем винчестера 1500 Гб объем ОЗУ 4 Гб	Wi-Fi системный блок порт ввода-вывода

Далее задается вопрос о дальнейшем режиме работы:

Режим обучения продолжается? Нет
Режим работы?
1 – запись априорной информации;
2 – режим обучения с учителем;
3 – режим распознавания;
4 – режим распознавания с самообучением;
5 – конец работы с программой.

Пусть следующим режимом работы будет режим распознавания с самообучением:

```
4
Введите атрибуты распознаваемого объекта
Введите атрибут 1? Процессор AMD
Введите атрибут 2? Системный блок
Введите атрибут 3? Порт ввода-вывода
Введите атрибут 4? Отдельный GPU
Введите атрибут 5?. Entrer
```

Вычисление оценочных функций F_1 и F_2 для предъявленного объекта дает, что $F_2 > F_1$, т.е. предъявлен второй объект. Однако при этом имеется атрибут (Отдельный GPU), которого нет ни в одном из списков, поэтому программа этот атрибут занесет в список второго объекта и выдаст сообщение:

```
Предъявлен объект PC2, неизвестный атрибут (Отдельный GPU)
внесен в список его атрибутов
Режим распознавания с самообучением продолжается? Да
Введите атрибуты любого объекта
Введите атрибут 1? WiFi
Введите атрибут 2? Объем ОЗУ 6 Гб
Введите атрибут 3? Порт ввода-вывода
Введите атрибут 4? Отдельный GPU
Введите атрибут 5?. Entrer
```

Вычисление оценочных функций F_1 и F_2 для предъявленного объекта дает, что $F_2 > F_1$, т.е. предъявлен второй объект, однако при этом числовой атрибут (объем ОЗУ 6 Гб) находится вне интервала [4, 4], полученного в режиме обучения, поэтому программа расширит интервал значений атрибута «объем ОЗУ» и выдаст сообщение:

```
Предъявлен объект PC2, изменен интервал атрибута объем ОЗУ с
[4, 4] Гб на [4, 6] Гб
Режим распознавания с самообучением продолжается? Нет
```

```
Режим работы?
1 – запись априорной информации;
2 – режим обучения с учителем;
3 – режим распознавания;
4 – режим распознавания с самообучением;
5 – конец работы с программой.
 5
```

Программа работу закончила.

Индивидуальные задания

№ по журналу	Способы обработки входной информации		Вид статистической обработки			
	Запоминание атрибутов	Запоминание атрибутов с выделением списка общих свойств	1	2	3	4
1	+	–	+	–	–	+
2	–	+	–	+	–	+
3	+	–	+	–	+	–
4	–	+	–	+	–	+
5	+	–	+	–	+	–
6	–	+	–	+	–	+
7	+	–	+	–	–	+
8	–	+	+	–	–	+
9	+	–	–	+	+	–
10	–	+	+	–	+	–
11	+	–	+	–	–	+
12	–	+	–	+	–	+
13	+	–	+	–	–	+
14	–	+	–	+	–	+
15	+	–	+	–	+	–
16	–	+	–	+	–	+
17	+	–	–	–	+	+
18	–	+	+	+	–	–
19	+	–	+	–	+	–
20	–	+	+	–	–	+

Содержание отчета

1. Тема лабораторного занятия.
2. Индивидуальное задание.
3. Описание применяемых методов решения.
4. Результаты выполнения индивидуального задания.
5. Выводы по работе.

Контрольные вопросы

1. Чем системы искусственного интеллекта отличаются от других систем?
2. Какие есть уровни обучения в системах ИИ?
3. Какова структура обучающейся системы?
4. Для чего создается список общих атрибутов объектов?
5. В чем отличие работы системы в режиме обучения с учителем от работы в режиме самообучения?
6. Какие виды статистической обработки применяются при обучении?
7. Когда оправдано применять интервальный вид атрибута объекта?
8. Для чего используется параметр *i* при описании атрибутов объекта?
9. На основании чего принимается решение о классе распознаваемого объекта?
10. Для чего применяется таблица общих свойств?
11. Определить к какому классу принадлежит объект X со свойствами (5, 8, 10, 0), если известны характеристики 2-х объектов (10, 5, 1, 5) и (5, 9, 15, 10).

Лабораторная работа № 3
АВТОМАТИЗАЦИЯ ЛОГИЧЕСКОГО ВЫВОДА

Цель работы: приобретение, закрепление знаний и получение практических навыков работы с языком Пролог: базы знаний, имена, переменные и списки; синтаксис фактов и правил; программа Пролога как совокупность фактов и правил. Построение программ для автоматического решения логических задач.

3.1. Теоретические основы языка Пролог

Пролог является языком исчисления предикатов. Предикат – это логическая формула от одного или нескольких аргументов. Можно сказать, что предикат – это функция, отображающая множество произвольной природы в множество {ложно, истинно} [3–6].

Обозначаются предикаты $F(x)$, $F(x, y)$, $F(x, y, z)$ и т. д.

Одноместный предикат $F(x)$, определенный на предметной области M, является истинным, если у объекта x есть свойство F, и ложным – если этого свойства нет.

Двухместный предикат $F(x, y)$ является истинным, если объекты x и y находятся в отношении F.

Многоместный предикат $F(x_1, x_2, x_3, ..., x_N)$ задает отношение между элементами $x_1, x_2, x_3, ..., x_N$ и интерпретируется как обозначение высказывания: «Элементы $x_1, x_2, x_3, ..., x_N$ находятся между собой в отношении F».

При разработке логических программ предикаты получают обычно названия, соответствующие семантике описываемой предметной области.

Примеры предикатов:

хищник (X).

супруги (X, Y).

фио (X, Y, Z).

Предикаты, которые нельзя разбить на отдельные компоненты,

называют атомарными. Сложные формулы строятся путем комбинирования атомарных предикатов логическими соединителями И, ИЛИ и НЕ. Одноместный предикат при подстановке в него значения переменной становится «нульместным», т.е. высказыванием-предложением, которое является истинным или ложным:

хищник (тигр).

При подстановке в *n*-местный предикат конкретного значения его местность становится равной *n* – 1. Таким образом, каждое высказывание порождается некоторым предикатом, а каждый предикат соответствует множеству высказываний.

Задача программы на Прологе заключается в том, чтобы доказать, является ли заданное целевое утверждение следствием из имеющихся фактов и правил или нет.

Решаемая задача представляется в виде совокупности утверждений (аксиом), цель задачи также записывается в виде утверждения. Тогда решение задачи представляет собой выяснение логического следования из аксиом $(A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_n) \rightarrow B$.

На практике удобно использовать доказательство от противного, т.е. доказывать невыполнимость. Если $A \rightarrow B$ истинно, то $\neg(A \rightarrow B)$ ложно и $A \rightarrow \neg B$ ложно.

На этом основан принцип резолюции, который требует представления формул исчисления предикатов в виде набора дизъюнктов, связанных операцией конъюнкции (конъюнктивная нормальная форма − КНФ).

Правило резолюции имеет вид

$$(X \vee A) \wedge (\neg A \vee Y) \rightarrow (X \vee Y)$$

и позволяет соединить две формулы, из одной удалив *A*, а из другой не *A*. В этом случае говорят, что *A* и не *A* резольвируют. Главная идея метода резолюции заключается в проверке того, содержит ли множество дизъюнктов *R* пустой (ложный) дизъюнкт. При положительном ответе формула, соответствующая *R*, невыполнима и соответствующее

утверждение противоречиво. Таким образом, доказав невыполнимость формулы

$$(A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_n) \rightarrow B,$$

можно сделать вывод, что B истинно.

Рассмотрим пример использования метода резолюции, применяя доказательство от противного. Пусть необходимо доказать истинность выражения

$$((P \vee Q) \wedge (P \rightarrow R) \wedge (Q \rightarrow S)) \rightarrow (R \wedge S).$$

Для использования метода резолюции нужно рассмотреть конъюнкцию совокупности посылок и отрицания заключения в конъюктивной нормальной форме. Для этого выполняется ряд шагов:

1. Приводим посылки к нормальной форме (без → и ↔):

$P \vee Q$.

$\neg P \vee R$.

$\neg Q \vee S$.

2. Записываем в нормальной форме отрицание заключения:

$\neg(R \vee S) = \neg R \wedge \neg S$.

$\neg R$.

$\neg S$.

3. Рассматриваем конъюнкцию пяти дизъюнктов:

$\neg R \wedge (R \vee \neg P) \wedge (P \vee Q) \wedge \neg S \wedge (\neg Q \vee S)$.

Первые два дизъюнкта дают $\neg P$, что в сочетании с третьим дизъюнктом дает Q. Четвертый и пятый дизъюнкты дают Q.

Таким образом, имеем

$Q \wedge \neg Q = False$.

Таким образом, доказана истинность выражения $((P \vee Q) \wedge (P \rightarrow R) \wedge (Q \rightarrow S)) \rightarrow (R \wedge S)$, поскольку противоположное неверно.

Любую программу на Прологе можно рассматривать как базу данных. Механизм обработки запросов в Прологе называется унифи-

кацией. После того как пользователь вводит запрос, интерпретатор приступает к анализу содержимого базы данных, выполняя допустимые подстановки фактов в целевое утверждение, чтобы обосновать его истинность.

3.2. Структура программы на Прологе

Программа на языке Пролог включает следующие основные разделы:

описание имен и структур объектов (domains);

описание предикатов – названий отношений, существующих между объектами (predicates);

раздел целевых утверждений (goal), который может отсутствовать; в этом случае программа будет запрашивать целевое утверждение при запуске;

описание фактов и правил, описывающих отношения (clauses).

Имена объектов (констант) в Прологе пишутся с маленькой буквы, а переменных – с большой.

3.3. Основные приемы программирования на Прологе

Рассмотрим программу 3.1, как можно описать на Прологе задачу из некоторой предметной области, например, географии.

```
/* Программа 3.1  это комментарий            */
domains
        gorod, strana = symbol       %  после процента − коментарий
predicates
        situ(gorod,strana)
clauses
%  Факты
        situ (london, england).
        situ (berlin, germanij).
```

```
    situ (kiev, ukraine).
    situ (pekin, cina).
    situ (warszawa, poland).
    situ (berlin, europe).
%  Правила
    situ (X, europe):- situ (X, england).
    situ (X, europe):- situ (X, poland).
```

Выражение

```
situ(kiev, ukraine).
```

описывает тот факт, что город Киев находится на Украине. Ранее в программе был введен соответствующий предикат, работающий с объектами символьного типа. Название города и страны здесь являются константами, поэтому по правилам пролога их можно писать с маленькой буквы или в двойных ковычках.

В конце раздела clauses программы 3.1 описаны два правила. Правило задает новый предикат через предикаты, определенные ранее. Правило состоит из головы (предиката) и тела – последовательности предикатов. Голова отделяется от тела значком :- , который можно интерпретировать как слово «if». Таким образом, заключение является головой правила, а тело правила состоит из набора посылок. Использование правил является основным способом представления знаний в интеллектуальных системах.

Смысл правила состоит в том, что цель, являющаяся головой, будет истинной, если интерпретатор пролога сможет показать, что все выражения (подцели) в теле правила являются истинными.

В правилах буква *X* (любая другая заглавная буква, или любое слово, начинающееся с заглавной буквы) обозначает переменную, которая может принимать разные значения.

Так, правило

```
situ (X, europe):- situ (X, poland).
```

означает, что любой польский город является одновременно европей-

ским городом.

Добавлением новых правил можно пополнять и модифицировать описание задачи. Если добавить одно правило, что все города Франции являются одновременно европейскими городами

```
situ (X, europe):- situ (X, france).
```

то можно будет по-прежнему использовать все остальные факты о городах Европы.

В Прологе можно использовать составные объекты. Составные объекты позволяют описывать иерархические структуры, в которых описание одного предиката включает в себя описание других предикатов.

```
% Программа 3.2
domains
    personal_library=book(title, author, publication)
    publication= publication(publisher, year)
    collector, title, author, publisher=symbol
    year=integer
predicates
    collection(collector, personal_library)
clauses
    collection("Иванов",book("Война и мир", "Лев Толстой",
    publication("Просвещение",1990))).
```

При описании правил часто возникает необходимость использовать логические связки И и ИЛИ. В качестве связки И используется запятая, а в качестве связки ИЛИ – точка с запятой.

```
gigant(X) :- rost(X,Y),Y>200.
star_or_mlad(X) :- X>70; X<10.
```

Пролог имеет большое количество встроенных предикатов, т.е. предикаты, определяемые автоматически. Например, встроенный пре-

дикат nl вызывает перевод строки, а встроенный предикат write применяется для вывода информации на экран. Встроенные предикаты используются так же, как и определяемые пользователем предикаты, но встроенный предикат не может являться головой правила или появляться в факте.

Часто используемыми встроенными предикатами являются = (унификация) и логическое отрицание not

```
student(X) :- X=”Петров”; X=”Иванов”.
xor_student(X) :- not(X=”Петров”), not(X=”Иванов”).
planeta(X) :- not(X=”солнце”).
```

Утверждение not(X = Y) эквивалентно X<>Y.

Иногда бывает полезно использовать предикаты, про которые заранее известно, истинны они или ложны. Для этих целей используют предикаты true и fail. Предикат true всегда истинен, в то время как fail всегда ложен. Последний предикат используется для управления процессом решения задачи на Прологе.

Запрос – это последовательность из одного предиката или множества предикатов, разделяемых запятыми (связка and) и завершающаяся точкой. С помощью запросов можно установить истинность соответствующего выражения. Предикат запроса называется целью (goal).

Простые запросы, не содержащие переменных, называют да-нет-вопросами. Они допускают лишь два возможных ответа: «Yes» или «No». В случае ответа «Yes» говорят, что запрос завершился успехом, цель достигнута.

При описании конкретной предметной области обычно имеется набор исходных фактов и правдоподобных допущений, на основании которых формулируются правила.

Рассмотрим классическую задачу о семейных отношениях.

Пусть имеются факты об отцовстве:

1) Иван – отец Игоря.

2) Иван – отец Сидора.

3) Сидор – отец Лизы.

Введем также три предиката:

Мужчина (x), означающий, что x – мужчина,

Единокровный (x, y), означающий единокровность x и y,

Брат (x, y), означающий, что x брат y.

Справедливы, очевидно, следующие правила:

1) «Все отцы – мужчины».

2) «Если у детей один отец, то они единокровны».

3) «Брат – это единокровный мужчина».

Рассмотрим вывод решения при ответе на вопрос «Есть ли братья у Игоря?».

```
%  Программа 3.3
domains
        person = symbol
predicates
        otec(person, person)
        man(person)
        brat(person, person)
clauses
        man(X) :- otec(X, _).
        brat(X,Y) :- otec(Z, Y), otec(Z, X), man(X), X<>Y.
        otec(ivan, igor). otec(ivan, sidor). otec(sidor, lisa).
```

Во втором правиле программы указано условие X<>Y. Это позволяет описать тот факт, что человек не может быть собственным братом.

После запроса goal brat(igor, X).

Программа выдаст следующий результат:

X = sidor

Это отвечает нашим представлениям о правильном решении.

Приведенные примеры примитивны, но они позволяют представить неожиданность и полезность решений, которые может сгенерировать Пролог при большом количестве фактов и правил в сложной предметной области.

Список – это упорядоченная структура элементов, причем эле-

ментами могут быть любые термы, включая другие списки.

Пустой список – [ ].

Непустые списки имеют голову и хвост. Они являются компонентами функтора, обозначаемого точкой «.». Например, список из одного элемента a представляется так (a, []), где a – голова и [] – хвост.

На Прологе такой список представляется так [a, b, c], где a – голова, b, c – хвост списка, который и сам является списком.

Например, информация о некоторых семьях может выглядеть как предикат со следующей структурой

семья (фамилия, имя_отца, имя_матери, [список _имен_ детей]).

семья (“Иванов”, “Иван”, “Людмила”, [“Антон”, “Наташа”, “Юрий”]).

В этом случае количество детей не имеет значения, а количество аргументов предиката – семья – всегда одинаково. Поэтому, если записать целевое утверждение

семья (X, Y, Z, Дети).

Переменные конкретизируются так:

X = ”Иванов” Y = ”Иван” Z = ”Людмила”
Дети = [“Антон”, ”Наташа”, ”Юрий”].

На Прологе есть специальная форма представления списка [X|Y].

Если такой спосок сопоставляется с другим списком, то X конкретизируется головой, а Y – хвостом того списка, например,

семья(X, Y, Z, [Det1|Deti].

Тогда переменные конкретизируются так

X = ”Иванов” Y = ”Иван” Z = ”Людмила”
Det1 = ”Антон” Deti = [”Наташа”, ”Юрий”].

В примере 3.4 показана программа, определяющая семью, где

имеется ребенок с именем Алекс.

```
/* Программа 3.4                */
domains
        detlist = person*   % Это список
         person = string
predicates
         member(person, detlist)
        semja(person, person, person, detlist)
clauses
        member(X, [X|_]).  member(X, [_|Y]):-member(X, Y).
        semja("Ivanov", "Ivan", "Ljda", ["Anton", "Nata", "Jri"]).
        semja("Sevs", "Piter", "Vera", []).
        semja("Pospelov", "Serg", "Ira", ["Aleks", "Tanj"]).
goal
        semja(F, I1, I2, Det), member("Aleks", Det),
        write("Aleks", F), nl, write("ОТЕЦ=", I1, "    МАТЬ=", I2).
```

При этом предикат member проверяет принадлежность элемента к списку.

Первый дизъюнкт предиката member означает, что искомый элемент находится в голове списка. Если это не так, то выбирается второй дизъюнкт, который отделяет хвост от головы списка и проверяет, находится ли искомый элемент в хвосте. Это происходит рекурсивно до тех пор, пока либо не найдено, что у остатка списка голова является искомым элементом (удача), либо не окажется, что после отделения хваста от головы остался пустой список ([]), который уже не расщепляется на голову и хвост (неудача).

В Прологе для управления процессом решения задачи реализован механизм поиска с возвратом (backtracking), при котором система пытается отыскать все возможные решения задачи. Механизм вывода программы запоминает те точки процесса унификации, в которых не были использованы все альтернативные решения, а затем возвращается в эти точки и ищет решение по иному пути. Однако поиск с возвратом выполняется автоматически только в тех случаях, когда програм-

ма решает задачу в результате диалога с пользователем. Если же цель указана в разделе goal программы, то поиск оканчивается после нахождения первого решения задачи. Для вывода всех решений используется предикат fail.

Предикат fail называют откатом после неудачи. Он вызывает искусственное неуспешное завершение поиска, что позволяет получить все возможные решения задачи.

```
% Программа 3.5
predicates
        gorod(symbol)
        show
clauses
        gorod(“Харьков”).   gorod(“Минск”).   gorod(“Киев”).
        gorod(“Полтава”).
        show :- gorod(X), write(X), nl, fail.
goal
        write(“Это города:”), nl, show.
```

После запуска будут напечатаны все названия городов.

Предикат отсечения cut обозначается с помощью символа !. Он позволяет получить доступ только к части данных, устраняя дальнейшие поисковые действия. В общем виде можно записать, например,

```
R : – A, B, !, C.
```

Это означает, что если для целей A и B найдено хотя бы одно решение, то дальнейший перебор возможных вариантов значений A и B не нужен и процесс прекращается.

Способность к описанию логических задач является наиболее сильной стороной языка Пролог. Многие логические задачи связаны с рассмотрением нескольких конечных множеств с одинаковым количеством элементов, между которыми устанавливается взаимно-однозначное соответствие. На языке Пролог эти множества можно описывать как базы данных, а зависимости между объектами устанавливать с помощью правил.

3.4. Примеры решения логических задач

Рассмотрим решение простой логической задачи. Беседует трое друзей: Белокуров, Рыжов, Чернов. Брюнет сказал Белокурову: «Любопытно, что один из нас блондин, другой брюнет, третий – рыжий, но ни у кого цвет волос не соответствует фамилии». Какой цвет волос у каждого из друзей?

Для решения построим вспомогательную таблицу 3.1.

Таблица 3.1 – Пример решения

Фамилия / Цвет	Белокуров	Рыжов	Чернов
блондин	–		
рыжий		–	
брюнет	–		–

Можно сделать выводы:

1) Белокуров не брюнет и не блондин;

2) Чернов не черный, цвет волос Чернова и Белокурова не совпадают;

3) Рыжов не рыжий, цвет волос у Рыжова и Белокурова, Рыжова и Чернова не совпадают.

