



Гибридные системы, основанные на мягких вычислениях

Монография

Дилноз Тулкуновна Мухамедиева

Гибридные системы, основанные на мягких вычислениях

В работе рассмотрены актуальные теоретико-методические проблемы разработки методов и алгоритмов построения гибридных моделей принятия решений при интеллектуальном анализе. Мягкие, или гибридные системы, такие как нечеткие нейронные сети с генетической настройкой параметров, демонстрируют взаимное усиление достоинств и погашение недостатков отдельных методов. Очевидно, что представление знаний в нейронных сетях в виде матриц весов не позволяет строить объяснение проделанного распознавания или прогнозирования, в то время как системы вывода на базе нечетких правил позволяют строить объяснения как обратные протоколы вывода. Нейронные сети обучаются с помощью универсального алгоритма, то есть трудоемкое извлечение знаний заменяется сбором достаточной по объему обучающей выборки. Изложение ведется в достаточно строгой и в то же время доступной форме. Все основные положения и операции иллюстрируются большим числом примеров. Книга рассчитана на широкий круг читателей, включающий специалистов по прикладной математике, инженеров, а также лиц, интересующихся вопросами обработки или анализа изображений и текстов.

Дилноз Тулкуновна Мухамедиева – доктор технических наук, профессор национального исследовательского университета «Ташкентский институт инженеров ирригации и механизации сельского хозяйства». Автор более 30 монографий, 350 статей в области искусственного интеллекта.



Дилноз Тулкуновна Мухамедиева

Гибридные системы, основанные на мягких вычислениях

Дилноз Тулкуновна Мухамедиева

**Гибридные системы,
основанные на мягких
вычислениях**

Монография

Palmarium Academic Publishing

Imprint

Any brand names and product names mentioned in this book are subject to trademark, brand or patent protection and are trademarks or registered trademarks of their respective holders. The use of brand names, product names, common names, trade names, product descriptions etc. even without a particular marking in this work is in no way to be construed to mean that such names may be regarded as unrestricted in respect of trademark and brand protection legislation and could thus be used by anyone.

Cover image: www.ingimage.com

Publisher:

Palmarium Academic Publishing

Is a trademark of

Dodo Books Indian Ocean Ltd. And Omni Scriptum S.R.L publishing group

120 High Road, East Finchley, London, N2 9ED, United Kingdom
Str.Armeneasca 28/1, office 1, Chisinau MD-2012, Republic of Moldova,
Europe

Printed at: see last page

ISBN: 978-620-2-39595-3

Copyright © Дилноз Тулкуновна Мухамедиева

Copyright © 2023 Dodo Books Indian Ocean Ltd. And OmniScriptum S.R.L publishing group

Содержание

Введение.....	6
Глава 1. НЕЧЕТКОЕ МНОЖЕСТВО.....	9
1.1. Нечеткая логика.....	9
1.2. Предварительная обработка изображений с использованием аппарата нечетких множеств.....	28
1.3. Вопросы улучшения контраста изображений с использованием аппарата нечетких множеств.....	46
1.4. Сегментация цветных изображений на основе кластеризации.....	62
1.5. Интуитивная нечеткая обработка изображений.....	72
Глава 2. ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ.....	99
2.1. Общие понятия искусственных нейронных сетей.....	99
2.2. Типы активационных функций.....	101
2.3. Проблема функции «исключающее или».....	103
2.4. Многослойные нейронные сети.....	107
2.5. Обучение нейронных сетей.....	111
2.6. Процедура обратного распространения.....	116
2.7. Сети встречного распространения.....	136
2.8. Нейронные сети Хопфилда и Хэмминга.....	145
2.9. Нейросетевое распознавание объектов на изображениях топологических слоев интегральных микросхем.....	155
2.10. Выделение объекта на изображении.....	157
Глава 3. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ.....	198
3.1. Анализ особенности генетических алгоритмов.....	198
3.2. Исследование сходимости генетических алгоритмов к глобальному оптимуму.....	219
3.3. Решение многокритериальной оптимизации с использованием генетических алгоритмов с искусственным отбором.....	233

3.4. Решения практических задач оптимизации в области телекоммуникаций и обработки изображений на основе использования генетического алгоритма с искусственным отбором.....	241
3.5. Применение генетического алгоритма для фрактального сжатия изображения.....	269
Глава 4. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ РАЗРАБОТАННЫХ ПОДХОДОВ И ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ИХ РАБОТЫ.....	292
4.1. Улучшение качества изображений на основе применения эволюционирующей нейронной сети, вейвлет-преобразования и генетического алгоритма.....	292
4.2. Построение моделей интеллектуального анализа состояний слабоформализуемых процессов.....	300
4.3. Методы и алгоритмы автоматического формирования нечетких систем управления.....	309
4.4. Применение алгоритмов природных вычислений к решению задач оптимизации.....	331
4.5. Решение задачи нечеткой многокритериальной оптимизации в условиях риска.....	352
4.6. Численное решение некорректно поставленных задач с применением природных алгоритмов.....	363
4.7. Принятие слабоструктурированных решений в задачах селекции на основе нечеткого интеграла.....	370
4.8. Как рисовать с помощью нейросетей.....	377
4.9. Сервисы для генерации текста.....	408
Заключение.....	460
Список использованной литературы.....	461

Введение

Искусственный интеллект (ИИ) - это область компьютерной науки, которая изучает, как компьютерные системы могут моделировать человеческий интеллект и поведение. Она включает в себя создание алгоритмов и моделей, которые могут обрабатывать информацию и принимать решения, которые обычно требуют человеческого интеллекта.

Искусственный интеллект использует множество технологий, включая машинное обучение, глубокое обучение, нейронные сети, алгоритмы обработки естественного языка и компьютерное зрение, чтобы создать системы, которые могут выполнять различные задачи и функции. Например, ИИ может быть использован для распознавания голоса и обработки естественного языка, обнаружения и классификации изображений, управления автоматическими системами, такими как роботы и автомобили, и многих других задач.

В настоящее время искусственный интеллект активно применяется во многих областях, таких как здравоохранение, финансы, промышленность, транспорт, образование и т.д. Он также имеет множество потенциальных применений в будущем, например, в различных аспектах научных исследований, создания новых технологий и улучшения качества жизни людей.

Термин «мягкие вычисления» введен Лофти Заде в 1994 году. Это понятие объединяет такие области, как нечеткая логика, нейронные сети, вероятностные рассуждения, сети доверия и эволюционные алгоритмы, которые дополняют друг друга и используются в различных комбинациях или самостоятельно для создания гибридных интеллектуальных систем [1].

Нечеткие множества (или нечеткие логики) - это раздел теории множеств и математической логики, который позволяет работать с

понятиями, которые не имеют четкой граничной области определения. В отличие от классической теории множеств, где каждый элемент принадлежит или не принадлежит множеству, нечеткие множества позволяют описывать понятия, которые могут быть частично и недостаточно определены.

Каждый элемент нечеткого множества описывается степенью принадлежности, которая является числовым значением, выражающим насколько данный элемент принадлежит множеству. Степень принадлежности может быть любым числом на интервале $[0,1]$, где 0 означает, что элемент не принадлежит множеству, а 1 - что элемент полностью принадлежит множеству. Значения между 0 и 1 выражают степень неопределенности.

Нечеткие множества широко используются в технике и искусственном интеллекте, где они позволяют описывать нечеткие и неопределенные понятия, такие как "высокая температура", "быстрый автомобиль" и т.д. Они также используются для построения нечетких логических систем, которые позволяют выражать нечеткие правила и принимать нечеткие решения.

Нейронные сети - это компьютерные системы, которые моделируют работу человеческого мозга и позволяют компьютеру "учиться" на основе большого количества данных. Они состоят из большого количества взаимосвязанных узлов (нейронов), которые обрабатывают информацию и передают сигналы друг другу.

Каждый нейрон получает входные данные, обрабатывает их и передает результаты следующему нейрону в сети. Информация обрабатывается параллельно во всей сети, что делает ее очень быстрой и эффективной в решении сложных задач.

Нейронные сети используются во многих областях, включая компьютерное зрение, обработку естественного языка,

рекомендательные системы, распознавание речи, управление автономными транспортными средствами, игры и многие другие. Например, нейронные сети могут быть использованы для определения объектов на изображениях, классификации текстовых документов, прогнозирования будущих значений финансовых показателей и т.д.

Обучение нейронных сетей происходит на основе большого количества данных и требует определенного количества вычислительных ресурсов и времени. Однако благодаря быстрому развитию технологий, нейронные сети становятся все более доступными и используются во многих современных технологиях.

Предложенные сравнительно недавно — в 1975 году — Джоном Холландом генетические алгоритмы (ГА) основаны на принципах естественного отбора Ч. Дарвина. ГА относятся к стохастическим методам [4]. Эти алгоритмы успешно применяются в различных областях деятельности (экономика, физика, технические науки и т.п.). Созданы различные модификации ГА и разработан ряд тестовых функций. Рассмотреть как работают ГА, и какие проблемы остаются неразрешенными — цель данной работы.

Идеи Холланда развили его ученики Кеннет Де Йонг (Kenneth De Jong) из университета Джорджа Мейсона (Вирджиния) [2-8] и Дэвид Голдберг (David E. Goldberg) из лаборатории ГА Иллинойса. Благодаря им, был создан классический ГА, описаны все операторы и исследовано поведение группы тестовых функций (именно алгоритм Голдберга и получил название «генетический алгоритм»).

Генетические алгоритмы — это адаптивные методы поиска, которые в последнее время используются для решения задач оптимизации. В них используются как аналог механизма генетического наследования, так и аналог естественного отбора.

Глава 1. НЕЧЕТКОЕ МНОЖЕСТВО

1.1 Нечеткая логика

Первые упоминания о нечеткой логике появились в 1965 году в работах Лофте Заде. Первоначально она разрабатывалась как средство моделирования неопределенности естественного языка, однако впоследствии круг задач, в которых нечеткая логика нашла применение, значительно расширился. В настоящее время она используется для управления линейными и нелинейными системами реального времени, при решении задач анализа данных, распознавания, исследования операций [1].

Нечеткая логика представляет собой надмножество классической булевой логики. Она расширяет возможности классической логики, позволяя оперировать не только значениями "ложь" и "истина", но и значениями в промежутке между ними. Несмотря на то, что нечеткая логика использует нечеткую информацию и основана на теории нечетких множеств, ее аппарат столь же строг и точен, как и классический.

Рассмотрим основные понятия определения из теории нечетких множеств.

Теория нечетких множеств. Одним из базовых понятий нечеткой логики является понятие нечеткого множества.

Пусть X - множество, состоящее из упорядоченных пар вещественных чисел. Если $x = (x_1, x_2)$ является элементом X , то этот факт записывается в символическом виде следующим образом:

$$x \in X .$$

В противном случае, т.е. когда x не является элементом X , используется запись

$$x \notin X .$$

Множество, не содержащее ни одного элемента, называется пустым множеством и обозначается символом \emptyset . Множество задается путем перечисления его содержимого в фигурных скобках: $\{\cdot\}$. Например, записывая выражение в форме $Z = \{z \mid z = n^2 + 2, n \in N\}$, мы имеем в виду, что множество Z состоит из элементов z , которые строятся умножением на себя каждого элемента множества N . Один из способов использования множеств в обработке изображений заключается в рассмотрении элементов множества в виде координат пикселей (упорядоченные пары целых чисел), представляющих объекты или другие интересующие признаки на изображении.

Если все элементы множества A являются также элементами другого множества B , то говорят, что A есть подмножество множества B , что символически обозначается так:

$$A \subseteq B.$$

Объединение двух множеств A и B , которое обозначается через

$$C = A \cup B,$$

есть по определению множество всех элементов, принадлежащих либо множеству A , либо множеству B , либо обоим множествам одновременно. Аналогично пересечение двух множеств A и B , которое обозначается через

$$D = A \cap B,$$

есть по определению множество всех элементов, принадлежащих одновременно обоим множествам A и B . Два множества A и B называются непересекающимися или взаимоисключающими, если у них нет общих элементов. В этом случае

$$A \cap B = \emptyset.$$

Множество всех элементов в данном приложении называют универсальным множеством. Например, при работе с множеством вещественных чисел универсальным множеством является числовая прямая, содержащая все вещественные числа. В цифровой обработке изображений в качестве универсального множества обычно определяется прямоугольник, содержащий все пиксели изображения.

Дополнение множества A есть множество элементов, не содержащихся в A :

$$\tilde{A} = \{z | z \notin A, z \in U\}.$$

Разность двух множеств A и B обозначается через $A \setminus B$ и определяется следующим образом:

$$C = A \setminus B = \{z | z \in A, z \notin B\}.$$

Видно, что это множество состоит из элементов A , которые не входят во множество B . Можно в качестве примера определить дополнение в терминах универсального множества и разности множеств: $\tilde{A} = U \setminus A$.

Следует отметить, что все элементы множества имеют одно и то же значение, поскольку в определениях теоретико-множественных операций не участвуют значения яркости (например, мы не определили, какую яркость имеют элементы пересечения двух множеств). Однако можно говорить о принадлежности пикселей множеству на основе только координат, предполагая, что все элементы множеств имеют одинаковую яркость.

При работе с полутоновыми изображениями изложенные выше понятия неприменимы, потому что необходимо указать значения всех пикселей для результата операции над множествами. На самом деле, в

случае полутонов операции объединения и пересечения обычно определяются как соответственно максимум и минимум для пары соответственных пикселей, а дополнение определяется как попарные разности между константой и яркостью каждого пикселя. Тот факт, что мы имеем дело с парами соответственных пикселей, указывает на поэлементный характер операций над полутоновыми множествами, как это определялось в подразделе 1.1.1. Рассмотрим пример по применению операции над множествами с учетом яркости изображений, который иллюстрирует теоретико-множественные операции над полутоновыми изображениями.

Пусть пиксели полутонового изображения представляются множеством A , элементами которого являются тройки вида (x, y, z) , где x и y – пространственные координаты, а z обозначает яркость в точке с указанными координатами. Можно определить дополнение A как множество $\tilde{A} = \{(x, y, L - z) | (x, y, z) \in A\}$, которое есть просто множество пикселей A с яркостями, полученными вычитанием их исходных значений из константы $L (L = 2^k - 1, \text{ где } k - \text{ число бит, используемых для представления яркости } z)$. Предположим, что в качестве A выступает 8-битовое полутоновое изображение и требуется сформировать негатив A с помощью операций над множествами. Для этого построим множество $\tilde{A} = \{(x, y, 255 - z) | (x, y, z) \in A\}$.

Объединение двух полутоновых множеств A и B определяется как множество

$$A \cup B = \left\{ \max_z(a, b) | a \in A, b \in B \right\},$$

где a, b – яркость изображений A и B соответственно в координатах (x, y) .

В результате применения операции объединения двух полутоновых изображений получается массив, сформированный из максимальных значений яркости каждой пары соответственных пикселей.

1.1.1. Логические операции

При работе с двоичными изображениями можно говорить о множествах пикселей фона (со значениями 0) и переднего плана (со значениями 1). Если определить объекты как области, состоящие из пикселей переднего плана, то операции над множествами превращаются в операции над координатами объектов в бинарном изображении. Имея дело с бинарными изображениями, обычно говорят об объединении, пересечении и дополнении как о логических операциях OR (ИЛИ, логическое сложение), AND (И, логическое умножение) и NOT (НЕ, отрицание). Слово «логические» указывает на происхождение из формальной логики, в которой 1 и 0 означают соответственно истинное и ложное значения.

Рассмотрим две области, т.е. множества A и B , состоящие из пикселей переднего плана. Операция OR над этими двумя множествами дает множество элементов (пар координат), принадлежащих либо A , либо B , либо обоим множествам одновременно. Операция AND дает множество элементов, являющихся общими для A и B . Операция NOT в применении к множеству A дает множество элементов, не принадлежащих A . Поскольку речь идет об изображении, где A есть данное множество пикселей переднего плана, множество NOT(A) состоит из всех пикселей изображения, не входящих в A , т.е. всех пикселей фона и, возможно, других пикселей переднего плана. Эту операцию можно рассматривать как установку значений всех элементов A в 0 (черное), а элементов вне A в 1 (белое).

Известно, что для методов, основанных на логических операциях, достаточно рассмотреть только три логические операции AND, OR и NOT, поскольку они образуют функционально полный класс.

Рассмотренные выше теоретико-множественные и логические действия оперируют четкими множествами, т.е. обрабатываемый элемент либо принадлежит, либо не принадлежит множеству. В некоторых приложениях это сильно ограничивает возможности применения методов обработки, основанных на теоретико-множественных и логических операциях.

Известно [1], что множество рассматривается как некоторая совокупность объектов (элементов), а теория множеств состоит из набора приемов, которые имеют отношение к операциям над и между множествами. Несомненно, что на базе теории множеств и математической логики построены аксиоматические основы классической математики. Одним из основных понятий в теории множеств является понятие принадлежности элемента к множеству.

Обычно при решении ряда задач используются так называемые «четкие» множества. Принадлежность их элементов может быть только истинной или ложной (истина обычно обозначается как 1, а ложь - как 0) (рис. 1.1). Например, пусть X означает множество всех представителей. Требуется определить подмножество A в Z , называемое «множеством молодых преподавателей».

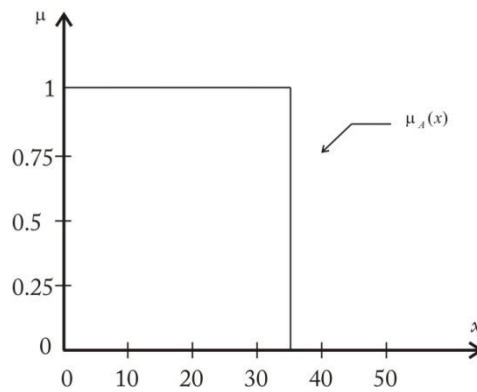


Рис. 1.1. Функция принадлежности, используемая при порождении четкого множества: μ - степень принадлежности; x - возраст; A - множество

Для определения этого подмножества необходимо задать функцию принадлежности элемента A . Заданная функция принадлежности определяет элементы A . При этом каждому элементу x из X присваивается определенное значение (точки 1 или 0). Известно, что в классической математике имеют дело с булевой логикой. Поэтому функция принадлежности задается просто порогом, определяемым в некоторой области; преподаватель, возраст которого ниже или совпадает со значением порога, признается молодым, а выше – не молодым. Этот подход, иллюстрируется рис. 1.68. На этом примере использован возрастной порог в 35 лет. На этом рисунке рассмотренная функция принадлежности обозначена через $\mu_A(z)$.

Необходимо отметить, что функции принадлежности в классической математике называются характеристическими функциями.

Нетрудно заметить некоторые трудности, которые возникают с этим определением: преподаватель возрастом в 35 лет признается молодым, а преподаватель в 30 лет и 1 минута уже не относится к множеству молодых специалистов. Данное обстоятельство является

основной проблемой в работе с четкими множествами, которая ограничивает использование классической теории множеств во многих практических приложениях. Для определения функции принадлежности требуется большая гибкость, использование которой поможет описать постепенный переход от молодого к не молодому. Один из вариантов этой функции приведен на рис. 1.2. Ключевым свойством функции принадлежности должна быть его многозначность, которая обеспечивает непрерывный переход между понятиями «молодой специалист» и «не молодой специалист». Это позволяет представить степень «молодости». После этого можно ввести такие понятия как: «молодой специалист» (верхняя плоская часть графика), «сравнительно молодой специалист» (верхняя часть склона), «не слишком молодой специалист» (нижняя часть склона) и так далее.

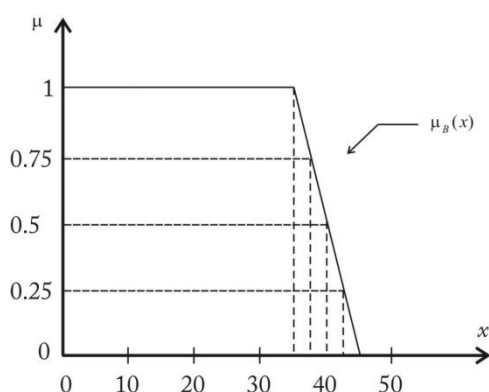


Рис. 1.2. Функция принадлежности, используемая при порождении нечетного множества: μ - степень принадлежности; x - возраст;

A - нечеткое множество

Отметим, что уменьшение крутизны спада функции принадлежности на рис. 1.2 дает больше неопределенности в том, что необходимо понимать под «молодым специалистом». Эти типы неопределенности (нечеткости) в определениях лучше согласуются с

тем, что имеется в виду, когда люди говорят неточно о возрасте. Таким образом, функцию принадлежности многозначности можно интерпретировать как основу нечеткой логики, а множества, возникающие в результате использования этой логики, могут быть рассмотрены как нечеткие множества. Эти идеи формализованы и развиты в работах [35-40].

Очень часто при описании какого-либо объекта возникают такие ситуации, когда мы не можем достаточно точно определить численные значения его параметров, а можем лишь задать их приблизительные значения. К примеру, глубина реки «больше 80» метров, Петрову «около 30» лет и т.д. Это все есть примеры так называемых нечетких чисел.

Нечеткое число \mathbf{A} – это нечеткое множество действительных чисел с нормальной, выпуклой и непрерывной функцией принадлежности ограниченного опорного множества.

В зависимости от того, как мы определяем нечеткое число $s \in \mathbf{S}$ – « s приблизительно равен a » или « s приблизительно лежит в интервале $[a;b]$ », – оно может быть представлено либо как триангулярное, либо как трапецеидальное.

Нечеткое множество \mathbf{A} называется триангулярным нечетким числом с вершиной (или центром) a , шириной слева $\alpha > 0$ и шириной справа $\beta > 0$, если его функция принадлежности имеет вид

$$A(t) = \begin{cases} 1 - (a-t)/\alpha, & a - \alpha \leq t \leq a \\ 1 - (t-a)/\beta, & a \leq t \leq a + \beta \\ 0, & t < a - \alpha, t > a + \beta \end{cases} \quad (1.1)$$

и мы используем обозначение $\mathbf{A} = (a, \alpha, \beta)$. Опорным множеством множества \mathbf{A} является $(a - \alpha, a + \beta)$. Данное нечеткое число с центром a можно понимать как нечеткую величину « s приблизительно равно a ».

Нечеткое множество \mathbf{A} называется трапецидальным нечетким числом с интервалом допуска $[a,b]$, шириной слева $\alpha > 0$ и шириной справа $\beta > 0$, если его функция принадлежности имеет вид

$$A(t) = \begin{cases} 1 - (a-t)/\alpha, & a - \alpha \leq t \leq a \\ 1, & a \leq t \leq b \\ 1 - (t-b)/\beta, & a \leq t \leq b + \beta \\ 0, & t < a - \alpha, t > b + \beta \end{cases}, \quad (1.2)$$

и мы используем обозначение $\mathbf{A} = (a, b, \alpha, \beta)$. Опорным множеством множества \mathbf{A} является $(a - \alpha, b + \beta)$. Данное нечеткое число можно понимать как нечеткую величину «с приблизительно находится в интервале $[a,b]$ ».

Нечеткие системы управления. Нечеткие системы управления (нечеткие контроллеры, НСУ) являются наиболее важным приложением теории нечетких множеств. Функционирование нечетких логических контроллеров отличается от работы обычных контроллеров. Это отличие состоит в том, что обычные контроллеры основаны на аналитическом выражении, описывающем объект управления, в то время как нечеткие контроллеры используют знания экспертов. Эти знания могут быть выражены естественным образом с помощью так называемых лингвистических переменных.

Лингвистические переменные могут рассматриваться либо как переменные, значения которых являются нечеткими числами, либо как переменные, значения которых определяются в лингвистических терминах [1].

Лингвистические переменные характеризуются пятеркой $(x, T(x), U, G, M)$, в которой x – имя переменной, $T(x)$ – множество термов x , т.е. множество имен лингвистических величин x , каждое значение которой есть нечеткое число, определенное на U , G – синтаксические правила для выработки имен величин x , M –

семантические правила для связывания каждой величины с ее смыслом.

Предназначение НСУ состоит в том, чтобы следить за значениями переменных состояния управляемой системы и получать величины переменных управления путем определенных связей, которые представляют собой БП системы.

Нечеткие логические системы управления обычно состоят из четырех основных частей: Интерфейс фаззификации (введение нечеткости на четких входных данных), База нечетких правил, Машина нечеткого вывода и Интерфейс дефаззификации (исключение нечеткости для обеспечения четкого выхода) [1].

Интерфейс фаззификации – оператор фаззификации переводит четкие данные в нечеткие множества.

База нечетких правил – содержит информацию о связях нечетких входных и выходных значений.

Машина нечеткого вывода – механизм сопоставления по БП нечетких значений входных параметров нечетким значениям выходных.

Интерфейс дефаззификации – оператор дефаззификации переводит нечеткие множества в четкие значения.

В общем виде принцип работу НСУ представлен на рисунке 1.3.

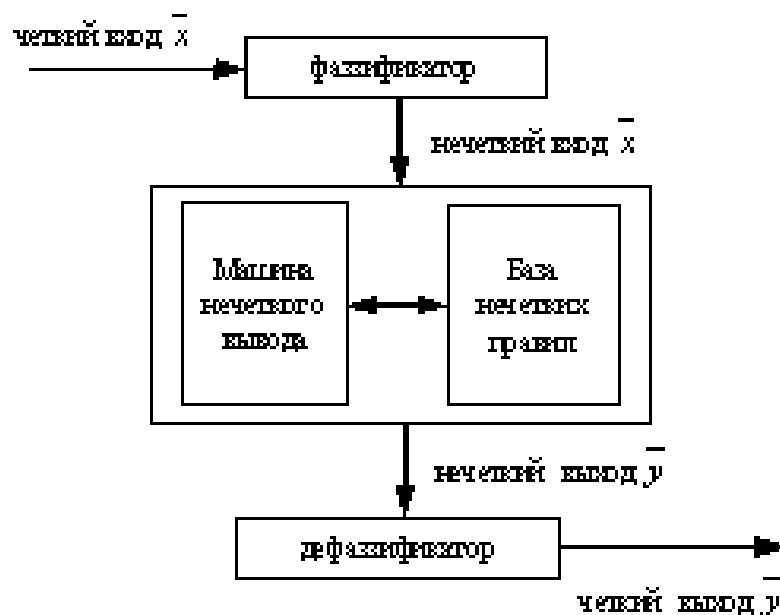


Рис. 1.3. Структурная схема нечеткого логического контроллера

1.1.2. Подавление шума упорядочивающими фильтрами

Упорядочивающие фильтры, основанные на порядковых статистиках, представляют собой пространственные фильтры, вычисление отклика которых требует предварительного ранжирования, т.е. упорядочивания значений пикселей, заключенных внутри обрабатываемой фильтром области S_{xy} изображения. Отклик фильтра определяется по результатам упорядочивания и не может быть представлен в виде свертки.

Наиболее известным из фильтров, основанных на порядковых статистиках, является медианный фильтр. Действие этого фильтра, как следует из его названия, состоит в присвоении пикселю восстановленного изображения с координатами (x,y) значения медианы упорядоченного множества значений яркости из окрестности S_{xy} обрабатываемого изображения [4]:

$$z'(x, y) = \underset{(s,t) \in S_{xy}}{\text{med}} \{z(s, t)\}. \quad (1.3)$$

Широкая популярность медианных фильтров обусловлена тем, что они прекрасно приспособлены для подавления некоторых видов случайных шумов и при этом приводят к меньшему размыванию контуров по сравнению с линейными сглаживающими фильтрами того же размера. Медианные фильтры эффективны при наличии как биполярного, так и униполярного импульсного шума, но мало эффективны при устранении гауссова шума (рис. 1.4). Существенным недостатком медианных фильтров является подавление и искажение мелких объектов, например, регулярных структур, показанных на рис. 1.4.

Для устранения данного вида шума («соль и перец») медианная фильтрация является одним из наиболее простых и эффективных методов. Размер локальной апертуры влияет на результат обработки, но не в такой степени, как в других рассмотренных нами методах. Одно из основных преимуществ медианной фильтрации состоит в том, что она приводит к устранению импульсных выбросов, не размывая границ объектов.

Исходное изображение



Исходное изображение



Шум типа "соль и перец"



Шум типа "соль и перец"



Медиана



Медиана



a)

б)

Рис. 1.4. Медиана при размере локального окна:

a - 3×3; *б* - 9×9

Хотя медианные фильтры принадлежат к числу наиболее часто используемых в обработке изображений фильтров, основанных на порядковых статистиках, это отнюдь не единственный пример таких фильтров. Медиана (1.3) представляет собой середину упорядоченного набора чисел, однако использование иных характеристик этого набора предоставляет много других

возможностей. Например, использование крайних значений ранжированного набора чисел приводит к фильтрам максимума и минимума, основанным на выборе максимального или минимального значения яркости из окрестности S_{xy} соответственно.

Фильтр максимума полезен при обнаружении наиболее ярких точек на изображении. Кроме того, поскольку униполярный «черный» импульсный шум принимает минимальные значения, применение этого фильтра приводит к уменьшению такого шума, так как в процессе фильтрации из окрестности S_{xy} выбирается максимальное значение:

$$z'(x, y) = \max_{(s,t) \in S_{xy}} \{z(s, t)\}.$$

Соответственно, фильтр минимума полезен при обнаружении наиболее темных точек на изображении и его применение приводит к уменьшению униполярного «белого» импульсного шума:

$$z'(x, y) = \min_{(s,t) \in S_{xy}} \{z(s, t)\}.$$

На рис. 1.5, 1.6 приведены результаты обработки изображений с помощью фильтра типа максимум (рис. 1.5) и минимум (рис. 1.6) по окрестности. Конечно, результаты такой обработки существенно зависят от размеров локальной апертуры.

Последовательное применение максимального и минимального фильтров одинаковым размером апертуры позволяет эффективно удалять дефекты (пятна, разрывы линии контура) на бинарных изображениях без изменения формы и размеров фрагментов, несущих полезную информацию, как это показано на рис. 1.5, 1.6. Максимальный размер удаляемых дефектов равен размеру окрестности S_{xy} .

Фильтр срединной точки, или Чебышевский, заключается в вычислении среднего между максимальным и минимальным значениями в окрестности S_{xy} :

$$z'(x, y) = \frac{1}{2} \left[\max_{(s,t) \in S_{xy}} \{z(s, t)\} + \min_{(s,t) \in S_{xy}} \{z(s, t)\} \right].$$

Исходное изображение



Исходное изображение



Шум типа "соль и перец"



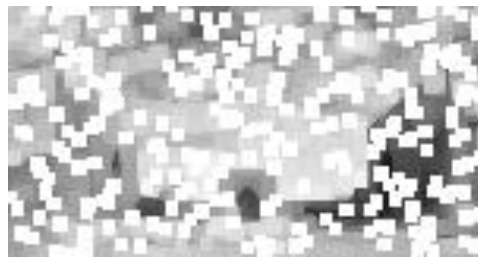
Шум типа "соль и перец"



Максимум по окрестности



Максимум по окрестности



a)

б)

Рис. 1.5. Максимум по окрестности при размере локального окна: *a* - 3×3; *б* - 9×9

Исходное изображение



Исходное изображение



Шум типа "соль и перец"



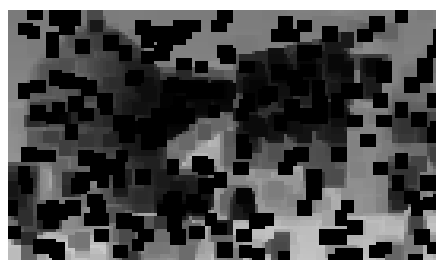
Шум типа "соль и перец"



Минимум по окрестности



Минимум по окрестности



a)

б)

Рис. 1.6. Минимум по окрестности при размере локального окна: *a* - 3×3 ; *б* - 9×9

Обработка изображений данным методом показывает его неэффективность к устранению шума типа «соль и перец». Также из представленных на рис. 1.7 результатов следует вывод, что работа фильтра типа срединная точка зависит от размеров локальной окрестности.

Исходное изображение



Исходное изображение



Шум типа "соль и перец"



Шум типа "соль и перец"



Срединная точка



Срединная точка



a)

б)

Рис. 1.7. Срединная точка при размере локального окна: *a* - 3×3; *б* - 9×9

1.1.3. Нечеткая обработка изображения

Рассмотрим нечеткое множество $\hat{A} = \{(z, \mu_A(z))\}$, где Z – дискретное множество элементов $\{z_1, \dots, z_n\}$.

При рассмотрении дискретных нечетких множеств и случайных событий следует учитывать, что функция принадлежности задается только на дискретном множестве элементов из Z или R^n . Приведем выражения для нечеткой обработки изображения:

$$z'(x, y) = \frac{\sum_{i=1}^n \sum_{(s,t) \in S_{xy}} \mu(z_i) z_i(s, t)}{\sum_{i=1}^n \mu(z_i)}.$$

На рис. 1.8 показана нечеткая обработка изображения при размере локального окна 3×3 и 9×9 .

Исходное изображение



Исходное изображение



Шум типа "соль и перец"



Шум типа "соль и перец"



Нечеткая обработка



Нечеткая обработка



а)

б)

Рис. 1.8. Нечеткая обработка изображения при размере локального окна: а - 3×3 ; б - 9×9

1.2. Предварительная обработка изображений с использованием аппарата нечетких множеств

1.2.1. Архитектура системы обработки изображений, основанной на аппарате нечетких множеств

Известно, что теория нечетких множеств оперирует качественными понятиями. Это характерно для человека, вместе с тем она дает им количественную оценку, что характерно для компьютера. Этим она объединяет достоинства человеческого оперирования знаниями и вычислительной мощностью компьютера. Нечеткая логика, которая служит основой для реализации методов нечеткой обработки изображений, при решении задач анализа и распознавания изображений более естественно описывает характер человеческого мышления и ход его рассуждений, чем традиционные формальные системы. Именно поэтому использование математических средств для представления нечеткой исходной информации позволяет строить модели, которые наиболее адекватно отражают различные аспекты неопределенности, постоянно присутствующей в окружающей нас реальности. Однако при всем своем потенциале теория нечетких множеств не была способна решать практические задачи, а существовала действительно всего лишь как теория. Ситуация изменилась в последние 2-3 десятка лет и это связано с появлением прикладных реализаций для решения интеллектуальных задач с использованием теории нечетких множеств [59-65]. Следует отметить, что при решении задач с использованием аппарата нечеткой логики аксиомы вводятся по ходу решения задачи и формируются, главным образом, из эмпирических знаний человека.

В данном разделе представлены состав и структура системы обработки изображений с использованием концепции нечетких множеств. Сначала, в целях простоты изложения, рассмотрим систему

нечеткой обработки изображений как черный ящик, который представляет собой удобную схему - преобразование входного пространства в выходные (рис. 1.9). При этом изображение рассматривается как нечеткое множество и обрабатывается методами «мягкого вычисления».