При описании задачи на Прологе получается программа 3.6.

```
% Программа 3.6
predicates
        surname(symbol)
        color(symbol)
        corresponds(symbol, symbol)
        answer
clauses
        surname("Белокуров").  surname("Рыжов").  surname("Чернов").
        color("рыжий").  color("брюнет").  color("блондин").
        corresponds(X, Y) :- surname(X), color(Y), X = "Белокуров",
                                not(Y="брюнет"), not(Y = "блондин").
        corresponds(X, Y) :- surname(X), color(Y), X = "Чернов",
```

```
                    not(Y="брюнет"), not(corresponds("Белокуров",Y)).
    corresponds(X, Y) :- surname(X), color(Y), X = "Рыжов",
                         not(corresponds("Белокуров", Y)),
                         not(corresponds("Чернов", Y)).
    answer :- corresponds(X,Y), write(X," – ", Y),nl, fail.
goal    answer.
```

Решение задачи:
Белокуров – рыжий,
Чернов – блондин,
Рыжов – брюнет.

Рассмотрим решение следующей задачи. В велосипедных гонках три первых места заняли Алеша, Петя и Коля. Какое место занял каждый из них, если Петя занял не второе и не третье место, а Коля – не третье?

Решение задачи можно описать в таблице (прочерки указывают известную информацию):

	1 место	2 место	3 место
Алеша			
Петя		–	–
Коля			–

Очевидно, Петя может занимать только первое место, тогда Коля – только второе, а Алеше достается третье:

	1 место	2 место	3 место
Алеша	–	–	+
Петя	+	–	–
Коля	–	+	–

Пример на Прологе дан в программе 3.7.

```
%    Программа 3.7
predicates
    name(symbol)
    mesto(symbol)
    prizer(symbol, symbol)
```

```
        solution(symbol, symbol, symbol, symbol, symbol, symbol)
clauses
        name(alex).    name(pier).    name(nike).
        mesto(odin).   mesto(dva).    mesto(tri).
        prizer(X, Y) :-
            name(X), mesto(Y), X = pier, not(Y = dva), not(Y = tri);
            name(X), mesto(Y), X = nike, not(Y = tri);
            name(X), mesto(Y), not(X = pier), not(X = nike).
        solution(X1, Y1, X2, Y2, X3, Y3) :-
            name(X1), name(X2), name(X3),
            mesto(Y1), mesto(Y2), mesto(Y3),
            prizer(X1, Y1), prizer(X2, Y2), prizer(X3, Y3),
            Y1 <> Y2, Y2 <> Y3, Y1 <> Y3,
            X1 <> X2, X2 <> X3, X1 <> X3, !.
goal
        solution(X1, Y1, X2, Y2, X3, Y3).
```

Результат работы программы:

```
X1=alex, Y1=tri, X2=pier, Y2=odin, X3=nike, Y3=dva
1 Solution
```

Программа 3.8. решает следующюю задачу. На скамейке сидели Петя, Боря, Коля. Петя справа от Бори, Боря справа от Коли. Кто сидел посередине? Кто сидел с правого (левого) края? Кто сидел между указанными объектами?

```
%     Программа 3.8
predicates
      rayd(symbol, symbol, symbol)
      spravo(symbol, symbol)
      seredina(symbol)
      kr_cl(symbol)
      kr_cpr(symbol)
clauses
      spravo(kolya, boray).       /*Справа от Коли - Боря*/
```

```
    spravo(boray, petay).
    rayd(X, Y, Z) :-  spravo(X, Y), spravo(Y, Z).
    seredina(X) :-  rayd(_, X, _).
    kr_cl(X) :-  rayd(X, _, _).
    kr_cpr(X) :-  rayd(_, _, X).
goal  seredina(X1), kr_cl(X2), kr_cpr(X3).
```

Результат работы программы:

```
X1=boray, X2=kolya, X3=petay
1 Solution
```

Индивидуальные задания

Написать программу на языке Пролог, используя или Turbo Prolog 2.0 или Visual Prolog 5.2.

1) В соревнованиях по бегу Юра, Гриша и Толя заняли три первых места. Какое место занял каждый ребенок, если Гриша занял не второе и не третье место, а Толя не третье?

2) Три подруги вышли в белом, зеленом и синем платьях и туфлях. Известно, что только у Ани цвета платья и туфель совпадали. Ни туфли, ни платье Вали не были белыми. Наташа была в зеленых туфлях. Определить цвета платья и туфель на каждой из подруг.

3) Коля и Саша носят фамилии Шилов и Гвоздев. Какую фамилию носит каждый из них, если Саша и Шилов живут в разных домах.

4) На заводе работали три друга: слесарь, токарь и сварщик. Их фамилии Борисов, Иванов и Семенов. У слесаря нет ни братьев, ни сестер. Он самый младший из друзей. Семенов женат на сестре Борисова, старше токаря. Назвать фамилии слесаря, токаря и сварщика.

5) В бутылке, стакане, кувшине и банке находятся молоко, лимонад, квас и вода. Известно, что вода и молоко не в бутылке, сосуд с лимонадом находится между кувшином и сосудом с квасом, в банке – не лимонад и не вода. Стакан находится около банки и сосуда с молоком. Как распределены эти жидкости по сосудам?

6) Воронов, Павлов, Левицкий и Сахаров – четыре талантливых

молодых человека. Один из них танцор, другой художник, третий – певец, а четвертый – писатель. О них известно следующее: Воронов и Левицкий сидели в зале консерватории в тот вечер, когда певец дебютировал в сольном концерте. Павлов и писатель вместе позировали художнику. Писатель написал биографическую повесть о Сахарове и собирается написать о Воронове. Воронов никогда не слышал о Левицком. Кто чем занимается?

7) Три друга заняли первое, второе, третье места в соревнованиях универсиады. Друзья разной национальности, зовут их по-разному, и любят они разные виды спорта. Майкл предпочитает баскетбол и играет лучше, чем американец. Израильтянин Саймон играет лучше теннисиста. Игрок в крикет занял первое место. Кто является австралийцем? Каким спортом увлекается Ричард?

8) Три девочки Маша, Рита, Лена пошли гулять. На улице было жарко, и они купили мороженое «Белка», «Стрелка», «Гагара». Какое мороженое купила каждая из девочек, если Лена купила не «Белку» и не «Гагару», а Рита – не «Гагару».

9) В комнате находятся Коля, Света, Оля. Каждый из них сидит на отдельной мебели (кровать, стул, диван). Известно, что Коля сидит не на стуле и не на кровати. Света не сидит на стуле. Кто где сидит?

10) На столе лежат ручка, карандаш, фломастер, красного, синего и зеленого цвета. Известно, что ручка лежит между предметом красного и зеленого цвета. Карандаш либо зеленый, либо синий.

11) За круглым столом оказалось пятеро ребят из Киева, Львова, Харькова, Полтавы, Днепра: Денис, Игорь, Иван, Алеша, Сергей. Киевлянин сидел между днепровцем и Сергеем, львовянин – между Денисом и Игорем, а напротив него сидел полтавчанин и Иван. Алеша ни разу не был во Львове, а Денис не бывал в Киеве и Днепре, а днепрянин с Игорем регулярно переписываются. Определите, в каком городе живет каждый из ребят.

12) На улице, встав в кружок, беседует четыре девочки: Аня, Валя, Надя, Галя. Девочка в зеленом платье – не Аня и не Валя – стоит между девочкой в голубом платье и Галей. Девочка в белом платье стоит между девочкой в розовом платье и Валей. Какого цвета платье у каждой из девочек?

13) Трое юношей: Коля, Дима и Юра влюблены в трех девушек: Аню, Лену, Вику. Но эта любовь без взаимности. Коля любит девуш-

ку, влюбленную в юношу, который любит Лену. Дима любит девушку, влюбленную в юношу, который любит Вику. Лена не любит Юру. Кто кого любит?

14) Составить базу знаний по сказке «Репка». Фактами в этой базе должны быть утверждения типа тянет (X, Y). Составить правила, определяющие: кто первый тянет репку, кто последний тянет репку, кто тянет после бабки, кто тянет на четвертом месте.

15) Даны числа X, Y, Z, T. X меньше Y и меньше T; Y больше Z и больше T; Z больше X и меньше T. В каком порядке расположены эти числа?

16) На одной улице стоят 5 домов, окрашенных в 5 разных цветов. В каждом доме живет гражданин одной страны. Каждый из них пьёт свой напиток, курит свои сигареты и содержит своё домашнее животное. Определите, кто из них содержит рыб? Британец живёт в красном доме. У шведа есть собака. Датчанин пьёт чай. Зелёный дом стоит слева от белого и вплотную к нему. Хозяин зелёного дома пьёт кофе. У того, кто курит Pall-Mall, есть птицы. Хозяин желтого дома курит Dunhills. Хозяин среднего дома пьёт молоко. Норвежец живёт в первом доме. Человек, который курит Blends, живёт рядом с хозяином котов. Тот, кто содержит лошадей, живёт рядом с тем, кто курит Dunhills. Тот, кто курит Blue Master, пьёт пиво. Немец курит Prince. Норвежец живёт рядом с синим домом. У того, кто курит Blends, есть сосед, который пьёт воду.

17) Пятеро студентов едут на велосипедах. Их зовут Сергей, Борис, Леонид, Григорий и Виктор. Велосипеды сделаны в пяти городах: Риге, Полтаве, Львове, Харькове и Киеве. Каждый из студентов родился в одном из этих городов, но ни один из студентов не едет на велосипеде, сделанном на его родине. Сергей едет на велосипеде, сделанном в Риге. Борис родом из Риги, у него велосипед из Полтавы. У Виктора велосипед из Киева. У Григория велосипед из Харькова. Виктор родом из Львова. Уроженец Полтавы едет на велосипеде, сделанном на родине Леонида. Кто из студентов родом из Киева?

Содержание отчета

1. Тема занятия.
2. Индивидуальное задание.

3. Описание применяемых методов решения или программа.
4. Результаты выполнения индивидуального задания.
5. Выводы по работе.

Контрольные вопросы

1. Что обозначают разделы domains, predicates, goal и clauses?
2. Как записывается логическое правило с несколькими посылками?
3. Какими способами могут выполняться запросы?
4. Для чего используются анонимные переменные в запросах?
5. С какой целью используется предикат fail?
6. С какой целью используется предикат cut (!)?
7. Какие правила называются рекурсивными?
8. Как описывается момент окончания рекурсивных вызовов?
9. Какой структуре можно поставить в соответствие список?
10. С помощью какой операции описывается рекурсивная обработка списков в Прологе?
11. Чему будет равно R после выполнения программы?

```
predicates
    f_a (unsigned, long, unsigned, long)
    f(unsigned, long)
clauses
        f(N, FN) :-
            f_a(N, FN,1,1).
        f_a(N, FN,I,P):-
            I<=N,!, NewP=P*I, NewI=I+1,
            f_a(N, FN, NewI,  NewP).
        f_a(N, FN,I,P) :-
            I>N,  FN=P.
 goal
      f(3,R), write ("   R=",  R).
```

Лабораторная работа № 4
ЭВОЛЮЦИОННОЕ МОДЕЛИРОВАНИЕ

Цель работы: приобритение и закрепление знаний, и получение практических навыков работы с эволюционными алгоритмами. Ознакомление с методами и подходами эволюционного моделирования на примерах прогнозирования последовательностей с помощью конечных автоматов.

4.1. Эволюционное моделирование как метод решения задач в условиях существенной априорной неопределенности

Словосочетание эволюционное моделирование применяется в двух значениях. Как направление в искусственном интеллекте, в основе которого лежат принципы, заимствованные из эволюционной биологии и популяционной генетики и объединяющие компьютерные методы (генетические алгоритмы, генетическое программирование, метод группового учета аргументов, эволюционное моделирование и их стратегии) для решения задач прогнозирования, оптимизации, апроксимации и др. в различных областях науки и техники [7].

Второе значение, в настоящее время уже почти забытое, связано с названием первого метода моделирования эволюционных процессов.

Моделирование процесса эволюции осуществлялось многократно, но только с появлением работы американских ученых Фогеля, Оуэнса, Уолша в науке об искусственном интеллекте появилось новое направление, заменяющее процесс моделирования интеллекта человека на моделирование процесса эволюции человеческого интеллекта. Творцы нового научного направления рассчитывали при этом на более глубокое изучение важнейших свойств интеллекта, а главное – на получение универсального средства «для синтеза машин, обнаруживающих большую «разумность», чем до сих пор удалось найти в природе». По мнению авторов, разумное поведение – это не что иное, как умение прогнозировать состояние внешней среды для того, чтобы на основе правильного прогноза оптимизировать поведение для достижения

заданных целей. В качестве объектов эволюции выбраны конечные автоматы, на входы которых поступают сигналы из внешней среды, представляющей собой источник последовательных сигналов из конечного алфавита.

На рис. 4.1 представлен конечный автомат *M* с тремя состояниями *A, B, C.* Алфавит входных символов автомата состоит из символов *a*, *b*, *c*, выходных – из α, β, γ (входные символы приведены слева от наклонных черт, а выходные – справа, стрелкой отмечено начальное состояние автомата).

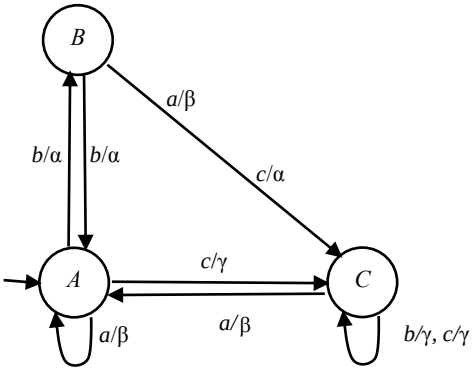

Рис. 4.1 Конечный автомат *M*

Рассмотрим работу автомата при поступлении на его вход последовательности символов *b, a, c, c, a, b, b, a, c, a*. Входной символ *b*, поданный на вход автомата в начальном состоянии *A*, приводит к появлению на выходе автомата символа α и переходу автомата в состояние *B.*

Следующий входной символ приходит на вход автомата только после того, как в автомате закончатся все переходные процессы. Второй символ *a* входной последовательности переводит автомат в состояние *C* с появлением на выходе автомата символа β.

В табл. 4.1 дана последовательность изменения состояний автомата и выходных символов при заданной входной последовательности.

Таблица 4.1 – Изменения состояний автомата

Текущее состояние	*A*	*B*	*C*	*C*	*C*	*A*	*B*	*A*	*A*	*C*
Входной символ	*b*	*a*	*c*	*c*	*a*	*b*	*b*	*a*	*c*	*a*
Следующее состояние	*B*	*C*	*C*	*C*	*A*	*B*	*A*	*A*	*C*	*A*
Выходной символ	α	β	γ	γ	β	α	α	β	γ	β

Отсюда видно, что автомат преобразует последовательность входных символов в последовательность выходных символов. При прогнозировании состояний внешней среды автомат должен иметь одинаковые входной и выходной алфавиты. Он может прогнозировать как следующий символ входной последовательности, поступающий на следующем такте работы автомата, так и любой другой символ с заданным числом тактов упреждения. Рассмотрим случай, когда автомат *M* прогнозирует следующий символ входной последовательности. При этом полагаем, что $a \equiv \alpha$, $b \equiv \beta$, $c \equiv \gamma$, а стоимость ошибки при любом неправильном предсказании следующего входного символа равна единице и равна нулю при правильном предсказании. При функционировании автомата видно (табл. 4.2), что автомат правильно предсказал четыре входных символа и сделал пять ошибок.

Таблица 4.2 – Функционирование автомата

Текущее состояние	*A*	*B*	*C*	*C*	*C*	*A*	*B*	*A*	*A*	*C*	
Входной символ	*b*	*a*	*c*	*c*	*a*	*b*	*b*	*a*	*c*	*a*	
Выходной символ		*a*	*b*	*c*	*c*	*b*	*a*	*a*	*b*	*c*	*b*
Стоимость ошибки		0	1	0	1	0	1	0	1	1	

В эволюционном моделировании потомки автоматов получают путем случайных мутаций исходного автомата – добавляют или убирают одно или несколько состояний конечного автомата, изменяют начальное состояние автомата, изменяют конечное состояние у одного или нескольких ребер, входные и (или) выходные символы, соответствующие случайно выбранным ребрам автомата, и так далее. Полученные таким образом потомки оценивают на той же последовательности, и если они превосходят родительский автомат по заданному критерию, то один или несколько лучших автоматов оставляют для последующей работы, а остальные автоматы, включая и родительский, отбрасывают. Процесс мутаций повторяется до тех пор, пока не будет достигнуто определенное значение критерия качества функционирова-

ния автомата на заданной предыстории. Полученный в результате такого процесса автомат может использоваться для предсказания среды в реальном времени.

Синтез прогнозирующих автоматов средствами эволюционного моделирования можно рассматривать и как синтез предсказывающих фильтров, поиск параметров которых ведется методом случайного поиска. Известно, что эволюционирующие фильтры обычно имеют детерминированную часть, связанную с анализом причин и следствий, которую можно рассчитать, и индетерминированную часть – случайный корректор.

Разумное сочетание детерминированных и индетерминированных вычислений позволяет существенно повысить эффективность алгоритмов эволюционного моделирования и расширить область их применения. В частности, это позволяет во многих случаях применять и более гибкий подход к эволюционному процессу, который дает возможность использовать конечные автоматы или непрерывные регуляторы в контурах управления объектов, находящихся в изменяющейся внешней среде. Он предполагает иерархическую двухуровневую процедуру, блок-схема которой приведена на рис. 4.2, где 1 – алгоритм структурной адаптации; 2 – алгоритм использования; 3 – алгоритм параметрической адаптации.

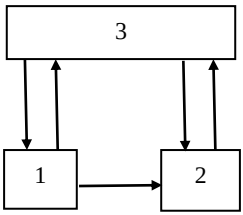

Рис. 4.2 Двухуровневая процедура синтеза регуляторов

На первом уровне постоянно протекают два процесса: процесс использования одного из k хранящихся в памяти синтезированных регуляторов в контуре управления и процесс синтеза множества моделей регуляторов.

При ухудшении качества работы функционирующего регулятора на i-м этапе он заменяется одним из синтезированных на $(i–1)$-м этапе регулятором либо одним из регуляторов, хранящихся в памяти. На

(i+1)-м этапе функционирует новая модель регулятора и опять выполняется синтез множества моделей регуляторов. Поскольку синтез эффективной структуры регулятора во многом зависит от таких параметров эволюционного процесса, как список возможных мутаций адаптируемых моделей, вероятности появления тех или иных режимов изменения, длины предыстории или объема используемой памяти на этапе адаптации и других параметров, то необходим алгоритм, который бы адаптировал параметры эволюционного процесса адаптации моделей регуляторов. Такая адаптация выполняется с помощью алгоритма параметрической адаптации, который фактически адаптирует алгоритмы случайного поиска моделей регуляторов по поступающей из внешней среды информации и результатам функционирования регуляторов в контуре управления.

Адаптация случайного поиска существенно повышает эффективность алгоритмов эволюционного синтеза моделей, однако, она не может полностью компенсировать отсутствие комплексного сочетания детерминированных и индетерминированных вычислений. Отсутствие такого сочетания приводит к тому, что там, где были бы эффективны простые детерминированные процедуры (например, модель находится на склонах «холма» или «пика» с глобальным экстремумом в многоэкстремальном пространстве возможных моделей и для получения лучшего решения необходимо только оптимизировать один-два параметра модели) применяется трудоемкий случайный поиск (при синтезе конечных автоматов генерируются автоматы во всем пространстве возможных решений, а не только на склонах «холма» или «пика», на которых находится лучший синтезированный на данный момент автомат). Это часто чрезмерно увеличивает время счета, а следовательно, и ограничивает область применения эволюционного моделирования.

Для решения задач синтеза или адаптации автоматов в случае предельной априорной неопределенности лучше вводить универсальные алгебры $A(M, D)$, отражающие специфику решаемой задачи. Здесь M – множество возможных переходов $S_k^{i\alpha} S_l^{i\beta}$ конечных автоматов $B_i = \left\langle X^i, Y^i, S^i, \varphi_Y^i, \varphi_S^i \right\rangle$ из состояния $S_k^i \in S^i$ в состояние $S_l^i \in S^i$ при

заданных конечных входном X^i и выходном Y^i алфавитах ($\alpha \in X^i$, $\beta \in Y^i$); $i = \overline{1,n}$ – индекс, указывающий на принадлежность к *i*-му автомату; S^i – конечное множество состояний; $y_j^i = \varphi_Y^i\left(x_j^i, S_j^i\right)$ – функция выходов; $S_{j+1}^i = \varphi_S^i\left(x_j^i, S_j^i\right)$ – функция переходов; y_j^i, x_j^i и S_j^i – соответственно выходной символ, входной символ и состояние *i*-го конечного автомата B_i в момент времени t_j, $j = i, 2, \ldots$; D – множество операций алгебры, которая должна содержать операции, позволяющие наиболее просто решать задачу синтеза или адаптации автомата из некоторого множества $W = \left(B_1, B_2, \ldots, B_n\right)$ исходных автоматов при заданном множестве I исходных данных.

При заданной последовательности входных сигналов x_1^i, x_2^i, … , x_l^i выходные сигналы (реакции) y_j^i, $j = \overline{1,l}$, объектов B_i, $i = \overline{1,n}$, переводят наборы (I, B_1, B_2, …, B_n) в информационную матрицу $\left\|\beta_{kj}\right\|$, $k = \overline{1,n}$, $j = \overline{1,l}$, где $\beta_{kj} = P_j\left(B_k\right)$, $P = (P_1, P_2, \ldots, P_l)$ – множество предикатов, определенных на множестве автоматов, $P_q = P_q(B)$, $q = \overline{1,l}$. Строка $(\beta_{i1}, \beta_{i2}, \ldots, \beta_{il})$ информационной матрицы) $\left\|\beta_{kj}\right\|_{nxl}$; является информационным вектором автомата B_i и составлена из элементов 0 и 1, т. е. из значений предикатов, вычисленных при функционировании автомата. Если все элементы строки являются единицами, то автомат называется корректным для решаемой задачи Z. На множестве значений выходных функций вводятся метрики $\rho = (\rho_1, \rho_2, .., \rho_\sigma)$, с помощью которых путем введения множества $\Phi = (\Phi_1, \Phi_2, \ldots, \Phi_r)$ функционалов на множестве значений выходных функций автоматов, оценивается отклонение функционирования некорректных автоматов от функционирования корректных. Если при функционировании автомата выполняются условия $\Phi_l \le \varepsilon_l$, $l = \overline{1,r}$, где $\varepsilon_l \ge 0$ – некоторые наперед заданные величины, то он называется автоматом с заданной мерой некорректности функционирования (при $\Phi_l = \varepsilon_l = 0$, $l = \overline{1,r}$ имеем

корректный автомат). При рациональном выборе множеств D, P, ρ, Φ обычно существуют признаки возможности или невозможности синтеза корректных автоматов или автоматов с заданной мерой некорректности функционирования из заданного множества W исходных объектов и множества I исходных данных. При наличии указанных признаков задача получения объекта для решения некоторой задачи Z выполняется следующим образом.

На первом этапе стандартным методом получается множество W^* автоматов, из которых по указанным признакам для данной задачи Z из множества W^* с помощью множества операций D алгебры $A(M, D)$ может быть получен корректный автомат (или автомат с заданной мерой некорректности).

На втором этапе выполняется композиция или преобразование полученных объектов, и для последующей работы отбирается подмножество лучших (в смысле заданного множества функционалов) автоматов, такое, что для данной задачи Z из них может быть получен корректный автомат (автомат с заданной мерой некорректности).

Затем выполняется композиция или преобразование вновь полученных автоматов и так далее – до тех пор, пока не будет получен один или подмножество автоматов, каждый из которых решает заданную задачу Z. При решении задачи адаптации автомата в исходное множество W^* включается и ранее функционировавший автомат.

Успешный синтез конечного автомата, а также затраты на его осуществление при использовании как алгебраического подхода, так и классических алгоритмов эволюционного моделирования, во многом зависят и от удачного определения исходного множества автоматов B^*, из которых для данной задачи с помощью выбранного алгоритма можно синтезировать корректный (или с заданной мерой некорректности функционирования) автомат.

В случае больших обучающих последовательностей и использования K-значных ($K >> 2$) входных и выходных алфавитов этап получения исходного множества автоматов может быть чрезвычайно трудоемким в связи с тем, что при его выполнении будет синтезировано множество бесперспективных частных описаний различной сложно-

сти, которые затем будут опробованы на обучающих выборках и отброшены как работающие неудовлетворительно. В то же время простой предварительный анализ исходных данных может существенно сократить необходимые переборы частных описаний. Действительно, если, например, имеется обучающая последовательность, содержащая группу из j символов α_i и m пар символов $\alpha_k\alpha_u$ ($u=\overline{1,m}$),

$$\alpha_1\alpha_2\alpha_3\underbrace{\alpha_i\ldots\alpha_i}_{j\text{ раз}}\alpha_k\alpha_1\ldots\alpha_k\alpha_2\ldots\alpha_k\alpha_3\ldots\alpha_k\alpha_m\alpha_p\ldots,$$

и требуется синтезировать автомат, который безошибочно предсказывает все символы обучающей последовательности, то такой автомат должен иметь число состояний, которое определяется большим из двух чисел j и m, и следовательно, на первом ряду селекции в множество B^* необходимо сразу вводить автоматы с $N = \max(j, m)$ состояниями, не рассматривая более простые автоматы. Для определения числа m необходим просмотр всех пар, следующих друг за другом входных символов, а для определения числа j – анализ групп входных символов, каждая из которых содержит j символов. В общем случае анализ входной последовательности можно представить следующим образом.

Имеется входная последовательность $I = \{i_1, i_2, ..., i_n\}$ из алфавита $A = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ входных символов и имеется множество предикатов $P = \{P_1, P_2, ..., P_l\}$, которые характеризуют неоднородность участия символов алфавита A в образовании различных комбинаций символов во входной последовательности. Наличие или отсутствие интересующего нас свойства R_q входной последовательности будем описывать предикатом P_q, аргументами которого являются группы или отдельные символы входного алфавита A:

$$P_q(\alpha_k,\ldots,\alpha_r)=\begin{cases}1, \text{если } \{i_1,i_2,\ldots,i_n\} \text{ обладает свойством } R_q,\\ 0 - \text{в противном случае.}\end{cases}$$

Рассмотрим, например, неоднородность участия в образовании входной последовательности символов α_k и α_r. Искомую неоднород-

ность можно характеризовать числами n_k, n_r вхождения каждого из символов во входную последовательность I и представить в виде диагональной матрицы F_1, в которой диагональные элементы $\alpha_{ii} = n_i, i = \overline{1,m}$. Максимальное число $n_{i\max} = \max\limits_{1,m} \alpha_{ii}$ на диагонали матрицы F_1 дает грубую оценку максимально необходимого числа состояний корректного конечного автомата, безошибочно предсказывающего по текущему символу следующий символ входной последовательности.

Диагональные элементы матрицы F_1 можно использовать и для грубой оценки числа n_c состояний автоматов с заданной мерой некорректности. Например, если задано, что число ошибок предсказания автомата не должно превышать $N_Э$, то используя соотношение

$$N_Э = \sum_{i=1}^{m} \varphi_i(n_i - n_c), \tag{4.1}$$

где

$$\varphi_i(n_i - n_c) = \begin{cases} n_i - n_c, \text{если } n_i \geq n_c, \\ 0, \text{если } n_i < n_c, \end{cases}$$

можно получить примерное значение числа n_c.

Видимо, оценки числа состояний корректных автоматов или автоматов с заданной мерой некорректности будут более точными, если наряду с числами n_i, $i = \overline{1,m}$, определять число пар входных символов, содержащих один из символов α_k или α_r, а также число упорядоченных пар $\alpha_k\alpha_r$ или $\alpha_r\alpha_k$. В общем случае, когда индексы k и r изменяются от единицы до m, можно построить частотную матрицу F_2 вида

$$F_2 = \begin{vmatrix} n_{11} & n_{12} & \dots & n_{1m} \\ n_{21} & n_{22} & \dots & n_{2m} \\ \dots & \dots & \dots & \dots \\ n_{m1} & n_{m2} & \dots & n_{mm} \end{vmatrix}, \tag{4.2}$$

где n_{ij} – число упорядоченных пар $\alpha_i\alpha_j$ символов во входной последовательности.

Вместо матрицы (4.2) иногда удобнее рассматривать матрицу F_{2p} из значений предикатов:

$$F_{2p} = \begin{vmatrix} P_{11} & P_{12} & \dots & P_{1m} \\ P_{21} & P_{22} & \dots & P_{2m} \\ \dots & \dots & \dots & \dots \\ P_{m1} & P_{m2} & \dots & P_{mm} \end{vmatrix}, \qquad (4.3)$$

где

$$P_{ij}(n_{ij}) = \begin{cases} 1, \text{если } n_{ij} \neq 0, \\ 0, \text{если } n_{ij} = 0, \quad i, j = \overline{1,m}. \end{cases} \qquad (4.4)$$

Наибольшее число $n_{\max}$, отличных от нуля элементов в строке F_{2pi} матрицы (4.3) или в соответствующей строке матрицы F_{2i} (4.2), определяет нижнюю границу числа состояний конечного автомата, необходимых для правильного предсказания всех пар символов, в том числе и всех пар $\alpha_i\alpha_p$ ($p \in \overline{1,m}$) с наиболее разнообразно входящим в последовательность I символом α_i . Минимально необходимое для правильного предсказания всех символов число состояний автомата зависит не только от N_{max}, но и от величины максимального элемента $n_{\max} = \max_{q,j}(n_{qj})$ матрицы F_2, поскольку в худших случаях (например, все пары элементов $\alpha_q\alpha_j$ следуют одна за другой) для безошибочного предсказания необходимо $n_{\max}$ состояний конечного автомата. Однако в других случаях, например, при циклической входной последовательности, когда группы пар символов $\alpha_q\alpha_j$ разделены одинаковыми подпоследовательностями символов, величина максимального элемента матрицы может существенно превышать минимально необходимое число состояний конечного автомата, требуемое для безошибочного воспроизведения входной последовательности. Такая неопределен-

ность в оценке необходимого числа состояний конечного автомата требует дальнейшего дополнения характеристик входной последовательности и рассмотрения неоднородности участия троек, четверок, ..., *L*-ек символов в образовании свойств входной последовательности.

Рассмотрим *L*-мерную матрицу $F_L = F(n_{i1,\,i2,\,\ldots,iL})$, $i_1, i_2, \ldots, i_L = \overline{1,m}.$

Места на каждой стороне матрицы пронумеруем числами от единицы до *m*. Каждому числу $j = \overline{1,m}$ поставим в соответствие символ α_j алфавита *A*, а каждому элементу $n_{i1,\,i2,\,\ldots,iL}$ – число упорядоченных *L*-ек символов, в которые входят символы алфавита, соответствующие номерам индексов $i_1, i_2, \ldots i_L \in \overline{1,m}$ выбранного элемента $n_{i1,\,i2,\,\ldots,iL}$ матрицы. Построенную таким образом матрицу называют *L*-мерной частотной матрицей входной последовательности.

Наличие *L*-мерной частотной матрицы F_L позволяет уточнить нижнюю оценку необходимого числа состояний конечных автоматов. Например, в случае, когда синтезируется автомат, который должен точно предсказывать все символы обучающей последовательности, наличие хотя бы одного, отличного от нуля элемента $n_{i1,\,i2,\,\ldots,iL}$, когда $i_1 = i_2 = \ldots = i_L$ и $i_1 \in \overline{1,m}$, указывает, что минимально необходимое число состояний конечного автомата в общем случае не меньше *L*.

Вместо *L*-мерных матриц можно использовать более наглядные двухмерные матрицы, пронумеровав их столбцы и строки таким образом, чтобы в них в удобной форме были перечислены все элементы *L*-мерной матрицы.

Отметим, что, по аналогии с матрицей (4.3), вводятся многомерные или соответствующие им двухмерные матрицы предикатов для *L*-ек символов.

Частотные матрицы целесообразно использовать и при синтезе конечных автоматов с заданной мерой некорректности функционирования.

Пример 4.1. Пусть задана последовательность из 22 символов:

$\alpha_1\ \alpha_2\ \alpha_2\ \alpha_2\ \alpha_3\ \alpha_1\ \alpha_1\ \alpha_4\ \alpha_2\ \alpha_2\ \alpha_2\ \alpha_4\ \alpha_1\ \alpha_4\ \alpha_5\ \alpha_2\ \alpha_2\ \alpha_2\ \alpha_1\ \alpha_3\ \alpha_1\ \alpha_5$.

Необходимо синтезировать конечный автомат, точно предсказывающий по текущему символу следующий символ входной последовательности. Матрицы F_1, F_2 в этом случае имеют вид

$$F_1 = \begin{vmatrix} 6 & 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{vmatrix}, \qquad F_2 = \begin{vmatrix} 1 & 1 & 1 & 2 & 1 \\ 1 & 6 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{vmatrix}.$$

Максимальный элемент матрицы F_1 указывает на то, что число состояний конечного автомата не превышает девяти, а из пяти ненулевых элементов первой строки матрицы F_2 следует, что автомат, безошибочно предсказывающий всю входную последовательность, не может иметь менее пяти состояний.

Из величины максимального элемента второй строки матрицы F_2 и общего числа символов α_2 (максимальный элемент матрицы F_1) без анализа троек символов не удается снизить верхнюю границу числа состояний конечного автомата. Анализ троек символов с помощью матрицы

$$F_3 = \begin{array}{c|cccccccccccc} & \alpha_1\alpha_1 & \alpha_1\alpha_2 & \alpha_1\alpha_3 & \alpha_1\alpha_4 & \alpha_1\alpha_5 & \alpha_2\alpha_1 & \alpha_2\alpha_2 & \dots & \alpha_3\alpha_1 & \dots & \alpha_4\alpha_1 & \dots \\ \alpha_1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & \dots & 1 & \dots & 0 & \dots \\ \alpha_2 & 0 & 0 & 1 & 0 & 0 & 1 & 3 & \dots & 1 & \dots & 1 & \dots \\ \alpha_3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 & \dots & 0 & \dots \\ \alpha_4 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & \dots & 0 & \dots & 0 & \dots \\ \alpha_5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \dots & 0 & \dots & 0 & \dots \end{array}$$

показывает, что имеется три тройки символов $\alpha_2\ \alpha_2\ \alpha_2$, отделенные друг от друга различными группами символов (все тройки символов $\alpha_2\ \alpha_2\ \alpha_k$, ($k$ = 1, 3, 4) различны). Поэтому число состояний автомата, безошибочно предсказывающего всю входную последовательность, не может быть меньше:

$$n = k_i\, n^*,$$

где k_i – число групп символов $\alpha_2\ \alpha_2\ \alpha_2$; n^* – число состояний конечного автомата, необходимых для безошибочного предсказания последовательности $\alpha_2\ \alpha_2\ \alpha_2\ \alpha_k$ (максимальный элемент матрицы F_3). Отсюда следует, что $n = 9$ и что автомат должен состоять из трех подавтоматов, правильно предсказывающих соответственно подпоследовательности символов $\alpha_2\ \alpha_2\ \alpha_2\ \alpha_1$; $\alpha_2\ \alpha_2\ \alpha_2\ \alpha_3$; $\alpha_2\ \alpha_2\ \alpha_2\ \alpha_4$. Такая информация на несколько порядков сокращает объем необходимых вычислений.

Пример 4.2. Пусть задана входная последовательность из примера 4.1 и требуется оценить число N ошибочных предсказаний автомата с пятью состояниями.

Из соотношения (4.1) и матрицы F_1 примера 4.1 следует, что

$$N = \sum_{i=1}^{5} \varphi_1(n_i - 5) = 1 + 4 = 5.$$

Анализ матрицы F_2 показывает, что автомат с пятью состояниями, предсказывая вторые элементы пар символов $\alpha_1\alpha_k$ ($k = \overline{1,5}$) может не допустить ни одной ошибки, т. е. верхняя оценка числа неправильных предсказаний, как будто может быть уменьшена на единицу.

Однако более глубокий анализ – анализ матрицы F_3 показывает, что тройки символов $\alpha_1\ \alpha_4\ \alpha_2$ и $\alpha_1\ \alpha_4\ \alpha_5$ можно правильно предсказать только в том случае, когда символ α_4 в них предсказывается разными состояниями автомата. Это свидетельствует о том, что величина оценки по первой строке матрицы F_1 числа ошибок с помощью соотношения (4.1) не может быть уменьшена.

Совместный анализ матриц F_2 и F_3, проведенный в примере 4.1, показывает, что девять состояний автомата, безошибочно предсказывающего все символы входной последовательности, обусловливаются наличием трех групп символов $\alpha_2\ \alpha_2\ \alpha_2\ \alpha_k$ (k = 1, 3, 4), для правильного предсказания каждой из которых требуется три состояния конечного автомата. Следовательно, автомат с пятью состояниями на подпо-

следовательностях, начинающихся с символа α_2, должен делать не менее двух ошибок. Поскольку величина оценки числа ошибок по третьей–пятой строкам матрицы F_1 с помощью соотношения (4.1) имеет нулевое значение, то весьма вероятно существование автомата с пятью состояниями, делающего всего три неправильных предсказания на заданной входной последовательности. Автомат, имеющий пять состояний и делающий всего три неправильных предсказания, действительно удается синтезировать.

Частотный метод анализа необходимо дополнять хотя бы простейшими элементами анализа во временной области. Дело в том, что важно не только наличие, например L-ки $\alpha_k \ \dots \ \alpha_k$ символов, но и где эта L-ка расположена во входной последовательности: если в начале или в середине входной последовательности, то для безошибочного предсказания всех символов α_k и следующего за ними символа необходимо L состояний конечного автомата, а если в конце последовательности, то безошибочное предсказание может быть выполнено одним состоянием конечного автомата.

Другой факт из пространственно-временного анализа, позволяющий часто во много раз сократить объем необходимых вычислений. Если входная последовательность имеет длину N символов и выполняется синтез предсказывающего автомата, то теоретически нет смысла рассматривать конечные автоматы с более чем $(N-1)$ переходами, а практически – более чем с $(0{,}5-0{,}7)(N-1)$ переходами. Следовательно, при алгебраической форме представления автоматов необходимо оценивать только автоматы, содержащие не более $(0{,}5-0{,}7)\,(N-1)$ одночленов, что ведет к резкому сокращению числа возможных автоматов-претендентов.

4.2. Работа с программой эволюционного синтеза конечных автоматов

Программа «Darvin» моделирует конечный автомат, задачей которого является имитация (прогнозирование некоторой входной по-

следовательности). Это достигается путем построения конечного автомата на вход которого поступает заданная последовательность (по умолчанию двоичная, но основание системы счисления может быть изменено), а на выходе получается последовательность, максимально приближенная ко входной. Т.е. после генерации конечного автомата, который предсказывает следующий символ входной последовательности, подав на его вход первый символ входной последовательности, на выходе получим спрогнозированный второй символ входной последовательности. Подав второй символ входной последовательности – получим спрогнозированный третий символ входной последовательности и т.д.

Перед началом работы программы задать основные константы программы:

MaxState – максимально возможное количество состояний конечного автомата, исходное значение 10;

Radix – основание системы счисления входной последовательности, исходное значение 2;

IdealKoef – коэффициент, задающий процент правильных предсказаний на входной последовательности синтезируемого автомата, исходное значение 0,9 (изменить на 1);

ValLen – длина входной последовательности, (задано 10).

При запуске программы на выполнение появляется меню:

1. Пошаговый режим …… Нет
2. Случайные мутации …… Да
3. Задание вероятностей мутаций ……... Нет
4. Задание входной последовательности …….. Нет
5. Периодический запрос подтверждения …… Нет
6. Задание исходного автомата …… Нет

Enter Начало генерации конечного автомата ……..

Esc Выход

Для изменения заданного в меню режима работы программы необходимо нажать на клавиатуре цифру соответствующего режима – это меняет содержимое правого столбца меню с «Нет» на «Да» или наоборот.

Пошаговый режим работы программы предусматривает вывод на экран полученного автомата после каждой мутации. Для его задания необходимо ввести «I», после чего в первой строке меню появится «Да» вместо «Нет». Режим может отменяться.

В программе предусмотрено пять типов мутаций конечных автоматов: добавление состояния, удаление состояния, изменение ребер автоматов, изменение выходных символов связей автоматов, изменение начального состояния. Вероятность применения каждого вида мутаций может быть определена пользователем в режиме «Задание вероятностей мутаций», в противном случае используются значения вероятностей, заданные в программе.

В четвертом режиме работы программы может быть задана необходимая входная последовательность, параметры которой (длина и алфавит) задаются соответственно константами программы ValLen и Radix. Константа Radix задает алфавит посредством определения основания системы счисления входной последовательности. Например, при Radix = 2 входной алфавит состоит из символов {0, 1}, а при Radix = K – из символов (0, 1, ..., K – 1).

В пятом режиме может быть задано число мутаций, после которого программа запрашивает о целесообразности дальнейшего поиска конечного автомата:

```
5
1. Пошаговый режим ……Нет
5. Периодически запрашивать подтверждение ….Да
Esc   Выход
Enter

Введите число мутаций, после которого запрашивать
подтверждение
1000 Enter
Please wait ...
Произошло 1000 мутаций. Продолжить? (Y/N)
Y
Произошло 2000 мутаций. Продолжить? (Y/N)
N
```

После этого на экране появляется лучший конечный автомат, по-

лученный после 2000 мутаций, входная и выходная последовательности и процент правильных предсказаний символов входной последовательности.