Рис. 1.9. Схема преобразования входных изображений в выходные

На основе цели и задач исследований определена архитектура системы обработки изображений, основанной на концепции нечетких множеств. Функциональная схема этой системы представлена на рис. 1.10.

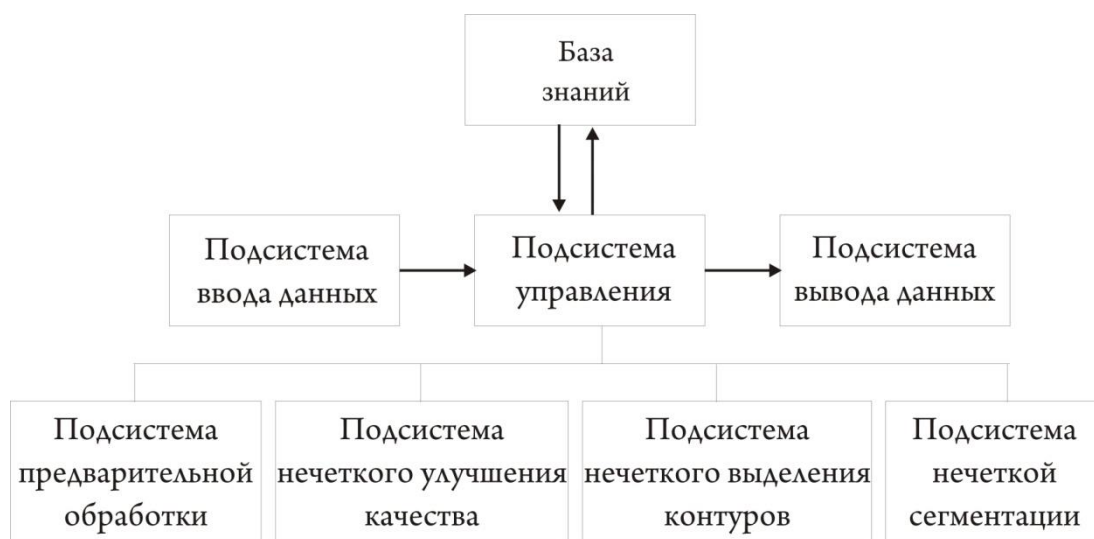


Рис. 1.10. Функциональная схема разрабатываемой системы «FuzzyIP»

В основу разрабатываемой системы обработки изображений заложены основные принципы разработки программной системы промышленного назначения. Эти принципы заключаются в следующем.

1. *Принцип эффективности.* Система должна обеспечить эффективный выбор схемы организации системы и математических методов решения задачи.

2. *Принцип простоты эксплуатации.* Максимальное удобство и простота общения пользователя с системой осуществляются входным языком, позволяющим компактно сформулировать задания.

3. *Принцип типизации и стандартизации.* Система должна учесть внесение корректив и доработок модулей при построении функциональных подсистем из стандартных элементов.

4. *Принцип формализации.* В системе задание на решение любой задачи должно производиться по общему правилу.

5. *Принцип надежности.* Система должна гарантировать достоверность результатов решения задач, что обеспечивается средством контроля вычислительного процесса на всех этапах работы ПК.

Реализация перечисленных принципов обеспечивает:

1) выбор соответствующей формальной модели в зависимости от условий задачи и исходного материала;

2) возможность работы всех функциональных подсистем системы в одном сеансе;

3) автономную работу функциональных подсистем системы;

4) простоту изменений, исправлений и расширений функциональных подсистем системы в рамках неизменности организации программ;

5) управление функционированием ПК с помощью центральной информационной программы управления.

Структура разрабатываемой системы «FuzzyIP» состоит из вспомогательных и функциональных подсистем. В состав системы входят следующие вспомогательные подсистемы:

- управления, осуществляющей организацию информационной связи в программной системе «FuzzyIP» с потребителем, координацию и управление работ составных подсистем в соответствии с заданием;

- ввода обрабатываемых изображений и исходных данных на ЭВМ для решения конкретных практических задач;

- вывода промежуточных и конечных результатов анализа и обработки изображений, а также значений требуемых параметров в удобной для потребителя форме, с соответствующими комментариями.

Под функциональной подсистемой понимается структурный единый комплекс, ориентированный на решение определенного класса задач обработки и анализа изображений. Разрабатываемая система будет состоять из следующих функциональных подсистем:

- предварительная обработка для улучшения результатов выделения контуров и сегментации изображений с использованием концепции нечетких множеств;

- выделение контуров изображений с использованием концепции нечетких множеств;

- сегментация изображений с использованием концепции нечетких множеств.

В функциональной подсистеме предварительной обработки изображений реализованы алгоритмы повышения качества изображений с использованием концепции нечетких множеств. Эти алгоритмы ориентированы на обработку исходного изображения с

целью улучшения результатов применения алгоритмов выделения границ (контуров) и сегментации.

В функциональной подсистеме выделения контуров изображений представлен набор процедур и функций, реализующих алгоритмы нечеткого приращения первого и второго порядков.

В функциональной подсистеме сегментации реализованы алгоритмы сегментации изображений на основе нечеткого критерия однородности области и нечеткой разрывности яркостных свойств изображения при переходе от одной однородной области к другой.

Все подпрограммы, реализованные в рассмотренных подсистемах, обладают следующими свойствами:

- упорядоченность: имеется один вход в подпрограмму и один выход из нее;

- автономность: передача управления осуществляется согласно значениям параметров, выработанным внутри каждой подпрограммы.

Основные подпрограммы предназначены для решения собственно задач анализа и обработки изображений на основе концепции нечетких множеств. При этом основное внимание уделяется задачам, связанным с выделением контуров и сегментацией изображений. Они составляют операционную часть алгоритмов системы «FuzzyIP».

Вспомогательные подпрограммы служат для приведения анализируемого изображения к некоторой стандартной форме и вычисления значений вспомогательных параметров, входящих в алгоритмы основных задач, с привлечением общеизвестных процедур.

Функциональное назначение разрабатываемой системы «FuzzyIP», включающей в себя подпрограммы (набор алгоритмов) и программу-диспетчер, заключается в

автоматизированном выборе оптимального, в некотором смысле, метода решения конкретной задачи с учетом ее специфики, и эта система предназначена для решения следующих задач обработки изображений с использованием концепции нечетких множеств:

- улучшение качества изображений;
- выделение краев области на анализируемом изображении;
- формирование области по свойствам пикселя анализируемого изображения;
- выделение некоторых реальных объектов на цветном изображении.

Все рассмотренные подсистемы, независимо от их функционального назначения, имеют унифицированную структуру. Это позволяет определить общую структуру для всех функциональных подсистем. Поэтому любую подсистему разрабатываемой системы FuzzyIP можно представить как схему преобразования, которая показана на рис. 1.11.

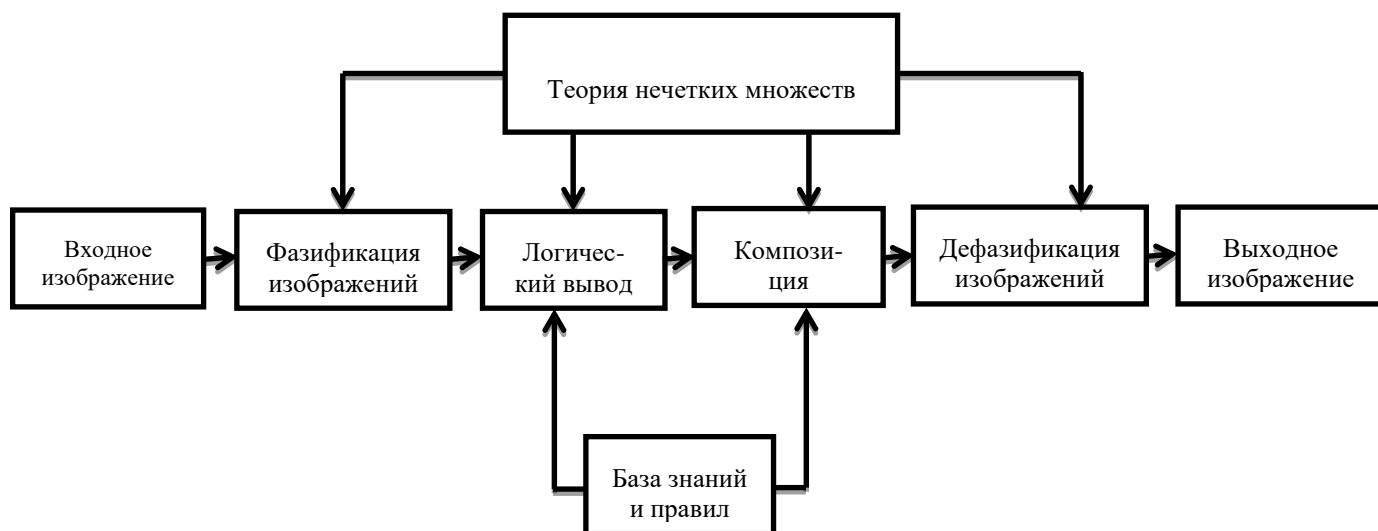


Рис. 1.11. Функциональная схема подсистемы разрабатываемой системы «FuzzyIP»

Вначале в каждой подсистеме происходит фаззификация изображения (т.е. приведение к нечеткости). Далее степень принадлежности элементов нечеткого множества (иными словами, пикселей изображения) изменяется в соответствии с базой нечетких правил. В последующем все переменные вывода объединяются для формирования одного нечеткого подмножества. И, наконец, происходит дефаззификация изображения (т.е. преобразование значений функций принадлежности в четкие значения яркости). Эти этапы обработки более подробно рассмотрены в разделе 1.2.2.

1.2.2. Основные этапы обработки изображений с использованием аппарата нечетких множеств

Предположим, что нас интересует установление правил для распознавания состояния фруктов, например, некоторого сорта абрикоса. Множество рассматриваемых абрикосов может быть разбито на три категории: незрелый, полуспелый и зрелый. При этом на основе наблюдения фруктов на разных стадиях зрелости можно сделать вывод, что незрелые абрикосы выглядят зелеными, полуспелые - желтыми, а зрелые - красными. Эти данные формализуются с помощью правил типа «ЕСЛИ-ТО». Вопросы формирования таких правил, которые представляют знания экспертов в предметной области, более подробно рассмотрены в разделе 1.2.3.

Далее рассмотрим определение способа использования входа (цвета) и базы знаний, представленной правилами «ЕСЛИ-ТО». Подобные процессы называются импликацией или логическими выводами. При этом исходные условия по каждому из правил должны быть обработаны и получено единое значение до осуществления импликации. В связи с тем, что мы имеем дело с нечеткими входами, то и выходы тоже являются нечеткими. Следовательно, для выходов

должны быть определены функции принадлежности. На рис. 1.12 приведены функции принадлежности нечетких выходов, которые мы планируем использовать в этом примере.

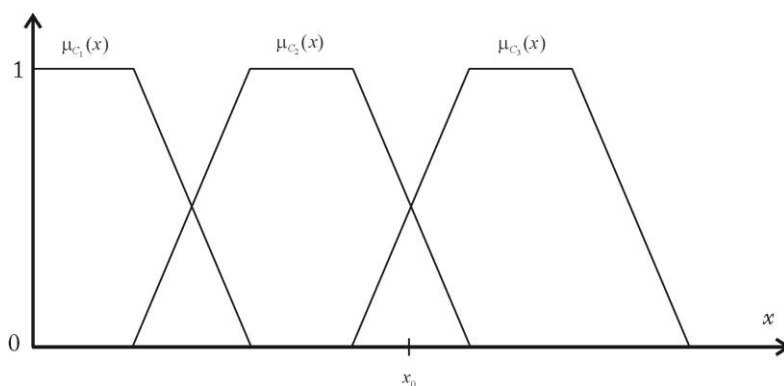


Рис. 1.12. Функции принадлежности, используемые при порождении нечеткого множества $C(C = \{C_1, C_2, C_3\})$: C - нечеткие множества цвета; C_1 - нечеткое подмножество «Зелёный»; C_2 - нечеткое подмножество «Желтый»; C_3 - нечеткое подмножество «Красный»

Определим, что независимой переменной на выходе является спелость, которая отличается от независимых переменных на входах.

Вся информация, необходимая для формирования связи между входами и выходами, находится в базе правил. Например, на рис. 1.13, 1.14 приведены функции принадлежности красного и спелого абрикоса. Для того чтобы найти результат операции «И», между этими двумя функциями определяется минимум этих функций принадлежности:

$$\mu_1(x,y) = \min\{\mu_{\text{зеленый}}(x), \mu_{\text{неспелый}}(y)\}, \quad (1.4)$$

где цифра 1 в индексе означает, что это результат применения правила D1 из базы знаний. Все правила, использованные в данном разделе, приведены в разделе 1.2.3.

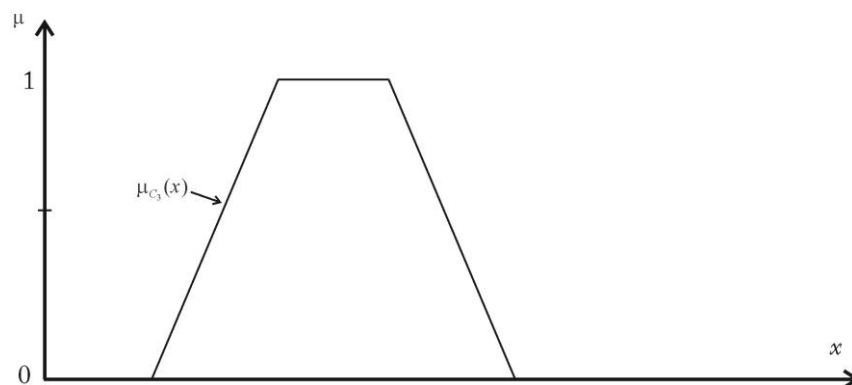


Рис. 1.13. Входная функция принадлежности, характеризующая красный цвет

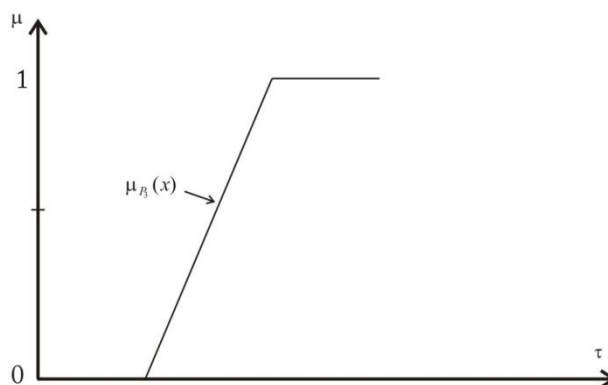


Рис. 1.14. Выходная функция принадлежности, соответствующая степени спелости абрикоса

Уравнение (1.4) представляет общую зависимость, включающую две функции принадлежности. При решении задачи распознавания состояния абрикосов нас интересует выход, получающийся в конкретном входе (например, зеленый).

Пусть x_0 – некоторое конкретное значение зеленого. Степень принадлежности зеленой цветовой компоненты в ответ на этот вход есть просто скалярная величина $\mu_{\text{зеленый}}(x_0)$. Требуется определить выход, соответствующий правилу D3 и конкретному значению x_0 . Искомое значение выхода получается после выполнения операции

«И» между $\mu_{\text{зеленый}}(x_0)$ и результатом общего вида $\mu_3(z, v)$, вычисленным при x_0 :

$$R_2(\tau) = \min\{\mu_{\text{зеленый}}(x_0), \mu_2(x_0, \tau)\}, \quad (1.5)$$

где $R_2(\tau)$ означает нечеткий выход в результате правила D1 и конкретного входа.

Единственной переменной в R_2 , как и ожидалось, является выходная переменная τ . При выполнении операции минимума данной функции с положительной константой ξ все значения $\mu_2(x, \tau)$, большие этой константы, будут урезаны. Однако нас интересует только значение x_0 на цветовой оси. Поэтому рассмотрим только профиль функции, урезанной плоскостью $x = x_0$ вдоль по оси спелости. Он соответствует правилу D2, следовательно, $r = \mu_{\text{зеленый}}(x_0)$. Уравнение (1.5) есть выражение для этого профиля.

Нетрудно определить нечеткие отклики по остальным двум правилам и выбранному входу x_0 , следуя аналогичному рассуждению:

$$R_2(\tau) = \min\{\mu_{\text{желтый}}(x_0), \mu_2(x_0, \tau)\}, \quad (1.6)$$

$$R_1(\tau) = \min\{\mu_{\text{зеленый}}(x_0), \mu_1(x_0, \tau)\}. \quad (1.7)$$

Отметим, что эти уравнения формируют выход, ассоциированный с конкретным правилом и заданным входом. Несмотря на то что значение на входе этих уравнений является скалярным, каждый из этих трех откликов образует нечеткое множество.

В целях формирования общего отклика решается вопрос группировки отдельных откликов. Например, три правила, сформулированные в начале данного раздела, присоединены операцией «ИЛИ». Поэтому конечный (сгруппированный) нечеткий выход задается в виде общего отклика, объединяющего три отдельных нечетких множества:

$$R = R_1 \text{ ИЛИ } R_2 \text{ ИЛИ } R_3. \quad (1.8)$$

Известно [1], что операция «ИЛИ» определяется как операция максимума. Исходя из этого конечного результата при $d = \{1, 2, 3\}$ и $r = \{\text{зеленый, желтый, красный}\}$, запишем общий отклик в виде

$$R(\tau) = \max_d \left\{ \min_r \{ \mu_r(x_0), \mu_d(r_0, \tau) \} \right\}. \quad (1.9)$$

Несмотря на то что данный результат получен в контексте некоторого примера, выражение (1.9) является совершенно общим. Достаточно принять $d = \{1, 2, \dots, n\}$ для расширения его на n правил. Точно также расширяется функция принадлежности до любого конечного объема. Отметим, что уравнения (1.8) и (1.9) утверждают одно и то же: «Откликом нечеткой системы (R) является объединение отдельных нечетких множеств, которые возникают в процессе импликации».

На рис. 1.15 приведены три входные функции принадлежности, оцененные в точке x_0 . Выходы в ответ на значение входа x_0 приведены на рис. 1.16. Эти нечеткие множества являются урезанными профилями. Необходимо отметить, что численно R_3 состоит только из нулей, поскольку $\mu_{\text{красный}}(x_0) = 0$. На рис. 1.17 представлен итоговый результат R , сформированный из объединения R_1 , R_2 и R_3 . Несмотря на то, что полный выход, соответствующий конкретному входу, получен, он является нечетким множеством.

Завершающим этапом является формирование четкого выхода τ_0 на основе нечеткого множества R путем использования процесса, который называется приведением к четкому (дефаззификация). Существует много способов дефаззификации R для получения четкого выхода. Одним из наиболее распространенных способов является вычисление центра тяжести этого множества [1]:

$$y_0 = \frac{\sum_{\tau=1}^k \tau R(\tau)}{\sum_{\tau=1}^k R(\tau)}. \quad (1.10)$$

Подставляя в это уравнение дискретные значения R , получаем $\tau_0 = 74,8$ (рис. 1.17). Этот результат свидетельствует о том, что заданный цвет x_0 означает приблизительно 75 % неспелости абрикосов.

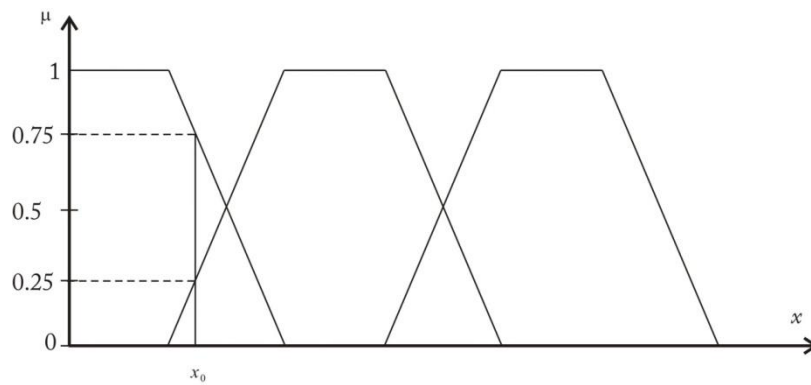


Рис. 1.15. Значения функции принадлежности, соответствующие заданному цвету x_0

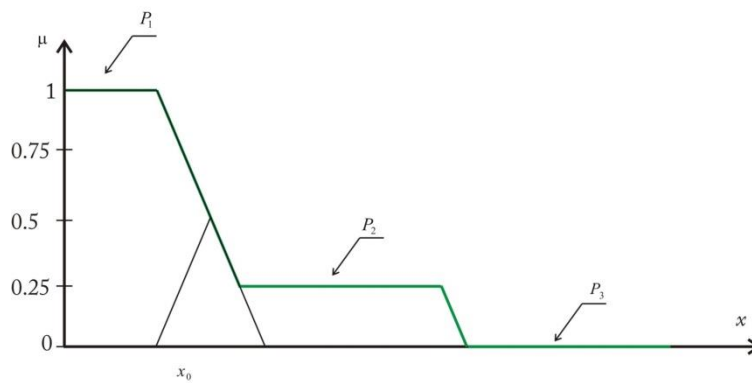


Рис. 1.16. Выделение нечеткого подмножества, сформированного в соответствии с уравнениями (1.6) и (1.7)

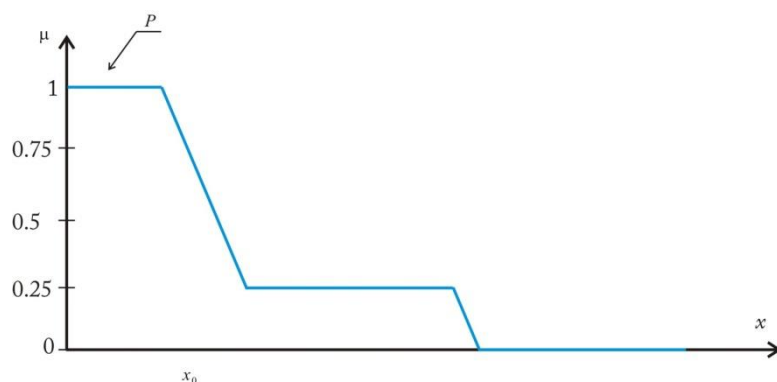


Рис. 1.17. Итоговое нечеткое подмножество, построенное с использованием уравнения (1.9)

До сих пор мы рассматривали правила «ЕСЛИ-ТО» (т.е. исходные условия), которые имеют только один параметр. Далее рассмотрим правила, содержащие несколько посылок. Например, вместо правила D1 даны его некоторые уточнения: «ЕСЛИ цвет рассматриваемого абрикоса зеленый ИЛИ его состояние твердое, ТО он неспелый». Для лингвистической переменной «твердый» должна быть определена функция принадлежности.

Для получения единственного числового результата по этому правилу требуется учитывать обе части исходных условий. Поэтому сначала оценивается заданное значение входного цвета «зеленый» с применением функции его принадлежности. Потом с применением функции принадлежности «твердый» определяется заданное значение состояния. Поскольку обе части этого условия связаны оператором «ИЛИ», выбирается максимум из двух результирующих значений. Затем полученное значение используется в процессе импликации для «урезания» выходной функции принадлежности.

Необходимо подчеркнуть, в случае использования m правил «ЕСЛИ–ТО», n входов (например, V_1, V_2, \dots, V_N) и одного выхода (τ) для формулирования правил нечеткой логики они могут иметь следующий вид:

$$\begin{aligned}
& \text{ЕСЛИ } (V_1, \mathfrak{R}_{11}) \text{ И } (V_2, \mathfrak{R}_{12}) \text{ И } \dots \text{ И } (V_n, \mathfrak{R}_{1n}) \text{ ТО } (\tau, B_1), \\
& \text{ЕСЛИ } (V_1, \mathfrak{R}_{21}) \text{ И } (V_2, \mathfrak{R}_{22}) \text{ И } \dots \text{ И } (V_n, \mathfrak{R}_{2n}) \text{ ТО } (\tau, B_2), \quad (1.11) \\
& \text{ЕСЛИ } (V_1, \mathfrak{R}_{m1}) \text{ И } (V_2, \mathfrak{R}_{m2}) \text{ И } \dots \text{ И } (V_n, \mathfrak{R}_{mn}) \text{ ТО } (\tau, B_m), \\
& \text{ИНАЧЕ } (\tau, B_E),
\end{aligned}$$

где \mathfrak{R}_{ij} – нечеткое множество, сформулированное с помощью i -го правила и j -й входной переменной; B_i – нечеткое множество, сформулированное с помощью выхода i -того правила. При этом предполагается, что составные части исходных условий правила связаны операторами «И»; B_E – нечеткое множество, сформулированное с помощью правила «ИНАЧЕ». Заметим, что правило «ИНАЧЕ» выполняется в случае, когда ни одно из предшествующих правил не удовлетворяется полностью.

При этом значения всех пар элементов исходных условий в каждом правиле должны быть определены, что позволяет определить единственное скалярное значение. В результате определения исходных условий i -го правила в (1.11) получаем скалярное значение χ_i , ($i = \overline{1, m}$):

$$\chi_i = \min_{1 \leq j \leq m} \{ \mu_{A_{ij}}(V_j) \}, \quad (1.12)$$

где $\mu_{A_{ij}}(V_j)$ является функцией принадлежности нечеткого множества \mathfrak{R}_{ij} , полученного по значению j -го входа с χ_i уровнем мощности (или уровнем включения) i -го правила.

Правило «ИНАЧЕ» выполняется в случае, когда условия всех правил «ТО» удовлетворяются слабо. Данное правило дает сильный отклик в случае слабых откликов всех остальных. Поэтому правило «ИНАЧЕ» можно рассматривать как выполнение операции «НЕ» по

условиям всех остальных правил. Используя этот способ комбинирования (операций «И») всех уровней условий правил «ТО», можно получить следующий уровень включения для правила «ИНАЧЕ»:

$$\chi_E = \min_{1 \leq i \leq m} \{1 - \chi_i\}. \quad (1.13)$$

Если хотя бы одно из правил «ТО» включается на «полный уровень» (его отклик равен 1), то отклик правила «ИНАЧЕ» равен 0. При понижении уровней откликов условий правил «ТО» мощность правила «ИНАЧЕ» увеличивается и является нечетким эквивалентом известных правил «ЕСЛИ-ТО-ИНАЧЕ», используемых в языках программирования [66,67].

При рассмотрении вопросов обработки цифровых изображений с помощью концепции нечетких множеств были использованы базы правил. Процесс формирования этих правил детально рассмотрен в следующем разделе.

1.2.3. Формирование базы правил

Нечеткие правила конструируются из функции принадлежности входов и выходов. Такие правила обеспечивают связь между предпосылкой и заключением, выраженными в форме «ЕСЛИ–ТО». Входы и выходы, приведенные к нечеткому виду, облегчают выделение правил и их обобщение. В общем случае правила могут быть получены из опыта одного или нескольких экспертов в рассматриваемой предметной области.

Несомненно, что определение хороших лингвистических правил зависит от объема и качества знаний эксперта. Однако в настоящее время не существует формализованного подхода к задаче точного и однозначного представления знаний в рамках теории нечетких

множеств. В связи с этим представление знаний в фаззифицированном виде, по существу, произвольно.

Предположим, что нас интересует установление правил для распознавания состояния некоторого фрукта, например, абрикоса. Несомненно, что в приведенных выводах цвета абрикоса, наблюдаемые на различных стадиях спелости, обозначены не слишком четкими описаниями цветовых ощущений. Поэтому в качестве отправной точки при этих обозначениях должны быть использованы описания в нечеткой форме. Это достигается введением функции принадлежности абрикосов к классу незрелых, полуспелых или зрелых на основе функции цвета (рис. 1.18, 1.19). В данном контексте цвет является лингвистической переменной, а конкретный цвет (т.е. красный фиксированный) – лингвистической величиной. Лингвистическая величина x определяется с использованием функций принадлежности (рис. 1.18). Приведенное знание может быть формализовано в форме следующих нечетких правил «ЕСЛИ–ТО»:

D1: ЕСЛИ цвет рассматриваемого абрикоса зеленый, ТО он незрелый
ИЛИ

D2: ЕСЛИ цвет рассматриваемого абрикоса желтый, ТО он полуспелый,
ИЛИ

D3: ЕСЛИ цвет рассматриваемого абрикоса красный, ТО он зрелый.

При определении этого значения использована функция принадлежности, приведенная на рис. 1.18.

Приведенные три правила представляют собой результаты нашего знания о рассматриваемой задаче распознавания. Эти правила не являются ничем, кроме формализма для процесса мышления об этой задаче. Информация о цвете является той частью правила,

которая находится слева от «ТО», и называется посылкой (или исходным условием). Правая часть называется следствием (или выводом).

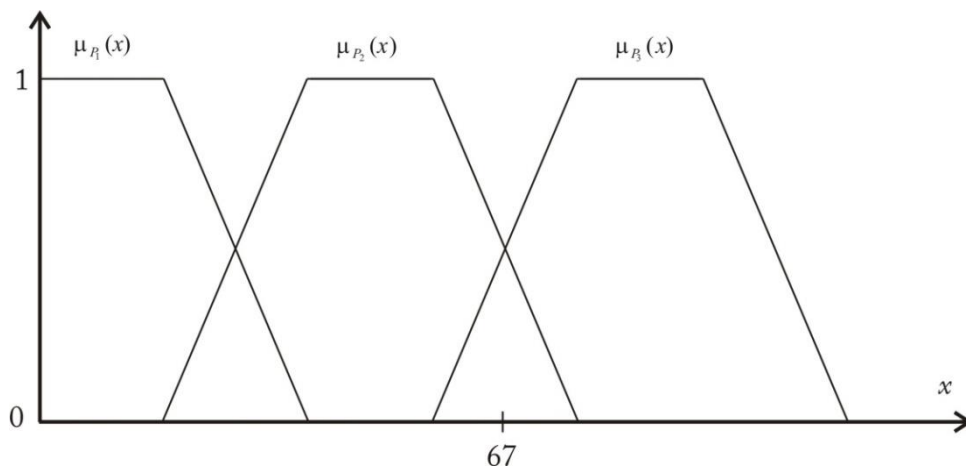


Рис. 1.18. Функции принадлежности, характеризующие спелость абрикоса: x - степень спелости абрикоса; P_1 - нечетное подмножество неспелого абрикоса; P_2 - нечеткое подмножество полуспелого абрикоса; P_3 - нечетное подмножество спелого абрикоса

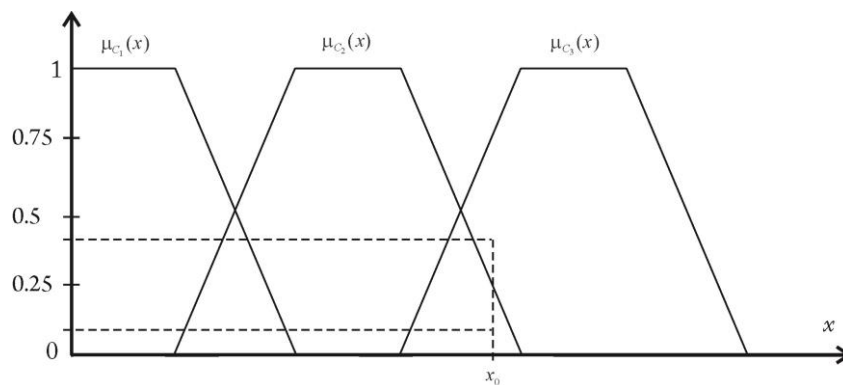


Рис. 1.19. Определение значения нечеткости для конкретного цвета x_0

Специфика четких правил диктует частые изменения, что вызывает значительные проблемы в оценке их поведения. Такие системы с часто изменяющимися параметрами не имеют практического значения. В то же время нечеткая реализация приводит

к тому, что активизация правил будет зависеть от нечетких множеств и от правил, связывающих свойства входа и выхода.

База знаний состоит из определенного набора нечетких правил. Обычно число правил, требуемых в системе, связано с числом управляемых параметров. Таким образом, в рассматриваемом примере оценка состояния некоторого абрикоса является единственной переменной выхода, зависящей от двух входных переменных: цвета и мягкости абрикоса. Если принять, что каждый вход имеет функцию принадлежности, состоящую из трех элементов, тогда возможно девять входных комбинаций цвета и мягкости. В большинстве случаев можно использовать меньшее количество правил. Например, в работах [55, 68, 69] приведены нечеткие правила для сглаживания резкости и выделения контуров на изображении в градациях серого, которые состоят из трех входов и стольких же выходов:

ЕСЛИ пиксель темнее соседних пикселей,

ТО сделать его светлее,

ИНАЧЕ

ЕСЛИ пиксель светлее соседних пикселей,

ТО сделать его темнее,

ИНАЧЕ оставить без изменения.

В этих правилах различия значений яркостей между рассматриваемым пикселем и его соседями являются входными данными, а увеличение/уменьшение значений яркостей этого пикселя - выходным данным. Если предположить, что значения яркости изображения находятся в интервале $[0, L-1]$ (где L - число градаций яркости), то простые треугольные нечеткие множества – среднее положительное и среднее отрицательное – определены на интервале $[-L+1, L-1]$ и представляют яркие и темные значения яркостей входных

переменных, а малый положительный, нулевой и небольшой отрицательный треугольные нечеткие множества определены как приращение указанного значения по следствиям составленных правил. Выходное значение добавляется к исходному значению яркости пикселя.

Правила или нечеткие ассоциации представляют собой знания, которые важны для системы с точки зрения отклика на всевозможные комбинации входов. Нечеткие системы запоминают наборы (банки) нечетких ассоциаций или правил.

1.3. Вопросы улучшения контраста изображений с использованием аппарата нечетких множеств

1.3.1. Локально-адаптивное улучшение качества изображений

Качество изображения на локальных участках можно улучшать, используя такие параметры интенсивностей пикселей, как среднее значение интенсивности и изменение интенсивности (или стандартное среднеквадратичное отклонение интенсивностей элементов локальной окрестности изображения). Среднее значение - это мера средней яркости. При вычислении и анализе средней яркости элементов изображения существует возможность ее коррекции, т.е. затемненные участки изображения делать более светлыми, а слишком светлые участки изображения затемнять. Однако в случае, если на изображении присутствуют темные и светлые области, то такой подход только ухудшит его визуальное восприятие. Поэтому целесообразно использовать еще один параметр, который характеризовал бы распределение яркостей элементов изображения в некоторой локальной окрестности. Другими словами, этот параметр характеризовал бы изменения интенсивностей или меру контрастности изображения.