Режим 6 используется в тех случаях, когда необходимо задать исходный автомат. Для начала работы этого режима необходимо ввести «6», после чего в шестой строке меню вместо «Нет» появится «Да», а затем Enter и на экране появится меню

1. Добавить состояние и выходы из него
2. Удалить последнее состояние
3. Перебросить связь на другое состояние
4. Изменить выходной символ связи
5. Принять другое состояние в качестве начального
ESC Принять этот автомат за исходный

Пусть после двукратного нажатия «1» получим на экране автомат

```
    1     2
1  0/1   -
   1/0
2  1/0  0/0
Начальное состояние: 1
```

Для демонстрации работы программы в режимах 3, 4 и 5 предположим, что необходимо заменить связь 1/0 из состояния 1 в состояние 1 на связь 1/1 из состояния 1 в состояние 2 и изменить при этом начальное состояние автомата с первого на второе. Вначале изменим начальное состояние, а затем – выходной символ:

```
5
Введите номер начального состояния
2 Enter
4
Изменить выходной символ связи:
из   1 Enter
в    1 Enter
номер связи (0-1)
1 Enter
```

```
выходной символ:    1 Enter
Все введенное Вами правильно? (Y/N)
Y
```

Если информация введена неправильно, то после ввода «N» программа предоставляет возможность повторить ввод.

Для изменения конечного состояния связи 1/1 в первой вершине автомата используем режим 3:

```
3
Связь:
из   1 Enter
в   1 Enter
номер (0-1)   1 Enter
Перебросить в   2 Enter
Все введенное Вами правильно? (Y/N)
Y
```

В результате на экране получим автомат:

```
    1     2
1  0/1  1/1
2  1/0  0/0
Начальное состояние: 2
```

Индивидуальные задания

В соответствии с индивидуальным заданием (табл. 4.3), проанализировать входную последовательность с помощью частотных матриц и использовать полученную информацию для задания исходного автомата. Определить влияние изменения вероятностей появления различного вида мутаций на число автоматов, синтезируемых в процессе поиска заданного автомата и процент правильных предсказаний лучших автоматов как функцию общего числа синтезированных автоматов. Сравнить результаты синтеза автомата с использованием априорной информации и без нее.

Таблица 4.3 – Индивидуальные задания

Номер варианта	Значность алфавита	Длина входной последовательности	Макс. кол-во рядом стоящих одинаковых симво-	Кол-во групп одинаковых сим-	Частотные матрицы
1	2	10	2	3	F_2, F_3
2	2	11	3	2	F_2
3	2	12	5	2	F_1
4	2	13	5	2	F_3
5	2	14	3	3	F_2
6	2	15	4	2	F_1, F_2
7	2	16	3	4	F_2
8	2	17	4	3	F_2, F_3
9	2	19	2	6	F_3
10	2	20	2	7	F_3
11	2	21	3	5	F_1, F_3
12	3	16	4	3	F_1, F_3
13	3	17	5	2	F_2
14	3	18	3	4	F_3
15	3	19	4	3	F_1
16	4	20	7	2	F_1, F_3
17	4	24	3	6	F_3
18	4	25	6	3	F_2, F_3
19	4	21	3	4	F_1, F_3
20	5	24	4	4	F_1, F_2

Содержание отчета

1. Тема и цель занятия.

2. Индивидуальное задание.

3. Результаты выполнения задания.

4. Нарисуйте схему графа автомата, синтезированного для вашей входной последовательности.

5. Выводы по работе.

Контрольные вопросы

1. Что такое конечный автомат?

2. Какие могут быть заданы мутации при синтезе автомата?

3. Какое количество мутаций необходимо для синтеза вашего автомата?

4. Одинаковые ли автоматы синтезируются при случайном характере мутаций?

5. Перечислите от чего будет зависеть синтезируемый автомат и его вид.

6. Нарисуйте автомат для входной последовательности: 1, 1, ..., 1.

7. Как изменится число состояний автомата, если уменьшать вероятность изменения числа состояний автомата?

8. Как использовать программу для поиска конечного автомата с заданным числом состояний автомата?

9. Приведите примеры практического применения конечных автоматов.

Лабораторная работа № 5
МЕТОД ГРУППОВОГО УЧЕТА АРГУМЕНТОВ

Цель работы: ознакомление с методом группового учета аргументов и получение практических навыков работы с алгоритмами метода самоорганизации математических моделей.

5.1. Основные типы задач, решаемых алгоритмами МГУА

Решение задач синтеза математических моделей реальных технических систем представляет собой сложный процесс, сочетающий умение глубоко разобраться в физике процессов исследуемого объекта и правильно поставить задачу с поиском оптимальных структур и параметров математических моделей с помощью компьютера. Часто оказывается невозможно традиционным подходом с участием человека довести модель до конечных соотношений из-за отсутствия информации о существенных характеристиках воздействия среды или о физике процессов в моделируемом объекте. В таких случаях одним из наиболее подходящих методов для синтеза математических моделей является метод самоорганизации математических моделей (или метод группового учета аргументов (МГУА)). Этот метод искусственного интеллекта как раз и предназначен для автоматического синтеза моделей в условиях существенной априорной неопределенности, когда традиционный подход с участием человека неработоспособен [7].

С помощью алгоритмов метода группового учета аргументов (или метода самоорганизации математических моделей) решается большое число разнообразных задач синтеза математических моделей. Рассмотрим несколько основных типов этих задач, где успешно применяются алгоритмы метода группового учета аргументов (МГУА). При этом первую задачу сформулируем не в классическом виде, а в терминах теории равновесных состояний.

Задача 5.1. Пусть векторная вещественная функция

$\varphi(y) = \{\varphi^1(y), \ldots, \varphi^q(y)\}$, $y \in \Omega \in E_n$ задана своими приближенными значениями $\bar{\varphi}_y$ в ограниченном числе точек y_v, $v = \overline{1,r}$, $y_v \in Y_0 \in \Omega$. Необходимо найти приближенное значение $\varphi(y_i)$ в любой точке y_i в виде

$$\varphi^i(y) = \sum_{k=0}^{m} a_k^i g_k(y), \ i = 1, 2, \ldots, q, \tag{5.1}$$

где Ω область определения функций g_k, φ^i; E_n – n-мерное векторное пространство; $g_k, k = \overline{1,m}$ – заданные или выбранные функции; Y_0 – множество, содержащее r исходных точек; $A_1 = \left\{a_0^1, \ldots, a_m^1, a_0^2, \ldots, a_m^2, \ldots, a_0^q, \ldots, a_m^q\right\}$ – вектор искомых коэффициентов.

Решение $z = (A_1^*, y^*)$, $A_1^* \in A(Y_1)$, $y^* \in Y_2$, $Y_1, Y_2 \in Y_0 \in \Omega$, $Y_1 \cup Y_2 = Y_0$, $Y_1 \neq Y_2$ ищется в предположении, что функция $\varphi_1(A_1, y)$, характеризующая точность модели на множествах Y_1 и y^*, и функция $\varphi_2(A_2, y)$, характеризующая ошибку модели на всем множестве Y_0 исходных данных и фактически определяющая подмножество точек $y^* \in Y_0$, в которых модель $\varphi(y)$ имеет максимальную погрешность, удовлетворяют следующим уравнениям

$$\varphi_1(A_1^*, y^*) = \max_{A_1 \in A(Y_1)} \varphi_1(A_1, y^*), \tag{5.2}$$

$$\varphi_2(A_1^*, y^*) = \max_{y_0 \in Y_2} \varphi_2(A_1^*, y_0), \tag{5.3}$$

где A – область допустимых значений вектора коэффициентов; Y_1 – множество точек исходных данных, на которых определяется вектор A_1 коэффициентов модели; y_0 – множество точек, на которых определяется ошибка моделирования исходных данных; Y_2 – множество точек исходных данных, на которых может определяться погрешность моде-

ли, полученной на множестве Y_1.

В качестве примера функции φ_1 могут служить функции вида $\varphi_1 = C - K,$ где C – положительная константа, K – заданный критерий, причем, всегда выполняется условие $C \geq K$. Приведем примеры возможных критериев.

Критерий регулярности [7]

$$\Delta^2(Y_2) = \frac{\sum_{i=1}^{N_{\text{пр}}} \left(\varphi(y_i) - \bar{\varphi}(y_i)\right)^2}{\sum_{i=1}^{N_{\text{пр}}} \left(\bar{\varphi}(y_i)\right)^2}, \qquad (5.4)$$

где $\varphi(y_i)$ – выходная величина модели, коэффициенты которой получены на множестве Y_1 точек обучающей последовательности, в точке $y_i \in Y_2$; Y_2 – множество точек проверочной последовательности, $Y_1 \cup Y_2 = Y_0$, $Y_1 \cap Y_2 = \emptyset$; $N_{\text{пр}}$ – число точек в множестве Y_2; $\bar{\varphi}(y_i)$ – фактическое значение функции $\varphi(y)$ в точке $y_i \in Y_2$.

Среднеквадратичная ошибка, вычисленная на отдельной проверочной последовательности

$$\delta^2(Y_2) = \frac{1}{N_{\text{пр}}} \sum_{i=1}^{N_{\text{пр}}} \left(\varphi(y_i) - \bar{\varphi}(y_i)\right)^2, \; y_i \in Y_2. \qquad (5.5)$$

Критерий минимума смешения

$$n_{\text{см}}^2 = \frac{\sum_{i=1}^{r} \left(\varphi_{Y_1}(y_i) - \varphi_{Y_2}(y_i)\right)^2}{\sum_{i=1}^{r} \left(\bar{\varphi}(y_i)\right)^2}, \qquad (5.6)$$

где $\varphi_{Y_1}(y_i)$ – выходная величина модели, коэффициенты которой получены на множестве Y_1, в точке y_i; $\varphi_{Y_2}(y_i)$ – выходная величина модели, коэффициенты которой получены на множестве Y_2, в точке

y_i ; r – число точек в множестве $Y_1 \cup Y_2$.

Несимметричный критерий стабильности

$$\delta^2(Y_2) = \frac{1}{r}\sum_{i=1}^{r}\left(\varphi(y_i) - \bar{\varphi}(y_i)\right)^2, \qquad (5.7)$$

где $\varphi(y_i)$ – выходная величина модели, коэффициенты которой получены на множестве Y_1, в точке y_i, $\bar{\varphi}(y_i)$ – фактическое значение функции $\varphi(y)$ в точке y_i.

В качестве функции $\varphi_1(A_1, y)$ могут служить и другие критерии, модифицированные, если нужно, с учетом наличия множества Y_0.

Примеры функций $\varphi_2(A_1, y)$.

Критерий

$$\Delta^2(Y_2) = \frac{\sum_{j=1}^{N_{21}}\left(\varphi(y_j) - \bar{\varphi}(y_j)\right)^2}{\sum_{j=1}^{N_{21}}\left(\bar{\varphi}(y_j)\right)^2}, \qquad (5.8)$$

где N_{21} – число точек, на которых оценивается полученная модель, $N_{21} \le N_2$; N_2 – число точек множества Y_2.

Среднеквадратичный критерий, вычисленный на точках множества Y_0:

$$\delta^2(Y_0) = \frac{1}{r_1}\sum_{j=1}^{r_1}\left(\varphi(y_j) - \bar{\varphi}(y_j)\right)^2, \qquad (5.9)$$

где r_1 – число точек множества Y_0, на которых оценивается модель, $r_1 \le r$; r – число точек множества Y_0.

Или среднеквадратичный критерий, вычисленный на точках множества Y_2

$$\delta^2(Y_2) = \frac{1}{N_2}\sum_{j=1}^{N_2}\left(\varphi(y_i) - \bar{\varphi}(y_i)\right)^2. \qquad (5.10)$$

Критерии

$$\delta_1 = \max_{y_j \in Y_0} \left| \varphi(y_j) - \bar{\varphi}(y_j) \right|, \tag{5.11}$$

$$\delta_2 = \max_{y_{j1},\ldots,y_{jq} \in Y_0} \sum_{k=j_1,\ldots,j_q} \left| \varphi(y_k) - \bar{\varphi}(y_k) \right|, \; q << r, \tag{5.12}$$

$$\delta_3 = \max_{y_{j1},\ldots,y_{jq} \in Y_2} \sum_{k=j_1,\ldots,j_q} (\varphi(y_k) - \bar{\varphi}(y_k))^2, \; q < N_2, \tag{5.13}$$

$$\delta_4 = \max_{y_{j1},\ldots,y_{jq} \in Y_2} \frac{1}{q} \sum_{k=j_1,\ldots,j_q} \left| \varphi(y_k) - \bar{\varphi}(y_k) \right|, \; q < N_2, \tag{5.14}$$

$$\delta_5 = \max_{y_j \in Y_2} \left| \varphi(y_j) - \bar{\varphi}(y_j) \right|, \tag{5.15}$$

и так далее [7]. В выражениях (5.11) − (5.15) операция максимума берется на множестве Y_2 или Y_0, поэтому эти выражения можно использовать в виде соотношения (5.3).

В зависимости от вида функций $g_k(y)$, $\varphi^i(y), i = \overline{1,q}$ и величины q в выражении (5.1) может порождаться целый ряд конкретных задач, решаемых различными алгоритмами МГУА.

Задача 5.2. Определение лучшей математической модели в виде скалярной функции (q = 1)

$$\varphi(y) = \sum_{k=0}^{m} a_k g_k(y). \tag{5.16}$$

При $\varphi(y) \equiv y$, $g_k(y) = x_k$, $k = \overline{1,m}$, имеем случай синтеза функции $y = f(x_1, x_2, \ldots, x_m)$.

Задача 5.3. Определение лучшей математической модели в виде системы линейных алгебраических уравнений: $q > 1$, $g_k(y) = y_k$; $\varphi^i(y) = y_i$, $k, i = \overline{1,q}$:

$$y_1 = a_0^1 + a_1^1 y_1 + \ldots + a_q^1 y_q.$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$y_q = a_0^q + a_1^q y_1 + \ldots + a_q^q y_q.$$

Задача 5.4. Определение лучшей математической модели в виде обыкновенного дифференциального уравнения с постоянными коэффициентами: $q = 1;\ g_0(y) = 1,\quad g_k(y) = y^k(t),\ k = \overline{1,m},\quad \varphi(y) = \dfrac{dy}{dt};$

$a_k,\ k = \overline{0,m}$ – константы,

$$\frac{dy}{dt} = a_0 + a_1 y^1 + a_2 y^2 + \ldots + a_m y^m.$$

Задача 5.5. Определение лучшей математической модели в виде системы обыкновенных дифференциальных уравнений с переменными коэффициентами:

$$q > 1,\ g_0(y) = 1,\ \ g_k(y) = y_k(t),\ k = \overline{1,m},\ \ \varphi^i(y) = \frac{dy_i}{dt},$$
$$\frac{dy_1}{dt} = a_0^1 + a_1^1 y_1 + a_2^1 y_2 + \ldots + a_q^1 y_q,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$\frac{dy_q}{dt} = a_0^q + a_1^q y_1 + a_2^q y_2 + \ldots + a_q^q y_q.$$

Если в задаче 5.5 положить, что $\varphi(y) = \int y_i(t)dt,$ то поиск лучшей математической модели будет вестись в классе систем интегральных уравнений. Не трудно также из задачи 5.1 получить задачи поиска математических моделей среди смешанных систем уравнений, например, содержащих алгебраические и дифференциальные уравнения или алгебраические и интегральные и т.д.

5.2. Схемы обработки данных многорядными алгоритмами МГУА

Обработку данных итерационными алгоритмами МГУА поясним на примере задачи 5.2, когда полное описание объекта ищется в виде некоторой нелинейной функции m переменных $y = f(x_1, x_2, \ldots, x_m)$.

В отличие от регрессионного анализа, предполагающего известным вид уравнения регрессии и определяющего только неизвестные коэффициенты уравнения, исходя из достижения минимума среднеквадратичной ошибки на всех точках исходных данных, метод самоорганизации математических моделей определяет не только параметры, но и структуру полного описания объекта. При этом выполняется последовательный синтез и селекция множеств постепенно усложняющихся моделей или решений − характерная особенность итерационных или многорядных алгоритмов самоорганизации математических моделей (или алгоритмов МГУА).

Схема обработки данных этим методом при синтезе математического описания в виде функции m переменных $y = f(x_1, x_2, \ldots, x_m)$. приведена на рис. 5.1.

Из схемы обработки данных следует, что структура и параметры нелинейной функции $y = f(x_1, x_2, \ldots, x_m)$. определяются посредством ряда последовательных шагов или рядов селекции. При этом все исходные данные делятся на две части − обучающую и проверочную последовательности.

На первом ряду селекции на точках обучающей последовательности генерируются простейшие частные модели как функции пар входных независимых переменных $x_i, i = 1, 2, \ldots, m,$ например вида

$$f(x_j, x_k) = a^1_{1\,jk} + a^1_{2\,jk} x_j + a^1_{3\,jk} x_k, \qquad (5.17)$$

или

$$f(x_j, x_k) = a^1_{1\,jk} + a^1_{2\,jk} x_j + a^1_{3\,jk} x_k + a^1_{4\,jk} x_j x_k, \qquad (5.18)$$

или

$$f(x_j, x_k) = a^1_{1\,jk} + a^1_{2\,jk} x_j + a^1_{3\,jk} x_k + a^1_{4\,jk} x_j^2 + a^1_{5\,jk} x_k^2 + a^1_{6\,jk} x_j x_k, \quad (5.19)$$

и т.д. Здесь $j = \overline{1, m-1}$, $k = \overline{2, m}$, $j \neq k$.

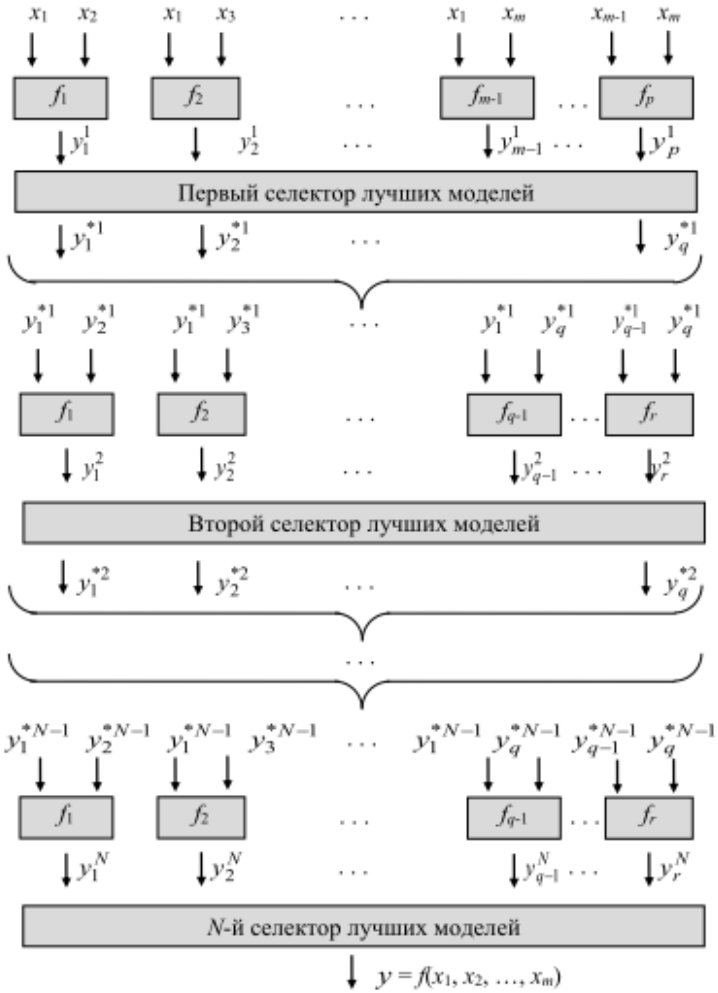

Рис. 5.1 Схема обработки данных многорядным алгоритмом МГУА

Лучшие по заданному критерию селекции q моделей первого ряда $y_1^{*1}, y_2^{*1}, \ldots, y_q^{*1}$ используются для синтеза моделей на втором ряду селекции с помощью соотношений (5.20) – (5.22), аналогичных, соответственно соотношениям (5.17) – (5.19):

$$y^2(y_r^{*1}, y_v^{*1}) = a_{1\,rv}^2 + a_{2\,rv}^2 y_r^{*1} + a_{3\,rv}^2 y_r^{*1}, \tag{5.20}$$

или

$$y^2(y_r^{*1}, y_v^{*1}) = a_{1\,rv}^2 + a_{2\,rv}^2 y_r^{*1} + a_{3\,rv}^2 y_r^{*1} + a_{4\,rv}^2 y_r^{*1} y_v^{*1}, \tag{5.21}$$

или

$$\begin{aligned} y^2(y_r^{*1}, y_v^{*1}) = a_{1\,rv}^2 + a_{2\,rv}^2 y_r^{*1} + a_{3\,rv}^2 y_v^{*1} + a_{4\,rv}^2 (y_r^{*1})^2 + \\ + a_{5\,rv}^2 (y_v^{*1})^2 + a_{6\,rv}^2 y_r^{*1} y_v^{*1}, \end{aligned} \tag{5.22}$$

и т.д. Здесь $r = \overline{1, q-1}, v = \overline{2, q}, r \neq v.$

Все полученные модели первого ряда оцениваются на точках проверочной последовательности и q описаний, лучших по заданному критерию селекции, используются для синтеза моделей на точках обучающей последовательности на втором ряду селекции.

Из полученного множества моделей второго ряда на точках проверочной последовательности опять отбирается подмножество лучших, которые используются для синтеза моделей третьего ряда, и т.д. до тех пор, пока происходит улучшение показателя качества получаемых математических описаний.

Приведенная задача синтеза математического описания в виде скалярной функции m переменных является простейшим примером применения итерационных алгоритмов МГУА, которые используются для решения обширного класса задач моделирования.

В рассмотренном примере из ряда в ряд при помощи порогового

отбора пропускаются только лучшие по принятому критерию селекции промежуточные описания. Как видно из рисунка и приведенных выражений, исходные аргументы в промежуточные переменные входят в соотношения попарно. Это справедливо для большинства алгоритмов МГУА, но известны и алгоритмы, в которых промежуточные модели получаются при использовании большего числа исходных аргументов или частных описаний предшествующих рядов селекции.

В зависимости от того, какая конкретная задача синтеза математической модели решается, какие используются критерии отбора моделей и соотношения для получения частных описаний, изменяется и общая схема алгоритма МГУА. Однако ряд блоков алгоритма остается при решении большинства задач автоматического синтеза матмоделей.

Рассмотрим эти блоки на примере синтеза модели в виде нелинейной функции m переменных $y = f(x_1, x_2, \ldots, x_m)$ с помощью алгоритма МГУА с линейными полиномами. В этом алгоритме на первом и последующих рядах селекции используются соотношения вида (5.17) и (5.20). Алгоритм предусматривает выполнение основных шагов.

1. Задание основных параметров (числа и вида синтезируемых моделей на первом ряду селекции, числа лучших описаний, пропускаемых в следующий ряд селекции, способа разделения исходных данных на части, критерия селекции, условий окончания работы алгоритма и т.д.).

2. Разделение исходных данных на обучающую и проверочную последовательности.

3. Получение нормальных уравнений Гаусса на точках обучающей последовательности.

4. Решение нормальных уравнений Гаусса.

5. Формирование массива частных описаний для текущего ряда селекции.

6. Оценка с помощью заданного критерия полученных частных описаний на множестве точек проверочной последовательности.

7. Отбор пропускаемых в следующий ряд селекции лучших частных описаний текущего ряда.

8. Проверка условий окончания процесса синтеза моделей. Если условия выполняются, то − переход на выполнение пункта 10.

9. Формирование частных описаний с линейными полиномами для следующего ряда селекции и переход на выполнение пункта 3.

10. Вывод информации о лучших полученных моделях.

11. Конец.

Все алгоритмы МГУА требуют разделения исходных данных на несколько частей. В наиболее простом и распространенном случае исходные данные делятся на обучающую и проверочную последовательности. Качество конечной модели во многом зависит от этого деления, однако универсального алгоритма, позволяющего наилучшим образом выполнить этот этап, нет.

Наиболее часто применяются следующие способы деления исходных данных на две части:

- все исходные точки нумеруются числами натурального ряда, а затем обучающая последовательность формируется из точек с нечетными номерами, а проверочная − из точек с четными номерами, или наоборот;

- исходные точки ранжируются по дисперсии и в обучающую последовательность включается примерно половина точек с большей дисперсией, а в проверочную − оставшиеся точки;

- в проверочной последовательности используется только одна точка исходных данных;

- исходные данные ранжируются по дисперсии и делятся на обучающую и проверочную последовательности таким образом, чтобы получить минимум числа рядов селекции.

В лучших универсальных программах МГУА, как правило, используется целый спектр различных способов деления исходных данных на две и большее число частей.

В настоящее время известны сотни различных алгоритмов самоорганизации математических моделей, которые успешно применялись на универсальных вычислительных машинах фон-неймановской архитектуры при решении разнообразных задач математического модели-

рования. Однако известны и трудности применения алгоритмов самоорганизации, связанные как с недостаточной теоретической обоснованностью алгоритмов, так и нехваткой быстродействия самых лучших последовательных компьютеров. Дальнейшее повышение эффективности применения метода самоорганизации математических моделей связано не только с развитием теории метода, но и разработкой новых принципов организации аппаратных и программных средств, связанных, прежде всего, с распараллеливанием вычислений.

Это одна из основных схем алгоритмов МГУА. Таких схем насчитывается несколько десятков. На основе каждой схемы обычно существуют десятки или даже сотни различных алгоритмов, ориентированных на решение различных задач автоматического синтеза математических моделей.

Для получения коэффициентов $a^1_{1jk}, a^1_{2jk}, a^1_{3jk}, a^1_{4jk}, \ldots$ частных описаний первого и последующего рядов селекции обычно используют метод нормальных уравнений Гаусса, позволяющий получить модели, сумма квадратов отклонений значений которых на используемых данных обучающей последовательности, минимальна.

Рассмотрим работу алгоритма МГУА с линейными полиномами на примере синтеза функции двух переменных.

Пример 5.1. Пример применения алгоритма МГУА с линейными полиномами для синтеза функций двух переменных. Исходные данные представлены в табл 5.1. Они получены с шагом 0,32 по переменным *x* и *z* полинома

$$y = 12 + 8x - 7x^2 - 2x^3 + x^4 + 0{,}5z + z^2, \qquad (5.23)$$

в прямоугольнике $-2{,}4 \le x \le 3.04,\ -1{,}6 \le z \le 1{,}6.$

В первой строке таблицы приведены значения переменной *x*, а в первом столбце – переменной *z*.

Во всех остальных элементах таблицы приведены значения функции, при этом четные столбцы содержат значения функции на точках обучающей последовательности, а нечетные – значения функции на

точках проверочной последовательности.

Частные описания первого ряда селекции имеют вид

$$y_q^1 = a_{0q}^1 + a_{1q}^1 v_i^1 + a_{2q}^1 v_j^1,\ v_i^1 \neq v_j^1,$$

где $v_i^1, v_j^1 \in M,\ M = (x, \ldots, x^{k1},\ z, \ldots, z^{k2},\ xz, \ldots,\ x^{k3} z^{k4});\ \ k^1, \ldots, k^4$ – заданные константы; $k^5 = \sum_{s=1}^{4} k^s;\ q = \overline{1, C_{k5}^2}$, C_{k5}^2 – число сочетаний по 2 из k^5, $C_{k5}^2 = \dfrac{9 \cdot 8}{2} = 36$.

При $k^1 = 5, k^2 = 3, k^3 = k^4 = 1$ на первом ряду селекции получается 36 частных описаний, приведенных в табл. 5.2, где цифрами 1, 2, …, 9 обозначены, соответственно, одночлены $x, x^2, \ldots, xz$.

При оценке частных описаний с помощью среднеквадратичной ошибки на точках проверочной последовательности величина ошибки δ_q^1 меняется существенно от $\delta_{10}^1 = 21{,}471$ до $\delta_{17}^1 = 82{,}070,$ но при этом имеются группы частных описаний: $y_5 - y_8$; $y_{12} - y_{15}$; $y_{18} - y_{21}$; $y_{23} - y_{26}$; y_2, $y_{27} - y_{30}$; y_{31}, y_{34}; y_9, y_{32}, y_{33}, y_{36}, разница в величинах критерия которых минимальна (не превышает 0,5 – 1, т.е. меньше 5 %). Анализ поведения частных описаний, принадлежащих к одной группе, показывает, что обычно частные описания одной группы ведут себя подобным образом. При изучении ошибок (таблицы ошибок не приведены):

$$h_q(x_v, z_w) = y(x_v, z_w) - y_q(x_v, z_w),\ q = \overline{12,15},\ v = \overline{1,18},\ w = \overline{1,11},$$

$$x_v \in X = \{x_1, x_2, \ldots, x_{18}\} = \{-2{,}40, -2{,}08, \ldots, 3{,}04\},$$

$$z_v \in Z = \{z_1, z_2, \ldots, z_{11}\} = \{-1{,}60, -1{,}28, \ldots, 1{,}60\},$$

определены группы частных описаний, имеющих лучшие критерии селекции.

Таблица 5.1 – Исходные данные для примера 5.1

z	*x*								
	-2,4	-2,08	-1,76	-1,44	-1,12	-0,8	-0,48	-0,16	0,16
-1,60	15,07	3,55	-1,50	-2,00	0,40	4,31	8,58	12,31	14,85
-1,28	14,30	2,79	-2,27	-2,77	-0,36	3,55	7,82	11,55	14,09
-0,96	13,75	2,23	-2,82	-3,32	-0,92	3,00	7,26	10,99	13,53
-0,64	13,40	1,88	-3,17	-3,67	-1,27	2,64	6,91	10,64	13,18
-0,32	13,25	1,73	-3,32	-3,82	-1,42	2,50	6,76	10,49	13,04
0,00	13,31	1,79	-3,26	-3,76	-1,36	2,55	6,82	10,55	13,09
0,32	13,57	2,05	-3,00	-3,50	-1,10	2,82	7,08	10,81	13,36
0,64	14,04	2,52	-2,53	-3,03	-0,63	3,28	7,55	11,28	13,82
0,96	14,71	3,19	-1,86	-2,36	0,04	3,96	8,22	11,95	14,49
1,28	15,58	4,07	-0,99	-1,49	0,92	4,83	9,10	12,83	15,37
1,60	16,67	5,15	0,10	-0,40	2,00	5,91	10,18	13,91	16,45

z	x								
	0,48	0,80	1,12	1,44	1,76	2,08	2,40	2,72	3,04
-1,60	15,82	15,07	12,70	9,09	4,85	0,84	-1,83	-1,78	2,61
-1,28	15,06	14,30	11,94	8,33	4,09	0,07	-2,59	-2,54	1,85
-0,96	14,50	13,75	11,38	7,77	3,53	-0,48	-3,15	-3,10	1,29
-0,64	14,15	13,40	11,03	7,42	3,18	-0,84	-3,50	-3,45	0,94
-0,32	14,00	13,25	10,89	7,28	3,03	-0,98	-3,65	-3,60	0,79
0,00	14,06	13,31	10,94	7,33	3,09	-0,92	-3,59	-3,54	0,85
0,32	14,32	13,57	11,21	7,60	3,35	-0,66	-3,33	-3,28	1,11
0,64	14,79	14,04	11,67	8,08	3,82	-0,20	-2,86	-2,81	1,58
0,96	15,76	14,71	12,34	8,73	4,49	0,48	-2,19	-2,14	2,25
1,28	16,34	15,58	13,22	9,61	5,37	1,35	-1,31	-1,26	3,13
1,60	17,42	16,67	14,30	10,69	6,45	2,44	-0,23	-0,18	4,21

Таблица 5.2 –Частные описания первого ряда селекции

q	v^1_i	v^1_j	δ^1_q	a^1_{0q}	a^1_{1q}	a^1_{2q}
1	1	2	29,476	9,0648	-0,60576	-1,07730
2	1	3	50,664	6,9212	6,19770	-1,45580
3	1	4	35,829	7,0252	-0,59964	-0,11411
4	1	5	78,421	7,0252	3,15010	-0,14344
5	1	6	39,856	6,1507	-0,95048	0,50000
6	1	7	39,294	5,1267	0,95048	1,00000
7	1	8	39,895	6,1507	-0,95048	0,23301
8	1	9	40,100	6,1507	-0,95048	0,02902
9	2	3	38,420	8,5995	-0,79115	-0,31960
10	2	4	21,471	10,924	-3,26790	0,29978
11	2	5	49,478	8,3531	-0,63331	-0,05706
12	2	6	24,981	9,2102	-1,16520	0,50000
13	2	7	24,419	8,1862	-1,16520	1,00000
14	2	8	25,020	9,2102	-1,16520	0,23301
15	2	9	25,226	9,2102	-1,16520	0,02902
16	3	4	42,695	7,2383	-0,3465	-0,05814
17	3	5	82,070	6,9742	0,83817	-0,197471
18	3	6	49,832	6,5128	-0,39103	0,50000
19	3	7	40,271	5,4888	-0,39103	1,00000
20	3	8	40,871	6,5128	-0,39103	0,23301
21	3	9	41,077	6,5128	-0,39103	0,02902
22	4	5	52,522	7,0046	-0,02364	-0,06411
23	4	6	31,821	7,7670	-0,12985	0,50000
24	4	7	31,260	6,7330	-0,12985	1,00000
25	4	8	31,860	7,7670	-0,12985	0,23301
26	4	9	32,066	7,7670	-0,12985	0,02902
27	5	6	51,237	6,7239	-0,06797	0,50000
28	5	7	50,675	5,6999	-0,06797	1,000000
29	5	8	51,275	6,7239	-0,06797	0,23301
30	5	9	51,481	6,7239	-0,06797	0,02902
31	6	7	37,282	4,9747	0,50000	1,00000
32	6	8	38,100	5,9987	0,50000	$0{,}14914\cdot10^{-7}$
33	6	9	38,100	5,9987	0,50000	$-0{,}73090\cdot10^{-9}$
34	7	8	37,320	4,9747	1,00000	0,23301
35	7	9	37,526	4,9747	1,00000	0,02902
36	8	9	38,138	5,9987	0,27269	0,00440

Во всех строках этих таблиц ошибка приближения четыре раза меняет свой знак (в первом, восьмом – тринадцатом и восемнадцатом столбцах ошибка положительная, а в третьем – шестом, пятнадцатом – семнадцатом столбцах – отрицательна, во втором, седьмом и четырнадцатом столбцах преобладают отрицательные ошибки), т.е. видно, что массивы ошибок приближения сходны не только по величинам критерия селекции, но и другим характеристикам.

Потомки частных описаний, получаемых за счет композиции частных описаний внутри выделенных групп, по величине среднеквадратичной ошибки мало отличаются от исходных описаний. В табл. 5.3 приведены среднеквадратичные ошибки δ_p^z и коэффициенты частных описаний второго ряда селекции

$$y_p^z = a_{0p}^z + a_{1p}^z y_i^1 + a_{2p}^z y_j^1, \;\; y_i^1 \neq y_j^1,$$

где y_i^1, y_j^1 – выбранные частные описания первого ряда селекции.

Частные описания $y_{11}^1, \ldots, y_{15}^1$ первого ряда селекции (в табл. 5.3 фигурируют как описания $y_2^1, \ldots, y_5^1$) дают шесть потомков $(y_9^2, y_{10}^2, y_{11}^2, y_{16}^2, y_{17}^2, y_{22}^2)$, численные значения критерия селекции находится в пределах от 24,312 до 25,020. Частные описания $y_{23}^1, \ldots, y_{25}^1$ (в табл. 5.3 фигурируют как описания $y_7^1, \ldots, y_9^1$) дают три потомка $(y_{34}^2, y_{35}^2, y_{36}^2)$, величины среднеквадратичных критериев которых на точках проверочной последовательности практически не отличаются от величин критериев описаний первого ряда.

Иначе может обстоять дело с потомками частных описаний, относящихся к различным исходным группам частных описаний.

Например, частные описания y_k^2 $(k = 1-8, 13, 20, 25, 26)$ имеют значения среднеквадратичного критерия на точках проверочной последовательности меньше, чем у самого лучшего частного описания первого ряда.

Таблица 5.3 – Частые описания второго ряда селекции

p	$y^1{}_i$	$y^1{}_j$	δ_p^2	a_{0p}^2	a_{1p}^2	a_{2p}^2
1	1	2	19,802	-0,4099	0,7524	0,3170
2	1	3	20,738	-0,1640	0,9005	0,1268
3	1	4	20,830	-0,1420	0,9138	0,1098
4	1	5	21,462	$-0{,}1743\cdot10^{-2}$	0,9989	$0{,}1348\cdot10^{-2}$
5	1	6	20,961	-0,6322	0,6950	0,4104
6	1	7	19,940	-0,6891	0,8757	0,2392
7	1	8	20,836	-0,2581	0,9534	0,0896
8	1	9	20,919	-0,2222	0,9599	0,0771
9	2	3	24,312	-0,1398	0,7672	0,2561
10	2	4	24,334	-0,1235	0,7943	0,2263
11	2	5	24,419	$-0{,}1740\cdot10^{-2}$	0,9971	$0{,}3189\cdot10^{-2}$
12	2	6	26,811	-0,3068	0,4888	0,5624
13	2	7	20,035	2,5349	2,7071	-2,1297
14	2	8	22,533	1,1000	1,4503	-0,6337
15	2	9	22,395	1,1999	1,4912	-0,6913
16	3	4	24,981	$-0{,}6976\cdot10^{-6}$	0,9999	$0{,}2544\cdot10^{-4}$
17	3	5	24,981	$-0{,}6970\cdot10^{-6}$	1,0000	$0{,}3419\cdot10^{-5}$
18	3	6	28,365	-0,1446	0,2303	0,7938
19	3	7	24,612	0,1621	1,0846	-0,1117
20	3	8	20,980	2,8511	2,7703	-2,2456
21	3	9	21,020	2,7716	2,6748	-2,1369
22	4	5	25,020	$-0{,}2630\cdot10^{-3}$	0,9983	$0{,}1708\cdot10^{-2}$
23	4	6	28,508	-0,1272	0,2027	0,8185
24	4	7	24,644	0,1624	1,0864	-0,1135
25	4	8	20,996	2,5354	2,5652	-1,9878
26	4	9	21,051	2,8757	2,7752	-2,2546
27	5	6	29,463	$-0{,}1741\cdot10^{-2}$	$0{,}2771\cdot10^{-2}$	0,9975
28	5	7	24,806	0,1641	1,0978	-0,1252
29	5	8	22,196	1,3949	1,8331	-1,0656
30	5	9	21,972	1,5568	1,9295	-1,1890
31	6	7	29,351	0,3821	1,1640	-0,2277
32	6	8	28,965	1,6182	1,6946	-0,9643
33	6	9	28,949	1,7597	1,7553	-1,0487
34	7	8	31,395	-0,2084	0,7699	0,2648
35	7	9	31,388	-0,1844	0,7964	0,2343
36	8	9	31,821	$-0{,}1469\ 10^{-5}$	0,9999	$0{,}2004\cdot10^{-4}$

Увеличения числа рядов селекции в группах частных описаний с близкими по величине критерия селекции не может существенно изменить характера частных описаний, приближающих исходные данные. Например, потомки девяти частных описаний y_k^2, $(k = 1-3, 5-8, 13, 20)$ на десятом ряду селекции дают полином

$$y = b_0 + b_1 x^2 + b_2 x^4 + b_3 z + b_4 z^2 + b_5 z^3,$$

где $b_0 = 9{,}6093$, $b_1 = -2{,}4945$, $b_2 = 0{,}1743$, $b_3 = 0{,}1750$, $b_4 = 0{,}8434$, $b_5 = -0{,}0251$, имеющий среднеквадратичный критерий на точках проверочной последовательности, равный 19,519. Если для сравнения из упорядоченного по величине критерия селекции множества частных описаний в следующий ряд селекции постоянно пропускать y_k^2, $k = \overline{1,11}$, $k = 1, 2, 4j+1$, $j = \overline{2,8}$, (на первом ряду в табл. 5.2 соответственно q равно 2, 7, 10, 13, 18, 25, 30, 33, 34), то на одинадцатом ряду селекции получим полином

$$y = b_0 + \sum_{i=1}^{5} b_i x^i + \sum_{j=1}^{3} b_{5+j} z^j + b_9 xz,$$

где $b_0 = 11{,}0360$, $b_1 = 10{,}4818$, $b_2 = -3{,}8049$, $b_3 = 0{,}1750$, $b_4 = 0{,}3741$, $b_5 = 0{,}2575$, $b_6 = 0{,}0$, $b_7 = 0{,}7983$, $b_8 = 0{,}2560$, $b_9 = 0{,}02686$, имеющий среднеквадратичный критерий, равный 2.844.

Полученные результаты показывают, что существующие в алгоритмах МГУА процедуры отбора частных описаний по минимуму величины критерия селекции на точках проверочной последовательности могут быстро сводится к уточнению одного локального экстремума в многоэкстремальном пространстве моделей.

Для того чтобы поиск имел глобальный характер, необходимо в следующий ряд селекции пропускать наравне с лучшими как можно больше отличающихся друг от друга частных описаний. Проще всего это сделать путем пропуска в следующий ряд селекции не только лучших, но и худших, а также занимающих промежуточное положение частных описаний. При формировании такого набора частных описа-

ний рационально использовать методы теории аналогий, позволяющие определять группы сходных и отличающихся частных описаний, и пропускать в следующий ряд селекции как можно более отличающиеся друг от друга описания.

Для осуществления глобального поиска в пространстве возможных моделей алгоритм предполагает многократное возвращение к первому ряду селекции и использование каждый раз нового способа отбора частных описаний для следующих рядов селекции. При формировании множества частных описаний, используемых для получения моделей-потомков, этот алгоритм в идейном отношении имеет определенное сходство с алгоритмами МГУА со случайным выбором партнеров. Недостатки этих алгоритмов также сходны – если на первом ряду селекции получены частные описания, которые при выбранном способе получения потомков не позволяют расщеплять критические точки, то эти алгоритмы не могут привести к удовлетворительной конечной модели ни при каких попытках изменить параметры эволюционного процесса синтеза конечной модели. В частности, не удается добиться приемлемого по точности приближения исходных данных и в рассматриваемом примере.