Типичное локальное преобразование, основанное на этих параметрах, переводит интенсивность исходного изображения L_{in} в интенсивность нового изображения L_{out} путем осуществления следующей операции над расположением (i,j) каждого пикселя:

$$L_{out}(i, j) = k \frac{\bar{L}}{\sigma(i, j)} [L_{in}(i, j) - \bar{L}_{lokal}(i, j)] + \bar{L}_{lokal},$$

где

\bar{L} - среднее значение интенсивностей элементов всего изображения L_{in} ;

$\sigma(i, j)$ - среднеквадратичное отклонение интенсивностей элементов локальной окрестности изображения в точке с координатами (i,j) ;

$L_{in}(i, j)$ - среднее значение интенсивности для окрестности с центром в точке (i,j) ;

k - некоторая константа, $0 < k < 1$.

Локальные изменения увеличиваются за счет умножения разности на

$$k \frac{\bar{L}}{\sigma(i, j)}.$$

Среднеквадратичное отклонение будет принимать меньшие значения в малококонтрастных окрестностях и более высокие значения в окрестностях с более высокой контрастностью. С учетом этого, а также того, что $\sigma(i, j)$ находится в знаменателе, участки с низкой контрастностью будут иметь большее усиление, чем участки с большей контрастностью. Среднее значение подставляется для восстановления среднего уровня интенсивности изображения на

локальном участке. На практике целесообразно ограничивать диапазон значений множителя $k \frac{\bar{L}}{\sigma(i, j)}$ во избежание больших отклонений интенсивностей на отдельных участках.

Результаты компьютерного моделирования рассмотренного метода представлены на рис. 1.20 - 1.23.



Рис. 1.20. Исходное изображение

Рис. 1.21. Обработанное изображение при $m=35$ и $k=0.7$

Проанализируем результаты моделирования. В методе есть два основных параметра, которые существенно влияют на результат обработки - размер локальной окрестности m и коэффициент k . Рассмотрим два изображения, которые представлены соответственно на рис. 1.21 и 1.23, которые представляют результат обработки при одинаковом коэффициенте k , но разных размерах локальных окон m . Из этих изображений видно, что уменьшение размера локального окна приводит к увеличению детальности обработки. Для анализа влияния коэффициента k при одинаковых размерах локальной окрестности m рассмотрим два других изображения, которые представлены на рис.

1.21 и 1.22. Уменьшение коэффициента k приводит к устранению резких перепадов на изображении и понижению его контрастности. Таким образом, используя различные значения параметров m и k , можно управлять уровнем контрастности и детальности обработки изображений.



Рис. 1.22. Обработанное изображение при $m = 35$ и $k = 0.3$



Рис. 1.23. Обработанное изображение при $m = 15$ и $k = 0.7$

1.3.2. Методы улучшения контраста изображений при нечеткой исходной информации

Концепция нечеткой обработки и идентификации изображений предполагает использование подходов проблемно-ориентированной предварительной обработки, сохраняющей информационные признаки объектов. Она позволяет упростить и ускорить процесс обучения и выделения (идентификации) объекта на изображении для нечеткой обработки за счет использования информации об иерархии признаков, что сокращает затраты времени на обработку. В этом разделе рассматривается нечеткое описание повышения контраста при нечеткой исходной информации.

Известно, что искаженные элементы изображения часто весьма заметно отличаются от соседних элементов. Это наблюдение послужило основой для многих алгоритмов, обеспечивающих подавление шума. Если яркость данного элемента превышает среднюю яркость группы ближайших элементов на некоторую пороговую величину, яркость элемента заменяется на нечеткую среднюю яркость.

При обработке изображений для визуализации получили распространение методы, в которых часто отсутствуют строгие математические критерии оптимальности, их заменяют качественные представления о целесообразности той или иной обработки, опирающиеся на субъективные оценки результатов.

Подавляющее большинство процедур обработки для получения результата в каждой точке изображения привлекает входные данные из некоторого множества точек исходного изображения, окружающих обрабатываемую точку. Однако имеется группа процедур, где осуществляется поэлементная обработка. При размытом изображении каждый элемент можно рассматривать как нечеткое множество.

Сущность поэлементной обработки изображений сводится к следующему. Пусть $f(x, y)$ и $g(x, y)$ - значения яркости исходного и получаемого после обработки изображений соответственно в точке кадра, имеющей декартовы координаты: x – номер строки и y – номер столбца.

Поэлементная обработка означает, что существует функциональная зависимость между этими яркостями:

$$g(x, y) = F(f(x, y)),$$

позволяющая по значению исходного сигнала определить значение выходного сигнала.

Задача контрастирования связана с улучшением согласования динамического диапазона изображения и экрана, на котором выполняется визуализация. Если для цифрового представления каждого отсчета изображения отводится 1 байт (8 бит) запоминающего устройства, то входной или выходной сигналы могут принимать одно из 256 значений. Обычно в качестве рабочего используется диапазон 0...255, при этом 0 соответствует при визуализации уровню черного, а значение 255 – уровню белого. Предположим, что минимальная и максимальная яркости исходного изображения равны f_{\min} и f_{\max} соответственно.

Если эти параметры или один из них существенно отличается от граничных значений яркостного диапазона, то визуализированное изображение выглядит как неудобное, утомляющее при наблюдении.

Часто бывает удобно рассматривать изображение как реализацию нечеткого случайного процесса. Введем порождающую изображения непрерывную случайную функцию $f(x, y)$ двух переменных пространственных координат x, y . Случайный процесс

$f(x, y)$ полностью описывается совместной плотностью вероятности $P[A]$.

Пусть $\{R^n, \sigma, P\}$ - вероятностное пространство, в котором R – пространство n -мерных вещественных векторов; σ – поле борелевских множеств в R^n ; P – вероятностная мера на R^n .

Нечеткое случайное событие A в R^n есть нечеткое множество, функция принадлежности которого $\mu_A(x) \in \{R^n \rightarrow [0,1]\}$ измерима по Борелю при $x \in X$. Вероятность нечеткого случайного события A равна математическому ожиданию функции принадлежности μ_A и определяется при помощи интеграла Лебега-Стилтьеса в виде

$$P[A] = \int_{R^n} \mu_A(x) dP(x) = M[\mu_A].$$

Другой способ описания случайного процесса состоит в вычислении средних по ансамблю.

Так как операции дополнения, объединения, пересечения, суммы и произведения нечетких событий A и B используют $1 - \mu_A$, $\max\{\mu_A, \mu_B\}$, $\min\{\mu_A, \mu_B\}$, $\mu_A + \mu_B$, $\mu_A \cdot \mu_B$, которые измеримы по Борелю, и поскольку измеримы $\mu_A(x)$ и $\mu_B(x)$, то можно сказать, что нечеткие события в отношении операций дополнения, объединения и пересечения образуют борелевскую σ -алгебру. Также можно определить нечеткое вероятностное пространство, индуцированное вероятностным пространством $\{R^n, \sigma, P\}$.

Это позволяет определить основные характеристики случайных нечетких событий, такие, как математическое ожидание, дисперсия, начальные и центральные моменты и т.д.:

$$M[A] = \frac{1}{P(A)} \int_{R^n} x \mu_A(x) dP(x),$$

$$\sigma^2[A] = \frac{1}{P(A)} \int_{R^n} [x - M[A]]^2 \mu_A(x) dP(x),$$

$$m_v[A] = \frac{1}{P(A)} \int_{R^n} x^v \mu_A(x) dP(x),$$

$$M_v[A] = \frac{1}{P(A)} \int_{R^n} [x - M[A]]^v \mu_A(x) dP(x).$$

При рассмотрении дискретных нечетких множеств и случайных событий следует учитывать, что функция принадлежности задается только на дискретном множестве элементов из X или R^n , а интеграл следует заменить соответствующей суммой. Приводим выражения для основных вероятностных характеристик нечеткого полного случайного события:

$$P[A_{\Theta}] = \sum_{j=1}^n \mu_j p_j,$$

$$M[A_{\Theta}] = \frac{\sum_{j=1}^n \theta_j \mu_j p_j}{\sum_{j=1}^n \mu_j p_j},$$

На рис. 1.24, 1.25 показано обработанное изображение с использованием локально-адаптивного метода и при нечеткой исходной информации.



Рис. 1.24. Обработанное изображение с использованием
локально-адаптивного метода



a)



б)

Рис. 1.25. Обработанное изображение при нечеткой исходной
информации

1.3.2.1. Функции неопределенности нечетких множеств

Рассмотрим нечеткое множество $A = \{(x, \mu_A(x))\}$, где X – дискретное множество элементов $\{x_1, \dots, x_n\}$.

Мера нечеткости A может быть определена по аналогии с энтропией Шеннона в виде [41]:

$$H(\mu_A) = -a \sum_{i=1}^n \{\mu_A(x_i) \ln \mu_A(x_i) + [1 - \mu_A(x_i)] \ln [1 - \mu_A(x_i)]\},$$

причем $H(\mu_A)$ удовлетворяет следующим свойствам.

1. $H(\mu_A) \geq 0, H(\mu_A) = 0 \Leftrightarrow \mu_A(x)$ вырождена (т.е. принимает значение 0 или 1).

2. $H(\mu_A)$ достигает максимума по $\mu_A \Leftrightarrow \mu_A(x) \equiv \frac{1}{2} \forall x \in X$.

3. $H(\mu_A)$ вогнута по $\mu_A \in [0, 1]$.

4. $H(\mu_A) = H(\mu_{\bar{A}})$.

5. $H(\mu_A) \geq H(\mu_{A^*}) \Leftrightarrow \mu_{A^*}(x) \geq \mu_A(x)$ при $\mu_A(x) \geq \frac{1}{2}, \mu_{A^*}(x) \leq \mu_A(x)$

при $\mu_A(x) \leq \frac{1}{2}$.

6. $H(\max[\mu_A, \mu_B]) + H(\min[\mu_A, \mu_B]) = H(\mu_A) + H(\mu_B), \forall \mu_A, \mu_B \in \{X \rightarrow [0, 1]\}$.

7. $H(\mu_{A \cdot B}) = |A|H(\mu_A) + |B|H(\mu_B)$, где $B = \{(x, \mu_B(x))\}_{x \in X}$, $|A|$ и $|B|$ -

мощности нечетких множеств A и B , соответственно

$$|A| = \sum_{i=1}^n \mu_A(x_i), \quad |B| = \sum_{i=1}^n \mu_B(x_i),$$

$A \cdot B$ – произведение нечетких множеств A и B .

Энтропию нечеткого множества $H(\mu_A)$ будем называть прямой функцией неопределенности четвертого рода, порожденной функцией неопределенности четвертого рода, порожденной функцией принадлежности нечеткого множества A [1].

Если коэффициент $a = 1/n$, то $H(\mu_A)$ может иметь интерпретацию, несколько отличную от общепринятого толкования энтропии случайного события как недоверности информационного предсказания результатов наблюдений. Например, $H(\mu_A)$ может рассматриваться как структурная мера неопределенности по составу структуры некоторой системы. И, кроме того, энтропия Шеннона и другие функции неопределенности первого рода определены на плоском множестве Δ_n n -мерного пространства, а $H(\mu_A)$ определяются на прямоугольном параллелепипеде $[0, 1]^n$.

Заметим, что возможно определение других прямых функций неопределенности четвертого рода, если при их конструировании исходить не из обобщения энтропии Шеннона, а из обобщения класса функций неопределенности первого рода.

1.3.2.2. Связанная и полная функции неопределенности четвертого рода

Рассмотрим нечеткое случайное событие $A = \{x, \mu_A(x) : P\{x = x_i\} = p_i (i = 1, \dots, n)\}$. Мера нечеткости события A может быть определена [70] как энтропия нечеткого подмножества A четкого множества X с распределением

$$p = (p_1, \dots, p_n) : H(\mu_A, p) = - \sum_{i=1}^n \mu_A(x_i) p_i \ln p_i .$$

Будем называть $H(\mu_A, p)$ связанной функцией неопределенности четвертого рода, соответствующей функции принадлежности $\mu_A(x)$ и распределению p на X .

В качестве другой связанной функции неопределенности четвертого рода, соответствующей $\mu_A(x)$ и p , является статически нормированная средняя мера нечеткости события A вида

$$H(p, \mu_A) = -\frac{1}{n} \sum_{i=1}^n p_i [\mu_A(x_i) \ln \mu_A(x_i) + (1 - \mu_A(x_i)) \ln(1 - \mu_A(x_i))].$$

Функции неопределенности четвертого рода $H(\mu_A, p)$ и $H(p, \mu_A)$ имеют концептуальное различие ввиду того, что $H(\mu_A, p)$ представляет собой по существу энтропию нечеткого подмножества состояний случайного вектора x с распределением p на X , а $H(p, \mu_A)$ – математическое ожидание прямой функции неопределенности – энтропии нечеткого множества, определенного на состояниях случайного вектора x с распределением p на X . При этом, если распределение p^0 вырождено, т.е. $p_{i_0}^0 = 1$, $p_i^0 = 0$ при $i \neq i_0$, то $H(p^0, \mu_A) \neq 0$, хотя $H(\mu_A, p^0) = 0$.

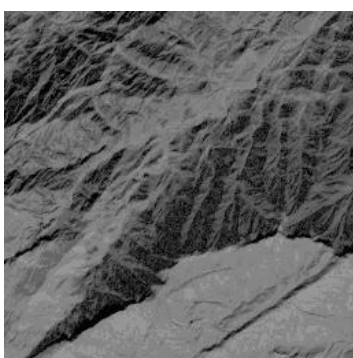
Полная функция неопределенности четвертого рода определяется как полная энтропия [4, 41] нечеткого случайного события A в виде $H(p \diamond \mu_A) = H(p) + H(p, \mu_A)$, где $H(p)$ – функция неопределенности первого рода для распределения p на X . Другие полные функции неопределенности четвертого рода определяются следующим образом:

$$\begin{aligned} H(p \mu_A) &= H(p) + H(\mu_A, p), & H(p * \mu_A) &= H(\mu_A) + H(p, \mu_A), \\ H(\mu_A * p) &= H(\mu_A) + H(\mu_A, p), & H(p, \mu_A) &= H(p) + H(p, \mu_A) + H(\mu_A), \\ H(\mu_A, p) &= H(\mu_A) + H(\mu_A, p) + H(p). \end{aligned}$$

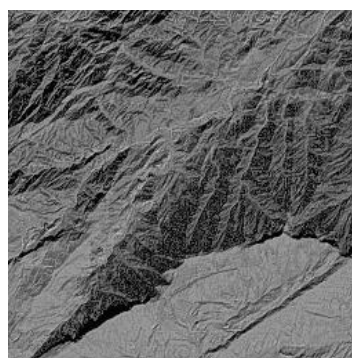
При этом можно определить, что приведенный класс функций неопределенности четвертого рода порождается рассмотрением бинарных отношений понятий случайности и нечеткости; естественно, что могут быть определены и более сложные классы функций неопределенности четвертого рода, основанные на рассмотрении других, более тонких отношений между понятиями случайности и нечеткости с привлечением классов функций неопределенности первого и второго рода.

Отметим, что энтропия локальной окрестности изображения определяется как сумма произведений вероятностей элементов окрестности с различными значениями яркостей на логарифм этих вероятностей, взятая с противоположным знаком (рис. 1.26).

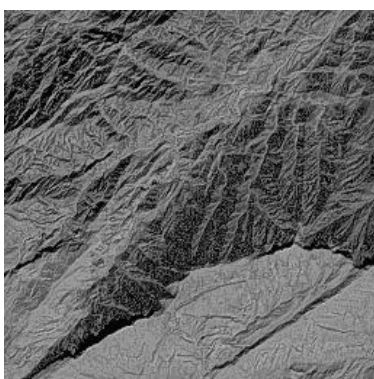
Локальную окрестность следует рассматривать как некоторую сложную систему, состоящую из простых подсистем - элементов окрестности, и уже с этих позиций искать энтропию окрестности. Кроме того, такой подход для определения энтропии локальной окрестности требует значительных вычислительных затрат.



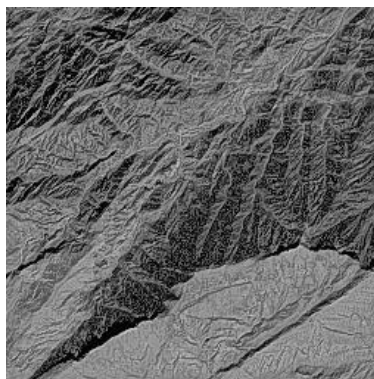
а)



б)



в)



г)

Рис. 1.26. Локально-адаптивный метод повышения визуального качества изображений с использованием энтропии: *а* - исходное изображение; *б* - изображение *а*, обработанное известным методом с использованием энтропии; *в* - изображение *а*, обработанное известным методом с использованием классического подхода к определению энтропии; *г* - изображение *а*, обработанное предложенным методом с использованием нечеткого подхода к определению энтропии

Далее приводим алгоритм линейного повышения контраста при нечеткой исходной информации.

Функции принадлежности $\mu^f(x, y)$ и $\mu^g(x, y)$ определяются следующим образом:

1. Нормализация:

$$u(x, y) = l \frac{f(x, y) - f_{\min}}{f_{\max} - f_{\min}}.$$

2. Фаззификация:

$$\mu_i^f(x, y) = \frac{1}{1 + \frac{u(x, y) - c_i}{\sigma_f}}, \quad i = \overline{1, k}.$$

3. Уточнение фаззификации:

$$\mu_i^f(x, y) = \begin{cases} 2(\mu_i^f(x, y))^2, & 0 \leq \mu_i^f(x, y) \leq \frac{1}{2}, \\ 1 - 2(1 - \mu_i^f(x, y))^2, & \frac{1}{2} < \mu_i^f(x, y) \leq 1. \end{cases}$$

4. Нормализация:

$$v(x, y) = l \frac{g(x, y) - g_{\min}}{g_{\max} - g_{\min}}.$$

5. Фаззификация:

$$\mu_i^g(x, y) = \frac{1}{1 + \frac{v(x, y) - c_i}{\sigma_g}}, \quad i = \overline{1, k}.$$

6. Уточнение фаззификации:

$$\mu_i^g(x, y) = \begin{cases} 2(\mu_i^g(x, y))^2, & 0 \leq \mu_i^g(x, y) \leq \frac{1}{2}, \\ 1 - 2(1 - \mu_i^g(x, y))^2, & \frac{1}{2} < \mu_i^g(x, y) \leq 1. \end{cases}$$

Здесь c_i , σ_f и σ_g - параметры функции принадлежности.

Изображения, вводимые в компьютер, часто оказываются малококонтрастными, т.е. у них изменения яркости малы по сравнению с ее средним значением. При этом яркость меняется не от черного к белому, а от серого к чуть более светлее серого. То есть реальный диапазон яркости оказывается намного меньше допустимого (шкалы яркости). Задача повышения контраста заключается в «растягивании» диапазона яркости изображения на всю шкалу.

Эту задачу можно решать при помощи поэлементного преобразования линейного контрастирования:

$$g(x, y) = af(x, y) + b,$$

т.е. берутся такие a и b , которые приводят нечеткие значения поля яркости к некоторым стандартным величинам. Здесь предварительно оцениваются $M[f(x, y)]$, $\sigma[f(x, y)]$, а коэффициенты a , b выбираются так, чтобы для выходного поля получить $M[g(x, y)]$, $\sigma[g(x, y)]$:

$$\begin{aligned} \bar{g}(x, y) &= \frac{f(x, y) - M[f(x, y)]}{\sigma[f(x, y)]} \cdot \sigma[g(x, y)] + M[g(x, y)] = \\ &= \frac{\sigma[g(x, y)]}{\sigma[f(x, y)]} f(x, y) + M[g(x, y)] - M[f(x, y)] \frac{\sigma[g(x, y)]}{\sigma[f(x, y)]}, \end{aligned}$$

т.е.

$$a = \frac{\sigma[g(x, y)]}{\sigma[f(x, y)]}; \quad b = M[g(x, y)] - M[f(x, y)] \frac{\sigma[g(x, y)]}{\sigma[f(x, y)]}.$$

Здесь

$$M[f(x, y)] = \frac{\sum_{i=1}^k f_i(x, y) \cdot \mu_i^f(x, y)}{\sum_{i=1}^k \mu_i^f(x, y)},$$

$$M[g(x, y)] = \frac{\sum_{i=1}^k g_i(x, y) \cdot \mu_i^g(x, y)}{\sum_{i=1}^k \mu_i^g(x, y)};$$

$$g(x, y) = F(f(x, y)) = \begin{cases} 0, & \bar{g}(x, y) < 0, \\ \bar{g}(x, y), & 0 \leq \bar{g}(x, y) \leq 255, \\ 255, & \bar{g}(x, y) > 255. \end{cases}$$

Таким образом, при обработке изображений требуется по некоторым признакам выделять некоторые однородные области изображения. Этапы предварительной обработки изображения позволяют уменьшить влияние искажений на процесс распознавания. Тем не менее, имеет место распознавание в условиях неполной и нечеткой информации. Для ее решения наиболее подходят технологии нечеткой логики, нечеткая логика при этом выступает в роли классификатора. Применение нечеткой логики в задачах обработки визуальной информации обосновывается также свойством обучаемости или адаптивности нечеткой логики к новым задачам, при этом сохраняются архитектура сети и алгоритм ее функционирования. Программная реализация и применение рассмотренного метода в задачах обработки изображений будут рассмотрены ниже, где предусмотрены вопросы разработки алгоритмов с использованием аппарата нечетких множеств.

На рис. 1.27 показан результат линейного повышения контраста при нечеткой исходной информации.



Рис. 1.27. Результат линейного повышения контраста при нечеткой исходной информации

1.4. Сегментация цветных изображений на основе кластеризации

1.4.1. Сегментация цветных изображений на основе кластеризации по методу k -средних

Одно из главных достоинств кластерного анализа в том, что его можно использовать циклически до тех пор, пока не будут достигнуты необходимые результаты. При этом каждый цикл дает информацию, которая влияет на применяемые в дальнейшем подход и направление. Этот процесс можно представить как систему с обратной связью. На этом основывается еще одно применение кластерного анализа - распознавание с самообучением.

Каждая единица совокупности в кластерном анализе считается «точкой в признаковом пространстве». Значение каждого из признаков у данной единицы совокупности служит ее координатой в этом «пространстве». Таким образом, «признаковое пространство» – это область варьирования всех признаков совокупности изучаемых явлений, а «вектор признаков» - объект (точка) в данном «пространстве».

Для решения задачи классификации массива большого объема многомерных наблюдений, таких как данные грозопеленгации, воспользуемся хорошо зарекомендовавшим себя методом кластерного анализа - k -средних (k -means). Суть данного метода кластеризации заключается в разбиении массива данных на кластеры таким способом, чтобы минимизировать сумму расстояний от объектов до соответствующих им центров кластеров. За меру близости примем евклидово расстояние, так как рассматриваемое явление, а именно грозовой очаг, является локализуемым в Евклидовой метрике [9, 10]. Тогда геометрическое расстояние в n - мерном признаковом пространстве между точками p и q с n - координатами (признаками) определяется так:

$$r_{p,q} = \sqrt{\sum_{i=1}^n (X_{ip} - X_{iq})^2},$$

где X – вектор признаков: $X = (X_1, X_2, \dots, X_n)$.

Квадрат евклидова расстояния между X_p и X_q представим в виде

$$r_{pq}^2 = (X_p - X_q)^T (X_p - X_q).$$

В таком случае алгоритм стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров.

Алгоритм работы k -средних имеет следующий вид:

1. Принимаем входные данные и предполагаемое число кластеров k .

2. Выбираем центры кластеров.

3. Рассчитываем расстояния от центров кластеров до всех объектов и ассоциируем каждый объект с ближайшим центром кластера.

4. Перемещаем центр кластера в центроид его точек данных, т.е. вычисленные средние становятся координатами нового центра каждого кластера.

5. Возвращаемся на шаг 3 пока не достигнута сходимость, например, центр кластера остается неподвижен, заняв устойчивое положение, т.е. пока состав кластеров не перестанет меняться.

Таким образом, алгоритм k -средних принимает в качестве входных данных вектор признаков X , содержащий n точек, а также параметр k , задающий требуемое количество кластеров. На выходе – набор из k центроидов кластеров, кроме того, всем точкам множества X присваиваются метки, относящие их к определенному кластеру, причем все точки в пределах данного кластера расположены ближе к своему центроиду, чем к любому другому центроиду.

На рис. 1.28 приведены совмещенные данные радиолокатора МРЛ-5 и грозопеленгатора LS8000.

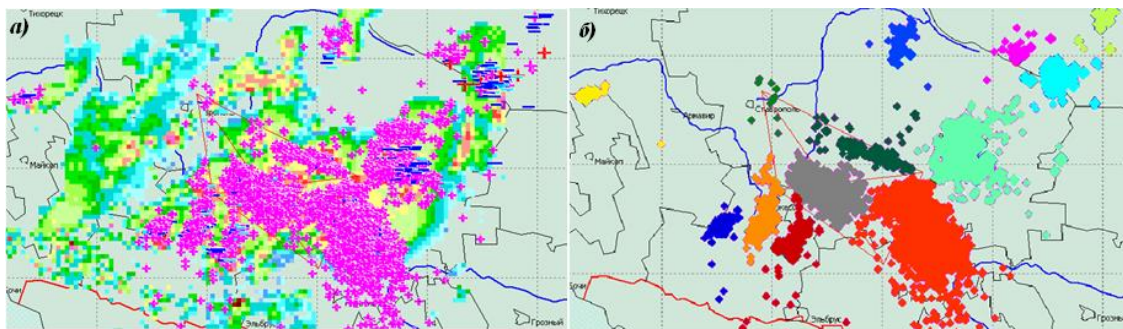


Рис. 1.28. Совмещенные данные радиолокатора МРЛ-5 и грозопеленгатора LS8000 (а). Изображены отражаемость облака и разряды внутриоблачные и между облаком и землей. Плюс обозначает положительную полярность молний, минус – отрицательную.

Полученные в результате обработки кластеры (б)

Для определения числа кластеров используется «разведочный» алгоритм: сначала совокупность делится на два кластера, затем на три и так до тех пор, пока не будет найдено оптимальное число кластеров.

После получения результатов следует проверить правильность кластеризации. Для этого рассчитываются средние значения для каждого кластера. При хорошей кластеризации должны быть получены сильно отличающиеся средние для всех измерений или хотя бы большей их части.

Далее рассмотрим задачу, основной целью которой является автоматическая сегментация на основе кластеризации (метод k -средних) цветных изображений, представленных в цветовом пространстве $L^*a^*b^*$.

1.4.2. Сегментация цветных изображений на основе кластеризации по методу с-средних

На основе нечеткого с-means алгоритм выполняет кластеризацию данных. Этот алгоритм кластеризации предложил Джеймс Бэздэк (James Bezdek) в 1981 г.

Задача нечеткой кластеризации ставится следующим образом.

Дано:

$X = (X_1, X_2, \dots, X_n)^T$ – объекты, подлежащие кластеризации (n – количество объектов).

Каждый объект $X_k = (x_{k_1}, x_{k_2}, \dots, x_{k_p})$ представляет собой точку в p -мерном пространстве признаков ($k = \overline{1, n}$); c – количество кластеров ($2 \leq c < n$).

Необходимо каждому элементу множества X поставить в соответствие степени принадлежности к классам.

Элементы одного кластера должны быть так близки каждый к каждому, как это только возможно, и, одновременно, кластеры должны быть на наибольшем удалении друг от друга. Для обеспечения управляемости процесса кластеризации необходимо использовать меру близости, в качестве которой обычно определяют расстояние между двумя объектами (точками в p -мерном пространстве) X_k и X_i в виде вещественной функции $d: X \times X \rightarrow R^+$ такой, что

$$d(X_k, X_{i1}) = d_{ki} \geq 0,$$

$$d_{ki} = 0 \Leftrightarrow X_k = X_i,$$

$$d_{ki} = d_{ik}.$$

Дополнительно, если функция d удовлетворяет правилу треугольника, т.е. $d_{ki} \leq d_{kj} + d_{ji}$, тогда эта функция является метрикой,

хотя выполнения этого свойства не всегда необходимо для задач кластеризации.

Любое разбиение множества $X = (X_1, X_2, \dots, X_n)^T$ на нечеткие подмножества S_i ($i = \overline{1, c}$) может быть полностью описано функцией принадлежности $\mu_{S_i} : X \rightarrow [0, 1]$.

Обозначим через μ_{ik} степень принадлежности объекта $X_k = (x_{k_1}, x_{k_2}, \dots, x_{k_p})$ к подмножеству S_i , т.е. $\mu_{ik} \equiv \mu_{S_i}(X_k)$, и через V_{cn} - множество всех действительных матриц размером $c \times n$. Тогда нечетким c -разбиением (или матрицей степеней принадлежности) называется матрица $M = [\mu_{ik}] \in V_{cn}$ при выполнении следующих условий:

1. $\mu_{ik} \in [0, 1], i = \overline{1, c}, k = \overline{1, n}$.
2. $\sum_{i=1}^c \mu_{ik} = 1, k = \overline{1, n}$.
3. $\sum_{k=1}^n \mu_{ik} \in (0, n), i = \overline{1, c}$.

В отличие от четкого, при нечетком c -разбиении любой объект одновременно принадлежит к различным кластерам, но с разной степенью. Условия 2) и 3) требуют только, чтобы сумма степеней принадлежности объекта ко всем кластерам была нормализована к 1, а также, чтобы количество кластеров, к которым принадлежит объект, не превышало c .

Обозначим центры кластеров, т.е. точки в p -мерном пространстве, вокруг которых сконцентрированы соответствующие объекты, через $V_i = (v_{i1}, v_{i2}, \dots, v_{ip}), i = \overline{1, c}$.

При использовании евклидова расстояния задача нечеткой кластеризации состоит в нахождении такой матрицы степеней принадлежности M и таких координат центров кластеров

$V = (V_1, V_2, \dots, V_c)$, которые обеспечивают минимум следующего критерия:

$$\sum_{i=1}^c \sum_{k=1}^n (\mu_{ik})^m \cdot \|X_k - V_i\|^2 \rightarrow \min ,$$

где $V_i = \frac{1}{\sum_{k=1}^n \mu_{ik}} \sum_{k=1}^n (\mu_{ik})^m \cdot X_k$ - центр i -го кластера, $i = \overline{1, c}$; m - так

называемый экспоненциальный вес ($m \geq 1$).

Значение экспоненциального веса устанавливается до начала кластеризации. Экспоненциальный вес m влияет на матрицу степеней принадлежности. Чем больше m , тем конечная матрица c -разбиения становится более «размазанной», и при $m \rightarrow \infty$ она примет вид $M = [1/c]$, что является очень плохим решением, так как все объекты принадлежат ко всем кластерам с одной и той же степенью. Также экспоненциальный вес позволяет при формировании координат центров кластеров усилить влияние объектов с большими значениями степеней принадлежности и уменьшить влияние объектов с малыми значениями степеней принадлежности. На сегодня не существует теоретически обоснованного правила выбора значения m . Обычно устанавливают $m = 2$.

Приводим алгоритм.

Шаг 1. Считывание изображения.

Шаг 2. Преобразование изображения из цветовой системы RGB в цветовую систему $L^*a^*b^*$.

Шаг 3. Классификация цветов в пространстве ' a^*b^* ' с использованием кластеризации (метод k -средних или метод c -средних).

Шаг 4. Присвоение меток каждому пикселю изображения на основе метода k -средних или метода c -средних.

Шаг 5. Создание сегментированного изображения на основе цветного.

Шаг 6. Сегментация ядер на основании отдельного изображения.

Шаг 1. Считывание изображения. Считаем файл, который содержит изображение. Здесь применен метод окрашивания для детального анализа патологий (рис. 1.29).



Рис. 1.29. Исходное изображение

*Шаг 2. Преобразование изображения из цветовой системы RGB в цветовую систему $L^*a^*b^*$.* Какое количество цветов видно на изображении, когда не принимать во внимание возможность комбинации яркостей? На самом деле их пять. Следует отметить различия этих цветов между собой. Цветовое пространство $L^*a^*b^*$ (оно еще известно как CIELAB или CIE $L^*a^*b^*$) позволяет различать эти визуальные различия.

Цветовое пространство $L^*a^*b^*$ получено на основе трехцветных значений CIE XYZ. Пространство $L^*a^*b^*$ включает информацию о значении интенсивности ' L^* ', значении цветности ' a^* ', которое показывает, какой цвет выбран на красно-зеленой оси, а значение цветности ' b^* ' показывает, какой цвет выбран на голубо-желтой оси. Вся информация о цветах содержится в значениях ' a^* ' и ' b^* '. Оценить

разницу между двумя цветами можно с использованием евклидового расстояния.

Преобразуем изображение в цветовое пространство $L^*a^*b^*$.

*Шаг 3. Классификация цветов в пространстве 'a*b*' с использованием кластеризации (метод k-средних или метод c-средних).*

Кластеризация приводит до разделения объектов на группы. Кластеризация методом k -средних или методом c -средних приводит также к локализации объектов в пространстве. Поиск разделения, т.е. какой объект к какому классу принадлежит, происходит на основе анализа метрического расстояния между объектами.

Далее на основании информации о цветах в пространстве 'a*b*' каждому пикселю объекта присваивается значение 'a*' и 'b*'. Используем кластеризацию методом k -средних для разделения объектов на пять кластеров и кластеризацию методом c -средних для разделения объектов на три кластера. Для этого используем евклидовую метрику.

Шаг 4. Присвоение меток каждому пикселю изображения на основе метода k-средних или метода c-средних. Для каждого объекта на исходном изображении метод k -средних или c -средних возвращает индекс соответствующего кластера. Значение параметра центра кластера, которое получено в результате применения метода k -средних или c -средних, будет использовано при дальнейшей демонстрации метода. Отметим пиксели, которые содержатся в индексе кластера (рис. 1.30, 1.31).



Рис. 1.30. Изображение, отмеченное кластерными индексами по методу k -средних



Рис. 1.31. Изображение, отмеченное кластерными индексами по методу s -средних

Шаг 5. Создание сегментированного изображения на основе цветного. Объекты можно разделить на изображения по цветам (рис. 1.32 – 1.39).



Рис. 1.32. Объекты в кластере 1 по методу k -средних



Рис. 1.33. Объекты в кластере 1 по методу c -средних

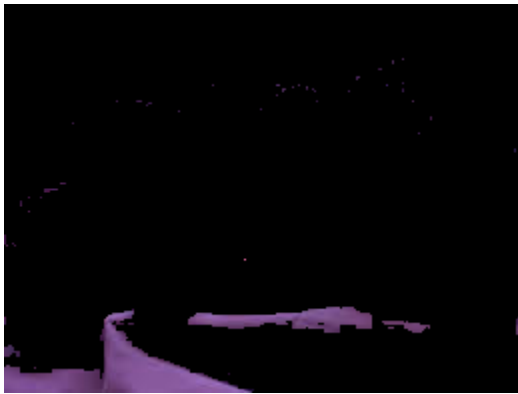


Рис. 1.34. Объекты в кластере 2 по методу k -средних



Рис. 1.35. Объекты в кластере 2 по методу c -средних



Рис. 1.36. Объекты в кластере 3 по методу k -средних



Рис. 1.37. Объекты в кластере 3 по методу c -средних



Рис. 1.38. Объекты в кластере 4 по методу k -средних

Рис. 1.39. Объекты в кластере 5 по методу k -средних

Шаг 6. Сегментация ядер на основании отдельного изображения. Рассмотрим изображение (рис. 5.35), которое содержит синие объекты. Отметим, что они являются темно-синими и светло-синими. Используя значение ' L^* ' в цветовом пространстве $L^*a^*b^*$, можно отделить темно-синие объекты от светло-синих.

1.5. Интуитивная нечеткая обработка изображений

Рассмотрим изображение Аразмера $M \times N$ пиксель, наличие серого уровня r в диапазоне между 0 и $L-1$. При применении интуитивного нечеткого множества (ИНМ) для обработки изображений они рассматриваются как массив нечетких синглтонов. Каждый элемент массива указывает на значение принадлежности $\mu_{\tilde{A}}(g_{ij})$ серого уровня g_{ij} , соответствующее (i,j) -му пикселю, в соответствии с predetermined свойствами изображения, такими как яркость, резкость, однородность [14].

В виде обобщения этого подхода введем следующее представление изображения в интуитивной нечеткой среде.

Изображение A , описанное ИНМ, имеет вид

$$A = \left\{ \left\langle g_{ij}, \mu_A(g_{ij}), \nu_A(g_{ij}) \right\rangle \middle| g_{ij} \in \{0, \dots, L-1\} \right\}, \quad (1.14)$$

где $i \in \{1, \dots, M\}$, $j \in \{1, \dots, N\}$, $\mu_A(g_{ij})$ и $\nu_A(g_{ij})$ обозначают соответственно степень принадлежности и непринадлежности (i, j) -го пикселя к множеству в соответствии со свойствами изображения.

Функции μ_{AI} и ν_{AI} соответствуют принадлежности и непринадлежности множествам компонентов изображения. Если вместо ИНМ мы рассмотрим нечеткое множество, то определение (1.14) приведет к одному из приведенных определений в [12-14].

Методы обработки изображений на основе теории ИНМ предоставляют гибкую математическую базу, для того чтобы справиться с «качественными» свойствами, такими, как контрастность изображения в условиях неоднозначности и расплывчатости, часто присутствующих в цифровых изображениях.

В терминах интуитивной нечеткой обработки изображений (ИНОИ) вопрос, который, естественно, возникает при попытке определить ИНМ яркости пикселей, можно сформулировать так: «как мы можем определить принадлежность и непринадлежность функции серых уровней для описания изображения в ИНМ или более понятным с человеческим восприятием: «как ярк серый уровень и как мы можем быть уверены, что он такой яркий?»».

Неопределенность в изображениях исходит из различных факторов. Они влияют на нашу уверенность в принятии решения, является ли пиксель «серым» или «резким», и поэтому вводят определенные сомнения, связанные с соответствующей точкой. Определение принадлежности компонента А-ИНМ, описывающего яркость пикселей изображения, является более простой задачей, которая может быть проведена аналогичным образом, как и в традиционных нечетких системах обработки изображений. В представленной эвристической системе мы рассматриваем

принадлежность значения уровня серого цвета g его нормализованному уровню интенсивности:

$$\mu_A(g) = \frac{g}{L-1}, \quad (1.15)$$

где $g \in \{0, \dots, L-1\}$.

Следует отметить, что любой другой метод расчета μ_A также может быть применен.

Шум квантования является неотъемлемой частью любой физической системы, которая включает в себя аналого-цифровое преобразование. Для того чтобы смоделировать неточность этого типа серых уровней, в [15] было предложено понятие нечеткой гистограммы на основе нечетких чисел.

Нечеткое число $\tilde{g} : R \rightarrow [0,1]$ является нормальным и выпуклым. Мы ограничиваем наш выбор для симметричных нечетких чисел, которые концептуально подходят для представления понятия «серый уровень примерно g ». Симметричное треугольное нечеткое число определяется как

$$\mu_{\tilde{g}}(x) = \max \left\{ 0, 1 - \frac{|x - g|}{p} \right\},$$

где положительный параметр p контролирует форму числа.

Рассматривая уровни серого в виде нечетких чисел, в [13] понятие гистограммы определено в нечеткой обстановке. Нечеткая гистограмма цифрового изображения является последовательностью $h_A^f(g)$ с $g \in \{0, \dots, L-1\}$ и определяется как

$$h_A^f(g) = \left\| \left\langle (i, j), \mu_{\tilde{g}_{ij}}(g) \right\rangle \mid i \in \{1, \dots, M\}, j \in \{1, \dots, N\} \right\|,$$

где $\|\cdot\|$ обозначает число элементов в нечетком множестве.

Более того, $h_A^f(g)$ представляет собой частоту появления уровня яркости «примерно g ». Однако, из-за его определения, нечеткой гистограмме не удастся быть функцией плотности вероятности.

Нормированная гистограмма имеет вид

$$\tilde{h}_A^f(g) = \frac{h_A^f(g)}{\sum_{g=0}^{L-1} h_A^f(g)},$$

где $g \in \{0, \dots, L-1\}$.

Влияние ошибок квантования можно увидеть путем сравнения четких и нечетких гистограмм. В случае «жесткого» первого порядка статистики существует ряд уровней серого с нулевой или почти нулевой частотой появления из-за шумов квантования, в то время как уровни серого в их окрестности обладают высокой частотой [15]. Это не тот случай, когда рассматривается нечеткая гистограмма. Поэтому для того чтобы моделировать неопределенности родом из шума квантования, колебания, соответствующие уровню серого изображения, должны быть пропорциональны нормированной абсолютной разнице между нормализованной четкой и нечеткой гистограммой:

$$\pi_A(g) \propto \frac{|\tilde{h}_A^c(g) - \tilde{h}_A^f(g)|}{\max_g \{|\tilde{h}_A^c(g) - \tilde{h}_A^f(g)|\}},$$

где \tilde{h}_A^c является нормированной гистограммой четких изображений.

Комбинируя все приведенные выше утверждения, смоделируем гистограмму $\pi_A(g)$, соответствующую серому уровню g изображения A , определенного в (1.14):

$$\pi_A(g) = (1 - \mu_A(g)) \frac{|\tilde{h}_A^c(g) - \tilde{h}_A^f(g)|}{\max \{|\tilde{h}_A^c(g) - \tilde{h}_A^f(g)|\}}. \quad (1.16)$$

Легко увидеть, что степень четкости, вычисленная по (1.16), удовлетворяет условиям, описанным в (1.14). ИНМ в союзе с изображением A определяют компоненты принадлежности и четкости (1.15) и (1.16) соответственно.

Размытость изображения - это мера серости двусмысленности, связанная с пикселями изображения. Иногда требуется в несколько раз уменьшить размытость, присутствующую в изображении, в целях повышения контрастности между яркой и темной областями. Алгоритм контрастного усиления, основанный на минимизации нечеткости, предлагается в [1]:

$$\mu_{\tilde{A}}(g) = \left(1 + \frac{g_{\max} - g}{F_d} \right)^{-F_e},$$

где g_{\max} обозначает максимально серый уровень желаемого, а F_e , F_d являются экспоненциальным и деноминационным фаззификаторами соответственно, которые контролируют неопределенность в нечетких плоскостях.

Фаззификатор F_d определяется как

$$F_d = \frac{g_{\max} - g}{\left(\frac{1}{2} \right)^{\frac{-1}{F_e}} - 1}.$$

Модификация принадлежности значений осуществляется с помощью оператора интенсификации [15] в следующем виде:

$$T_1(\mu_{\tilde{A}}(g)) = 1 \begin{cases} 2(\mu_{\tilde{A}}(g))^2, & \text{если } 0 \leq \mu_{\tilde{A}}(g) \leq \frac{1}{2}, \\ 1 - 2(1 - \mu_{\tilde{A}}(g))^2 & \text{если } \frac{1}{2} < \mu_{\tilde{A}}(g) \leq 1. \end{cases}$$

Оператор интенсификации T последовательно применяется по следующей схеме:

$$T_r(\mu_{\tilde{A}}(g)) = T_1\{T_{r-1}(\mu_{\tilde{A}}(g))\},$$

где $r = 1, 2, \dots$, - индекс результата дальнейшего уменьшения размытости изображения. В предельном случае $r \rightarrow \infty$ T_r производит двухуровневые (бинарные) изображения.

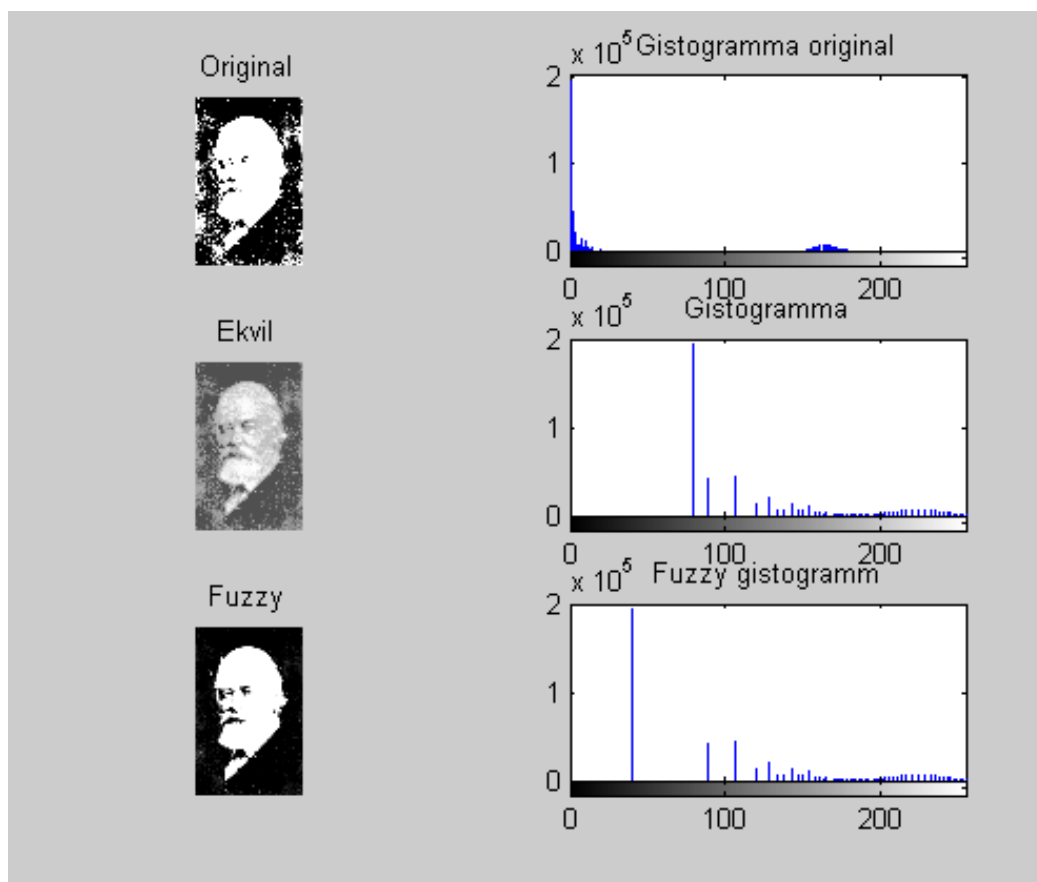
После модификации значений принадлежности дефаззификация выполняется в следующем виде:

$$g' = \begin{cases} 0 & \text{если } \bar{g}' < 0, \\ \bar{g}' & \text{если } 0 \leq \bar{g}' \leq 255, \\ 255 & \text{если } \bar{g}' > 255, \end{cases}$$

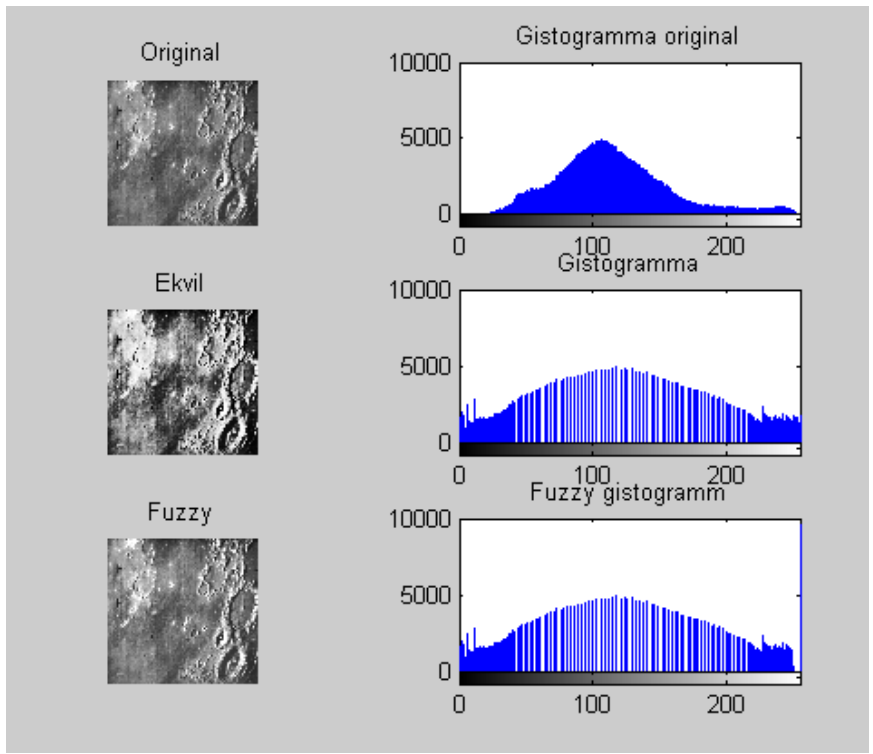
где g' - новый серый уровень и \bar{g}' получается из обратной функции принадлежности как

$$\bar{g}' = g_{\max} - F_d \left((\mu'_A(g))^{\frac{-1}{F_c}} - 1 \right).$$

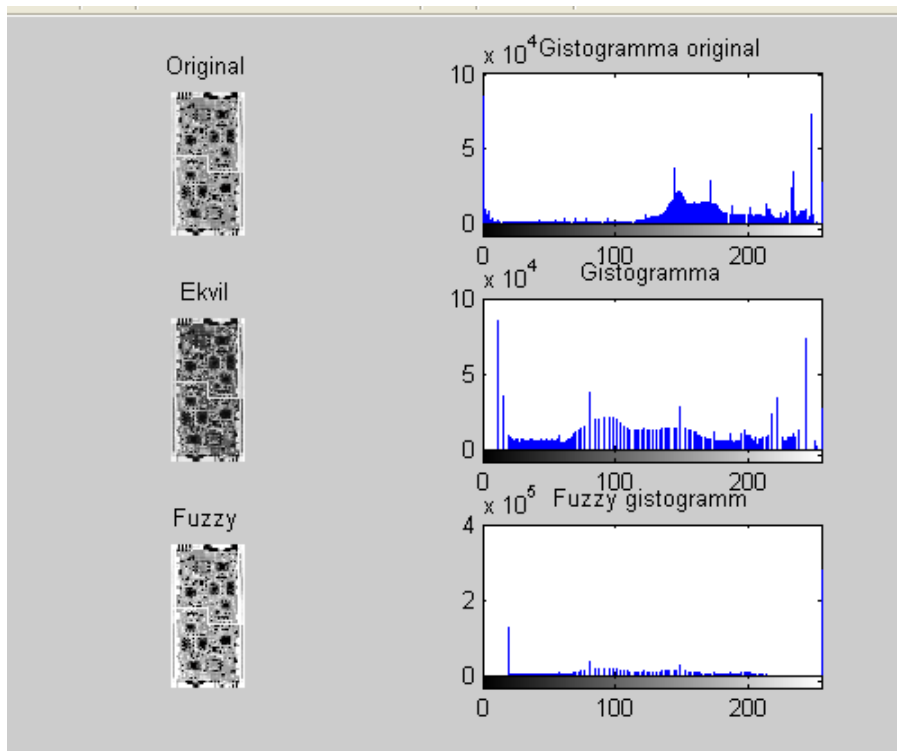
На рис. 1.40 приведены результаты работы рассмотренных алгоритмов на разных изображениях.



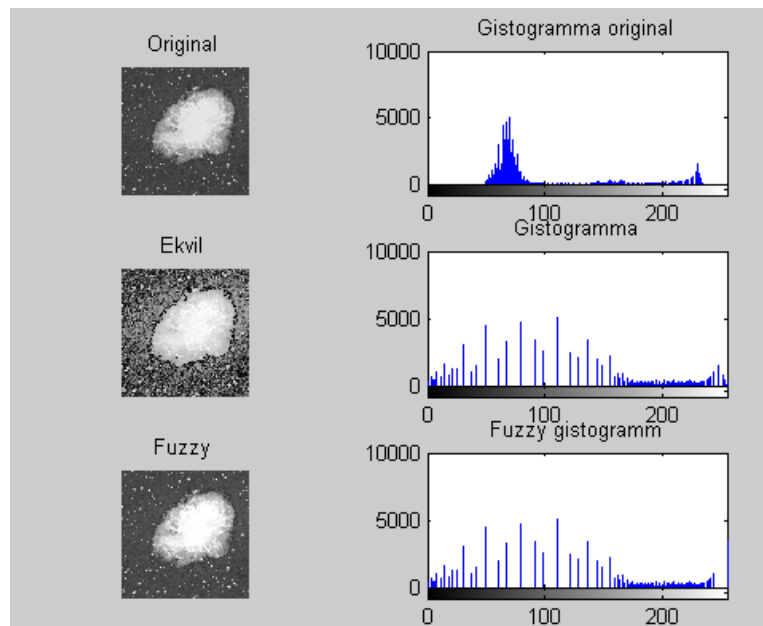
a)



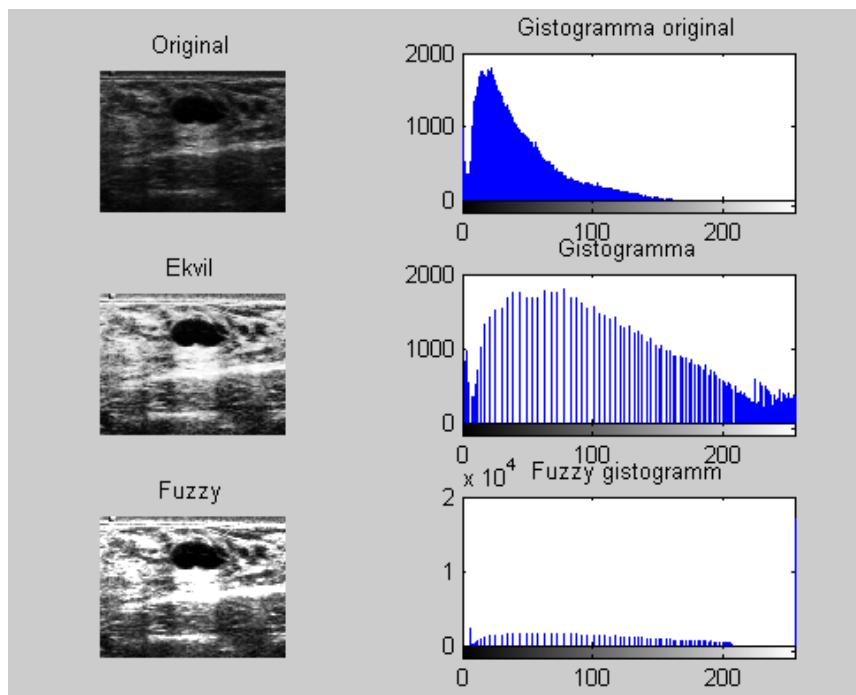
b)



b)



з)



д)

Рис. 1.40. Результаты работы рассмотренных алгоритмов:
a - на первом изображении; *б* - на втором изображении;
в - на третьем изображении; *г* - на четвертом изображении;
д - на пятом изображении

Таким образом, при обработке изображений требуется по некоторым признакам выделять некоторые однородные области изображения. Этапы предварительной обработки изображения позволяют уменьшить влияние искажений на процесс распознавания. На основе полученных результатов можно заключить, что предложенный интуитивный нечеткий подход вводит новый потенциал в решение различных задач обработки изображений, так как он предлагает гибкий и адаптируемый способ обработки неопределенности, присутствующей в цифровых изображениях.

1.5. 1. Исследование алгоритма нечеткой кластеризации для сегментации изображения

В задачах разработки алгоритмов сегментации изображений возникают некоторые сложности, связанные с определением понятия однородности области. Кроме того, структуризация сложных образов требует учитывать тот факт, что существует множество реальных объектов, не имеющих четких границ по своей природе. В таком случае вопрос о необходимости обеспечения однозначности при сегментации является неадекватным, особенно при необходимости учета незначительных различий или для сегментов сложной формы, перекрывающихся между собой. Поэтому применение существующих методов и алгоритмов не решает задачу сегментации изображений в реальных условиях.

Методы сегментации можно разделить на два класса: автоматические, не требующие взаимодействия с пользователем, и интерактивные, использующие пользовательский ввод непосредственно в процессе работы. Здесь рассматриваем только автоматические методы.

Задачи автоматической сегментации делятся на два класса:

- выделение областей изображения с известными свойствами;

- разбиение изображения на однородные области.

Между этими двумя постановками задачи есть принципиальная разница. В первом случае задача сегментации состоит в поиске определенных областей, о которых имеется априорная информация (например, мы знаем цвет, форму областей, или интересующие нас области представляют собой изображения известного объекта). Методы этой группы узкоспециализированы для каждой конкретной задачи. Сегментация в такой постановке используется в основном в задачах машинного зрения (анализ сцен, поиск объектов на изображении).

Во втором случае никакая априорная информация о свойствах областей не используется, зато на само разбиение изображения накладываются некоторые условия (например, все области должны быть однородны по цвету и текстуре). Так как при такой постановке задачи сегментации не используется априорная информация об изображенных объектах, то методы этой группы универсальны и применимы к любым изображениям. В основном сегментация в этой постановке применяется на начальном этапе решения задачи, для того чтобы получить представление изображения в более удобном виде для дальнейшей работы.

Задача разбиения изображения на однородные области поставлена некорректно. На рис. 1.41 приведены три варианта сегментации одного и того же изображения [4]. Далеко не всегда для изображения есть единственно «правильная» сегментация, и далеко не всегда задача сегментации имеет единственное решение. По той же причине нет и объективного критерия оценки качества разбиения изображения.

Поскольку сегментация обычно используется не самостоятельно, а как часть некоторой системы (например, системы

машинного зрения), то с практической точки зрения качество работы метода оценивается исходя из работы системы в целом. Поэтому один и тот же метод сегментации может оказаться хорошим для одной задачи и плохим для другой.

Для оценки качества метода в конкретной задаче обычно фиксируют несколько свойств, которыми должна обладать хорошая сегментация.

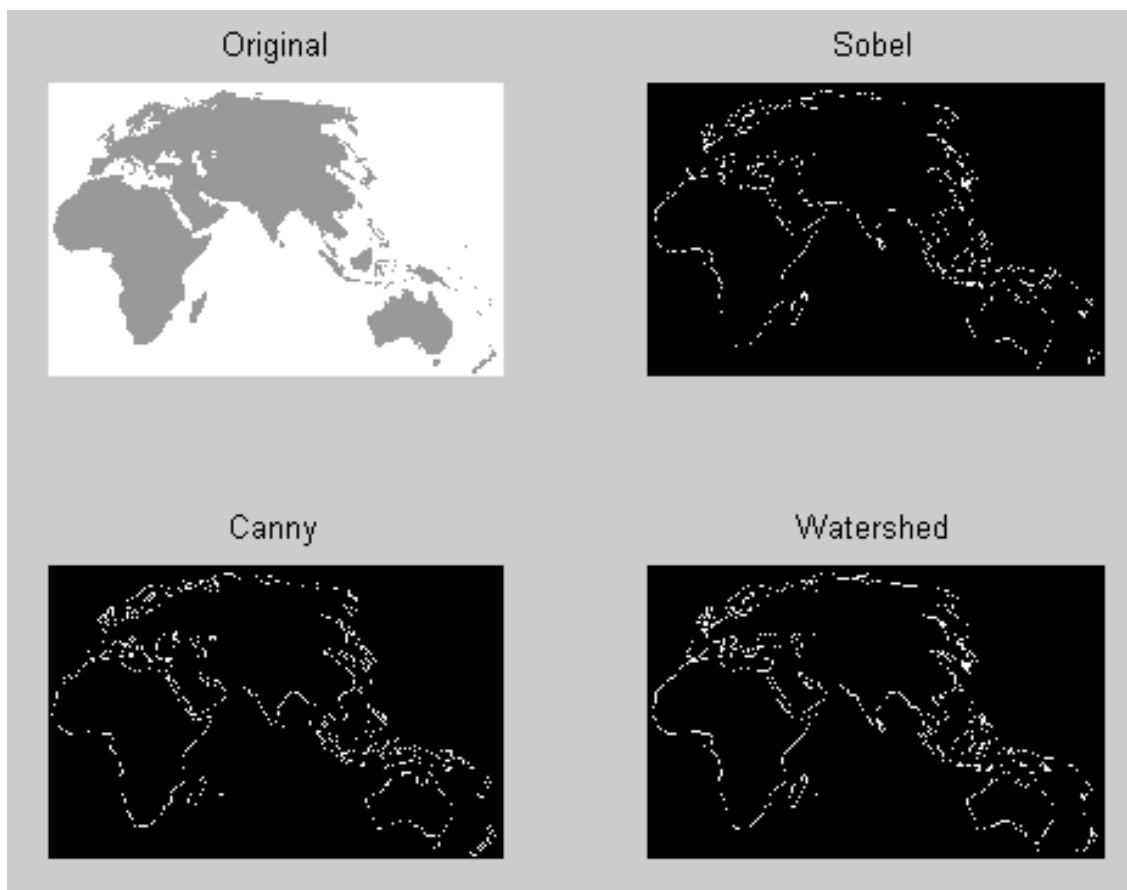


Рис. 1.41. Варианты сегментации изображения

Качество работы метода оценивается в зависимости от того, насколько полученная сегментация обладает этими свойствами. Наиболее часто используются следующие свойства [1]:

- однородность регионов (однородность цвета или текстуры);
- непохожесть соседних регионов;
- гладкость границы региона;

- маленькое количество мелких «дырок» внутри региона.

Разные методы сегментации ориентированы на разные свойства разбиения. Поэтому при выборе метода сегментации для решения конкретной задачи следует определиться, какие свойства разбиения действительно важны. В некоторых прикладных задачах достаточно того, чтобы разбиение обладало лишь первыми двумя из перечисленных свойств.

Методы выращивания регионов и дробления-слияния учитывают пространственное расположение точек напрямую.

Методы выращивания регионов основаны на следующей идее. Сначала по некоторому правилу выбираются центры регионов (seeds), к которым поэтапно присоединяются соседние точки, удовлетворяющие некоторому критерию. Процесс выращивания регионов (regiongrowing) останавливается, когда ни одна точка изображения не может быть присоединена ни к одному из регионов.

Применяются разные критерии, на основании которых точка присоединяется или не присоединяется к региону: близость (в некотором смысле) точки к центру региона; близость к соседней точке, присоединенной к региону на предыдущем шаге; близость по некоторой статистике региона; стоимость кратчайшего пути от точки до центра региона и т.п.

В основном процедура выращивания региона используется для получения отдельных регионов, однако, применяя эту процедуру последовательно или одновременно для нескольких регионов, можно получить разбиение всего изображения. Существуют различные стратегии выращивания регионов.

Методы дробления-слияния состоят из двух основных этапов: дробления и слияния. Дробление начинается с некоторого разбиения изображения, не обязательно на однородные области. Процесс

дробления областей происходит до тех пор, пока не будет получено разбиение изображения (пересегментация), удовлетворяющее свойству однородности сегментов. Затем происходит объединение схожих соседних сегментов до тех пор, пока не будет получено разбиение изображения на однородные области максимального размера.

Конкретные методы различаются алгоритмами, используемыми на этапах дробления и слияния. Для получения пересегментации изображения используются алгоритмы k -средних [1], watershed [2], на втором этапе также используются алгоритмы k -средних [1], самоорганизующиеся карты Кохонена [29]. На этапе слияния регионов используются relaxationprocess [28], k -средних [1], SIDE-уравнения, самоорганизующиеся карты Кохонена [29] и т.д.

При использовании методов, основанных на операторах выделения краев задача сегментации формулируется как задача поиска границ регионов. Методы поиска границ хорошо разработаны для полутоновых изображений. Полутоновое изображение рассматривается как функция двух переменных $(x \text{ и } y)$, и предполагается, что границы регионов соответствуют максимумам градиента этой функции. Для их поиска применяется аппарат дифференциальной геометрии (в простейшем случае это фильтры Roberts, Kirsch, Prewitt, Sobel).

Для повышения устойчивости к шуму перед применением фильтрации изображение обычно размывают. Благодаря коммутативности оператора Лапласа и Гауссова фильтра, можно одновременно осуществлять размытие и поиск границ. В методе Canny комбинируются результаты поиска границ при разной степени размытия.

Основной проблемой методов поиска границ является неустойчивость к шуму. Кроме того, поскольку понятие границы свое для каждой задачи, каждый раз при применении методов поиска границ требуется дополнительно выбирать метод доработки результатов фильтрации (edgeling, edgerelaxation).

1.5.2. Нечеткий алгоритм кластеризации для множества вещественных чисел

Пусть даны вещественное множество $X = \{x_1, x_2, \dots, x_m\} \subset R$ и k вещественные числа s_1, s_2, \dots, s_k в возрастающем порядке, а именно:

$$s_1 < s_2 < \dots < s_k.$$

Нечеткое разбиение A_1, A_2, \dots, A_k для множества X может быть организовано с помощью треугольной функции.

Пусть $a \neq b$ - два действительных числа. Открытая треугольная функция имеет вид

$$\mu(x, b; a) = \max \left[\min \left(1, \frac{x-a}{b-a} \right), 0 \right]. \quad (1.17)$$

Если $a < b$, то формула (1.17) определяет правую открытую треугольную функцию, и если $a > b$, то формула (1.17) определяет левую открытую треугольную функцию.

Отметим, что функция μ проверяет следующие четыре свойства:

$$\begin{aligned} \mu(a, b; a) &= 0, \\ \mu(b, b; a) &= 1, \\ \mu\left(\frac{a+b}{2}, b; a\right) &= \frac{1}{2}, \\ \mu(x, b; a) + \mu(x, a; b) &= 1. \end{aligned} \quad (1.18)$$

Выражение (1.18) показывает, что функции $\mu(x, b; a)$ и $\mu(x, a; b)$ определяют нечеткую область пространства R .

Пусть $a < b < c$ - три действительных числа. Треугольная функция определяется как

$$t(x, b; a, c) = \mu(x, b; a) \wedge \mu(x, b; c), \quad (1.19)$$

где \wedge относится к функции "min" или алгебраическому произведению "·".

Другими словами, нечеткое множество определяется треугольной функцией принадлежности и представляет пересечение между двумя открытыми нечеткими множествами.

Можно найти и другие формулы вычисления для треугольной функции. Эта функция может иметь следующий вид:

$$\mu(x, b; a) = \alpha \cdot |x - a| + \beta |x - b| + \gamma |x - c|.$$

Параметры функции определяются с помощью значений функций $\mu(a)$, $\mu(b)$, $\mu(c)$ в точках a , b , c . Для этого надо решить систему

$$\begin{cases} \beta |a - b| + \gamma |a - c| = \mu(a, b; a) \\ \alpha |b - a| + \gamma |b - c| = \mu(b, b; a) \\ \alpha |c - a| + \beta |c - b| = \mu(c, b; a) \end{cases}. \quad (1.20)$$

Система (1.20) имеет следующее решение:

$$\begin{cases} \alpha = \frac{1}{2} \cdot \left(\frac{\mu(c, b; a) + \mu(a, b; a)}{c - a} + \frac{\mu(b, b; a) - \mu(a, b; a)}{b - a} \right) \\ \beta = \frac{1}{2} \cdot \left(\frac{\mu(c, b; a) - \mu(b, b; a)}{c - b} + \frac{\mu(b, b; a) - \mu(a, b; a)}{b - a} \right) \\ \gamma = \frac{1}{2} \cdot \left(\frac{\mu(c, b; a) + \mu(a, b; a)}{c - a} - \frac{\mu(c, b; a) - \mu(b, b; a)}{c - b} \right) \end{cases}.$$

Для конкретных значений функции $\mu(a) = 0$, $\mu(b) = 1$, $\mu(c) = 0$ решение имеет вид

$$\begin{cases} \alpha = \frac{1}{2} \cdot \frac{1}{b-a} \\ \beta = \frac{1}{2} \cdot \left(\frac{1}{b-a} - \frac{1}{c-b} \right) \\ \gamma = \frac{1}{2} \cdot \frac{1}{c-b} \end{cases}$$

и

$$\mu(x, b; a) = \frac{1}{2} \cdot \frac{|x-a|}{b-a} + \frac{1}{2} \cdot \left(\frac{1}{b-a} - \frac{1}{c-b} \right) \cdot |x-b| + \frac{1}{2} \cdot \frac{|x-c|}{c-b},$$

или

$$\mu(x, b; a) = \frac{1}{2} \cdot \frac{|x-a| - |x-b|}{b-a} + \frac{1}{2} \cdot \frac{|x-c| - |x-b|}{c-b}. \quad (1.21)$$

Точно так же мы можем получить эквивалентную формулу для открытой треугольной функции. Для двух действительных чисел $a \neq b$ функция

$$\mu(x, b; a) = \frac{1}{2} \cdot \frac{|x-a| - |x-b|}{b-a} + \frac{1}{2} \quad (1.22)$$

определяет открытую треугольную функцию. С практической точки зрения мы можем вместо формулы (1.22) использовать следующую:

$$\mu(x, b; a) = \frac{1}{2} \cdot \frac{|x-a| - |x-b| + \varepsilon}{|b-a| + \varepsilon} + \frac{1}{2}, \quad (1.23)$$

где

$$1 \gg \varepsilon > 0. \quad (1.24)$$

Более обобщенно нечеткое разбиение A_1, A_2, \dots, A_k для множества X может быть организовано с помощью L - R функции.

Нечеткое число \tilde{x} называется нечетким числом L - R типа, если

$$\mu(x, x_R(\alpha), x_L(\alpha)) = \begin{cases} \mu_L(x) = 1 - \frac{x - x_L(\alpha)}{u_L}, \\ \mu_R(x) = 1 - \frac{x_R(\alpha) - x}{x_R}, \end{cases}$$

где x - четкое значение числа \tilde{x} , т.е. $x = x_L(1) = x_R(1)$; x_L и x_R - соответственно левое и правое растяжения нечеткого числа \tilde{x} ;

$x_L(\alpha)$ и $x_R(\alpha)$ - соответственно левое и правое значения нечеткого числа \tilde{u} четкости α .