Индивидуальное задание

Для произвольной кривой, заданной набором амплитуд (Внимание!! Не задавайте линейно изменяющихся последовательностей), с помощью программы определите вид и коэффициенты аппроксимирующего полинома:

1) по критерию 1;
2) по критерию 2.

Сравните результаты работы программы

Оцените точность полученных результатов и скорость решения.

Содержание отчета

1. Тема и цель занятия.
2. Индивидуальное задание.

3. Результаты выполнения задания.

4. Выводы по лабораторной работе.

Контрольные вопросы

1. Какой алгоритм МГУА использовался в программе?

2. Какой вид частных описаний используется в программе?

3. Как производится нормировка исходных данных в программе?

4. Где и как в программе определяется максимальное число рядов селекции?

5. Какие две последовательности входных данных использовались в Ваших экспериментах в программе?

6. Как влияет число аргументов на первом ряду селекции (константа Margum) на точность прогнозирования?

7. Как влияет число пропускаемых в следующий ряд селекции частных описаний (константа Prop) на точность прогнозирования?

8. Как влияет критерий отбора частных описаний на результаты прогнозирования?

9. Почему сумма числа точек обучающей и проверочной последовательностей меньше числа *N* точек исходных данных?

10. Какие методы разделения исходных данных на обучающую и проверочную последовательности существуют?

11. Сколько частных описаний будет синтезировано на втором ряду селекции если на второй ряд пропущено 11 лучших моделей?

12. Зачем нужна в программе процедура Maxim?

13. С помощью какого метода определяются коэффициенты моделей на точках обучающей последовательности?

14. Для синтеза каких математических моделей могут использоваться алгоритмы МГУА?

15. Какой метод решения систем линейных уравнений используется в программе?

Лабораторная работа № 6
РЕШЕНИЕ ЗАДАЧ С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Цель работы: получение знаний и практических навыков по решению задач поиска экстемумов с помощью генетических алгоритмов.

6.1 Основные понятия генетических алгоритмов

Генетические алгоритмы (ГА) – модели эволюции в природе, реализованные в виде компьютерных программ, которые воспроизводят механизмы естественного отбора и генетического наследования [8].

Основное назначение генетических алгоритмов заключается в решении поисковых задач, которые характеризуются большой размерностью пространства поиска решений.

Естественный отбор – это дифференциальное выживание и размножение особей, которые отличаются друг от друга генетически детерминированными признаками. Более приспособленные к условиям окружающей среды особи оставляют больше потомков, чем менее приспособленные, что, в конечном счете, влияет на приспособленность всей популяции.

- *Хромосома* – вектор (последовательность) из нулей и единиц.
- *Ген* – одна позиция (бит) хромосомы.
- *Аллель* – значение гена.
- *Особь* (генотип, индивидуум, структура) – вариант решения задачи в закодированном виде. Иначе, это точка в многомерном пространстве для оптимизируемой функции

$$f(x_1, x_2, \ldots, x_n) \rightarrow \max.$$

- *Популяция* – множество особей, образованное на i-м шаге выполнения генетических алгоритмов.
- *Фенотип* – множество декодированных значений, соответ-

ствующих генотипу.

- *Кроссовер* (скрещивание) – операция, при которой хромосомы обмениваются своими частями.
- *Локус* – позиция (номер гена) в хромосоме.
- *Мутация* – случайное изменение одной или нескольких позиций в хромосоме.
- *Селекция* – отбор для производства потомков.
- *Поколение* – очередная популяция на *i*-м шаге.
- *Функция приспособленности* (fitness function) (ФП) – функция оценки, которая определяет меру приспособленности данной особи в популяции и позволяет выбрать из множества особей наиболее приспособленную в соответствии с эволюционным принципом выживания сильнейших.

6.2. Кодирование хромосом

Хромосома должна содержать все данные, которые дают решение, и все данные, от которых зависит это решение. Длина хромосомы зависит от количества данных и может достигать больших размеров.

Обычно используют бинарную кодировку данных. Иногда для этого применяют масштабирование.

Кроме обычной двоичной кодировки часто используется код Грея, который в некоторых случаях позволяет увеличить эффективность генетического алгоритма. Отличительной характеристикой кода Грея является то, что соседние целые числа отличаются друг от друга только на один бит.

Масштабирование предназначено для предотвращения доминирования какого-либо элемента популяции над остальными на ранних стадиях эволюции и для расширения диапазона значений оптимальности на финальных стадиях эволюции, когда разнообразие в популяции существенно снижается.

Среди данного класса методов выделяется динамическое параметрическое перекодирование, которое реализуется следующим обра-

зом: когда наблюдается сходимость в популяции, максимальные и минимальные значения диапазона пересчитываются для меньшего интервала (окна), и процесс итерируется далее. Этим достигается динамическое изменение диапазона решений при приближении к оптимуму. Таким образом, масштабирование контролирует селективный отбор внутри популяции.

Возможны также другие методы масштабирования, такие как, например, линейное динамическое масштабирование, логарифмическое масштабирование, экспоненциальное масштабирование и др.

6.3. Виды функций приспособленности

Если решается задача оптимизации, то функция приспособленности – сама целевая функция. При этом, если ищется максимум, то функция не модифицируется.

Если же ищется минимум, то функция приспособленности модифицируется так, чтобы опять свести задачу к максимизации.

Если решается задача теории управления, то функция приспособленности – это функция ошибки ε.

Если задача теории игр, то функция приспособленности – это стоимостная функция.

6.4. Виды селекции хромосом

Селекция (отбор) хромосом может осуществляться по трем вариантам: *пропорциональный*, *равномерный* и *линейный равномерный* отбор. Например, *турнирный отбор* реализует *n* турниров, чтобы выбрать *n* особей. Каждый турнир построен на выборке *k* элементов из популяций и выбора лучшей особи среди них. Наиболее распространен турнирный отбор с $k = 2$.

Элитные методы отбора, когда лучшие по значению функции приспособленности особи переносятся в следующее поколение и гарантируют, что обязательно выживут лучшие члены популяции.

6.5. Стандартный генетический алгоритм

Существует много способов реализации идеи биологической эволюции в рамках ГА. Стандартным считается ГА, представленный ниже с использованием псевдокода. Прописными буквами представлены управляющие операторы алгоритма, а курсивом – генетические операторы.

```
НАЧАЛО /*генетический алгоритм*/
        Создать начальную популяцию
        Оценить приспособленность каждой особи
        останов := ЛОЖЬ
        ПОКА НЕ останов ВЫПОЛНЯТЬ
          НАЧАЛО /*создать популяцию нового поколения */
        ПОВТОРИТЬ (размер_популяции/2) РАЗ
                НАЧАЛО /*цикл воспроизводства */
        Выбрать две особи с высокой приспособленностью из
        предыдущего поколения для скрещивания
        Скрестить выбранные особи и получить двух потомков
        Поместить потомков в новое поколение
                КОНЕЦ
        ЕСЛИ приспособленность лучшего потомка > порога ТО
        останов := ИСТИНА
          КОНЕЦ
КОНЕЦ
```

На рис. 6.1 показана схема стандартного генетического алгоритма.

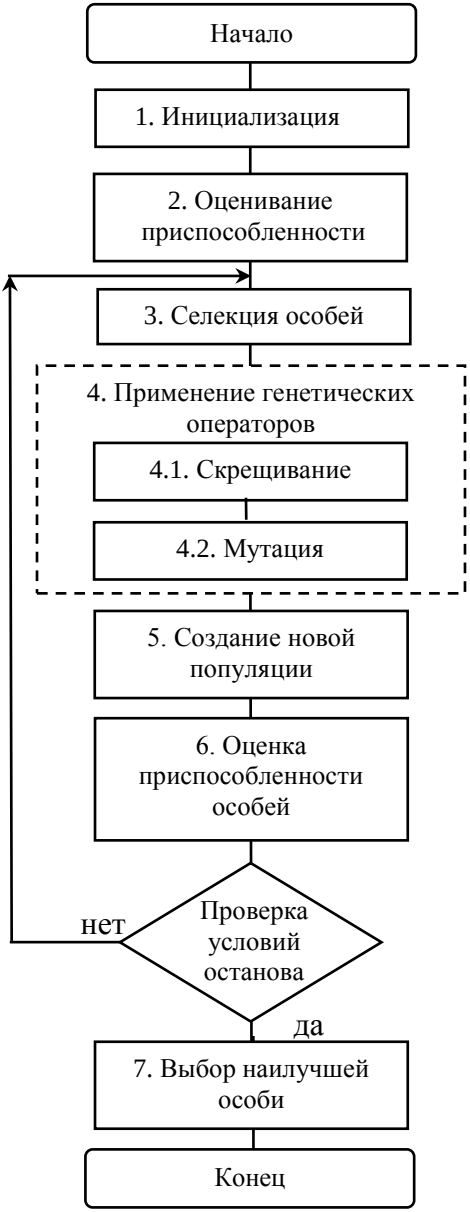

Рис. 6.1 Стандартный генетический алгоритм

Данный алгоритм оказался эффективным не только при решении реальных инженерных проблем, но и при исследовании биологических проблем, например, при моделировании иммунной системы. Канонический генетический алгоритм характеризуется следующими особенностями работы:

1) Задается критерий, определяющий эффективность каждого найденного решения.

2) В соответствии с определенными ограничениями инициализируется исходная популяция потенциальных решений. Каждое решение кодируется как вектор x, который называется хромосомой. Его элементами являются части вектора – гены, изменяющиеся значения в определенных позициях которых называются аллелями. Наиболее часто используется двоичное представление переменных.

3) Каждая хромосома x_i, $i = l, ..., P$ в популяции декодируется в необходимую форму для последующей оценки и затем ей присваивается значение эффективности $\mu(x_i)$ в соответствии с функцией приспособленности.

4) Каждой хромосоме присваивается вероятность воспроизведения ρ_i, $i = l, ..., P$, которая зависит от эффективности данной хромосомы.

5) В соответствии с вероятностями *воспроизведения* ρ_i создается новая популяция хромосом, причем с большей вероятностью воспроизводятся наиболее эффективные элементы. Хромосомы производят потомков, используя операции рекомбинации: кроссинговер (хромосомы скрещиваются, обмениваясь частями строк) и мутация (вероятностное изменение аллелей).

6) Процесс останавливается, если получено удовлетворительное решение либо если исчерпано отведенное на эволюцию время. Если процесс не окончен, то вновь повторяются процессы синтеза и оценки особей и воспроизведения новой популяции.

Формально ГА можно описать следующим образом:

$$\text{ГА} = (\text{P}^0, a_j^0\ (j = \overline{1,\lambda}), l, s, p, f, t),$$

где $P^0 = (a_1^0, ..., a_\lambda^0)$ – исходная популяция; a_j^0 $(j = \overline{1, \lambda})$ – решение задачи, представленное в виде хромосомы; λ – целое число (размер популяции); l – целое число (длина каждой хромосомы популяции); s – оператор отбора; p – отображение, определяющее рекомбинацию (кроссинговер, мутация); f – функция оптимальности; t – критерий останова.

6.6. Применение генетических операторов

К генетическим операторам относятся *кроссовер* и *мутация*. *Кроссовер* – операция скрещивания хромосом, при котором хромосомы обмениваются своими частями. Операция скрещивания (рис. 6.2) происходит следующим образом:

1) генерируются пары хромосом случайным образом;

2) для каждой пары хромосом подбирается случайным образом локус – точка разреза хромосом;

3) производится обмен частями хромосом между двумя родителями.

Родитель 1 →10011 Потомок 1 → 100***01***
Родитель 2 →10101 Потомок 2 → 101***11***

Рис. 6.2 Операция скрещивания

Мутация – случайное изменение одной или нескольких позиций в хромосоме (рис. 6.3). Оператор мутации применяется с определенной вероятностью P_m к особям популяции.

10111 → 10101

Рис. 6.3 Операция мутации хромосом

Получение новых особей и популяций происходит до тех пор, пока не будет выполняться условие останова, которое, в зависимости от условий задачи может быть: по времени; по количеству итераций; по

отсутствию улучшения функции приспособленности; по достижению максимума (если он известен).

Особенность генетических алгоритмов состоит в том, что первоначально задается множество точек, а не одна, следовательно, дальнейший поиск происходит в несколько потоков, что иногда трактуется как параллельные процессы вычисления.

Согласно схеме стандартного генетического алгоритма (рис. 6.1), следующий шаг – это создание новой популяции путем отбора лучших особей, т.е селекции. Суть селекции в том, что родителями могут стать только те особи, значение приспособленности которых не меньше пороговой величины, например, среднего значения приспособленности по популяции. Такой подход обеспечивает быструю сходимость алгоритма, однако не подходит для определения нескольких экстремумов, поскольку для таких задач алгоритм, как правило, быстро сходится к одному из решений или дает квазиоптимальное решение.

Этот недостаток можно отчасти компенсировать использованием подходящего механизма отбора. Поэтому в литературе по ГА выделяют различные вариации селекции. Наиболее известные из них — это турнирный и рулеточный (пропорциональный) отборы.

При *турнирном отборе* из популяции, содержащей N особей, выбираются случайным образом t особей, и лучшая из них особь записывается в промежуточный массив. Эта операция повторяется N раз. Особи в полученном промежуточном массиве затем используются для скрещивания (также случайным образом). Размер группы строк, отбираемых для турнира, часто равен 2. В этом случае говорят о двоичном (парном) турнире. Преимуществом способа является то, что он не требует дополнительных вычислений.

При применении *метода рулетки* особи отбираются с помощью N «запусков» рулетки, где N – размер популяции. Каждой хромосоме может быть сопоставлен сектор колеса рулетки, величина которого устанавливается пропорционально значению функции приспособленности (ФП) данной хромосомы, и чем больше значение, тем больше сектор на колесе рулетки [8].

Все колесо рулетки соответствует сумме значений ФП всех хро-

мосом рассматриваемой популяции.

Каждой хромосоме ch_i для $i = 1, 2, \ldots, K$ (K – численность популяции) соответствует сектор колеса $d(ch_i)$, выраженный в процентах

$$d(ch_i) = p_s 100\%, \quad p_s(ch_i) = f(ch_i) \Big/ \sum_{j=1}^{K} f(ch_j),$$

где $f(ch_i)$ – ФП i-й хромосомы; $p_s(ch_i)$ – вероятность воспроизведения i-й хромосомы.

При таком отборе особи популяции с более высокой приспособленностью с большей вероятностью будут выбираться, чем особи с низкой приспособленностью. Другие способы отбора можно получить на основе модификации вышеприведенных. Так, например, в рулеточном отборе можно изменить формулу для вероятности попадания особи в новую популяцию.

6.7. Пример работы генетического алгоритма

Пусть необходимо максимизировать функцию $F(x) = 2x + 1$ на отрезке [0; 31].

1. Проведем инициализацию популяции.

1.1. Решение задачи подвергаем бинарному кодированию – для представления решения (числа от 0 до 31) достаточно 5 битов, т.е. хромосома будет длиной 5 битов.

1.2. Выберем размер популяции $N = 8$. Для реальных задач значение N значительно увеличивают до сотен и даже тысяч.

Произвольно выбираем 8 точек на отрезке [0; 31], например

6 5 13 21 26 18 8 5.

Переводим значения в двоичный вид:

ch_1= [00110], ch_2= [00101], ch_3= [01101], ch_4= [10101],
ch_5= [11010], ch_6= [10010], ch_7= [01000], ch_8= [00101].

Получили первоначальную популяцию из 8 особей.

2. Рассчитываем ФП каждой особи:

$$f_1(ch_1) = 2\,ch_1 + 1 = 13, \qquad f_2(ch_2) = 2\,ch_2 + 1 = 11,$$
$$f_3(ch_3) = 2\,ch_3 + 1 = 27, \qquad f_4(ch_4) = 2\,ch_4 + 1 = 43,$$
$$f_5(ch_5) = 2\,ch_5 + 1 = 53, \qquad f_6(ch_6) = 2\,ch_6 + 1 = 37,$$
$$f_7(ch_7) = 2\,ch_7 + 1 = 17, \qquad f_8(ch_8) = 2\,ch_8 + 1 = 11.$$

3. Следующий шаг – селекция хромосом, например, методом рулетки. На основании формул

$$d(ch_i) = p_s(ch_i) \cdot 100\,\%, \qquad p_s = f(ch_i) \Big/ \sum_{j=1}^{K} f(ch_j),$$

получаем данные, приведенные на рис. 6.4 и в табл. 6.1.

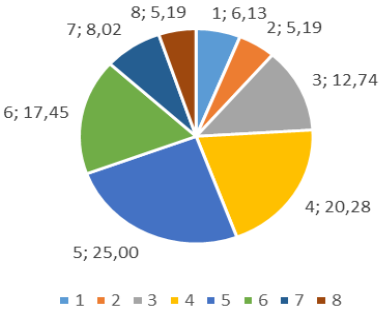

Рис. 6.4 Демонстрация метода рулетки,
где первое число – номер особи, второе – проценты особи (d)

Таблица 6.1 – Значения функций f, p_s, d

	№ особи							
	1	2	3	4	5	6	7	8
f	13	11	27	43	53	37	17	11
p_s	0,06	0,05	0,13	0,20	0,25	0,17	0,08	0,05
d	6,13	5,19	12,74	20,28	25	17,45	8,02	5,19

4. Далее разыгрываются 8 чисел из интервала от 0 до 100:

79 44 9 74 45 86 48 23,

которые сопостовляются на рулетке с соответствующим номером особи и осуществляется выбор хромосом (родительский пул) (рис. 6.5).

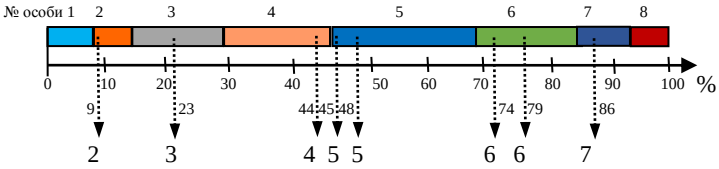

Рис. 6.5 Схема выбора особей методом рулетки

Получаем номера особей для родительского пула:

6 4 2 6 5 6 5 3.

5. Допустим, что ни одна из хромосом не подвергается мутации $p_m = 0$.

6. Проведем только скрещивание с вероятностью $p_c = 0{,}75$. Объединяем пары, так как они находятся в родительском пуле.

Для каждой пары случайным образом выбираем число из интервала [1, 0], например:

0,12 0,73 0,65 0,33.

Все выбранные числа меньше $p_c = 0{,}75$, значит, скрещивается каждая пара.

Находим для каждой пары локус – точку разреза хромосом t_k путем случайного выбора цифр из диапазона [1, 4] – (табл. 6.2).

Получили следующее, второе, поколение популяции.

Вычисляем f_i (ch_i) для каждой новой особи:

35 45 9 39 53 37 59 21.

Получили популяцию с более высоким уровнем приспособленности. Цикл повторяется до выполнения условия останова.

Таблица 6.2 – Скрещивание хромосом

t_k = 3 Первая пара родителей ch_6= [100**10**] ch_4= [101**01**]	Первая пара потомков [100**01**] [101**10**]
t_k = 4 Вторая пара родителей ch_2= [00101] ch_6= [10010]	Вторая пара потомков [00100] [10011]
t_k = 3 Третья пара родителей ch_5= [110**10**] ch_6= [100**10**]	Третья пара потомков [110**10**] [100**10**]
t_k = 2 Четвертая пара родителей ch_5= [11**010**] ch_3= [01**101**]	Четвертая пара потомков [11**101**] [01**010**]

Преимуществом генетических алгоритмов является: 1) отсутствие ограничения на дифференцируемость функций, в частности, ГА работает и тогда, когда функции нет вообще; 2) гибкость – хорошо работает при минимуме информации об окружающей среде (при высокой степени априорной неопределенности); 3) в ряде случаев ГА может находить только логический минимум (максимум). Несмотря на это, дает быстрое нахождение приемлемого решения; 4) комбинируется с другими методами искусственного интеллекта, и его эффективность может повышаться; 5) применяются для решения поисковых задач, которые имеют большое пространство поиска решений. Наиболее распространенное применение – решение задач оптимизации; 6) возможность распараллеливания вычислений.

Содержание отчета

1. Тема и цель занятия.
2. Индивидуальное задание.
3. Результаты выполнения задания.
4. Выводы по работе.

Индивидуальное задание

Задание предусматривает выполнение одного из типов задач. Тип задачи, I или II, задается преподавателем на занятии.