Из определения следует, что если

$$\tilde{x}(\alpha) = \{x, x_L(\alpha), x_R(\alpha)\},$$

то

$$x_L(\alpha) = x - (1 - \alpha)x_L; \quad x_R(\alpha) = x + (1 - \alpha)x_R.$$

Рассмотрим алгебраическое действие над нечеткими числами $L-R$ типа.

Сложение:

$$\tilde{u} + \tilde{v} = \{u + v - (1 - \alpha)(u_L + v_L); u + v + (1 - \alpha)(u_R + v_R)\}.$$

Вычитание:

$$\tilde{u} - \tilde{v} = \{u - v - (1 - \alpha)(u_L + v_L); u - v + (1 - \alpha)(u_R + v_R)\}.$$

Умножение:

1) для $u > 0; v > 0$

$$\tilde{u} \cdot \tilde{v} = \{u \cdot v; (1 - \alpha)(uv_L + vu_L) - (1 - \alpha)u_L v_L; (1 - \alpha)(uv_R + vu_R) + (1 - \alpha)u_R v_L\};$$

2) для $u > 0; v < 0$

$$\tilde{u} \cdot \tilde{v} = \{u \cdot v; (1 - \alpha)(uv_R + vu_L) - (1 - \alpha)u_L v_L; (1 - \alpha)(uv_L + vu_R) + (1 - \alpha)u_R v_L\};$$

3) для $u < 0; v < 0$

$$\tilde{u} \cdot \tilde{v} = \{u \cdot v; (1 - \alpha)(uv_R + vu_L) - (1 - \alpha)u_R v_R; (1 - \alpha)(uv_L + vu_L) - (1 - \alpha)u_L v_L\}.$$

Деление:

$$\frac{\tilde{u}}{\tilde{v}} = \tilde{u} \cdot \frac{1}{\tilde{v}}.$$

При разработке алгоритма нечеткой кластеризации для сегментации изображения, во-первых, мы рассматриваем функции принадлежности:

$$\mu_i(x) = \mu(x, s_1; s_2),$$

для $i=2,3, \dots, k-1$

$$\mu_i(x) = \mu(x, s_i; s_{i-1}) \wedge \mu(x, s_i; s_{i+1}),$$

$$\mu_k(x) = \mu(x, s_k; s_{k-1}).$$

Функции $\mu_1, \mu_2, \dots, \mu_k$ определяют раздел разбиения по следующему равенству:

$$\mu_1 + \mu_2 + \dots + \mu_k = 1.$$

Во-вторых, определяется оператор дефаззификации $\tau(\mu, \gamma)$, который применяется к нечеткому разбиению:

$$\tau(\mu, \gamma) = (\tau_1, \tau_2, \dots, \tau_k),$$

где

$$\tau(x, \mu, \gamma) = \frac{\mu_i^\gamma(x)}{\sum_{j=1}^k \mu_j^\gamma(x)}. \quad (1.25)$$

Пусть v_i и F_i определяются следующим образом:

$$v_i(x) = \mu_i(x) \wedge \tau_i(x, \mu; \gamma), \quad (1.26)$$

$$F_i(s_i) = \frac{\sum_{j=1}^m v_i(x_j) \cdot x_j}{\sum_{j=1}^m v_i(x_j)}. \quad (1.27)$$

Рассмотрим теперь следующее ограничение для параметров s_1, s_2, \dots, s_k :

$$s_i = F_i(s_i). \quad (1.28)$$

Нечеткое множество A_i определяется уравнением (1.28). Центр s_i принадлежит выпуклой оболочке множества X , и это есть фиксированная точка для функции F_i .

Результаты получаются с применением следующего нечеткого алгоритма кластеризации.

Шаг 1. Инициализируются количество кластеров k , параметр дефаззификации γ , процедура остановки параметра δ , индекс итерации $l = 0$ и центры кластеров. Далее, вычисляются нечеткие функции

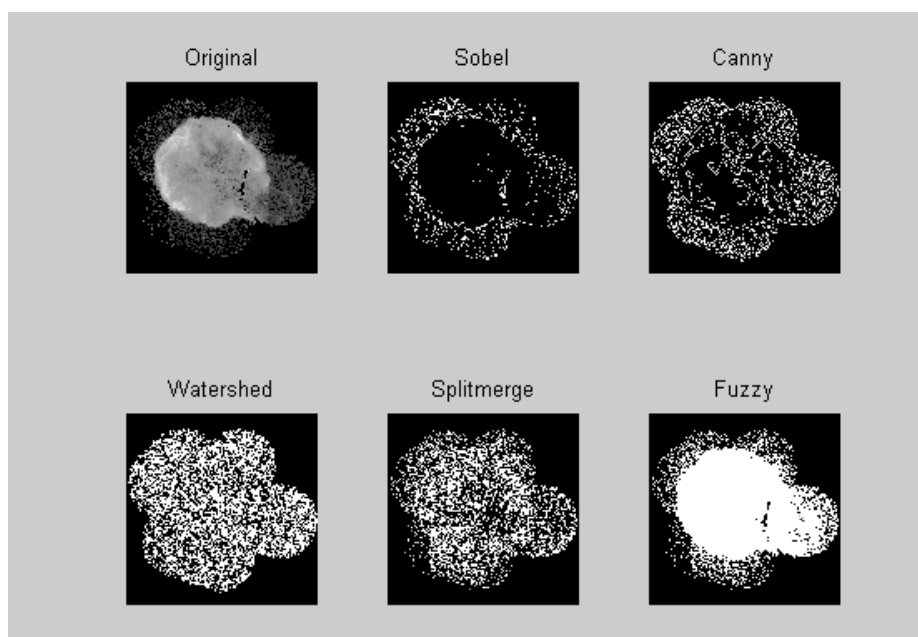
принадлежности $\mu_1^{(0)}, \mu_2^{(0)}, \dots, \mu_k^{(0)}$ и компоненты дефаззификации функции $\tau_1^{(0)}, \tau_2^{(0)}, \dots, \tau_k^{(0)}$.

Шаг 2. Индекс итерации увеличивается, т.е. $l \rightarrow l+1$. Мы рассчитываем центры кластера $s_1^{(l)} = F_1(s_1^{(l-1)}), s_2^{(l)} = F_2(s_2^{(l-1)}), \dots, s_k^{(l)} = F_k(s_k^{(l-1)})$, нечеткие функции принадлежности $\mu_1^{(l)}, \mu_2^{(l)}, \dots, \mu_k^{(l)}$, компоненты дефаззификации $\tau_1^{(l)}, \tau_2^{(l)}, \dots, \tau_k^{(l)}$ и функции $v_1^{(l)}, v_2^{(l)}, \dots, v_k^{(l)}$.

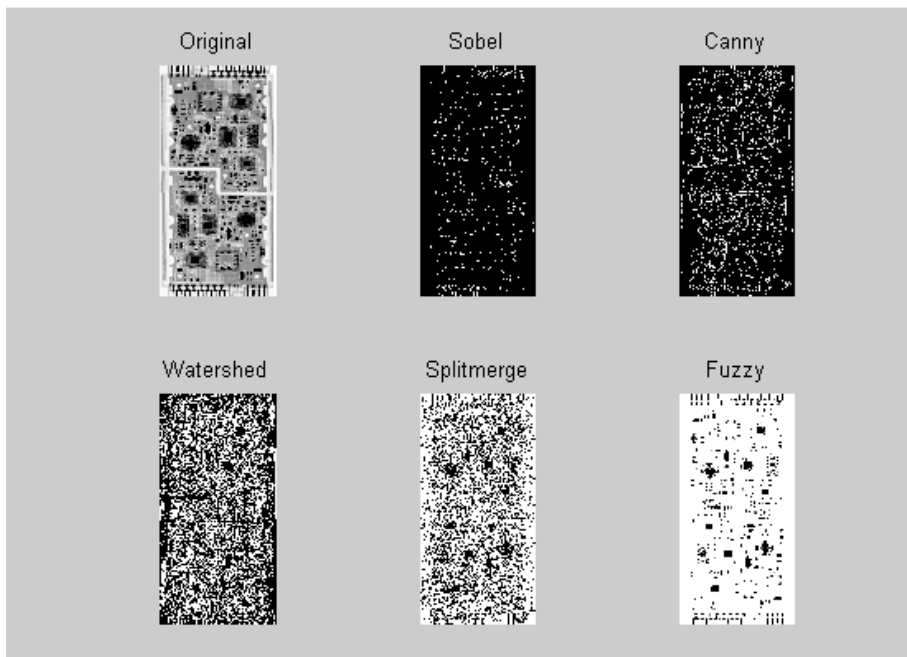
Шаг 3. Вычисляем $d = \sum_{i=1}^k |\mu_i^{(l)} - \mu_i^{(l-1)}|$. Если $d > \delta$, то вернемся к шагу 2, в противном случае - перейти к шагу 4.

Шаг 4. Сохранение данных и конец.

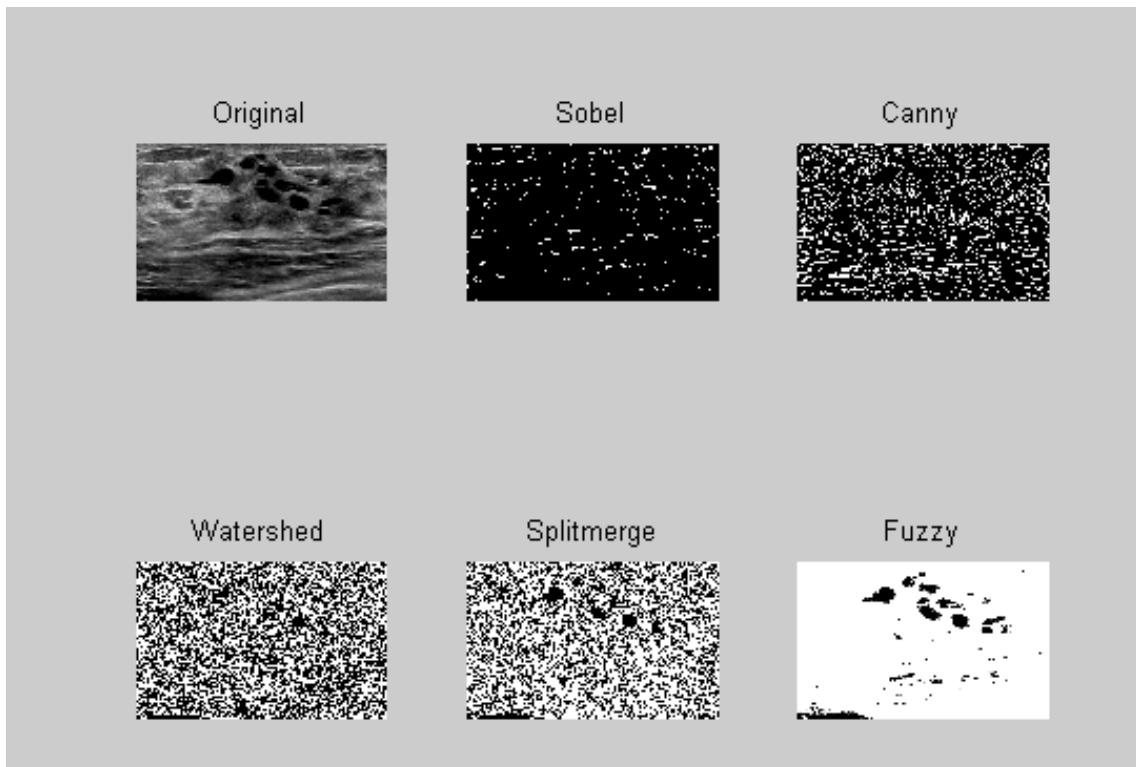
На рис. 1.42 приведены результаты работы рассмотренных алгоритмов на разных изображениях [4].



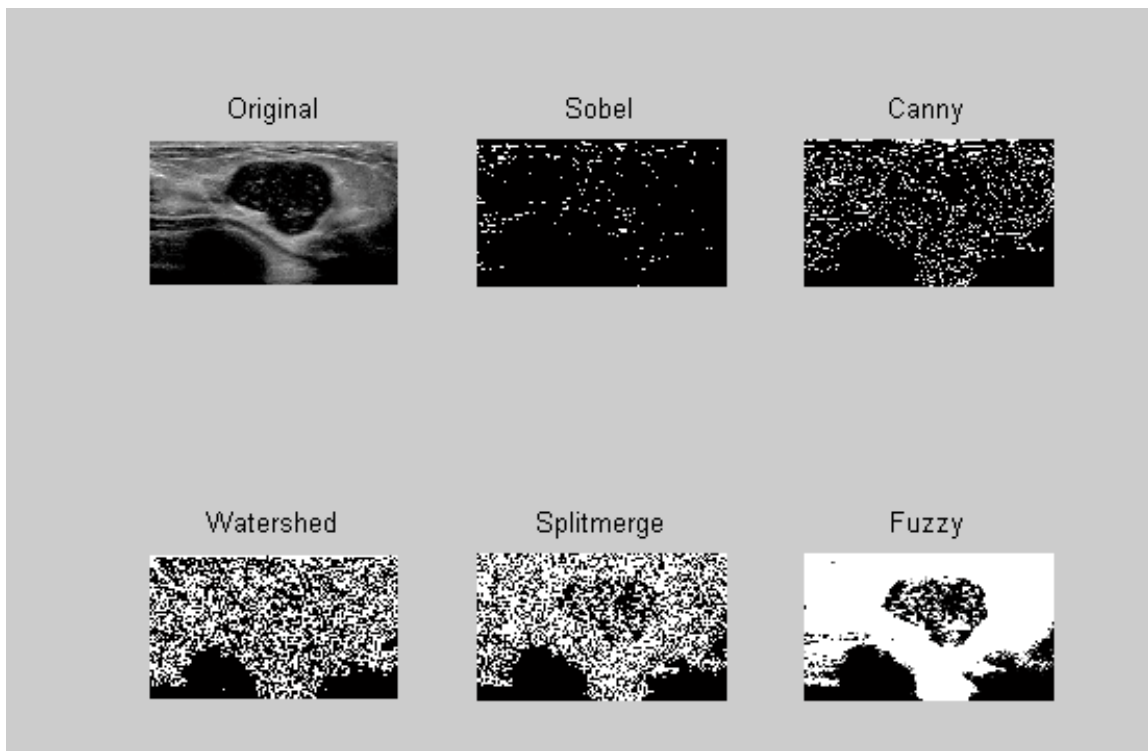
a)



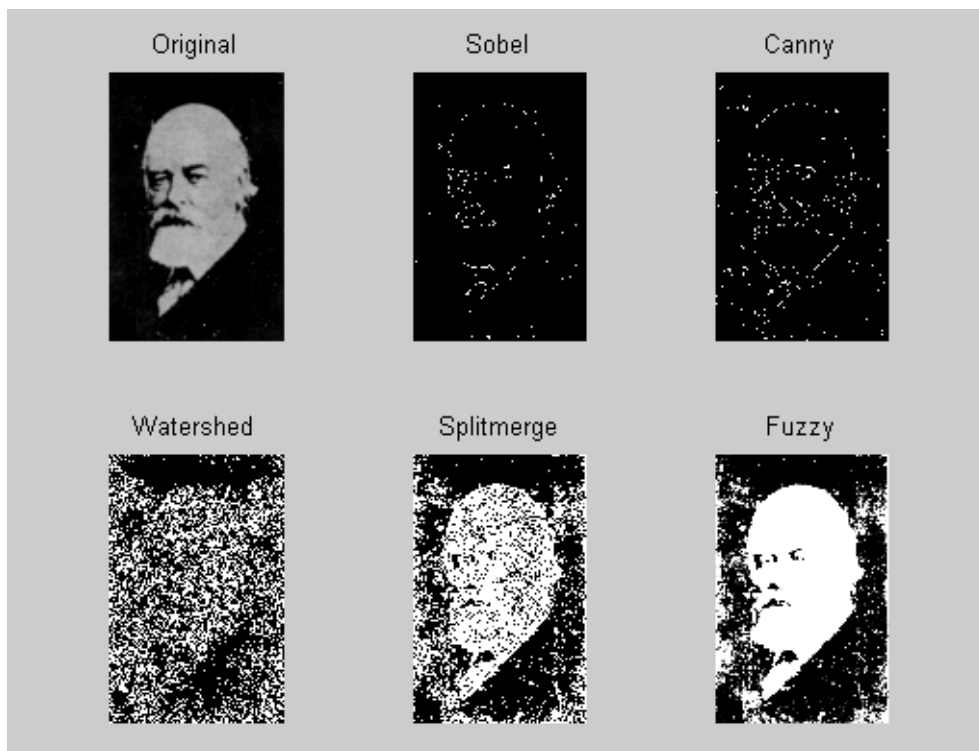
b)



b)



з)



д)

Рис. 1.42. Результаты работы рассмотренных алгоритмов на:
a - первом; *б* - втором; *в* - третьем; *з* – четвертом;
д - пятом изображениях

В работе представлены нечеткие алгоритмы кластеризации для пространства R , исследуется метод сегментации изображений, которые используют алгоритмы кластеризации. Достоинствами такого подхода являются:

- широкий спектр применения. За счёт выбора отображений и меры можно приспособить алгоритм под различные задачи;
- гибкость. Изменяя порог и меру, можно эффективно менять чувствительность алгоритма;
- скорость. Алгоритм работает существенно быстрее алгоритма с поиском границы метода;
- устойчивость. Алгоритм более устойчив к ошибкам, чем методы, основанные на нахождении границ, так как при ошибке мы теряем не весь регион, а лишь его небольшую часть.

1.5.3. Обнаружение объекта на основе цвета

Существует большое количество задач, одной из составляющих которых является вопрос автоматического распознавания лиц на изображениях. Подходов к решению этого вопроса существует много, однако в рамках данного материала рассмотрим метод автоматического обнаружения лиц на изображениях на основе анализа цвета.

Считаем некоторое исходное изображение (рис. 1.43 - 1.46).



Рис.1.43. 1-е исходное изображение



Рис.1.44. 2-е исходное изображение



Рис.1.45. 3-е исходное изображение



Рис.1.46. 4-е исходное изображение

Далее необходимо на изображении лица выбрать пиксели с характерным цветом, определить среднее значение их интенсивности и среднеквадратичное отклонение.

На основе знаний о значениях интенсивностей пикселей лица и их возможных вариациях проводится сегментация изображения (рис. 1.47 – 1.50).



Рис. 1.47. 1-е исходное изображение после сегментации



Рис. 1.48. 2-е исходное изображение после сегментации



Рис. 1.49. 3-е исходное изображение после сегментации



Рис. 1.50. 4-е исходное изображение после сегментации

Целью сегментации является выделение лица на изображении. Однако, вполне естественно, что на изображении присутствовали и другие объекты, значения интенсивностей пикселей которых совпали с интенсивностью пикселей лица. В результате на сегментированном изображении кроме лица выделились и другие объекты. Теперь на сегментированном изображении предстоит найти изображение нужного объекта, т.е. лица. Критерии поиска могут быть разными. Это

может быть площадь, форма и др. В данном примере в качестве критерия для поиска лица выберем площадь. Исходя из выбранного ранее способа сегментации, можно предположить, что лицо занимает наибольшую площадь. Поэтому выбор критерия для поиска лица на основе площади позволит удалить другие объекты, которые меньше за площадью.

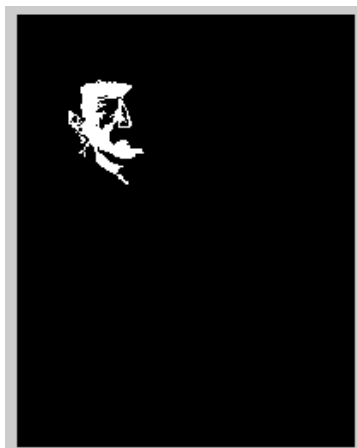


Рис.1.51- Исходное изображение
после удаления шума



Рис.1.52- Исходное изображение
после удаления шума



Рис.1.53- Исходное изображение
после удаления шума



Рис.1.54- Исходное изображение
после удаления шума

Удалим также и другие объекты на сегментированном изображении лица.



Рис.1.55- Исходное изображение
после удаления шума небольших
объектов

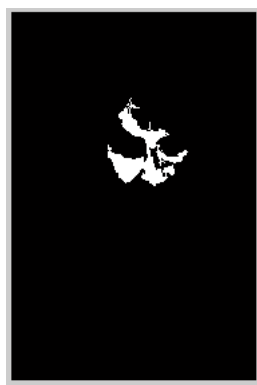


Рис.1.56- Исходное изображение
после удаления шума небольших
объектов



Рис.1.57- Исходное изображение
после удаления шума небольших
объектов



Рис.1.58- Исходное изображение
после удаления шума небольших
объектов

Здесь следует отметить, что выбранного нами критерия может быть недостаточно для достоверного определения объекта поиска, т.е. изображения лица. Такое может произойти в том случае, если на изображении присутствуют другие объекты с похожим цветом, лицо может быть в тени, быть под наклоном и т.п. Поэтому необходимо использовать также другие критерии для поиска изображения лица. Такие критерии могут базироваться на априорной информации о форме лица.

Таким образом, для повышения достоверности распознавания критерий поиска лица на изображении должен быть комплексным.

Далее проводим выделение выбранного лица, например, прямоугольником.



Рис.1.59. Выделение выбранного
1- лица прямоугольником



Рис.1.60. Выделение выбранного
2- лица прямоугольником



Рис.1.61. Выделение выбранного
3- лица прямоугольником



Рис.1.62. Выделение выбранного 4
-лица прямоугольником

Если же на изображении присутствует несколько лиц одновременно, то рассмотренный выше метод необходимо модифицировать. Рассмотрим это более детально. Первые несколько шагов (считывание исходного изображения, выбор характерных пикселей, сегментация и фильтрация) аналогичны, как и в предыдущем методе, поэтому мы приведем только результаты выполнения этих операций.

Глава 2. ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

2.1. Общие понятия искусственных нейронных сетей

Искусственная нейронная сеть является некоторой моделью естественной нейронной сети. Каждый элемент искусственной сети (нейрон) является прототипом, имитирующим свойства и работу биологического нейрона [2-4].

Рассмотрим работу искусственной однослойной нейронной сети(рис. 2.1). На синапсы (однаправленные входы) искусственного нейрона поступает некоторое множество сигналов. Каждый сигнал умножается на соответствующий вес, характеризующий синаптическую силу. Входные сигналы после синапсов поступают в ячейки нейронов (блоки суммирования), выходом которых являются аксоны, с которых сигналы (возбуждения или торможения), определяющие уровень активации нейрона, поступают на синапсы следующих нейронов (если сеть многослойная) или на основе которых принимается решение (например, пороговым мажоритарным блоком). Хотя сетевые парадигмы весьма разнообразны, в их основе почти всегда лежит именно эта конфигурация.

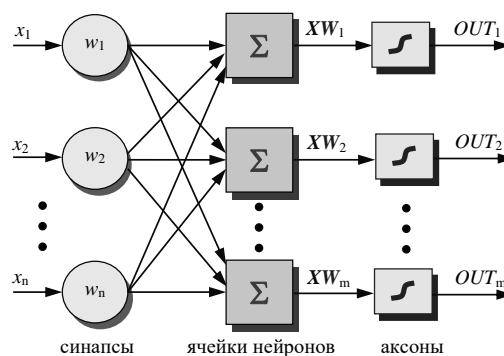


Рис. 2.1. Искусственная однослойная нейронная сеть

Отметим, что синапсы служат лишь для распределения входных сигналов, а аксоны лишь для выдачи сигналов, поэтому они не считаются слоем (слоем являются ячейки нейронов, выполняющие вычисления с комбинациями входных сигналов). В искусственных и биологических сетях многие соединения могут отсутствовать, все соединения показаны в целях общности. Могут иметь место также соединения между выходами и входами элементов в одном слое.

Условно эту часть нейронной сети может представить искусственным нейроном (рис. 2.2). Эта простая модель искусственного нейрона игнорирует многие свойства своего биологического двойника. Например, она не принимает во внимание задержки во времени, которые воздействуют на динамику системы. Входные сигналы сразу же порождают выходной сигнал. И, что более важно, она не учитывает воздействия функции частотной модуляции или синхронизирующей функции биологического нейрона, которые ряд исследователей считают решающими.

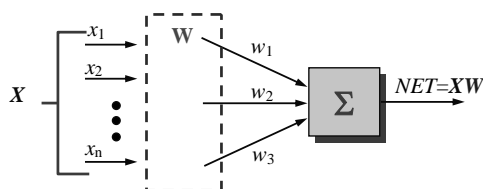


Рис. 2.2. Входная часть искусственного нейрона

Сигнал NET далее, как правило, преобразуется некоторой активационной функцией F и дает выходной нейронный сигнал OUT .

Активационная функция – это нелинейная усилительная характеристика искусственного нейрона, определяющая его уровень активации.

Коэффициент усиления активационной функции вычисляется как отношение приращения величины OUT к вызвавшему его

небольшому приращению величины $NET - \frac{\Delta OUT}{\Delta NET}$. Он выражается наклоном кривой при определенном уровне возбуждения и изменяется от малых значений при больших отрицательных возбуждениях (кривая почти горизонтальна) до максимального значения при нулевом возбуждении и снова уменьшается, когда возбуждение становится большим положительным.

Если активационная функция F сужает диапазон изменения величины NET так, что при любых значениях NET значения OUT принадлежат некоторому конечному интервалу, то F называется «сжимающей» функцией.

Качество работы искусственной нейронной сети во многом зависит от весов синаптических связей и от типа применяемых активационных функций. Рассмотрим типы активационных функций.

2.2. Типы активационных функций

Различают следующие виды активационных функций:

- единичная функция;
- линейный порог (функция гистерезиса);
- сигмоидальная функция (гиперболический тангенс).

Единичная функция. Данная функция является самой простой, ее график представлен на рис. 2.3.

Для этой функции $OUT = 1$, если $NET = XW$ больше некоторой величины T и $OUT = 0$ в остальных случаях, где T – некоторая постоянная пороговая величина.

Если в однослойной нейронной сети применяется единичная активационная функция, то такая сеть называется перцептроном [2, 3, 4]. Первое систематическое изучение перцептронов было предпринято Маккаллоком и Питтсом в 1943 г.

[2]. В 60-е годы перцептроны вызвали большой интерес и оптимизм. Розенблатт доказал замечательную теорему об обучении перцептронов [4]. Далее первоначальная эйфория сменилась разочарованием, когда оказалось, что перцептроны не способны обучиться решению ряда простых задач. Минский [3] строго проанализировал эту проблему и показал, что имеются жесткие ограничения на то, что могут выполнять однослойные перцептроны, и, следовательно, на то, чему они могут обучаться.

Функция линейного порога. Отличием от предыдущей функции является не скачкообразное изменение выхода *OUT*, а в виде линейного тренда. График функции представлен на рис. 2.4.

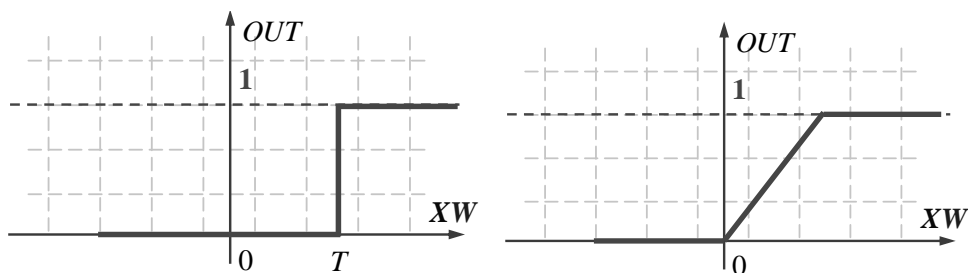


Рис. 2.3. График единичной активационной функции
Рис. 2.4. График линейной активационной функции

Сигмоидальная функция. В качестве «сжимающей» функции часто используется логистическая или «сигмоидальная» (S-образная) функция. Эта функция математически выражается как $F(x) = 1/(1 + e^{-x})$. Таким образом,

$$OUT = 1/(1 + e^{-NET}) . \quad (2.1)$$

Данная функция позволяет обрабатывать нейронной сети как слабые, так и сильные сигналы. Слабые сигналы нуждаются в большом сетевом усилении, чтобы дать пригодный к использованию выходной сигнал. Однако усилительные каскады с большими коэффициентами усиления могут привести к насыщению выхода

шумами усилителей (случайными флуктуациями), которые присутствуют в любой физически реализованной сети. Сильные входные сигналы в свою очередь также будут приводить к насыщению усилительных каскадов, исключая возможность полезного использования выхода. Сигмоидальная функция решает эту проблему. Впервые это обнаружил Гроссберг [28].

В качестве сигмоидальных функций может выступать как логистическая функция (2.1), так и функция гиперболического тангенса $OUT = th(NET)$ (рис. 2.5). Отличием логистической функции от функции гиперболического тангенса является диапазон выходных значений. У первой он находится в диапазоне $(0; 1)$, а у второй в диапазоне $(-1; 1)$.

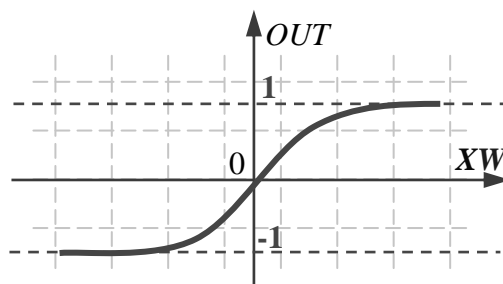


Рис. 2.5. График сигмоидальной активационной функции

2.3. Проблема функции «исключающее или»

Один из самых пессимистических результатов Минского [23] показывает, что однослойный персептрон не может воспроизвести такую простую булевскую функцию, как «*исключающее или*». Это функция от двух аргументов, каждый из которых может быть нулем или единицей. Она принимает значение единицы, когда один из аргументов равен единице (но не оба). Проблему можно проиллюстрировать с помощью однослойной однеионной системы с двумя входами, показанной на рис. 2.6. Обозначим один вход через

x , а другой через y , тогда все их возможные комбинации будут состоять из четырех точек на плоскости XOY , как показано на рис. 2.7. Например, точка $x = 0$ и $y = 0$ обозначена на рисунке как точка A_0 . Таблица 2.1 показывает требуемую связь между входами и выходом, где входные комбинации, которые должны давать нулевой выход, помечены A_0 и A_1 , а единичный выход – B_0 и B_1 .

В сети на рис. 2.6 активационная функция F является обычным порогом, так что OUT принимает значение ноль, когда NET меньше 0.5, и единица в случае, когда NET больше или равно 0.5. Персептрон выполняет следующее вычисление:

$$NET = xw_1 + yw_2. \quad (2.2)$$

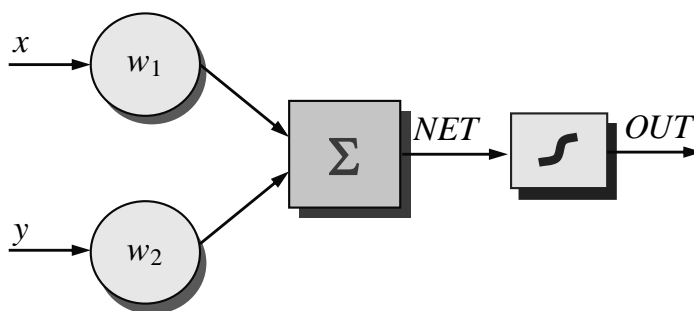


Рис. 2.6. Однонейронная система

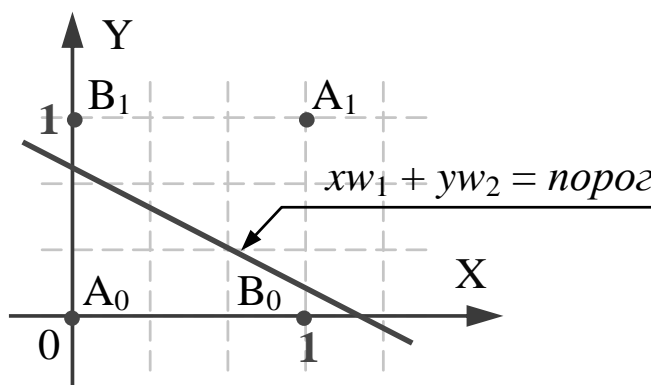


Рис. 2.7. Проблема «исключающее или»

Никакая комбинация значений двух весов не может дать соотношения между входом и выходом, задаваемого табл. 2.1 (таблица истинности для булевой функции). Чтобы понять это

ограничение, зафиксируем NET для (2.2) на величине порога 0.5. Персептрон в этом случае описывается уравнением (2.3). Это уравнение линейно по x и y , т.е. все значения по x и y , удовлетворяющие этому уравнению, будут лежать на некоторой прямой в плоскости XOY

$$xw_1 + yw_2 = 0.5. \quad (2.3)$$

Любые входные значения для x и y на этой линии будут давать пороговое значение 0.5 для NET . Входные значения с одной стороны прямой обеспечат значения NET больше порога, следовательно, $OUT = 1$. Входные значения по другую сторону прямой обеспечат значения NET меньше порогового значения, делая OUT равным 0. Изменения значений w_1 , w_2 и порога будут менять наклон и положение прямой. Для того чтобы сеть реализовала функцию «исключающее или», заданную табл. 2.1, нужно расположить прямую так, чтобы точки А были с одной стороны прямой, а точки В – с другой. Попытавшись нарисовать такую прямую на рис. 2.7, убеждаемся, что это невозможно. Это означает, что какие бы значения ни приписывались весам и порогу, сеть неспособна воспроизвести соотношение между входом и выходом, требуемое для представления функции «исключающее или».

Таблица 2.1

Таблица истинности для функции «исключающее или»

Точки	Значения x	Значения y	Требуемый выход
A_0	0	0	0
B_0	1	0	1
B_1	0	1	1
A_1	1	1	0

Невозможность нарисовать прямую линию, разделяющую плоскость XOY так, чтобы реализовывалась функция «*исключающее или*», является не единственным случаем. Имеется обширный класс функций, не реализуемых однослойной нейронной сетью. Об этих функциях говорят, что они являются линейно неразделимыми, и они накладывают определенные ограничения на возможности однослойных сетей.

Линейная разделимость ограничивает однослойные нейронные сети задачами классификации, в которых множества точек (соответствующих входным значениям) могут быть разделены геометрически. И поскольку линейная разделимость ограничивает возможности персептронного представления, то важно знать, является ли данная функция разделимой. К сожалению, не существует простого способа определить это, если число переменных велико.

Нейрон с n двоичными входами может иметь 2^n различных входных образов, состоящих из нулей и единиц. Так как каждый входной образ может соответствовать двум различным бинарным выходам (единица и ноль), то всего имеется 2^{2^n} функций от n переменных (см. зависимость в таблице 2.2) [21].

Как видно из табл. 2.2, вероятность того, что случайно выбранная функция окажется линейно разделимой, весьма мала даже для умеренного числа переменных. По этой причине однослойные персептроны на практике ограничены простыми задачами.

Линейно разделимые функции

n	2^{2^n}	Число линейно разделимых функций
1	4	4
2	16	14
3	256	104
4	65536	1882
5	4294967296	94572

2.4. Многослойные нейронные сети

К концу 60-х годов проблема линейной разделимости была хорошо понята. К тому же было известно, что это серьезное ограничение представляемости однослойными сетями можно преодолеть, добавив дополнительные слои.

Рассмотрим простую двухслойную сеть с двумя входами, подведенными к двум нейронам первого слоя, соединенными с единственным нейроном в слое 2 (рис. 2.8). Пусть порог выходного нейрона равен 0,75, а оба его веса равны 0,5. В этом случае для того, чтобы порог был превышен и на выходе появилась единица, требуется, чтобы оба нейрона первого уровня на выходе имели единицу. Таким образом, выходной нейрон реализует логическую функцию «и».

Каждый нейрон слоя 1 разбивает плоскость XOY на две полуплоскости, один обеспечивает единичный выход для входов ниже

верхней линии, другой – для входов выше нижней линии. На рис. 2.9 показан результат такого двойного разбиения, где выходной сигнал нейрона второго слоя равен единице только внутри V-образной области. Аналогично во втором слое может быть использовано три нейрона с дальнейшим разбиением плоскости и созданием области треугольной формы. Включением достаточного числа нейронов во входной слой может быть образован выпуклый многоугольник любой желаемой формы. Так как они образованы с помощью операции «и» над областями, задаваемыми линиями, то все такие многогранники выпуклы, следовательно, только выпуклые области и возникают. Точки, не составляющие выпуклой области, не могут быть отделены от других точек плоскости двухслойной сетью.

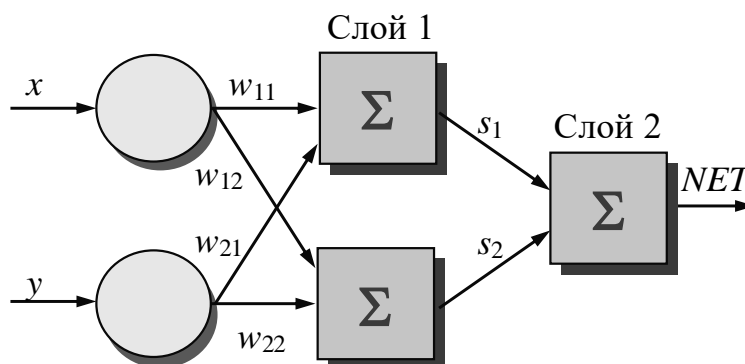


Рис. 2.8. Простейшая двухслойная

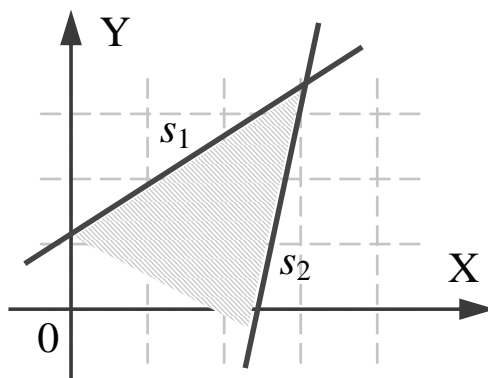


Рис. 2.9. Выпуклая область решений,
нейронная сеть, задаваемая двухслойной сетью

Нейрон второго слоя не обязательно ограничен функцией «и», он может реализовывать многие другие функции при подходящем выборе весов и порога. Например, можно сделать так, чтобы единичный выход любого из нейронов первого слоя приводил к появлению единицы на выходе нейрона второго слоя, реализовав тем самым логическое «или». Имеется 16 двоичных функций от двух переменных. Если выбирать подходящим образом веса и порог, то можно воспроизвести 14 из них (все, кроме «исключающее или» и «исключающее нет»).

Входы не обязательно должны быть двоичными. Вектор непрерывных входов может представлять собой произвольную точку на плоскости XOY . В этом случае мы имеем дело со способностью сети разбивать плоскость на непрерывные области, а не с разделением дискретных множеств точек. Для всех этих функций, однако, линейная делимость показывает, что выход нейрона второго слоя равен единице только в части плоскости XOY , ограниченной многоугольной областью. Поэтому для разделения плоскостей P и Q необходимо, чтобы все P лежали внутри выпуклой многоугольной области, не содержащей точек Q (или наоборот).

Трехслойная сеть, однако, является более общей (рис. 2.10). Ее классифицирующие возможности ограничены лишь числом искусственных нейронов и весов. Ограничения на выпуклость отсутствуют. Теперь нейрон третьего слоя принимает в качестве входа набор выпуклых многоугольников, и их логическая комбинация может быть невыпуклой.

На рис. 2.11 иллюстрируется случай, когда два треугольника A и B , скомбинированные с помощью функций « A и не B », задают невыпуклую область. При добавлении нейронов и весов число сторон многоугольников может неограниченно возрастать. Это позволяет

аппроксимировать область любой формы с любой точностью. Вдобавок не все выходные области второго слоя должны пересекаться. Возможно, следовательно, объединять различные области, выпуклые и невыпуклые, выдавая на выходе единицу всякий раз, когда входной вектор принадлежит одной из них.

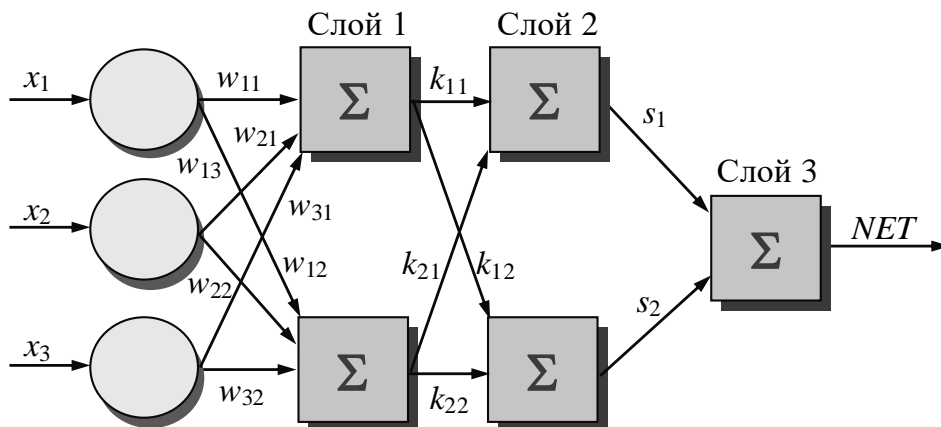


Рис. 2.10. Простейшая двухслойная нейронная сеть

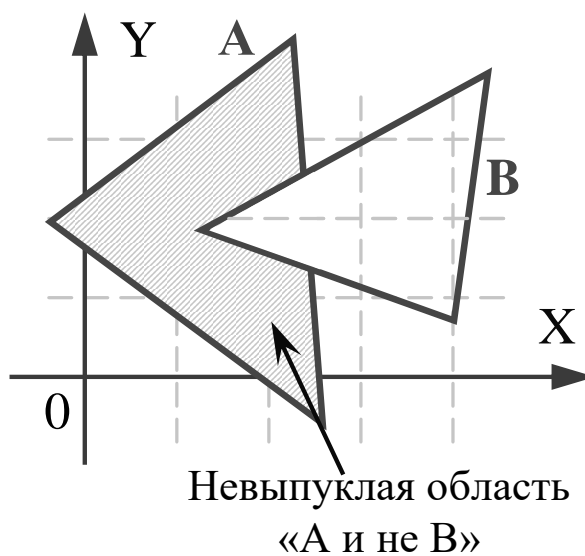


Рис. 2.11. Вогнутая область решений, задаваемая трехслойной сетью

Более крупные и сложные нейронные сети обладают, как правило, и большими вычислительными возможностями. Хотя созданы сети всех конфигураций, какие только можно себе представить, послойная организация нейронов копирует слоистые структуры определенных отделов мозга.

Существуют даже трехмерные и многомерные слоистые нейронные структуры. Однако, многослойные сети могут привести к увеличению вычислительной мощности по сравнению с однослойной сетью лишь в том случае, если активационная функция между слоями будет нелинейной. Вычисление выхода слоя заключается в умножении входного вектора на первую весовую матрицу с последующим умножением (если отсутствует нелинейная активационная функция) результирующего вектора на вторую весовую матрицу[2]

$$NET = (XW_1)W_2.$$

Так как умножение матриц ассоциативно, то

$$NET = X(W_1W_2). \quad (2.4)$$

Это показывает, что двухслойная линейная сеть эквивалентна одному слою с весовой матрицей, равной произведению двух весовых матриц. Следовательно, любая многослойная линейная сеть может быть заменена эквивалентной однослойной сетью.

2.5. Обучение нейронных сетей

Способность искусственных нейронных сетей обучаться является их наиболее интригующим свойством. Подобно

биологическим системам, которые они моделируют, эти нейронные сети сами моделируют себя в результате попыток достичь лучшей модели поведения.

Используя критерий линейной разделимости, можно решить, способна ли однослойная нейронная сеть реализовывать требуемую функцию. Даже в том случае, когда ответ положительный, это принесет мало пользы, если у нас нет способа найти нужные значения весовых коэффициентов. Чтобы сеть представляла практическую ценность, нужен систематический метод (алгоритм) для вычисления этих значений. Розенблатт [4] сделал это в своем алгоритме обучения персептрона вместе с доказательством того, что персептрон может быть обучен всему, что он может реализовывать.

Целью обучения нейронной сети является такая подстройка ее весов, чтобы приложение некоторого множества входов приводило к требуемому множеству выходов.

Для краткости эти множества входов и выходов будут называться *векторами*. При обучении предполагается, что для каждого входного вектора существует парный ему целевой вектор, задающий требуемый выход. Вместе они называются *обучающей парой*. Как правило, сеть обучается на многих парах. Например, входная часть обучающей пары может состоять из набора нулей и единиц, представляющего двоичный образ некоторой буквы алфавита. Выход может быть числом, представляющим букву «А», или другим набором из нулей и единиц, который может быть использован для получения выходного образа. При необходимости распознавать с помощью сети все буквы алфавита, потребовалось бы 26 обучающих пар. Такая группа обучающих пар называется *обучающим множеством*.

Обучение может быть с учителем или без него. Для обучения с учителем нужен «внешний» учитель, который оценивал бы поведение системы и управлял ее последующими модификациями. При обучении без учителя сеть путем самоорганизации делает требуемые изменения.

Рассмотрим обучение персептрона на задаче распознавания образов (рис. 2.12). Обучение персептрона является обучением с учителем.

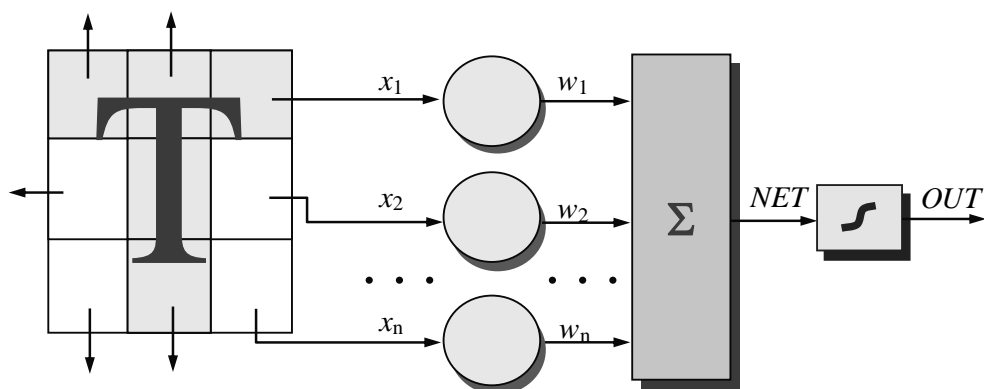


Рис. 2.12. Персептронная система распознавания образов

Персептрон обучают, подавая множество образов по одному на его вход и подстраивая веса до тех пор, пока для всех образов не будет достигнут требуемый выход.

Пусть, для простоты, изображение будет монохромным. Тогда если пиксель содержит точку изображения, то от него подается единица, в противном случае – ноль. Множество пикселей создает вектор входа персептрона.

Допустим, что вектор X содержит значения признаков распознаваемого образа (пикселей) – (x_1, x_2, \dots, x_n) , которые умножаются на соответствующие компоненты вектора весов $W = (w_1, w_2, \dots, w_n)$. Эти произведения суммируются. Если сумма превышает

порог Θ , то выход нейрона OUT равен единице, в противном случае – нулю.

Для обучения сети образ X подается на вход и вычисляется выход OUT . Если OUT правилен, то ничего не меняется. Однако если выход неправилен, то веса, присоединенные к входам, усиливающим ошибочный результат, модифицируются, чтобы уменьшить ошибку.

Метод обучения состоит из следующих шагов:

1. Выбирается очередной входной образ.
2. Значения признаков образа подаются на персептрон и вычисляется OUT .
3. Если выход правильный, то переходят на первый шаг.
4. Если выход неправильный и равен нулю, то добавить значения всех входов к соответствующим им весам; или если выход неправильный и равен единице, то вычесть значение каждого входа из соответствующего ему веса.
5. Перейти на первый шаг.

За конечное число шагов сеть научится разделять образы. Отметим, что это обучение *глобально*, т. е. сеть обучается на всем множестве образов. Возникает вопрос о том, как это множество должно предъявляться, чтобы минимизировать время обучения. Должны ли элементы множества предъявляться последовательно друг за другом или их следует выбирать случайно?

Существует несколько правил изменения шага и подачи образов.

Дельта-правило. Важное обобщение алгоритма обучения персептрона, называемое дельта-правилом, переносит этот метод на непрерывные входы и выходы. Вводится величина δ , которая равна

разности между требуемым или целевым выходом T и реальным выходом OUT

$$\delta = (T - OUT). \quad (2.5)$$

Случай, когда $\delta = 0$, соответствует шагу 3, т.е. когда выход правилен и в сети ничего не изменяется. Шаг 4 соответствует случаю $\delta \neq 0$.

В любом из этих случаев персептронный алгоритм обучения сохраняется: δ умножается на величину каждого входа x_i и это произведение добавляется к соответствующему весу. С целью обобщения вводится коэффициент «скорости обучения» η , который умножается на δx_i , что позволяет управлять средней величиной изменения весов.

В алгебраической форме записи

$$\begin{aligned} \Delta_i &= \eta \delta x_i, \\ w_i(n+1) &= w_i(n) + \Delta_i, \end{aligned}$$

где Δ_i – коррекция, связанная с i -м входом x_i ; $w_i(n+1)$ – значение i -го веса после коррекции; $w_i(n)$ – значение i -го веса до коррекции.

Дельта-правило модифицирует веса в соответствии с требуемым и действительным значениями выхода каждой полярности как для непрерывных, так и для бинарных входов и выходов.

2.6. Процедура обратного распространения

На рис. 2.13 изображена двухслойная сеть, которая может обучаться с помощью процедуры обратного распространения (рисунок упрощен).

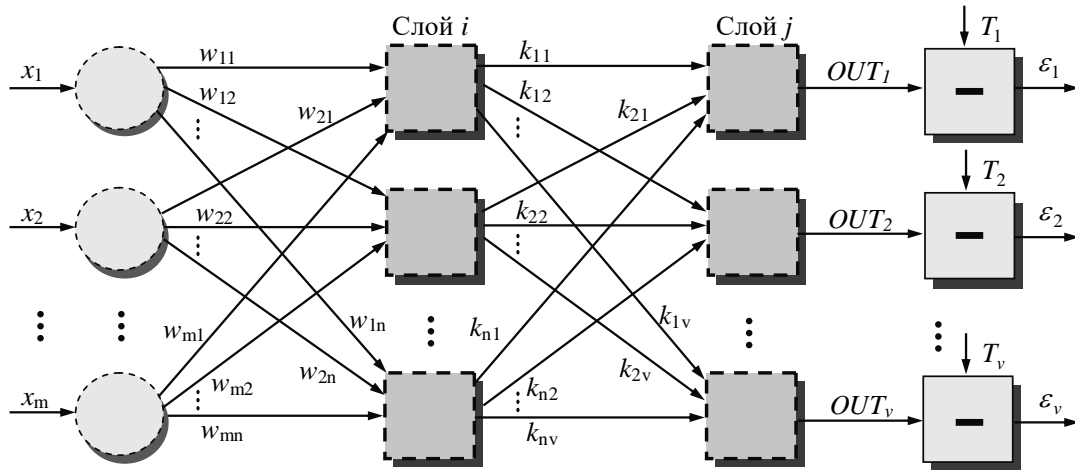


Рис. 2.13. Двухслойная сеть обратного распространения

На вход сети поступает входной вектор сигналов $X(x_1, x_2, \dots, x_m)$, который умножается на матрицу весов W (на рис. 13 показаны все возможные соединения, хотя в реальности ряд из них может отсутствовать). На рис. 13 пунктирными кружками обозначены блоки умножения скаляра на вектор. Далее сигналы поступают на первый слой нейронной сети (слой i), где они суммируются и модифицируются активационными функциями каждого нейрона, а затем полученные выходы умножаются на вектор весов для передачи сигналов на второй слой нейронной сети (слой j). В слое j нет блока умножения выхода нейронов на вектор. Каждый из нейронов в этих слоях на рисунке представлен пунктирным квадратом.

Процедура обратного распространения применима к сетям с любым числом слоев. Однако для того, чтобы продемонстрировать алгоритм, достаточно двух слоев. Сейчас будут рассматриваться лишь

сети прямого действия, хотя обратное распространение применимо и к сетям с обратными связями.

Фактически процедура обратного распространения является алгоритмом обучения сети без учителя.

Перед началом обучения всем весам должны быть присвоены небольшие начальные значения, выбранные случайным образом. Это гарантирует, что в сети не произойдет насыщения большими значениями весов, и предотвращает ряд других патологических случаев. Например, если всем весам придать одинаковые начальные значения, а для требуемого функционирования нужны неравные значения, то сеть не сможет обучиться.

Обучение сети процедурой обратного распространения требует выполнения следующих операций:

1. Выбрать очередную обучающую пару из обучающего множества и подать входной вектор на вход сети.
2. Вычислить выход сети.
3. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).
4. Подкорректировать веса сети так, чтобы минимизировать ошибку.
5. Повторять шаги с 1 по 4 для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

Операции, выполняемые шагами 1 и 2, сходны с теми, которые выполняются при функционировании уже обученной сети, т. е. подается входной вектор и вычисляется получающийся выход. Вычисления выполняются послойно. На рис. 2.13 сначала вычисляются выходы нейронов слоя i , затем они используются с

некоторыми коэффициентами в качестве входов слоя j . После вычисляются выходы нейронов слоя j , которые и образуют выходной вектор сети.

На шаге 3 каждый из выходов сети, которые на рис. 2.13 обозначены OUT_q , вычитается из соответствующей компоненты целевого вектора T_q , чтобы получить ошибку ε_q . Эта ошибка используется на шаге 4 для коррекции весов сети, причем знак и величина изменений весов определяются алгоритмом обучения.

После достаточного числа повторений этих четырех шагов разность между действительными выходами и целевыми выходами должна уменьшиться до приемлемой величины, при этом говорят, что сеть обучилась. Тогда сеть может использоваться для распознавания и веса больше не изменяются.

Шаги 1 и 2 можно рассматривать как «проход вперед», так как сигнал распространяется по сети от входа к выходу. Шаги 3, 4 составляют «обратный проход», здесь вычисляемый сигнал ошибки распространяется обратно по сети и используется для подстройки весов.

Проход вперед. Шаги 1 и 2 могут быть выражены в векторной форме следующим образом: подается входной вектор X и на выходе получается вектор OUT . Векторная пара вход-цель X и T берется из обучающего множества. Вычисления проводятся над вектором X , чтобы получить выходной вектор OUT .

Величина NET каждого нейрона первого слоя вычисляется как взвешенная сумма входов нейрона. Затем активационная функция F «сжимает» NET и дает величину OUT для каждого нейрона в этом слое. Когда множество выходов слоя получено, оно является входным множеством для следующего слоя. Процесс повторяется слой за

слоем, пока не будет получено заключительное множество выходов сети.

$$OUT = F_2(F_1(XW_1)W_2), \quad (2.6)$$

где W_1 – матрица весовых коэффициентов входного слоя, W_2 – матрица весовых коэффициентов передачи выходов первого слоя нейронной сети на входы второго слоя, F_1 – вектор активационных функций нейронов первого слоя, F_2 – вектор активационных функций нейронов второго слоя.

Обратный проход. На данном этапе осуществляется *подстройка весов выходного слоя*. Так как для каждого нейрона выходного слоя задано целевое значение, то подстройка весов этого слоя легко осуществляется с использованием модифицированного дельта-правила (5). Внутренние слои называют «скрытыми слоями», для их выходов не имеется целевых значений для сравнения. Поэтому обучение усложняется.

Выход нейрона слоя j , вычитаясь из целевого значения T_q , дает сигнал ошибки ε_q . Он умножается на производную активационной функции (для логистической функции (1) она равна $OUT_q(1 - OUT_q)$) вычисленную для этого нейрона q слоя j , давая, таким образом, величину δ_q

$$\delta_q = OUT_q(1 - OUT_q)(T_q - OUT_q). \quad (2.7)$$

Затем δ_q умножается на величину OUT_p нейрона слоя i , из которого выходит рассматриваемый вес. Это произведение в свою очередь умножается на коэффициент скорости обучения η (обычно от 0.01 до 1.0), и результат прибавляется к весу. Такая же процедура выполняется для каждого веса от нейрона скрытого слоя к нейрону в выходном слое.

Следующие уравнения иллюстрируют это вычисление

$$\Delta k_{pq} = \eta \delta_q OUT_p, \quad (2.8)$$

$$k_{pq}(n+1) = k_{pq}(n) + \Delta k_{pq}, \quad (2.9)$$

где $k_{pq}(n)$ – величина веса от нейрона p в скрытом слое к нейрону q в выходном слое на шаге n (до коррекции); $k_{pq}(n+1)$ – величина веса на шаге $n + 1$ (после коррекции); δ_q – величина δ для нейрона q , в выходном слое j ; OUT_p – величина OUT для нейрона p в скрытом слое i .

Подстройка весов скрытого слоя. Рассмотрим один нейрон в скрытом слое i , предшествующем выходному слою j . При проходе вперед этот нейрон передает свой выходной сигнал нейронам в выходном слое через соединяющие их веса. Во время обучения эти веса функционируют в обратном порядке, пропуская величину δ_q от выходного слоя назад к скрытому слою. Каждый из этих весов умножается на величину δ_q нейрона, к которому он присоединен в выходном слое. Величина δ_p , необходимая для нейрона скрытого слоя, получается суммированием всех таких произведений и умножением на производную сжимающей функции

$$\delta_p = OUT_p (1 - OUT_p) \left[\sum_q \delta_q k_{pq} \right]. \quad (2.10)$$

Когда значение δ_p получено, веса, первого скрытого слоя, могут быть подкорректированы с помощью уравнений

$$\Delta w_{xp} = \eta \delta_p x_p, \quad (2.11)$$

$$w_{xp}(n+1) = w_{xp}(n) + \Delta w_{xp}, \quad (2.12)$$

где $w_{xp}(n)$ – величина веса от входа x к нейрону p в скрытом слое на шаге n (до коррекции); $w_{xp}(n+1)$ – величина веса на шаге $n + 1$ (после коррекции); δ_p – величина δ для нейрона p , в скрытом слое i ; x_p – величина входа для нейрона p в скрытом слое i .

Для каждого нейрона в данном скрытом слое должно быть вычислено δ_p и подстроены все веса, ассоциированные с этим слоем. Этот процесс повторяется слой за слоем по направлению к входу, пока все веса не будут подкорректированы.

С помощью векторных обозначений операция обратного распространения ошибки может быть записана значительно компактнее. Обозначим множество величин δ_q выходного слоя через D_q и множество весов выходного слоя как массив K_q . Чтобы получить D_j , δ -вектор выходного слоя, достаточно следующих двух операций:

1. Умножить вектор выходного слоя D_q на транспонированную матрицу весов K_q^T , соединяющую скрытый уровень с выходным уровнем.

2. Умножить каждую компоненту полученного произведения на производную сжимающей функции соответствующего нейрона в скрытом слое.

В символьной записи

$$D_j = D_q K_q^T * [OUT_p * (I - OUT_p)] \quad (2.13)$$

где $*$ – оператор поэлементного произведения векторов, OUT_p – выходной вектор слоя i , I – вектор, все элементы которого равны единице.

Импульс. В работе [21] описан метод ускорения обучения для алгоритма обратного распространения, увеличивающий также устойчивость процесса. Этот метод, названный импульсом, заключается в добавлении к коррекции веса члена, пропорционального величине предыдущего изменения веса. Как только происходит коррекция, она «запоминается» и служит для модификации всех последующих коррекций. Уравнения коррекции

(2.8), (2.9) и по аналогии (2.11), (2.12) модифицируются следующим образом:

$$\Delta k_{pq}(n+1) = \eta \delta_q OUT_p + \alpha \Delta k_{pq}(n), \quad (2.14)$$

$$k_{pq}(n+1) = k_{pq}(n) + \Delta k_{pq}(n+1), \quad (2.15)$$

где α – коэффициент импульса, обычно устанавливается около 0.9.

Используя метод импульса, сеть стремится идти по дну узких оврагов поверхности ошибки (если таковые имеются), а не двигаться от склона к склону. Этот метод, по-видимому, хорошо работает на некоторых задачах, но дает слабый или даже отрицательный эффект на других.

В работе [23] описан сходный метод, основанный на экспоненциальном сглаживании, который может иметь преимущество в ряде приложений. В этом случае формулы (2.14), (2.15) будут иметь вид

$$\Delta k_{pq}(n+1) = (1-\alpha) \delta_q OUT_p + \alpha \Delta k_{pq}(n), \quad (2.16)$$

$$k_{pq}(n+1) = k_{pq}(n) + \eta \Delta k_{pq}(n+1), \quad (2.17)$$

где α – коэффициент сглаживания, варьируемый в диапазоне от 0.0 до 1.0. Если α равен 1.0, то новая коррекция игнорируется и повторяется предыдущая. В области между 0 и 1 коррекция веса сглаживается величиной, пропорциональной α .

Метод обратного распространения имеет ряд недостатков:

- **паралич сети** – когда в процессе обучения значения весов становятся большими величинами, т.е. нейроны будут иметь большие значения выходов, а поскольку производная активационной функции при больших значениях мала, то производная в (2.7), (2.10), (2.13) будет также мала. В результате процесс обучения «замрет». Для предотвращения паралича обычно уменьшают размер шага η , но это

увеличивает время обучения. Можно также попытаться ввести эвристические правила обнаружения паралича;

– **локальные минимумы** – в процессе обучения будут найдены не наилучшие значения весов, поскольку обратное распространение представляет собой разновидность градиентного спуска, т. е. осуществляет спуск вниз по поверхности ошибки, непрерывно подстраивая веса в направлении к минимуму. Если поверхность ошибки имеет сложную форму, то обучение застревает в локальном экстремуме. Для избегания подобной ситуации используют статистические методы обучения, но они медленны;

– **размер шага** – в [21] говорится о том, что коррекция весов должна осуществляться бесконечно малым шагом, но на практике шаг должен быть конечным и достаточно большим. В тоже время размер шага не должен приводить к параличу сети и постоянной неустойчивости;

– **временная неустойчивость** – заключается в том, что нейронная сеть забывает, чему ее уже обучили, при дальнейшем обучении. Процесс обучения должен быть таким, чтобы сеть обучалась на всем обучающем множестве без пропусков того, что уже выучено. Рекомендует перед коррекцией весов на вход сети подать все обучающее множество. К сожалению, если входные образы не могут быть повторены данный подход не применим, поскольку будут постоянные осцилляции.

2.6.1. Программная реализация нейросети обратного распространения ошибки

Алгоритм реализован в виде класса `NeuroNetwork`. Для описания нейрона создана структура `neuron_type`, содержащая следующие поля:

- список весовых коэффициентов для связей между данным нейроном и всеми нейронами предыдущего слоя (или входными данными, если нейрон находится во входном слое). Каждый весовой коэффициент - действительное число (по одному весовому коэффициенту на нейрон предыдущего слоя);
- пороговый уровень;
- значение ошибки, используется только на стадии обучения;
- изменение ошибки, также используется только во время обучения.

Выходной сигнал нейрона хранится в поле a (так называемая активность нейрона). Нейрон должен отреагировать на входной сигнал, поступающий по взвешенным связям, вычислив при этом выходной сигнал. Для трансформации входных сигналов в выходные необходима функция активации. Функция активации $x_i^{[n+1]} = f(a_i^{[n]})$ реализована в открытом методе `sigmoid`. В данном методе

используется формула
$$f(x) = \frac{1}{1 + e^{-ax}}$$

Биологические нейроны не срабатывают (не выдают выходной сигнал) до тех пор, пока уровень входного сигнала не достигнет некоторого порогового значения, т.е. на вход нейрона поступает сумма взвешенных сигналов минус некоторая величина. Полученное значение проходит через активационную функцию.

Каждая активность нейрона предыдущего слоя умножается на соответствующий весовой коэффициент, результаты умножения

суммируются $a_i^{[n]} = \sum_j w_{ij}^{[n]} x_j^{[n]}$, вычитается пороговое значение, вычисляется значение сигмоидной функции $x_i^{[n+1]} = f\left(\sum_j w_{ij}^{[n]} x_j^{[n]}\right)$.

Вычисление суммы взвешенных сигналов для скрытого и выходного слоев (методы `run_hidden_layer` и `run_output_layer`) производится аналогично.

Каждый слой нейронов базируется на выходе предыдущего слоя (за исключением входного слоя, базирующегося непосредственно на предъявляемых сети входных данных (в коде – массив `test_pat`). Это значит, что значения входного слоя должны быть полностью рассчитаны до вычисления значений скрытого слоя, которые в свою очередь должны быть рассчитаны до вычисления значений выходного слоя.

Выходы нейронной сети – значения активностей (поле `a`) нейронов выходного слоя. Программа, симулирующая работу нейронной сети, в процессе обучения будет сравнивать их со значениями, которые должны быть на выходе сети.

Полный алгоритм обучения НС с помощью процедуры обратного распространения строится следующим образом:

1. Инициализировать пороговые значения и весовые коэффициенты небольшими случайными величинами (не более 0.4). Инициализация весовых коэффициентов случайными вещественными значениями с помощью класса `Random` производится в функции `random_weights`.

2. Подать на входы сети один из возможных образов и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних. Метод `run_the_networ`.

3. Вычислить ошибки для выходного слоя (`calculate_output_layer_errors`). При этом используем формулу $E_i = (t_i - a_i) \cdot a_i \cdot (1 - a_i)$ (здесь E_i – ошибка для i -го узла выходного слоя, a_i – активность данного узла, t_i – требуемая активность для требуемого выходного значения) для каждого i -го значения выходного слоя. Ниже представлена соответствующая программа в виде функции.

Вычисление ошибки для скрытого и входного слоев производится методами `calculate_input_layer_errors` и `calculate_hidden_layer_errors` по формуле $E_i = a_i \cdot (1 - a_i) \cdot \sum_j E_j \cdot w_{ij}$.

Используя формулы, получаем функцию, обучающую весовые коэффициенты и пороговые уровни.

Далее объединяем указанные выше функции в одном методе `back_propagate()`.

Для очистки предыдущих значений используется функция `blank_changes`.

Для упрощения временной сложности работы сети полученные весовые коэффициенты будем записывать в отдельные файлы (метод `AddWeightsToFile()`), имена которым даются программой автоматически. Для считывания сохраненных параметров будет применяться метод `ExtractWeights()`.

Класс перевода текста в двоичный вид предназначен для бинаризации исходных данных (слов), т.е. перевода слов с естественного языка в набор единиц и нулей. Данная процедура является необходимой, так как нейронная сеть обратного распространения ошибки работает только с двоичными данными.

Параметром конструктора является массив строк для перевода в двоичный вид.

Алгоритм работы методов `GetBinarizeWord` и `GetBinarizeText` данного класса в общем состоит из следующих этапов:

- кодировка слова: суммирование произведений ASCII-кодов букв на $i+4$, где i – номер буквы в слове;

- перевод полученного десятичного числа в двоичный вид при помощи метода DecToBin;

- обработка полученных данных.

Класс распознавания является главным. Его задача – распознавание. В нем используются объекты всех описанных выше классов.

Параметрами конструктора являются:

- текст для анализа – sText;

- параметры нейросети N_HID, beta, m, Epoch;

- индикатор необходимости обучения нейросети – flag.

Из всех перечисленных выше методов наиболее важными являются конструктор, Scanning и GetNeuroResult.

Анализируемый текст сначала подается в конструктор. Там он разбивается на отдельные лексемы, тип которых либо определяется сразу (если это знак препинания или имя собственное), либо посредством метода Scanning. Метод Scanning логически можно разделить на три блока:

- поиск слова в хеш-таблицах;

- идентификация слова по окончанию и определение его типа;

- идентификация слова по окончанию и проведение углубленного анализа с помощью нейронной сети.

При решении задачи распознавания печатных символов можно выделить два этапа:

- локализация символа на изображении;

- распознавание символов на знаке.

Для упрощения дальнейших расчетов и программной реализации преобразуем исходное цветное изображение в полутоновое (рис. 2.14).

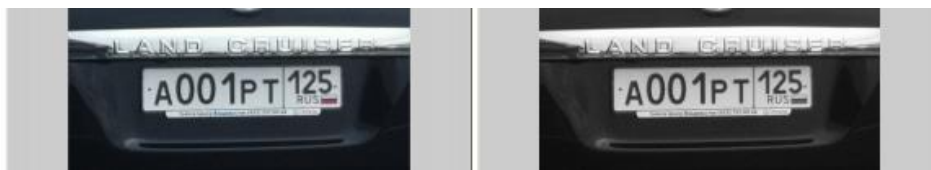


Рис. 2.14. Преобразование изображения в оттенки серого

Теперь, как и при решении большинства других задач, необходимо провести предварительную обработку исходных изображений. Этот этап особенно важен при решении задач распознавания. От качества решения этапа предварительной обработки во многом зависит эффективность решения задачи распознавания в целом.

К наиболее распространенным дефектам, которые могут присутствовать на исходных изображениях, относятся шум и низкий уровень контрастности. Для устранения импульсных выбросов используем медианную фильтрацию.

Результат обработки представлен на рис. 2.15.



Рис. 2.15. Изображение с импульсными шумами и после устранения импульсных шумов

Фильтр, повышающий резкость изображения, имеет маску, определяемую следующим выражением:

$$h = \frac{1}{a+1} \begin{bmatrix} -a & a-1 & -a \\ a-1 & a+5 & a-1 \\ -a & a-1 & -a \end{bmatrix},$$

где параметр a выбирается в пределах от 0 до 1.