I тип. Решение задачи комбинаторной оптимизаци. Решить задачу комивояжера согласно табл. 6.3. Тестовые наборы представлены в трех формах:

1. Эвклидовы координаты городов заданы в формате: №_города, координата *x*, координата *y* (через пробел).

2. Полная матрица расстояний между всеми городами.

3. Диагональная матрица расстояний, которую необходимо транспонировать, после чего заполнить верхнюю половину матрицы расстояний (от главной диагонали). Нижняя половина заполняется из верхней $dist_{ij} = dist_{ji}$.

Таблица 6.3. – Варианты заданий I типа

№ п.п.	Тестовые данные	Задача
1	Wi29.tsp	Представление соседства
2	Dj89.tsp	Представление соседства
3	Att48.tsp	Представление соседства
4	Eil51.tsp	Представление порядка
5	Bayg29.tsp	Представление порядка
6	Berlin52.tsp	Представление порядка
7	Eil51.tsp	Представление пути
8	Wi29.tsp	Представление пути
9	Dj89.tsp	Представление пути
0	Bayg29.tsp	Представление пути

II тип. Составить программу для получения с помощью ГА значений параметров для минимума функции *f*, заданной таблицей 6.4, согласно номеру зачетки. Принять размер начальной популяции 100 особей, вероятность скрещивания 0.4, вероятность мутации 0,06. Построить график зависимости значения ФП от количества поколений.

Таблица 6.4. – Варианты заданий II типа

№	Вид функции *f*
1	Определить минимум $1/(x-0{,}3)^2 + 1/(x-0{,}9)^2$ в интервале [−10; 20] с точностью 0,01
2	Определить минимум функции $-2x(\sin 10\pi x) + 1$ в интервале [−1; 2] с точностью 0,001
3	Определить минимум функции $x_1^2 + x_2^2$ в интервале [−15; +20] с точностью 0,001
4	Определить минимум функции $x_1^2 + 1/2(x_2^2 + 3)$ в интервале [−10; 10] с точностью 0,01
5	Определить минимум функции $(x_1^2 + x_2 - 11)^2 + (x_1^2 + x_2^2 - 7)^2$ в интервале [−1; 10] с точностью 0,001
6	Определить минимум функции $2(1-x_1)^2 e^{-x_1^2-(x_2+1)^2}$ в интервале [−30; +30] с точностью 0,01
7	Определить максимум функции $7(x_1/3 - x_1^3 - x_2^2)\, e^{-x_1^2 - x_2^2}$ в интервале [−5; +5] с точностью 0,001
8	Определить максимум функции $-7(x_1/3 - x_1^3 - x_2^2) - e^{-(x_1+1)^2 - x_2^2}$ в интервале [−1; +1] с точностью 0,01
9	Определить максимум $2(1-x_1)^2 e^{-x_2^2}$ в интервале [−13; +13] с точностью 0,001
0	Определить максимум функции $-1/5x_2\, e^{-(x_1+1)}$ в интервале [−10; +10] с точностью 0,01

Контрольные вопросы

1. Какие задачи решают с помощью ГА?
2. Что входит в понятие хромосома и функция приспособленности и от каких факторов зависит вид и размер хромосомы?
3. Какие существуют способы кодирования хромосом?
4. Что такое ген, особь, популяция, поколение, мутация, скрещивание и селекция?
5. Как связаны понятия популяция, генотип и фенотип?
6. Перечислите достоинства и недостатки ГА.

Лабораторная работа № 7
ИСКУССТВЕННАЯ НЕЙРОННАЯ СЕТЬ ХЕББА

Цель работы: ознакомление на примере работы нейронной сети Хебба с основными понятиями искусственных нейронных сетей.

7.1. Бинарные и биполярные нейроны

Искусственные нейронные сети, предназначенные для решения разнообразных конкретных задач, могут содержать от нескольких нейронов до тысяч и даже миллионов элементов. Однако уже отдельный нейрон с биполярной или бинарной функцией активации может быть использован для решения простых задач распознавания и классификации изображений. Выбор биполярного (1, –1) или бинарного (1, 0) представления сигналов осуществляется исходя из решаемой задачи и во многих случаях он равноценен. Имеется спектр задач, в которых бинарное кодирование сигналов более удобно, однако в общем биполярное представление информации более предпочтительно [10].

Поскольку выходной сигнал у двоичного нейрона принимает только два значения, то нейрон можно использовать для классификации предъявляемых изображений на два класса.

Пусть имеется множество M изображений, для которых известна корректная классификация на два класса:

$$X^1 = \{X^{11}, X^{12}, \ldots, X^{1q}\},$$

$$X^2 = \{X^{21}, X^{22}, \ldots, X^{2p}\},$$

$$X^1 \bigcup X^2 = M, X^1 \bigcap X^2 = \emptyset,$$

и пусть первому классу X^1 соответствует выходной сигнал $y = 1$, а классу X^2 – сигнал $y = -1$. Если, например, предъявлено некоторое

изображение $X^{\alpha} = (X_1^{\alpha}, ..., X_n^{\alpha}), \quad X^{\alpha} \in M$ и его взвешенная сумма входных сигналов превышает нулевое значение

$$S = \sum_{i=1}^{n} X_i^{\alpha} w_i + w_0 > 0,$$

то выходной сигнал $y = 1$ и, следовательно, входное изображение X^{α} принадлежит классу X^1. Если $S \leq 0$, то $y = -1$ и изображение принадлежит второму классу.

Возможно использование отдельного нейрона и для выделения из множества классов

$$M = \{X^1 = \{X^{11}, ..., X^{1k}\}, ..., X^i = \{X^{i1}, ..., X^{iq}\}, ..., X^p = \{X^{p1}, ..., X^{pm}\}\}$$

изображений единственного класса X^i. В этом случае полагают, что один из двух возможных выходных сигналов нейрона (например, 1) соответствует классу X^i, а второй – всем остальным классам. Поэтому, если входное изображение X^{α} приводит к появлению сигнала $y = 1$, то $X^{\alpha} \in X^i$, если $y = -1$ (или $y = 0$, если используется бинарное кодирование), то это означает, что предъявленное изображение не принадлежит выделяемому классу [9].

Система распознавания на основе единственного нейрона делит все пространство возможных решений на две области с помощью гиперплоскости

$$x_1 w_1 + x_2 w_2 + ... + x_n w_n + w_0 = 0.$$

Для двухмерных входных векторов границей между двумя классами изображений является прямая линия: входные векторы, расположенные выше этой прямой, принадлежат к одному классу, а ниже – к другому.

7.2. Правило Хебба

Для адаптации, настройки или обучения весов связей нейрона может использоваться несколько методов. Рассмотрим один из них, получивший название правило Хебба. Д. Хебб, исследуя механизмы функционирования центральной нервной системы, предположил, что обучение происходит путем усиления связей между нейронами, активность которых совпадает по времени. Хотя в биологических системах это предположение выполняется далеко не всегда и не исчерпывает всех видов обучения, однако при обучении однослойных нейросетей с биполярными сигналами оно весьма эффективно.

В соответствии с правилом Хебба, если предъявленному биполярному изображению $X = (x_1, \ldots, x_n)$ соответствует неправильный выходной сигнал y, то веса w_i, $i = \overline{1,n}$ связей нейрона адаптируются по формуле

$$w_i(t+1) = w_i(t) + x_i y, \quad i = \overline{0,n},, \tag{7.1}$$

где $w_i(t)$, $w_i(t + 1)$ соответственно вес i-й связи нейрона до и после адаптации; x_i ($i = \overline{1,n}$) – компоненты входного изображения; $x_0 = 1$ – сигнал смещения; y – выходной сигнал нейрона.

В более полной и строгой форме алгоритм настройки весов связей нейрона с использованием правила Хебба выглядит следующим образом:

Шаг 1. Задается множество $M = \{(X^1, t^1), \ldots, (X^m, t^m)\}$, состоящее из пар (входное изображение $X^k = \left(x_1^k, \ldots, x_n^k\right)$, необходимый выходной сигнал нейрона t^k), $k = \overline{1,m}$). Инициируются веса связей нейрона:

$$w_i = 0, \quad i = \overline{0,n}.$$

Шаг 2. Для каждой пары (X^k, t^k), $k = \overline{1,m}$ пока не соблюдаются условия останова, выполняются шаги 3–5.

Шаг 3. Инициируется множество входов нейрона:

$$x_0 = 1,\ x_i = x_i^k,\ \ i = \overline{1,n}.$$

Шаг 4. Инициируется выходной сигнал нейрона $y = t^k$.

Шаг 5. Корректируются веса связей нейрона по правилу

$$w_i\ (new) = w_i\ (old) + x_i\, y, \quad i = \overline{0,n}\,.$$

Шаг 6. Проверка условий останова.

Для каждого входного изображения X^k рассчитывается соответствующий ему выходной сигнал y^k:

$$y^k = \begin{cases} 1, & \text{если} \quad S^k > 0, \\ -1, & \text{если} \quad S^k \le 0, \end{cases} \qquad k = \overline{1,m}\,,$$

где $S^k = \sum_{i=1}^{n} x_i^k w_i + w_0$.

Если вектор $(y^1, \ldots, y^m)$ рассчитанных выходных сигналов равен вектору $(t^1, \ldots, t^m)$ заданных сигналов нейрона, т.е. каждому входному изображению соответствует заданный выходной сигнал, то вычисления прекращаются (переход к шагу 7), если же $(y^1, \ldots, y^m) \neq (t^1, \ldots, t^m)$, то переход к шагу 2.

Шаг 7. Останов.

Пример 7.1. Пусть требуется обучить биполярный нейрон распознаванию изображений X^1 и X^2, приведенных на рис. 7.1.

1	2	3
4	5	6
7	8	9

X^1

1	2	3
4	5	6
7	8	9

X^2

Рис. 7.1 Входные изображения X^1 и X^2

При этом нужно, чтобы изображению X^1 соответствовал выходной сигнал нейрона "+1", а изображению X^2 – сигнал "–1".

Применение алгоритма Хебба дает следующие результаты.

Шаг 1. Задается множество

$$M = \{(X^1 = (1, -1, 1, 1, 1, 1, -1, -1, 1), 1),$$
$$(X^2 = (1, 1, 1, 1, -1, 1, 1, -1, 1), -1)\};$$

инициируются веса связей нейрона: $w_i = 0,\ i = \overline{1, 9}.$

Шаг 2. Для каждой из двух пар (X^1, 1), (X^2, –1), выполняются шаги 3–5.

Шаг 3. Инициируется множество входов нейрона для изображения первой пары: $x_0 = 1,\ x_i = x_i^1,\ \ i = \overline{1, 9}.$

Шаг 4. Инициируется выходной сигнал нейрона для изображения первой пары: $y = t^1 = 1.$

Шаг 5. Корректируются веса связей нейрона по правилу Хебба $w_i = w_i + x_i^1 y \quad (i = \overline{1, n})$:

$$w_0 = w_0 + x_0 y = 0 + 1 \cdot 1 = 1;$$
$$w_1 = w_1 + x_1^1 y = 0 + 1 \cdot 1 = 1;$$
$$w_1 = w_3 = w_4 = w_5 = w_6 = w_9 = 1;$$
$$w_2 = w_2 + x_2^1 y = 0 + (-1) \cdot 1 = -1;$$
$$w_2 = w_7 = w_8 = -1.$$

Шаг 3. Инициируется множество входов нейрона для изображения X^2 второй пары: $x_0 = 1,\ x_i = x_i^2,\ i = \overline{1, 9}.$

Шаг 4. Инициируется выходной сигнал нейрона для изображения второй пары (X^2, t^2) $y = t^2 = -1.$

Шаг 5. Корректируются веса связей нейрона:

$$w_0 = w_0 + x_0 y = 1 + 1 \cdot (-1) = 0;$$
$$w_1 = w_1 + x_1^2 y = 1 + 1 \cdot (-1) = 0;$$

$$w_1 = w_3 = w_4 = w_6 = w_9 = 0;$$
$$w_2 = w_2 + x_2^2 y = -1 + 1 \cdot (-1) = -2;$$
$$w_2 = w_7 = -2;$$
$$w_5 = w_5 + x_5^2 y = 1 + (-1) \cdot (-1) = 2;$$
$$w_8 = w_8 + x_8^2 y = -1 + (-1) \cdot (-1) = 0.$$

Шаг 6. Проверяются условия останова.

Рассчитываются входные и выходной сигналы нейрона при предъявлении изображения X^1:

$$S^1 = \sum_{i=1}^{9} x_i^1 w_i + w_0 = 1 \cdot 0 + (-1) \cdot (-2) + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 2 + 1 \cdot 0 +$$
$$+ (-1) \cdot (-2) + (-1) \cdot 0 + 1 \cdot 0 + 0 = 6,$$
$$y^1 = 1, \text{ так как } S^1 > 0.$$

Рассчитываются входной и выходной сигналы нейрона при предъявлении изображения X^2

$$S^2 = \sum_{i=1}^{9} x_i^2 w_i + w_0 = 1 \cdot 0 + 1 \cdot (-2) + 1 \cdot 0 + 1 \cdot 0 + (-1) \cdot 2 +$$
$$+ 1 \cdot 0 + 1 \cdot (-2) + (-1) \cdot 0 + 1 \cdot 0 + 0 = -6,$$
$$y^2 = -1, \text{ так как } S^2 < 0.$$

Поскольку вектор $(y^1, y^2) = (1, -1)$ равен вектору (t^1, t^2), то вычисления прекращаются, так как цель достигнута – нейрон правильно распознает заданные изображения [9].

Основная идея правила (7.1) – усиливать связи, которые соединяют нейроны с одинаковой по времени активностью, и ослаблять связи, соединяющие элементы с различной активностью, может быть использована и при настройке нейросетей с бинарными элементами. Правило Хебба для однослойных бинарных сетей можно записать в виде

$$w_i(t+1) = w_i(t) + \Delta w_i, \tag{7.2}$$

где

$$\Delta w_i = \begin{cases} 1, & \text{если} \quad x_i y = 1, \\ 0, & \text{если} \quad x_i = 0, \\ -1, & \text{если} \quad x_i \neq 0 \quad \text{и} \quad y = 0. \end{cases} \tag{7.3}$$

Пример 7.2. Пусть требуется обучить бинарный нейрон распознаванию изображений X^1 и X^2 примера 7.1. При этом изображению X^1 пусть соответствует выходной сигнал нейрона «+1», а изображению X^2 – «0». Применение правила Хебба в этом случае дает следующие результаты:

Шаг 1. Задается множество

$$M = \{(X^1 = (1, 0, 1, 1, 1, 1, 0, 0, 1), 1),$$
$$(X^2 = (1, 1, 1, 1, 0, 1, 1, 0, 1), 0)\}$$

и инициируются веса связей нейрона $w_i = 0$, $i = \overline{0, 9}$.

Шаг 2. Для пар $(X^1, 1)$, $(X^2, 0)$, выполняются шаги 3–5.

Шаг 3. Инициируется множество входов нейрона элементами изображения X^1:

$$x_0 = 1, \; x_i = x_i^1, \; i = \overline{1,9}.$$

Шаг 4. Инициируется выходной сигнал нейрона для изображения X^1:

$$y = t^1 = 1.$$

Шаг 5. Корректируются веса связей нейрона с помощью соотношений (7.2), (7.3):

$$w_0 = w_0 + \Delta w_0 = 0 + 1 = 1;$$
$$w_1 = w_1 + \Delta w_1 = 0 + 1 = 1;$$
$$w_1 = w_3 = w_4 = w_5 = w_6 = w_9 = 1;$$
$$w_2 = w_2 + \Delta w_2 = 0 + 0 = 0;$$
$$w_2 = w_7 = w_8 = 0.$$

Шаг 3. Инициируется множество входов нейрона элементами изображения X^2:

$$x_0 = 1,\ x_i = x_i^2,\ \ i = \overline{1, 9}.$$

Шаг 4. Инициируется выходной сигнал нейрона для изображения X^2: $y = t^2 = 0.$

Шаг 5. Корректируются веса связей нейрона с помощью соотношений (7.2), (7.3):

$$w_0 = w_0 + \Delta w_0 = 1 + (-1) = 0;$$

$$w_1 = w_1 + \Delta w_1 = 1 + (-1) = 0;$$

$$w_1 = w_3 = w_4 = w_6 = w_9 = 0;$$

$$w_2 = w_2 + \Delta w_2 = 0 + (-1) = -1;$$

$$w_5 = w_5 + \Delta w_5 = 1 + 0 = 1;$$

$$w_7 = w_2 = -1;$$

$$w_8 = w_8 + \Delta w_8 = 0 + 0 = 0.$$

Шаг 6. Проверка условий останова.

Рассчитываются входные и выходные сигналы нейрона при предъявлении изображений X^1, X^2:

$$S^1 = \sum_{i=1}^{9} x_i^1 w_i + w_0 = 1 \cdot 0 + 0 \cdot (-1) + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 1 + 1 \cdot 0 + 0 \cdot (-1) +$$

$$+ 0 \cdot 0 + 1 \cdot 0 + 0 = 1;$$

$y^1 = 1$, так как $S^1 > 0$;

$$S^2 = \sum_{i=1}^{9} x_i^2 w_i + w_0 = 1 \cdot 0 + 1 \cdot (-1) + 1 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 0 + 1 \cdot (-1) +$$

$$+ 0 \cdot 0 + 1 \cdot 0 = -2;$$

$y^2 = 0$, так как $S^2 < 0$.

Поскольку вектор $(y^1, y^2) = (1, 0)$ равен заданному вектору

(t^1, t^2) = (1, 0), то цель достигнута и вычисления прекращаются.

Использование группы из m биполярных или бинарных нейронов A_1, …, A_m (рис. 7.2) позволяет существенно расширить возможности нейронной сети и распознавать до 2^m различных изображений. Однослойная нейронная сеть с двоичными нейронами, приведенная на рис. 7.2, может быть обучена с помощью алгоритма на основе правила Хебба. В этом случае она называется сетью Хебба.

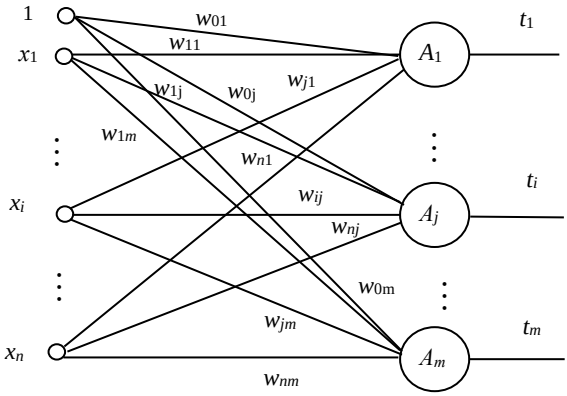

Рис. 7.2 Нейронная сеть из m элементов

Правда, применение этой сети для распознавания 2^m (или близких к 2^m чисел) различных изображений может приводить к неразрешимым проблемам адаптации весов связей нейросети. Поэтому часто рекомендуют использовать данную архитектуру для распознавания только m различных изображений, задавая каждому из них единичный выход только на выходе одного A-элемента (выходы остальных при этом должны принимать значение «–1» для биполярных нейронов или «0» – для бинарных).

Для биполярного представления сигналов возможно обучение нейросети с помощью следующего алгоритма:

Шаг 1. Задается множество $M = \{(X^1, t^1), \ldots, (X^m, t^m)\}$, состоящее из пар входных изображений $X^k = \left(x_1^k, \ldots, x_n^k\right)$ и необходимых выходных векторов $t^k = (t_1^k, \ldots, t_m^k)$, $k = \overline{1, m}$. Инициируются веса связей

нейронов:

$$w_{ij} = 0, \quad i = \overline{0, n}, \quad j = \overline{1, m}.$$

Шаг 2. Каждая пара (X^k, t^k), проверяется на правильность реакции нейронной сети на входное изображение. Если полученный выходной вектор сети (y_1^k, ..., y_m^k), отличается от заданного $t^1 = (t_1^k, \ldots, t_m^k)$, то выполняют шаги 3 – 5.

Шаг 3. Инициируется множество входов нейронов: $x_0 = 1$, $x_i = x_i^k$, $i = \overline{1, n}$.

Шаг 4. Инициируются выходные сигналы нейронов: $y_j = t_j^k$, $j = \overline{1, m}$.

Шаг 5. Корректируются веса связей нейронов по правилу:

$$w_{ij}(new) = w_{ij}(old) + x_i\, y_j, \quad i = \overline{0, n}, \quad j = \overline{1, m}.$$

Шаг 6. Проверяются условия останова, т.е. правильность функционирования сети при предъявлении каждого входного изображения. Если условия не выполняются, то переход к шагу 2 алгоритма, иначе – прекращение вычислений (переход к шагу 7).

Шаг 7. Останов.

Индивидуальные задания

Разработайте структуру сети Хебба, способную распознавать четыре различные буквы Вашего имени или фамилии. При этом обоснуйте выбор:

- числа рецепторных нейронов (число *n* *x*-элементов сети должно быть в пределах $12 \le n \le 30$);
- числа выходных нейронов;
- выбор векторов выходных сигналов.