Кроме этого, для улучшения качества изображения можно воспользоваться гистограммой. На рис. 2.16 показано, как устраняется размытость изображения.



Рис. 2.16. Устранение размытости изображения

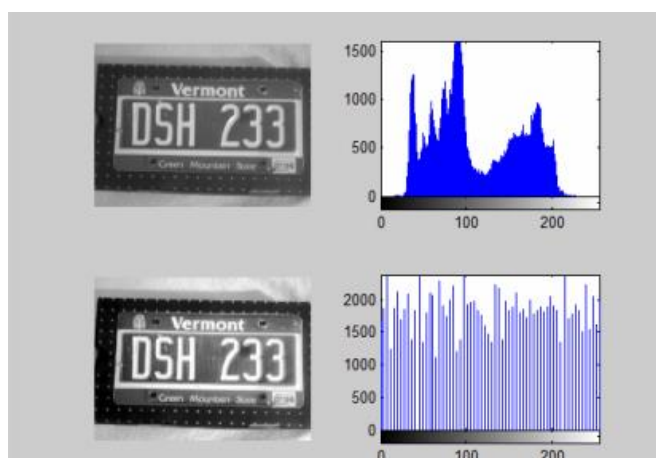


Рис. 2.17. Выравнивание гистограммы

Гистограмма улучшает контраст изображения с помощью преобразования значений пикселей исходного изображения таким образом, чтобы гистограмма яркостей пикселей результирующего изображения соответствовала некоторой предопределенной гистограмме (рис. 2.17). Данная функция предназначена для преобразования полутоновых изображений или палитровых изображений.

Определение расположения печатных символов на изображении. После проведения предварительной обработки изображения необходимо определить расположение печатных

символов. Для этого на изображении ищутся так называемые связанные области пикселей и создается матрица L , каждый элемент которой равен номеру объекта, которому принадлежит соответствующий пиксель изображения. Размер матрицы номеров объектов L равен размеру изображения. Объекты нумеруются по порядку, начиная с 1. Элементы, имеющие значение 1, относятся к первому объекту, имеющие значение 2 - ко второму объекту и т.д. Если элемент в матрице L равен 0, то это означает, что соответствующий пиксель исходного изображения относится к фону. Параметр n указывает критерий связности, используемый для нахождения связанных областей-объектов. Параметр n может принимать значения 4 или 8 (значение по умолчанию).

На рис. 2.18 показан результат выполнения программы.



Рис. 2.18. Результат выполнения программы

Следующий этап – вычисление признаков найденных объектов. Элементы матрицы L , имеющие значение 1, относятся к первому объекту, имеющие значение 2 - ко второму объекту и т.д. Если элемент в матрице L равен 0, то он относится к фону.

Пусть N - количество пикселей, относящихся к объекту. Все множество пикселей $p(x, y)$, относящихся к объекту, обозначим через Q . Тогда координаты центра масс объекта вычисляются как

$$x_c = \frac{1}{N} \sum_{p(x,y) \in Q} x \quad y_c = \frac{1}{N} \sum_{p(x,y) \in Q} y.$$

Затем вычисляется несколько вспомогательных величин:

$$U_x = \frac{1}{12} + \frac{1}{N} \sum_{p(x,y) \in \Omega} (x - x_c)^2,$$

$$U_y = \frac{1}{12} + \frac{1}{N} \sum_{p(x,y) \in \Omega} (y - y_c)^2,$$

$$c = \sqrt{(U_x - U_y)^2 + 4 \cdot U_{xy}^2}.$$

Тогда длины максимальной A_{\max} и минимальной A_{\min} осей инерции вычисляются следующим образом:

$$A_{\min} = 2\sqrt{2} \cdot \sqrt{U_x + U_y - c}.$$

Длины главных осей инерции используются для вычисления эксцентриситета и ориентации объекта. Эксцентриситет определяется с помощью соотношения

$$E = \frac{2 \cdot \sqrt{(0.5 \cdot A_{\max})^2 - (0.5 \cdot A_{\min})^2}}{A_{\max}}.$$

Ориентация определяется как угол в градусах между максимальной осью инерции и осью X. Если $U_y > U_x$, то ориентация O вычисляется с помощью формулы

$$O = \frac{180}{\pi} \cdot \arctg \left(\frac{U_y - U_x + c}{2 \cdot U_{xy}} \right),$$

в противном случае O вычисляется как

$$O = \frac{180}{\pi} \cdot \arctg \left(\frac{2 \cdot U_{xy}}{U_y - U_x + c} \right).$$



Рис. 2.19. Цифрами обозначены найденные объекты

Приведем результаты вычислений признаков для всех объектов изображения. В качестве параметра был выбран коэффициент заполнения, который равен отношению площади объекта к площади ограничивающего прямоугольника.

2.6.2. Распознавание символов

После локализации печатных символов на изображении выполняется второй этап – распознавание символов. Первый шаг – выделение части изображения. Результат выполнения функции показан на рис. 2.20.



Рис. 2.20. Исходное изображение с номерным знаком и изображение с выделенной областью

Получаем координаты расположения первого символа на изображении (рис. 2.21).

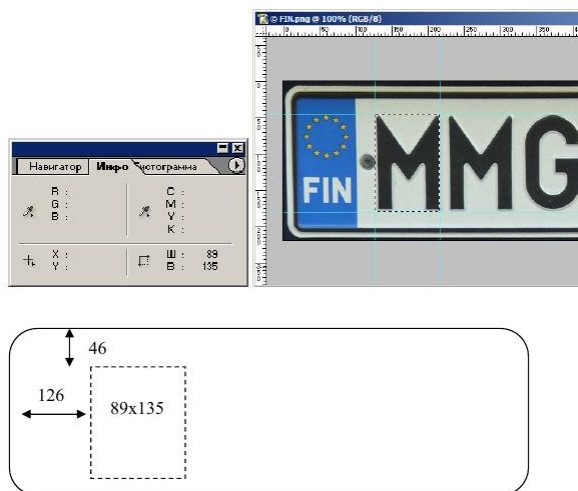


Рис. 2.21. Определение координат первого символа

Используя исходное изображение и программу Photoshop, создаем эталонное изображение (рис. 2.22).

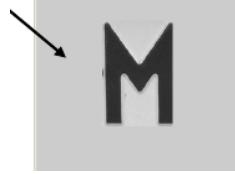


Рис. 2.22. Исходное и эталонное изображения

В настоящей работе распознаются только бинарные образы, поэтому на втором этапе после получения картинка она бинаризуется. При работе с цветной камерой преобразование из цвета в черно-белый происходит по стандартной формуле:

$$Y:=0.3*R+0.59*G+0.11*B.$$

Далее алгоритм довольно прост: есть некоторая планка, если цвет оттенка серого выше - он считается белым, если ниже - считается черным. На стадии бинаризации преобразуется объект изображения в бинарную матрицу данных, после чего начинается работа уже не с объектом картинка, а с бинарной матрицей.

Если не предусматривать некоторое разбиение общего изображения на части, то ни один из описанных выше алгоритмов не сможет корректно работать. Разбиение изображения на части, каждая из которых содержит свой уникальный объект, называется сегментацией.

Следует заметить, что в сегментации четко разделяются черно-белые изображения на бинарные и с оттенками серого. Здесь работают совершенно разные по скорости и сложности алгоритмы, однако интуитивно понятно, что любое изображение с оттенками серого можно бинаризовать по некоторым правилам.

После успешного завершения сегментации каждый сегмент попадает в модуль распознавания. Для того чтобы образы

распознавались инвариантно относительно положения и поворота, надо привязаться к их структуре. У каждого бинарного образа можно вычислить несколько признаков, не зависящих от его поворота или размера.

Опишем применение нейронных сетей (НС) для распознавания изображений.

НС состоит из элементов, называемых формальными нейронами, которые сами по себе очень просты и связаны с другими нейронами. Каждый нейрон преобразует набор сигналов, поступающих к нему на вход в выходной сигнал. Именно связи между нейронами, кодируемые весами, играют ключевую роль. Одно из преимуществ НС (а также недостаток при реализации их на последовательной архитектуре) это то, что все элементы могут функционировать параллельно, тем самым существенно повышая эффективность решения задачи, особенно в обработке изображений. Кроме того, что НС позволяют эффективно решать многие задачи, они предоставляют мощные гибкие и универсальные механизмы обучения, а это является их главным преимуществом перед другими методами [2, 21] (вероятностные методы, линейные разделители, решающие деревья и т.п.). Обучение избавляет от необходимости выбирать ключевые признаки, их значимость и отношения между признаками. Но, тем не менее, выбор исходного представления входных данных (вектор в n -мерном пространстве, частотные характеристики, вэйлеты и т.п.) существенно влияет на качество решения и является отдельной темой. НС обладают хорошей обобщающей способностью (лучше, чем у решающих деревьев [2]), т.е. могут успешно распространять опыт, полученный на конечном обучающем наборе, на все множество образов.

Исследование методов и программно-аппаратных систем оптического распознавания символов позволяет сформулировать следующие выводы:

1. Современное состояние технологии автоматического распознавания печатных текстов (OCR) позволяет решать задачу автоматизации ввода информации при необходимом уровне надежности.

2. При построении системы OCR, включающей оптическое устройство оцифровки изображений, блоки локализации и выделения элементов текста, предобработки изображения, выделения признаков, распознавания символов и постобработки результатов распознавания, необходимо использовать методы и алгоритмы, обладающие высокой робастностью к яркостно-геометрическим искажениям и сложным текстурным фонам.

3. В качестве таких методов и алгоритмов могут быть использованы: процедуры определения строк, знакомест на основе модификаций преобразования Hough; методы, основанные на исследовании устойчивых статистических распределений точек; методы, использующие интегральные преобразования, а также структурный анализ символов.

4. При разработке современных систем OCR для повышения качества распознавания символов и слов необходимо учитывать контекстную информацию, использование которой позволяет не только находить ошибки, но и исправлять их.

Переходя к программе, разработанной в ходе выполнения исследований, необходимо отметить, что хотя в ней и не применяются системы искусственного интеллекта (перцептроны и нейросети), а используется довольно простой метод сравнения с эталонными

символами, алгоритм дает приемлемый результат на заранее известном наборе эталонов.

Применение этого метода уместно в тех случаях, когда необходимо распознавать большие объемы текстов, напечатанных одним шрифтом в едином размере. При таких условиях результаты распознавания могут конкурировать с методами, основанными на использовании нейросетей, и не уступать им по скорости распознавания.

При всем этом метод сравнения с эталоном гораздо проще других алгоритмов, использует простой математический аппарат. Однако небольшие отклонения входных данных от эталонных значений приводят к резкому падению качества распознавания.

2.7. Сети встречного распространения

Возможности сети встречного распространения превосходят возможности однослойных сетей, а время обучения, по сравнению с обратным распространением, может уменьшаться на несколько порядков. Встречное распространение не столь общо, как обратное распространение, но оно может давать решение в тех приложениях, где долгая обучающая процедура невозможна.

Во встречном распространении объединены два хорошо известных алгоритма: самоорганизующаяся карта Кохонена [29] и звезда Гроссберга [28]. Их объединение ведет к свойствам, которых нет ни у одного из них в отдельности.

Сеть встречного распространения обладает одним из важных свойств для распознавания образов – обобщающая способность сети позволяет получать правильный выход даже при приложении входного вектора, который является неполным или слегка неверным.

Структура сети. На рис. 2.23 показана упрощенная версия сети встречного распространения. На нем иллюстрируются функциональные свойства этой парадигмы.

Как видно из рис. 2.23 сеть встречного распространения очень похожа на двухслойную сеть обратного распространения. Различие заключается в операциях, выполняемых нейронами Кохонена и Гроссберга.

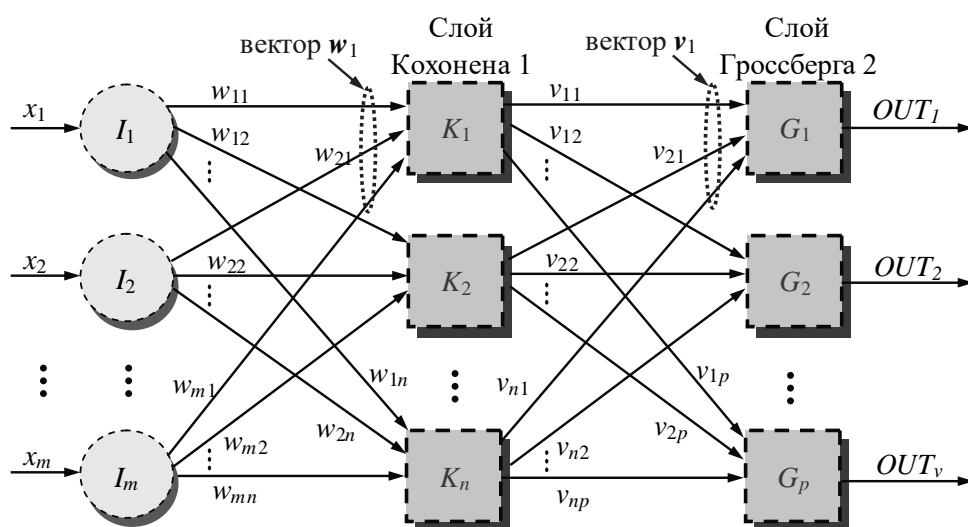


Рис. 2.23. Сеть встречного распространения без обратных связей

Слой Кохонена. В своей простейшей форме слой Кохонена функционирует следующим образом: только один нейрон из слоя выдает на выходе логическую единицу, все остальные выдают ноль для заданного входного вектора.

Ассоциированное с каждым нейроном Кохонена множество весов соединяет его с каждым входом. Например, на рис. 2.23 нейрон Кохонена K_1 имеет веса $w_{11}, w_{21}, \dots, w_{m1}$, составляющие весовой вектор w_1 . Они соединяются через входной слой с входными сигналами x_1, x_2, \dots, x_m , составляющими входной вектор X . Подобно нейронам большинства сетей выход NET каждого нейрона Кохонена является просто суммой взвешенных входов

$$NET_j = \sum_i x_i w_{ij}, \quad (2.18)$$

где NET_j – это выход NET нейрона Кохонена j .

Выход нейрона Кохонена с максимальным значением NET равен единице, у остальных он равен нулю[29].

Слой Гроссберга. Слой Гроссберга функционирует в сходной манере. Его выход OUT является взвешенной суммой выходов нейронов K_1, K_2, \dots, K_n слоя Кохонена

$$OUT_j = \sum_i NET_i v_{ij}, \quad (2.19)$$

где NET_i – выход i -го нейрона Кохонена, OUT_j – выход j -го нейрона Гроссберга.

Если слой Кохонена функционирует таким образом, что лишь у одного нейрона величина NET равна единице, а у остальных равна нулю, то лишь один элемент векторов v_1, v_2, \dots, v_p отличен от нуля, и вычисления очень просты. Фактически каждый нейрон слоя Гроссберга лишь выдает величину веса, который связывает этот нейрон с единственным ненулевым нейроном Кохонена [28].

Обучение слоя Кохонена. Слой Кохонена классифицирует входные векторы в группы схожих. Это достигается с помощью такой подстройки весов слоя Кохонена, что близкие входные векторы активируют один и тот же нейрон данного слоя. Затем задачей слоя Гроссберга является получение требуемых выходов.

Обучение Кохонена является самообучением, протекающим без учителя. Поэтому трудно (и не нужно) предсказывать, какой именно нейрон Кохонена будет активироваться для заданного входного вектора. Необходимо лишь гарантировать, чтобы в результате обучения разделялись несхожие входные векторы.

Предварительная обработка входных векторов. Весьма желательно (хотя и не обязательно) нормализовать входные векторы перед тем, как предъявлять их сети. Это выполняется с помощью деления каждой компоненты входного вектора на длину вектора. Эта длина находится извлечением квадратного корня из суммы квадратов компонент вектора. В алгебраической записи

$$x'_i = \frac{x_i}{\sqrt{x_1^2 + x_2^2 + \dots + x_m^2}}. \quad (2.20)$$

Это превращает входной вектор в единичный вектор с тем же самым направлением, т. е. в вектор единичной длины в m -мерном пространстве.

При обучении слоя Кохонена на вход подается входной вектор и вычисляются его скалярные произведения с векторами весов, связанными со всеми нейронами Кохонена. Далее веса нейрона с максимальным значением скалярного произведения подстраиваются. Так как скалярное произведение, используемое для вычисления величин *NET*, является мерой сходства между входным вектором и вектором весов, то процесс обучения состоит в выборе нейрона Кохонена с весовым вектором, наиболее близким к входному вектору, и дальнейшем приближении весового вектора к входному. Сеть самоорганизуется таким образом, что данный нейрон Кохонена имеет максимальный выход для данного входного вектора. Уравнение, описывающее процесс обучения имеет следующий вид:

$$w_n = w_c + \alpha(x - w_c), \quad (2.21)$$

где w_n – новое значение веса, соединяющего входную компоненту x с нейроном, имеющим единицу на выходе; w_c – предыдущее значение этого веса; α – коэффициент скорости обучения, который может варьироваться в процессе обучения.

Каждый вес, связанный с нейроном Кохонена, дающим единицу на выходе, изменяется пропорционально разности между его величиной и величиной входа, к которому он присоединен. Направление изменения минимизирует разность между весом и его входом.

На рис. 2.24 этот процесс показан геометрически в двумерном виде. Сначала находится вектор $X - W_c$. Затем этот вектор укорачивается умножением его на скалярную величину α , меньшую единицы, в результате чего получается вектор изменения δ . Окончательно новый весовой вектор W_n является отрезком, направленным из начала координат в конец вектора δ . Отсюда можно видеть, что эффект обучения состоит во вращении весового вектора в направлении входного вектора без существенного изменения его длины.

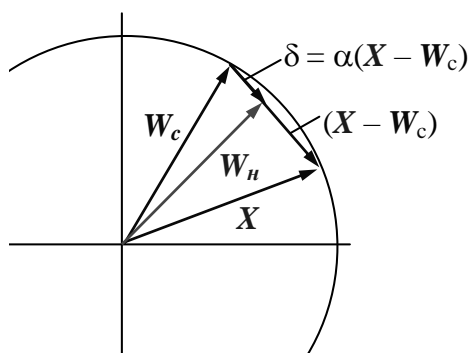


Рис. 2.24. Вращение весового вектора в процессе обучения

Коэффициентом скорости обучения вначале обычно равен 0,7 и может постепенно уменьшаться в процессе обучения. Это позволяет делать большие начальные шаги для быстрого грубого обучения и меньшие шаги при подходе к окончательной величине.

Если бы с каждым нейроном Кохонена ассоциировался один входной вектор, то слой Кохонена мог бы быть обучен с помощью одного вычисления на вес. Веса нейрона с единицей на выходе приравнялись бы к компонентам обучающего вектора ($\alpha = 1$). Как

правило, обучающее множество включает много сходных между собой входных векторов, и сеть должна быть обучена активировать один и тот же нейрон Кохонена для каждого из них. В этом случае веса этого нейрона должны получаться усреднением входных векторов, которые должны его активировать. Постепенное уменьшение величины α уменьшает воздействие каждого обучающего шага, так что окончательное значение будет средней величиной от входных векторов, на которых происходит обучение. Таким образом, веса, ассоциированные с нейроном, примут значение вблизи «центра» входных векторов, для которых данный нейрон дает единицу на выходе.

Выбор начальных значений весовых векторов. Всем весам сети перед началом обучения следует придать начальные значения. Общепринятой практикой при работе с нейронными сетями является присваивание весам небольших случайных значений. При обучении слоя Кохонена случайно выбранные весовые векторы следует нормализовать. Окончательные значения весовых векторов после обучения совпадают с нормализованными входными векторами. Поэтому нормализация перед началом обучения приближает весовые векторы к их окончательным значениям, сокращая, таким образом, обучающий процесс.

Рандомизация весов слоя Кохонена может породить проблему в процессе обучения, так как в результате ее весовые векторы распределяются равномерно по поверхности гиперсферы. Из-за того, что входные векторы, как правило, распределены неравномерно и имеют тенденцию группироваться на относительно малой части поверхности гиперсферы, большинство весовых векторов будут так удалены от любого входного вектора, что они никогда не будут давать наилучшего соответствия. Эти нейроны Кохонена будут всегда иметь

нулевой выход и окажутся бесполезными. Более того, если начальная плотность весовых векторов в окрестности обучающих векторов слишком мала, то может оказаться невозможным разделить сходные классы из-за того, что не будет достаточного количества весовых векторов на интересующей поверхности гиперсферы, чтобы приписать по одному из них каждому классу входных векторов. В тоже время, если несколько входных векторов получены незначительными изменениями из одного и того же образца и должны быть объединены в один класс, то они должны включать один и тот же нейрон Кохонена. Если же плотность весовых векторов очень высока вблизи группы слегка различных входных векторов, то каждый входной вектор может активировать отдельный нейрон Кохонена. Это не приводит к проблеме, поскольку слой Гроссберга может отобразить различные нейроны Кохонена в один и тот же выход, но это расточительная трата нейронов Кохонена.

Наиболее желательное решение состоит в том, чтобы распределять весовые векторы в соответствии с плотностью входных векторов, которые должны быть разделены, помещая тем самым больше весовых векторов в окрестности большого числа входных векторов. На практике это невыполнимо, однако существует несколько методов приближенного достижения тех же целей.

Одно из решений, известное под названием метода выпуклой комбинации (convexcombinationmethod), состоит в том, что все веса приравняются одной и той же величине $w_i = 1/\sqrt{m}$, где m – число входов и, следовательно, число компонент каждого весового вектора. Благодаря этому все весовые векторы совпадают и имеют единичную длину. Каждой же элементу входного вектора X придается значение $x_i = \alpha x_i + (1-\alpha)/\sqrt{m}$, где m – число входов. В начале α очень мало, вследствие чего все входные векторы имеют длину, близкую к $1/\sqrt{m}$,

и почти совпадают с векторами весов. В процессе обучения сети α постепенно возрастает, приближаясь к единице. Это позволяет разделять входные векторы и окончательно приписывает им их истинные значения. Весовые векторы отслеживают один или небольшую группу входных векторов и в конце обучения дают требуемую картину выходов. Метод выпуклой комбинации хорошо работает, но замедляет процесс обучения, так как весовые векторы подстраиваются к изменяющейся цели.

Другой подход состоит в добавлении шума к входным векторам. Тем самым они подвергаются случайным изменениям, схватывая в конце концов весовой вектор. Этот метод также работоспособен, но еще более медленен, чем метод выпуклой комбинации.

Третий метод начинает со случайных весов, но на начальной стадии обучающего процесса подстраивает все веса, а не только связанные с нейроном Кохонена, дающим на выходе единицу. Тем самым весовые векторы перемещаются ближе к области входных векторов. В процессе обучения коррекция весов начинает производиться лишь для ближайших к нейрону с выходом равным единице. Этот радиус коррекции постепенно уменьшается, так что в конце концов корректируются только веса, связанные с нейроном имеющим единицу на выходе.

Еще один метод разделяет время обучения. Если нейрон Кохонена чаще других дает единицу на выходе (более $1/n$ раз, где n – число нейронов Кохонена), он временно увеличивает свой порог, что уменьшает его шансы иметь единицу на выходе, давая тем самым возможность обучаться и другим нейронам.

Во многих приложениях точность результата существенно зависит от распределения весов. К сожалению, эффективность

различных решений исчерпывающим образом не оценена и остается проблемой.

Обучение слоя Гроссберга. Слой Гроссберга обучается относительно просто. Входной вектор, являющийся выходом слоя Кохонена, подается на слой нейронов Гроссберга, и выходы слоя Гроссберга вычисляются, как при нормальном функционировании. Далее, каждый вес корректируется лишь в том случае, если он соединен с нейроном Кохонена, имеющим ненулевой выход. Величина коррекции веса пропорциональна разности между весом и требуемым выходом нейрона Гроссберга, с которым он соединен. В символьной записи

$$v_{ijn} = v_{ijc} + \beta(OUT_j - v_{ijc})NET_i, \quad (2.22)$$

где NET_i – выход i -го нейрона Кохонена (только для одного нейрона Кохонена он отличен от нуля); OUT_j – j -ая компонента вектора желаемых выходов, v_{ijc} – старое значение настраиваемого веса, v_{ijn} – новое значение веса.

Первоначально β берется равным около 0,1 и затем постепенно уменьшается в процессе обучения.

Из (2.22) видно, что веса слоя Гроссберга будут сходиться к средним величинам от желаемых выходов, тогда как веса слоя Кохонена обучаются на средних значениях входов. Обучение слоя Гроссберга – это обучение с учителем, алгоритм располагает желаемым выходом, по которому он обучается. Обучающийся без учителя, самоорганизующийся слой Кохонена дает выходы в недетерминированных позициях. Они отображаются в желаемые выходы слоем Гроссберга.

2.8. Нейронные сети Хопфилда и Хэмминга

Среди различных конфигураций искусственных нейронных сетей (НС) встречаются такие, при классификации которых по принципу обучения, строго говоря, не подходят ни обучение с учителем [21], ни обучение без учителя [2]. В таких сетях весовые коэффициенты синапсов рассчитываются только однажды перед началом функционирования сети на основе информации об обрабатываемых данных, и все обучение сети сводится именно к этому расчету. С одной стороны, предъявление априорной информации можно расценивать, как помощь учителя, но с другой – сеть фактически просто запоминает образцы до того, как на ее вход поступают реальные данные, и не может изменять свое поведение, поэтому говорить о звене обратной связи с "миром" (учителем) не приходится. Из сетей с подобной логикой работы наиболее известны сеть Хопфилда и сеть Хэмминга, которые обычно используются для организации ассоциативной памяти. Далее речь пойдет именно о них.

Структурная схема сети Хопфилда приведена на рис.2.25. Она состоит из единственного слоя нейронов, число которых является одновременно числом входов и выходов сети. Каждый нейрон связан синапсами со всеми остальными нейронами, а также имеет один входной синапс, через который осуществляется ввод сигнала. Выходные сигналы, как обычно, образуются на аксонах.

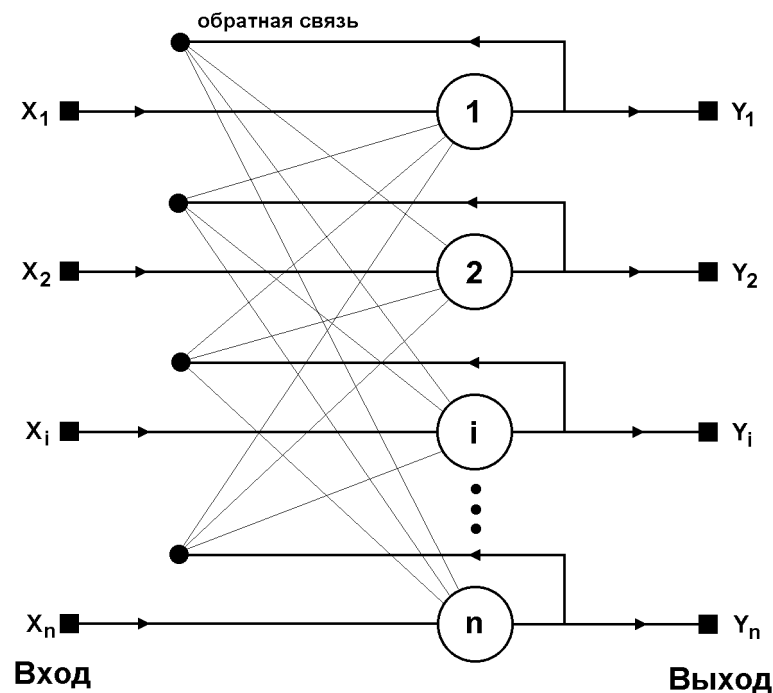


Рис.2.25. Структурная схема сети Хопфилда

Задача, решаемая данной сетью в качестве ассоциативной памяти, как правило, формулируется следующим образом. Известен некоторый набор двоичных сигналов (изображений, звуковых оцифровок, прочих данных, описывающих некие объекты или характеристики процессов), которые считаются образцовыми. Сеть должна уметь из произвольного неидеального сигнала, поданного на ее вход, выделить ("вспомнить" по частичной информации) соответствующий образец (если такой есть) или "дать заключение" о том, что входные данные не соответствуют ни одному из образцов. В общем случае, любой сигнал может быть описан вектором $\mathbf{X} = \{ x_i: i=0..n-1 \}$, n – число нейронов в сети и размерность входных и выходных векторов. Каждый элемент x_i равен либо $+1$, либо -1 . Обозначим вектор, описывающий k -ый образец, через \mathbf{X}^k , а его

компоненты, соответственно, – x_i^k , $k=0\dots m-1$, m – число образцов. Когда сеть распознаёт (или "вспомнит") какой-либо образец на основе предъявленных ей данных, ее выходы будут содержать именно его, то есть $\mathbf{Y} = \mathbf{X}^k$, где \mathbf{Y} – вектор выходных значений сети: $\mathbf{Y} = \{ y_i: i=0,\dots,n-1 \}$. В противном случае, выходной вектор не совпадет ни с одним образцовым.

Если, например, сигналы представляют собой некие изображения, то, отобразив в графическом виде данные с выхода сети, можно будет увидеть картинку, полностью совпадающую с одной из образцовых (в случае успеха) или же "вольную импровизацию" сети (в случае неудачи).

На стадии инициализации сети весовые коэффициенты синапсов устанавливаются следующим образом [3]:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, & i \neq j \\ 0, & i = j \end{cases} \quad (2.23)$$

Здесь i и j – индексы, соответственно, предсинаптического и постсинаптического нейронов; x_i^k, x_j^k – i -ый и j -ый элементы вектора k -ого образца.

Алгоритм функционирования сети следующий (p – номер итерации):

1. На входы сети подается неизвестный сигнал. Фактически его ввод осуществляется непосредственной установкой значений аксонов:

$$y_i(0) = x_i, \quad i = 0\dots n-1, \quad (2.24)$$

поэтому обозначение на схеме сети входных синапсов в явном виде носит чисто условный характер. Ноль в скобке справа от y_i означает нулевую итерацию в цикле работы сети.

2. Рассчитывается новое состояние нейронов

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p), j=0 \dots n-1 \quad (2.25)$$

и новые значения аксонов

$$y_j(p+1) = f[s_j(p+1)], \quad (2.26)$$

где f – активационная функция в виде скачка, приведенная на рис.2.26а.

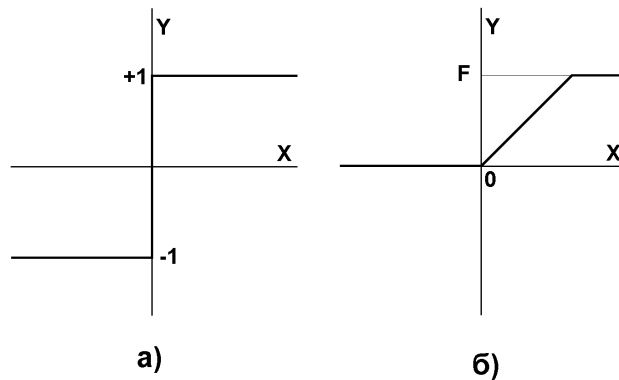


Рис.2.26. Активационные функции

3. Проверка, изменились ли выходные значения аксонов за последнюю итерацию. Если да – переход к пункту 2, иначе (если выходы застabilizировались) – конец. При этом выходной вектор представляет собой образец, наилучшим образом сочетающийся с входными данными.

Как говорилось выше, иногда сеть не может провести распознавание и выдает на выходе несуществующий образ. Это связано с проблемой ограниченности возможностей сети. Для сети Хопфилда число запоминаемых образов m не должно превышать величины, примерно равной $0.15 \cdot n$. Кроме того, если два образа А и Б сильно похожи, они, возможно, будут вызывать у сети перекрестные ассоциации, то есть предъявление на входы сети вектора А приведет к появлению на ее выходах вектора Б и наоборот.

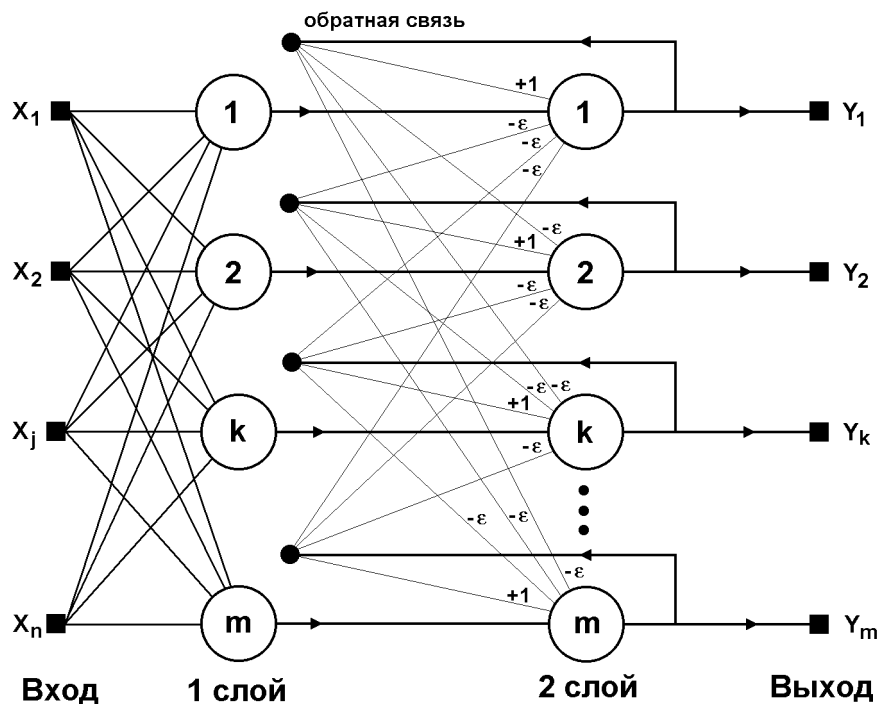


Рис.2.27 Структурная схема сети Хэмминга

Когда нет необходимости, чтобы сеть в явном виде выдавала образец, то есть достаточно, скажем, получать номер образца, ассоциативную память успешно реализует сеть Хэмминга. Данная сеть характеризуется, по сравнению с сетью Хопфилда, меньшими затратами на память и объемом вычислений, что становится очевидным из ее структуры (рис. 2.27).

Сеть состоит из двух слоев. Первый и второй слои имеют по m нейронов, где m – число образцов. Нейроны первого слоя имеют по n синапсов, соединенных со входами сети (образующими фиктивный нулевой слой). Нейроны второго слоя связаны между собой ингибиторными (отрицательными обратными) синаптическими связями. Единственный синапс с положительной обратной связью для каждого нейрона соединен с его же аксоном.