Разработайте и создайте програму для обучения НС Хебба.

Проведите эксперименты по распознавании идеальных образцов входных данных и по распознаванию зашумленных данных. Сделайте выводы по экспериментам и отразите их в отчете.

Содержание отчета

1. Тема и цель занятия.
2. Индивидуальное задание.
3. Результаты выполнения задания.
4. Выводы по работе.

Контрольные вопросы

1. Точки {(4, –1), (8, –2), (1, 1), (3, 6)} принадлежат к классу *A*, а точки {(–8, 4), (–2, –3), (–1, –1), (2, –9)} – классу *B*. Какой будет минимальная сеть, правильно классифицирующая эти точки: а) нейрон с двумя входами; б) нейрон с четырьмя входами; в) однослойная сеть из двух нейронов с четырьмя входами; г) двухслойная сеть с двумя входами, двумя нейронами в скрытом слое и одним нейроном в выходном слое?

2. От чего зависит выбор числа входных нейронов для задачи распознавания образов?

3. От чего зависит выбор числа выходных нейронов для задачи распознавания образов?

4. Сколько весовых коэффициентов равны единицы после первой эпохи обучения нейронной сети Хебба для двух образов (см. рис. 7.3 ниже), если первый образ кодируется как y1 = 0, второй y2 = 1?

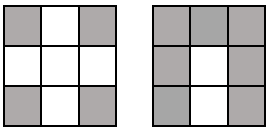

Рис. 7.3 Распознаваемые образы

Лабораторная работа № 8
МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ

Цель работы: знакомство с мультиагентной платформой JADE, освоить протокол передачи данных между агентами на платформе, запуск главного контейнера платформы и нового агента на примере агентов [10, 11].

8.1. Инструкция по выполнению лабораторной работы

Перед началом работы необходимо запустить среду разработки Eclipse IDE, загрузить и установить JDK и JADE. Запустить платформу можно двумя способами: из командной строки или в среде разработки. Ииспользовать среду разработки Eclipse IDE.

Перед тем как создавать агента, требуется установить платформу JADE. Скачать архив с платформой можно с официального сайта JADE по ссылке [11]. Скачайте последнюю версию платформы. Распакуйте архив и откройте его. Архив содержит 3 папки:

- JADE-bin – папка с исполняемыми файлами;
- JADE-lib – папка с исполняемыми файлами;
- JADE-example – папка с примерами.

Создайте отдельную папку, например, с названием Jade, и скопируйте содержимое всех трех папок из архива в новую папку (для упрощения адреса лучше всего новую папку создать в корне диска).

В результате получаем готовую платформу. Проверьте расположение таких файлов, как:

- build.xml – находится в корне папки Jade (созданной нами);
- jade.jar – находится в папке lib/;
- папки исходных файлов examples/ и jade/ – находятся в src/.

Основными элементами мультиагентной системы являются агенты. Мультиагентная платформа состоит из контейнеров, в которых живут агенты. Контейнер может не содержать ни одного агента или содержать сколько угодно агентов. Каждый контейнер обладает своим

сетевым адресом и именем.

Есть особый контейнер – главный контейнер – без него платформа не работает. Главный контейнер имеет несколько особенностей:

- он должен быть создан первым;
- он должен включать в себя двух специальных агентов:

AMS (agent management system) – агент, который управляет остальными агентами; он способен создавать и останавливать агентов;

DF (directory facilitation) – по сути представляет собой желтые страницы, в которых записываются адреса, имена и возможности агентов в системе.

8.2. Ход выполнение работы

Для запуска платформы необходимо запустить Eclipse IDE, создать новую рабочую среду и в ней открыть новый Java проект (рис. 8.1).

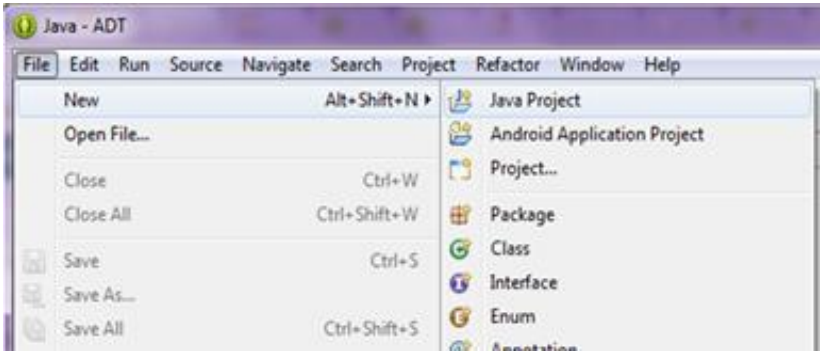

Рис. 8.1 Новый Java проект в Eclipse IDE

Далее переходим в окно создания проекта, задаем имя и нажимаем Next. В следующем окне необходимо подключить к проекту библиотеки Jade (рис. 8.2) [12].

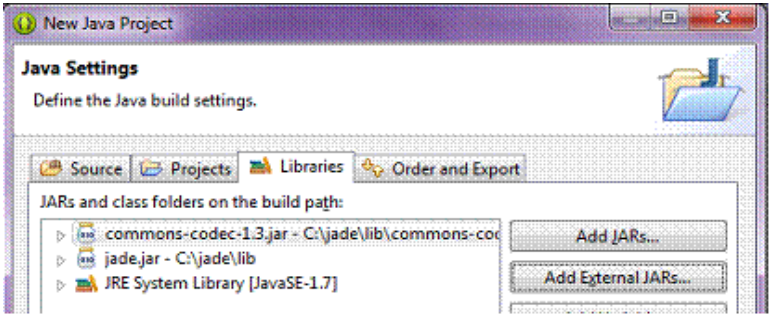

Рис. 8.2 Библиотеки Jade в проекте

Для подключения библиотек необходимо нажать Add External JARs и указать путь к библиотекам. Они находятся в папке \lib, куда были скопированы все файлы архивов платформы Jade. После этого, нажимаем Finish. Теперь к проекту подключены библиотеки JADE.

Создаем нового агента из примера «PingAgent». Для этого необходимо создать новый класс в папке \src (рис. 8.3).

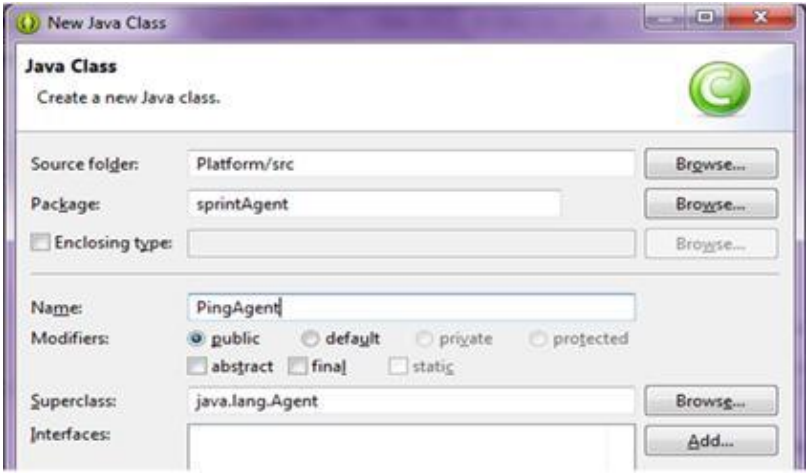

Рис. 8.3 Новый класс

Далее вводим информацию, создаем новый класс в папку Project/src в пакете sprintintel с именем PingAgent. Также на этом этапе

необходимо записать в поле "Superclass" следующую информацию: java.lang.Agent и нажать Finish.

Сейчас есть новый агент и подключены необходимые библиотеки. Необходимо задать программный код агента. Для этого открываем PingAgent.java из папки examples\PingAgent, которая находится в директории, в которую распакованы архивы Jade. Копируем все содержимое примера в нашего агента, меняем имя пакета на sprintintel и сохраняем.

Основная задача агента PingAgent – это отправлять ответ в виде слова «Ping» при запросе в виде слова «Pong».

В коде эта часть выглядит так.

```
public void action() {
  ACLMessage  msg = myAgent.receive();
  if(msg != null){
    ACLMessage reply = msg.createReply();
    if(msg.getPerformative()== ACLMessage.REQUEST){
        String content = msg.getContent();
        if ((content != null) &&
     (content.indexOf("ping") != -1)){
        myLogger.log(Logger.INFO,"Agent "
+getLocalName()+" - Received PING Request
from "+msg.getSender().getLocalName());
          reply.setPerformative(ACLMessage.INFORM);
          reply.setContent("pong");
      }
    }
  }
}
```

Для того чтобы запустить агента и мультиагентную платформу переходим к меню Run Configuration. В нем добавляем новую конфигурацию запуска Java-приложения. На рис. 8.4 и рис. 8.5 показаны параметры главного класса (MainClass) – jade.Boot и название проекта.

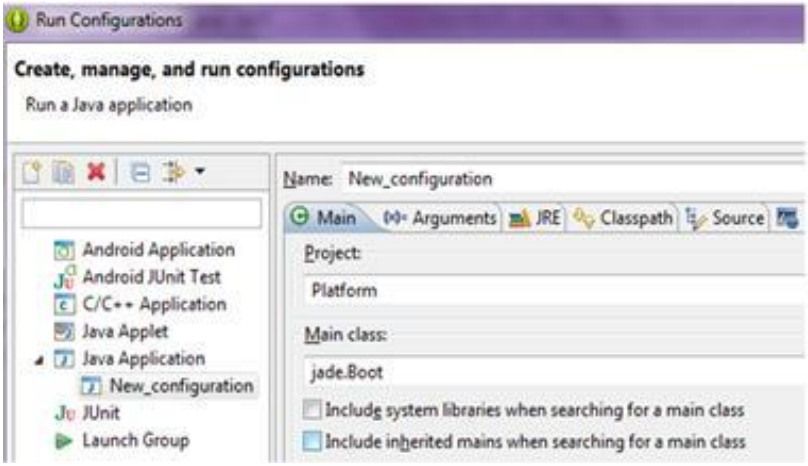

Рис. 8.4 Конфигурация запуска, вкладка Main

В поле Program Argument вкладки Arguments вводим «–gui». Таким образом, определили, что необходимо запустить платформу и главный контейнер, который уже содержит дополнительный класс агента PingAgent.

Рис. 8.5 Конфигурация запуска, вкладка Arguments

Нажимаем на кнопку Run и запускаем нашу платформу. Платформа запущена, и виден пользовательский интерфейс, в котором запущена платформа и два базовых агента AMS и DF. Чтобы добавить в главный контейнер агента «Ping», необходимо войти во вкладку Action и выбрать класс нашего агента (рис. 8.6, 8.7).

При запуске агента в главном контейнере он обращается в желтые страницы – агент DF, который записывает у себя адрес этого агента. Все запросы осуществляются через DF агента. Таким образом, в си-

стему могут входить и выходить новые агенты и быть на связи.

Рис. 8.6 Пользовательский интерфейс платформы Jade

На рис. 8.7 показано создание агента с именет «Ping».

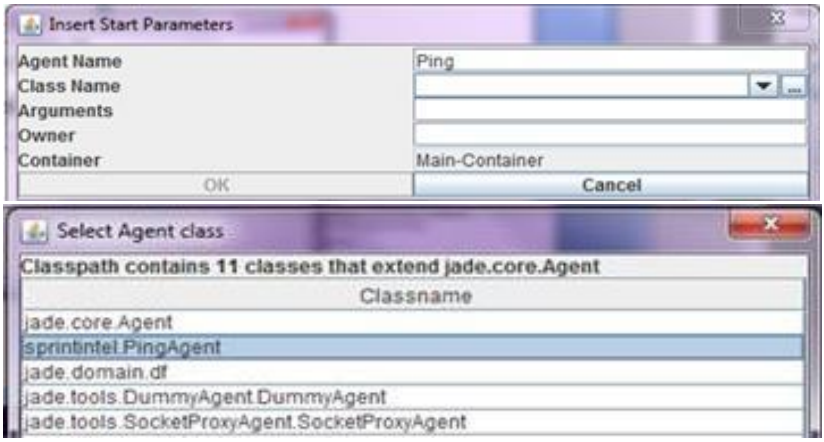

Рис. 8.7 Class Агента Ping Agent в пакете sprintintel

Проверим работу нашего агента и отправим ему запрос с помощью DummyAgent – готовый агент с пользовательским интерфейсом, посредством которого возможно отправлять сообщения другим агентам и получать от них ответы. Для запуска DummyAgent необходимо нажать кнопку StartDummyAgent в ряду кнопок быстрого доступа пользовательского интерфейса платформы. Необходимо составить запрос в форме, как это показано на рис. 8.8 и отправить нажатием кнопки: «отправить сообщение».

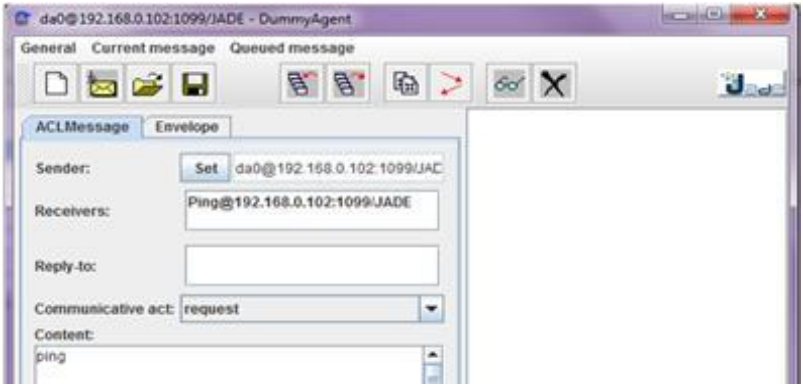

Рис. 8.8 DummyAgent, запрос на Ping агента

DummyAgent позволяет отправлять запрос с одного точного адреса агента на точный адрес другого агента, даже если они находится в различных контейнерах. Но эти агенты должны быть прописаны в одной мультиагентной системе (т.е. известны DF агентам). Справа, нажав на значок «очки», в белом поле пользовательского интерфейса DummyAgent можно увидеть, пришел ли ответ и какого он содержания (рис. 8.9).

Для того, чтобы проще было отслеживать отправку сообщений, можно запустить еще одного агента Sniffer, в котором отражаются стрелками отправленные между агентами сообщения (рис. 8.10).

Самостоятельно создайте еще одного агента Ping1 и отправьте запрос двум агентам одновременно.

Измените текст сообщения запроса и ответа.

Создайте Container1 (не главный) и создайте в нем агента.

Запустите агентов в примере папки src/examples/bookTrading. Этот пример является основой для выполнения индивидуального задания.

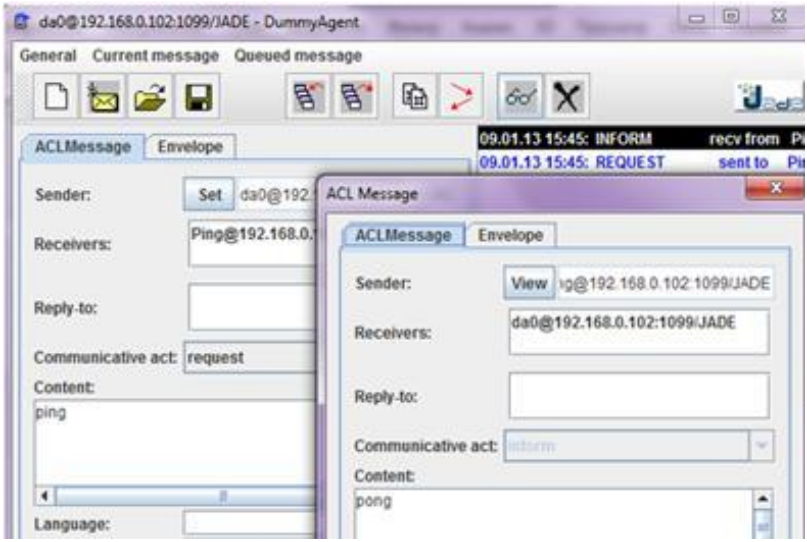

Рис. 8.9 Проверка ответа на запрос DummyAgent

Рис. 8.10 Результат работы агента Sniffer

Индивидуальные задания

Выбрать индивидуальное задание № 1 для нечетного номера (согласно номера в журнале группы и № 2 для четного номера).

1. Разработайте мультиагентное приложение «Заказы-перевозчики» в предметной области «Логистика». В фирму, выполняющую перевозки крупногабаритных грузов, поступают заказы на транспортировку разнообразных грузов на различные расстояния. Фирма располагает некоторым множеством средств перевозки (перевозчиками). Заказы характеризуются весом груза и средствами, которыми заказ располагает для оплаты перевозки.

Характеристиками перевозчика являются его максимальная грузоподъемность и продолжительность перевозки. Для выполнения каждого заказа на перевозку требуется подобрать одного перевозчика, имеющего максимальную грузоподъемность и/или обеспечивающего минимальную длительность перевозки.

В приложении должны функционировать 5-6 агента заказов и 3-5 агента перевозчиков со следующими значениями атрибутов.

Agent	Values
Project Demand_1	CargoWeight = 1000; Account = 1000
Project Demand_2	CargoWeight = 2000; Account = 1500
Vessel Resource_1	MaxWeight = 1000; CruiseTime = 5
Vessl Resource_2	MaxWeight = 2000; CruiseTime = 6
Vessel Resource_3	MaxWeight = 2500; CruiseTime = 10

2. Разработайте мультиагентное приложение для предметной области «Размещение заказов на оборудование». Двум фирмам требуется закупить оборудование. С помощью информации двух заводов-изготовителей были определены показатели функционирования необходимого оборудования.

Подберите для каждой фирмы завод-изготовитель.

Фирма	Требования фирмы		
	Стоимость, грн.	Производительность, у.е.	Надежность, у.е.
1	5	100	8
2	6	150	5

Варианты оборудования	Показатели эффективности оборудования изготовителя	
	Производительность, у.е.	Надежность, у.е.
1	120	9
2	200	6

Содержание отчета

1. Тема лабораторного занятия.
2. Индивидуальное задание.
3. Результаты выполнения индивидуального задания.
4. Выводы по работе.

Контрольные вопросы

1. Что такое платформа и контейнер JADE?
2. Для чего предназначены агенты AMS и DF?
3. Какие функции выполняет класс «Агент» JADE?
4. Какие функции выполняет класс «Поведение агента» JADE?
5. Как запланировать операцию агента JADE на заданое время?
6. Какие функции выполняет класс «Взаимодействие между агентами» JADE?
7. Как реализовать ожидание сообщения агентом?
8. Как реализовать выбор сообщения с указанными характеристиками из очереди сообщений агента?
9. Какие функции выполняет сервис «желтых страниц»?
10. Как опубликовать сервис?

Список рекомендуемой литературы

1. Bow S.-T. Pattern Recognition and Image Preprocessing / S.T. Bow. – NewYork : Dekker. – 2002. – 698 p.

2. Николенко С.И. Самообучающиеся системы / С.И. Николенко, А.Л. Тулупьев. – Москва : МЦНМО. – 2009. – 288 с.

3. Бессмертный И.А. Системы искусственного интеллекта : учебное пособие / И.А. Бессмертный. – Москва : Издательство Юрайт. – 2018. – 130 с.

4. Павлов Н.С. Системы искусственного интеллекта : учеб. пособие. В 2-х частях. / С.Н. Павлов. – Томск : Эль Контент. – 2011. – Ч. 1. – 176 с.

5. Кравець В.О. Вступ до експертних систем / В.О. Кравець, І.П. Хавіна. – Харьков : НТУ ХПІ. – 2006. – 232 с.

6. Гаврилов А.В. Системы искусственного интеллекта: Учебное пособие: / А.В. Гаврилов. – Новосибирск : НГТУ. – 2001. – Ч. 1. – 67 с.

7. Верлань А.Ф. Эволюционные методы компьютерного моделирования / А.Ф. Верлань, В.Д. Дмитриенко, Н.И. Корсунов, В.А. Шорох. – Київ : Наукова думка. – 1992. – 256 с.

8. Гладков Л.А. Генетические алгоритмы / Л.А. Гладков, В.П. Курейчик, В.М. Курейчик. – Москва : Физматлит. – 2006. – 320 с.

9. Дмитриенко В.Д. Основы теории нейронных сетей / В.Д.Дмитриенко, Н.И. Корсунов. – Белгород : БИИММАП. – 2001. – 159 с.

10. Швецов А.Н. Агентно-ориентированные системы: виртуальные сообщества : монография / А.Н. Швецов. – Вологда : ВоГУ. – 2014. – 168 с.

11. JAVA Agent Development Framework [Электронный ресурс] Java Site. – 2012. – Режим доступа: http://jade.tilab.com/.

12. JADE Tutorials & Guides. [Электронный ресурс] Jade Site. – 2018. – Режим доступа: http://jade.tilab.com/documentation/tutorials-guides/