Идея работы сети состоит в нахождении расстояния Хэмминга от тестируемого образа до всех образцов. Расстоянием Хэмминга

называется число отличающихся битов в двух бинарных векторах. Сеть должна выбрать образец с минимальным расстоянием Хэмминга до неизвестного входного сигнала, в результате чего будет активизирован только один выход сети, соответствующий этому образцу.

На стадии инициализации весовым коэффициентам первого слоя и порогу активационной функции присваиваются следующие значения:

$$w_{ik} = \frac{x_i^k}{2}, i=0\dots n-1, k=0\dots m-1, \quad (2.27)$$

$$T_k = n/2, k = 0\dots m-1. \quad (2.28)$$

Здесь x_i^k – i -ый элемент k -ого образца.

Весовые коэффициенты тормозящих синапсов во втором слое берут равными некоторой величине $0 < \varepsilon < 1/m$. Синапс нейрона, связанный с его же аксоном имеет вес $+1$.

Алгоритм функционирования сети Хэмминга следующий:

1. На входы сети подается неизвестный вектор $\mathbf{X} = \{x_i; i=0\dots n-1\}$, исходя из которого рассчитываются состояния нейронов первого слоя (верхний индекс в скобках указывает номер слоя):

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j, j=0\dots m-1. \quad (2.29)$$

После этого полученными значениями инициализируются значения аксонов второго слоя:

$$y_j^{(2)} = y_j^{(1)}, j = 0\dots m-1. \quad (2.30)$$

2. Вычислить новые состояния нейронов второго слоя:

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p), k \neq j, j = 0\dots m-1 \quad (2.31)$$

и значения их аксонов:

$$y_j^{(2)}(p+1) = f[s_j^{(2)}(p+1)], j = 0 \dots m-1 \quad (2.32)$$

Активационная функция f имеет вид порога (рис. 2.26б), причем величина F должна быть достаточно большой, чтобы любые возможные значения аргумента не приводили к насыщению.

3. Проверить, изменились ли выходы нейронов второго слоя за последнюю итерацию. Если да – перейди к шагу 2. Иначе – конец.

Из оценки алгоритма видно, что роль первого слоя весьма условна: воспользовавшись один раз на шаге 1 значениями его весовых коэффициентов, сеть больше не обращается к нему, поэтому первый слой может быть вообще исключен из сети (заменен на матрицу весовых коэффициентов), что и было сделано в ее конкретной реализации, описанной ниже.

Приведена тестовая программа для проверки сети. Здесь конструируется сеть со вторым слоем из пяти нейронов, выполняющая распознавание пяти входных образов, которые представляют собой схематичные изображения букв размером 5 на 6 точек (см.рис.2.28а). Обучение сети фактически сводится к загрузке и запоминанию идеальных изображений. Затем на ее вход поочередно подаются зашумленные на 8/30 образы (см.рис.2.28б), которые она успешно различает.

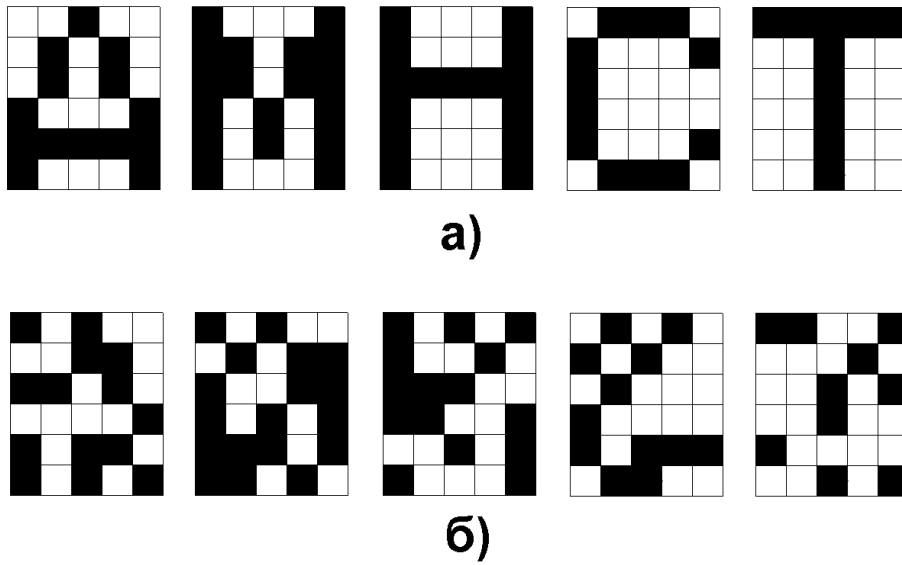


Рис. 2.28. Образцовые и тестовые образы

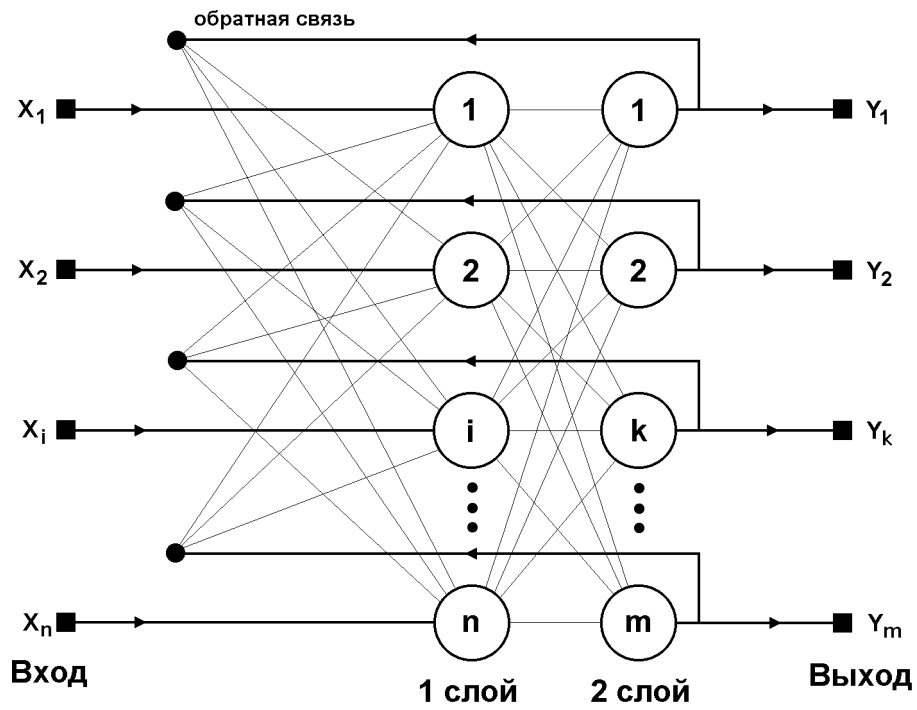


Рис.2.29. Структурная схема ДАП

В проект кроме файлов NEURO_HN и NEUROHAM входят также SUBFUN и NEURO_FF, описанные в [1]. Программа тестировалась в среде Borland C++ 3.1.

Предложенные классы позволяют моделировать и более крупные сети Хэмминга. Увеличение числа и сложности распознаваемых образов ограничивается фактически только объемом ОЗУ. Следует отметить, что обучение сети Хэмминга представляет самый простой алгоритм из всех рассмотренных до настоящего времени алгоритмов.

Обсуждение сетей, реализующих ассоциативную память, было бы неполным без хотя бы краткого упоминания о двунаправленной ассоциативной памяти (ДАП). Она является логичным развитием парадигмы сети Хопфилда, к которой для этого достаточно добавить второй слой. Структура ДАП представлена на рис.2.29. Сеть способна запоминать пары ассоциированных друг с другом образов. Пусть пары образов записываются в виде векторов $X^k = \{x_i^k: i=0..n-1\}$ и $Y^k = \{y_j^k: j=0..m-1\}$, $k=0..r-1$, где r – число пар. Подача на вход первого слоя некоторого вектора $P = \{p_i: i=0..n-1\}$ вызывает образование на входе второго слоя некоего другого вектора $Q = \{q_j: j=0..m-1\}$, который затем снова поступает на вход первого слоя. При каждом таком цикле вектора на выходах обоих слоев приближаются к паре образцовых векторов, первый из которых – X – наиболее походит на P , который был подан на вход сети в самом начале, а второй – Y – ассоциирован с ним. Ассоциации между векторами кодируются в весовой матрице $W^{(1)}$ первого слоя. Весовая матрица второго слоя $W^{(2)}$ равна транспонированной первой $(W^{(1)})^T$. Процесс обучения, также как и в случае сети Хопфилда, заключается в предварительном расчете элементов матрицы W (и соответственно W^T) по формуле:

$$w_{ij} = \sum_k x_i y_j, i = 0 \dots n-1, j = 0 \dots m-1 \quad (2.33)$$

Эта формула является развернутой записью матричного уравнения

$$\mathbf{W} = \sum_k \mathbf{X}^T \mathbf{Y} \quad (2.34)$$

для частного случая, когда образы записаны в виде векторов, при этом произведение двух матриц размером соответственно $[n \times 1]$ и $[1 \times m]$ приводит к (2.33).

В заключении можно сделать следующее обобщение. Сети Хопфилда, Хэмминга и ДАП позволяют просто и эффективно разрешить задачу воссоздания образов по неполной и искаженной информации. Невысокая емкость сетей (число запоминаемых образов) объясняется тем, что, сети не просто запоминают образы, а позволяют проводить их обобщение, например, с помощью сети Хэмминга возможна классификация по критерию максимального правдоподобия [2,21]. Вместе с тем, легкость построения программных и аппаратных моделей делают эти сети привлекательными для многих применений.

2.9. Нейросетевое распознавание объектовна изображениях топологических слоевинтегральных микросхем

Составной частью современной технологии проектирования и производства интегральных сем (ИС) являются системы технического зрения. Несмотря на ряд успехов в области идентификации и распознавания объектов на основе цифровой обработки изображений, к числу нерешенных можно отнести проблему использования теории нейронных сетей (НС) для решения задач обработки изображений топологии ИС. Основные нерешенные задачи здесь состоят в адекватном отображении предметной области на нейронную систему, выборе моделей используемых НС и их интеграции в единую интеллектуальную систему.

При обработке изображений топологии требуется по некоторым признакам выделять некоторые однородные области изображения, причем, как правило, подобие нечеткое и часто нарушается. Этапы предварительной обработки изображения позволяют уменьшить влияние искажений на процесс распознавания [2]. Тем не менее, имеет место распознавание в условиях неполной и нечеткой информации. Наиболее подходят для ее решения нейросетевые технологии, НС при этом . выступает в роли классификатора. Применение НС в задачах обработки визуальной информации обосновывается также свойством обучаемости или адаптивности НС к новым задачам, при этом сохраняются архитектура сети и алгоритм ее функционирования.

Концепция нейросетевой обработки и идентификации видеоизображений предполагает использование следующих подходов к обработке изображений топологических слоев ИС:

- 1) проблемно-ориентированной предварительной обработки, сохраняющей информационные признаки топологических объектов,

что позволяет сократить число связей НС, упростить и ускорить процесс обучения;

2) выделения (идентификации) объекта на изображении для нейросетевой обработки за счет использования информации об иерархии признаков, что сокращает затраты времени на обработку (осуществляется поиск лишь в идентифицированной области);

3) использования набора классификаторов, в котором по результатам классификации на тестовой выборке производится выбор наилучшей модели классификатора для обработки всех изображений.

Существует ряд НС, позволяющих решать задачу распознавания образов с определенной степенью точности [2-4,21]: многослойный персептрон, сеть на основе радиальной базисной функции, сети АРТ, сети Хопфилда, самоорганизующееся отображение Кохонена, неокогнитрон. Наиболее подходящим для этой цели является неокогнитрон. Для повышения точности иерархического распознавания и увеличения производительности НС исследования проводятся в следующих направлениях: модификация правил выделения признаков за счет введения новых дополнительных инвариантов относительно искажений [2]; модификация структуры и принципов послойной обработки [21]; разработка алгоритмов самообучения [2].

В этом параграфе рассмотрены нейросетевая реализация названных выше второго и третьего подходов, а также представлена структура системы, реализующей нейросетевую технологию распознавания объектов топологии интегральных микросхем.

2.10. Выделение объекта на изображении

Нейросетевой подход к идентификации (поиску) объектов предполагает разработку НС, алгоритма ее обучения и технологии ее применения. Поиск элементов топологии на основе НС состоит в сканировании изображения окном, размер которого равен размеру обучающего изображения, и определения по отклику НС, является ли изображение внутри окна элементом. Выбор архитектуры НС определяется типом обрабатываемого слоя и формой его представления на разных этапах обработки и осуществляется, как правило, путем экспериментальных просчетов и оценок их качества проектировщиком ИС. Учитывая многообразие топологических слоев предлагается для решения задачи идентификации совокупность НС, включающая многослойный персептрон, НС с радиальной базисной функцией и неокогнитрон, которые апробированы и рекомендуются для использования.

Многослойный персептрон состоит из множества слоев нейронных элементов, причем известно, что достаточно трех слоев для создания сколь угодно сложной решающей граничной поверхности в пространстве обучающих образов. При ее построении количество n нейронов первого слоя определяется в соответствии с количеством информативных признаков (яркостные значения (r, g, b) , семантические дескрипторы и др.). Количество нейронов промежуточных слоев определяется эмпирически. Выходной слой имеет p нейронов, активность которых определяет принадлежность изображения окна к элементу в виде некоторой функции, значение которой сравнивается с некоторым заданным порогом. Каноническим методом обучения такой сети считается алгоритм обратного распространения ошибки.

При использовании НС с радиальной базисной функцией не существует проблемы выбора количества скрытых нейронов, так как они определяются количеством групп (кластеров), на которые разбивается пространство обучающих образов при помощи радиальной базисной функции, что положительно отличает их от сетей обратного распространения ошибки.

Неокогнитрон (рис. 2.30) позволяет выполнить при поиске элементов сравнение изображения в окне сканирования с эталонным образцом с учетом иерархии признаков, формируемой в процессе обучения.

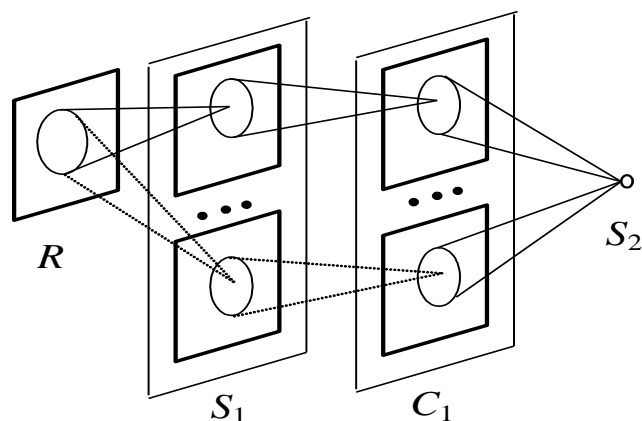


Рис. 2.30. Используемая архитектура сети

Слой R является входным. S_1 -слой предназначен для выделения общих признаков, встречающихся у всех элементов ИС, таких, как линейные границы перепадов яркостей различной ориентации. Все подслои этого слоя состоят из нейронов с одинаковыми размерами подгрупп рецепторного поля 4×4 нейрона. Обучающие образы для этих подслоев изображены на рис. 2.31.



Рис. 2.31. Обучающие образы S_1 -слоя

S_1 -слой предназначен для обобщения признаков, выделяемых в S_1 -слое, а рецепторные подгруппы его нейронов организованы таким образом, чтобы объединять такие парные признаки, как перепады яркости с темного на светлый или наоборот (объединены на рис. 2.31 горизонтальной прямоугольной скобкой). В результате получаем четыре подслоя в S_1 -слое. Размер подгрупп рецепторного поля выбран такой, чтобы активность нейронов из этого слоя была инвариантна по отношению к малым сдвигам признаков, выделяемых в предыдущем слое, и составляет поле 2×2 . S_2 -слой предназначен для выделения совокупности признаков, присущих одному конкретному элементу ИС. Он формирует выходное значение сети и состоит из одного нейрона, а размеры подгрупп рецепторного поля этого нейрона совпадают с размерами подслоев S_1 -слоя.

Входными данными для тестирования являлись полутонные фрагменты фотоизображения поликремниевого слоя ИС (рис. 2.32). Идентифицировались два типа фрагментов. Соответственно производились два варианта обучения НС, обучающие выборки для которых показаны на рис. 2.33.

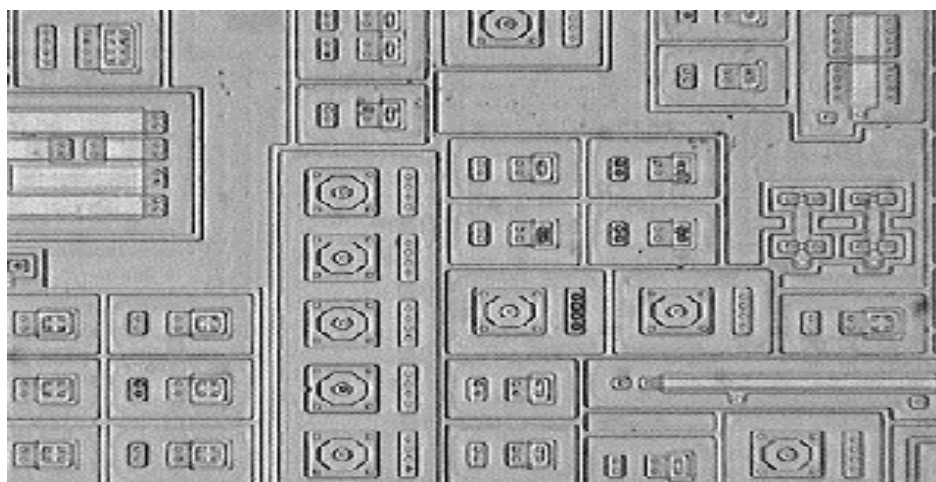


Рис. 2.32. Фрагмент поликремниевого слоя ИС

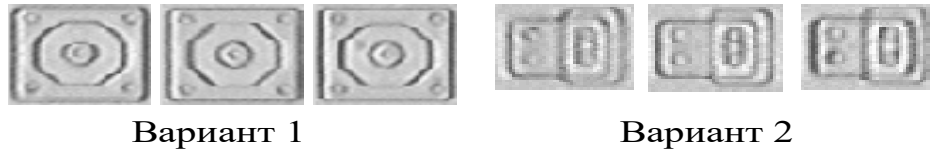


Рис. 2.33. Обучающие образы S_2 -слоя НС

Результаты тестирования показали, что метод формирования критерия схожести двух изображений с помощью описанной НС является лучшим по сравнению с методом расчета корреляции этих двух изображений по формуле:

$$\mu = \frac{\sum_{x,y} a_{x,y} \cdot b_{x,y}}{\sqrt{\sum_{x,y} a_{x,y}^2 \cdot \sum_{x,y} b_{x,y}^2}}, \quad (2.35)$$

где x, y – координаты точки сравниваемых изображений; a, b – значение цвета точки для усредненного изображения и изображения в скользящем окне соответственно (табл. 2.3) . Этот результат был достигнут вследствие того, что НС является инвариантной к искажениям формы и яркости изображения распознаваемого объекта, а также требует меньшего количества обучающих образцов для достижения необходимой точности идентификации.

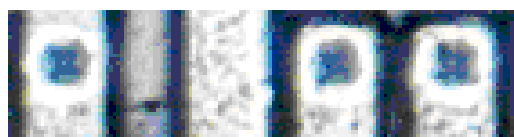
Для демонстрации пригодности персептрона ниже приведены тесты с сетью с сигмоидной функцией активации для идентификации контактных площадок (КП) двух типов (рис. 2.34). НС состояла из 35 входных нейронов, скрытого слоя из 2.34 нейронов и выходного слоя из одного нейрона (для поиска каждого типа объекта формируется своя сеть).

Результаты тестирования

Объем обучающей выборки	Точность идентификации			
	Корреляционный метод		Нейронная сеть	
	Вариант 1	Вариант 2	Вариант 1	Вариант 2
1	63,1	67,4	87,0	89,0
3	72,3	73,6	96,3	97,0
6	81,4	84,2	98,5	99,0



а)



б)

Рис. 2.34. Примеры исходных обучающих изображений КП:

а) первый тип; б) второй тип

В табл. 2.4 приведены результаты идентификации на пяти тестах, время обучения НС, которые показывают стабильность и высокое качество идентификации.

Результаты тестирования

Объем обучающей выборки	Количество тестируемых КП	Число итераций обучения	Суммарная квадратичная ошибка	Процент идентификации
18	20	2868	0,098	95
30	20	4225	0,099	100
40	50	3656	0,099	100
70	70	19906	0,099	100
70	70	18091	0,099	100

Для эффективной идентификации топологических объектов целесообразно сформировать библиотеку элементов, которая включает в себя различные варианты их расположения (ориентации). Так, предварительный анализ изображений показал целесообразность определить следующие типы элементов металлизации, показанные на рис. 2.35.

Под элементами металлизации понимается часть изображения, которая вмещает в себя изображение типа: контактная площадка, пересечение дорожек, дорожка. Каждый тип в свою очередь может быть представлен множеством его модификаций. К примеру, в зависимости от расположения на дорожке, КП могут быть левые или правые (расположены на левом или правом концах конце горизонтальной дорожки) и др. (рис. 2.36).



a) б) в) г) д)

Рис. 2.35. Типы элементов изображения металлизации:

- a)* КП; *б)* часть дорожки; *в)* изгиб дорожки; *г)* дефект дорожки;
д) пересечение дорожек текущего и предыдущего слоя



a) б) в) г) д)

Рис. 2.36. Виды КП изображения металлизации:

- a)* верхняя; *б)* правая; *в)* левая; *г)* нижняя; *д)* нижняя правая

Результаты идентификации КП различных видов трехслойным перцептроном с числом входов, равным числу пикселей эталонного изображения КП (40×40), скрытым слоем из 10 нейронов и выходным слоем, количество нейронов которого определяется числом видов идентифицируемых КП (до 10) показали, что большинство КП были правильно идентифицированы (77 из 83 КП, что составляет 93,9 %).

Типовой алгоритм идентификации (поиска) объектов на изображениях топологических слоев ИС имеет следующий вид.

1. Начало. Подготовить (выбрать из БД) образцы изображений искомых объектов.
2. Сформировать (выбрать из БД) архитектуру НС для поиска элементов на изображении ИС и выполнить ее настройку (обучение).
3. Сохранить информацию об архитектуре НС в БД в виде таких ее полей, как размер рецепторного слоя и матрицы весовых коэффициентов.

4. Выполнить сканирование изображения топологии ИС окном соответствующего размера и определить координаты мест расположения искомым объектов топологии.

5. Конец.

2.10.1. Анализ и классификация

С целью улучшения характеристик обработки информации и точности распознавания был проведен анализ структуры неокогнитрона и разработана архитектура многослойной НС, которая реализует многоуровневый процесс распознавания. В основе правила активации нейронов лежит метод сравнения матриц яркостей изображений, который позволяет сравнивать матрицы с нечетким позиционированием соответствующих элементов матриц [4]. При этом функция активации определяется формулой

$$c = 1 + \frac{n \sum_{x,y} (w_{x,y} a'_{x,y}) - \sum_{x,y} w_{x,y} \sum_{x,y} a'_{x,y}}{\sqrt{\left(n \sum_{x,y} (w_{x,y} a'_{x,y}) - \left(\sum_{x,y} w_{x,y} \right)^2 \right) \left(n \sum_{x,y} (w_{x,y} a'_{x,y}) - \left(\sum_{x,y} a'_{x,y} \right)^2 \right)}}, \quad (2.36)$$

где n – количество точек в эталонной матрице; $w_{x,y}$ – значения точек в эталонной матрице; $a'_{x,y}$ – значения соответствующих точек на анализируемом изображении,

$$a'_{x,y} = \begin{cases} 0, & \left[\min_{rx=-R, \dots, R; ry=-R, \dots, R} (a_{x+rx, y+ry}); \max_{rx=-R, \dots, R; ry=-R, \dots, R} (a_{x+rx, y+ry}) \right] \cap \\ & \left[\min_{rx=-R, \dots, R; ry=-R, \dots, R} (w_{x+rx, y+ry}); \max_{rx=-R, \dots, R; ry=-R, \dots, R} (w_{x+rx, y+ry}) \right] \neq \emptyset; \\ a_{x,y}, & \left[\min_{rx=-R, \dots, R; ry=-R, \dots, R} (a_{x+rx, y+ry}); \max_{rx=-R, \dots, R; ry=-R, \dots, R} (a_{x+rx, y+ry}) \right] \cap \\ & \left[\min_{rx=-R, \dots, R; ry=-R, \dots, R} (w_{x+rx, y+ry}); \max_{rx=-R, \dots, R; ry=-R, \dots, R} (w_{x+rx, y+ry}) \right] = \emptyset. \end{cases}$$

Параметр R – радиус геометрических искажений, определяющий максимально допустимое смещение пикселей эталона на изображении (образе), вычисляется по формуле $R = [0,3 \times 1/F]$, где

$$F = \frac{F_h + F_v}{2},$$

$$F_h = \frac{1}{X} \left(1 + \frac{1}{Y} \sum_{x=1, y=1}^{X-1, Y} \delta(g_{x,y}^h, g_{x+1,y}^h) \right),$$

$$F_v = \frac{1}{Y} \left(1 + \frac{1}{X} \sum_{x=1, y=1}^{X, Y-1} \delta(g_{x,y}^v, g_{x,y+1}^v) \right),$$

$g_{x,y}^h, g_{x,y}^v$ – значения вертикального и горизонтального градиентов в точке (x, y) ,

$$\delta(a,b) = \begin{cases} 1, & \text{sign}(a) \neq \text{sign}(b), \\ 0, & \text{в остальных случаях.} \end{cases}$$

В результате применения (2.36) получаем значения в интервале $c \in [0; 2]$, где значения $c \in [0; 1)$ говорят об обратной корреляции, значения $c = 1$ – об отсутствии корреляции, а значения $c \in (1; 2]$ – о прямой корреляции.

В рассматриваемой здесь постановке задачи распознавания предполагается, что существует набор изображений объектов, в котором каждый объект представлен несколькими изображениями, отличающимися друг от друга. Чтобы сформировать такое множество, разделим все изображения объектов на три множества: обучающее, тестовое для обучения и тестовое. Обучающее множество – это множество изображений объектов, которое формируется как результат алгоритма формирования обучающего множества. Главной характеристикой является его репрезентативность, т.е. достаточность различных представлений объекта на изображении для того, чтобы классификатор смог распознать все остальные возможные вариации

представления объекта. Тестовое множество для обучения – это некоторая небольшая часть общего множества изображений, которая используется при формировании обучающего множества. Будем считать, что тестовое множество для обучения является репрезентативным, но в то же время избыточным. Избыточность заключается во включении в множество таких элементов, которые можно исключить из процесса обучения. Тестовое множество – это большая часть общего множества изображений.

Теперь для формирования обучающего множества выполняются следующие действия.

1. Выбрать в обучающее множество по одному экземпляру изображений каждого класса из тестового множества для обучения.
2. Произвести обучение.
3. Выполнить распознавание на тестовом множестве для обучения.
4. Добавить в обучающее множество экземпляры изображений каждого класса, которые не были распознаны.
5. Повторить пп. 2-4 до тех пор, пока не будет достигнут требуемый уровень распознавания.

После построения обучающего множества для каждого слоя решается задача кластеризации входных данных слоя, к которой сводится задача обучения слоя.

Поиск лучшей архитектуры НС осуществлялся следующим методом. Для различных вариантов архитектуры она обучалась на обучающем множестве, затем тестировалась на тестовом множестве для обучения. По результатам тестов выбиралась лучшая архитектура.

Структура системы обработки изображений показана на рис. 2.37.



Рис. 2.37. Структура нейросетевой системы

Разработаны алгоритмы идентификации и распознавания объектов топологии на основе нейросетевого подхода с использованием следующих НС: многослойного персептрона, НС с радиальной базисной функцией и неокогнитрона, которые обеспечивают стабильность и снижение вычислительной сложности идентификации при нечеткой информации об объектах топологии. Предложенные алгоритмы являются составной частью нейросетевой технологии обработки топологии ИС и используется как для поиска собственно объектов топологии (которые затем классифицируются, и выполняется их анализ на наличие дефектов), так и для поиска специфических областей топологии для проведения их детального анализа.

2.10.2. Методы видеонаблюдения, сегментации и сопровождения движущихся объектов

Главным признаком, положенным в основу разрабатываемых методов видеонаблюдения, сегментации и сопровождения, является движение. Его наиболее информативная оценка – вектора движения.

Основой определения векторов движения является уравнение оптического потока:

$$\langle \nabla \mathbf{L}, \mathbf{V} \rangle + I_t = 0, \quad (2.37)$$

где $\nabla \mathbf{L} = \left(\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y} \right)$ – яркостный вектор-градиент по пространственным координатам; $I_t = \frac{\partial L}{\partial t}$ – производная яркости по времени; $\mathbf{V} = \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t} \right)$ – вектор движения.

Анализ уравнения оптического потока (2.37) показывает:

1. Уравнение является плохо обусловленным.

2. Однозначное определение вектора движения возможно только в случае, если компоненты яркостного вектора - градиента

$\nabla \mathbf{L} = \left(\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y} \right)$ отличны от нуля (имеют место изменения яркости по

горизонтали и вертикали). В случае однородной, нетекстурированной поверхности или текстуры только в одном направлении достоверную оценку векторов найти нельзя.

3. Уравнение (2.37) получено из предположения о постоянной яркости точки (пикселя) при ее движении. Подсветка, тени, блики, прозрачные и зеркальные поверхности нарушают это утверждение и приводят к ошибкам при определении векторов движения.

Вектора движения, методы и алгоритмы их определения были разработаны для стандартов видеокompрессии MPEG 1,2,4 для

обработки изображений студийного качества, где соблюдаются требования текстурированности и постоянства яркости.

При работе в сложных условиях наблюдения с определенным в рамках решаемой задачи классом объектов требования существования пространственных яркостных производных и постоянства яркости при движении точки (пикселя) вдоль траектории нарушаются. Изображение объектов интереса имеет низкую текстурированность. При видеонаблюдении на открытом воздухе высока вероятность изменения уровня освещенности, появления солнечных бликов, теней и др. Не соблюдение обоих требований порождает появление аномальных векторов – векторов, не отражающих реальное движение в кадре.

Тяжесть последствий от аномальных векторов движения в прикладных ТВ системах значительна: сегментация объектов интереса, снижение точности моделей движения, потеря объекта при сопровождении и др. Это определяет критерий оценки эффективности методов и алгоритмов определения векторов движения в прикладных ТВ системах – уровень достоверности:

$$K_{\partial} = 1 - \frac{Q_{ан}}{Q},$$

где Q – общее число найденных векторов, $Q_{ан}$ – число аномальных векторов в общем числе найденных

Проведенные исследования показали, что даже при использовании корреляционных методов (наиболее робастных по данным литературы) с алгоритмом полного перебора для поиска лучшего соответствия уровень достоверности $K_{\partial} = 0,6$. Этот уровень недостаточен для эффективного применения векторов движения в прикладных ТВ системах. Необходима разработка новых методов

определения векторов движения, учитывающих особенности объектов интереса, и свойства видеоданных, получаемых в сложных условиях наблюдения.

В методе совмещения блоков задачу определения векторов движения решают путем минимизации целевой функции, характеризующей степень соответствия (совпадения) двух блоков, на множестве различных положений обрабатываемого блока в области поиска. Результаты решения этой задачи определены видом целевой функции, которая зависит от уровня детальности изображения в блоке (таблица 2.5).

Под уровнем детальности изображения в блоке будем понимать:

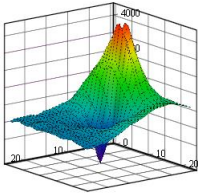
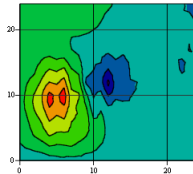
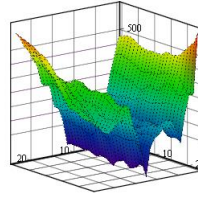
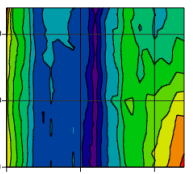
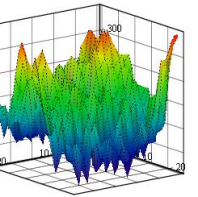
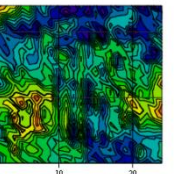
$$D(k,l) = \sum_{j=1}^M \sum_{i=1}^N \Lambda(x_k + i, y_l + j), \quad (2.38)$$

где $\Lambda(x, y)$ – яркость пикселя в препарате, полученном из исходного изображения $L(x, y)$ в результате операции подчеркивания высокочастотной составляющей (пространственного дифференцирования, многомасштабного морфологического дифференцирования и др.); x_k, y_l – координаты левого верхнего угла блока изображения; k, l – номер блока по горизонтали и вертикали; N, M – число пикселей блока по горизонтали и вертикали.

На основе проведенных экспериментальных исследований и регрессионного анализа найдена функция достоверности, позволяющая определить априорную вероятность корректного нахождения вектора движения $P_{c_v}(k, l)$ по модифицированной оценке уровня детальности в блоке $D_M(k, l)$:

$$P_{c_v}(k, l) = 1 - \exp[-\eta * D_M^4(k, l)], \quad (2.39)$$

Таблица 2.5

$D(k,l)$ <ul style="list-style-type: none"> • $D_m(k,l)$	Описание блока	Рельеф (ось X – смещение блока по горизонтали ; ось Y – смещение блока по вертикали; ось Z – значение целевой функции)	Линии равного уровня (ось X – смещение блока по горизонтали ; ось Y – смещение блока по вертикали)	Характер функции
1546 <ul style="list-style-type: none"> • 7	Блок изображения объекта «самолет» тестовая последовательность «Самолеты»			Унимодальная целевая функция с ярко выраженным минимумом
137 <ul style="list-style-type: none"> • 1	Блок фона «море» тестовая последовательность «Корабль - катер»			Унимодальная целевая функция с существенно плоским участком в области минимума («овраг»)
181 <ul style="list-style-type: none"> • 1	Блок фона «небо» тестовая последовательность «Самолеты»			Существенно мультимодальная целевая функция с плохо выраженным глобальным минимумом

где $\eta = 0,1$ – параметр модели, численное значение которого определено на основе экспериментальных данных методом нелинейного программирования.

Модифицированная оценка уровня детальности в блоке D_M :

$$D_M = \frac{D(k,l) - D_{\min}}{D_{\min}},$$

где D_{\min} – минимальный уровень детальности в блоке, обусловленный шумами

$$D_{\min} = \text{moda}\{D(k,l)\} \\ k = 1..KK, l = 1..LL \quad ,$$

где $KK * LL$ – число блоков в изображении.

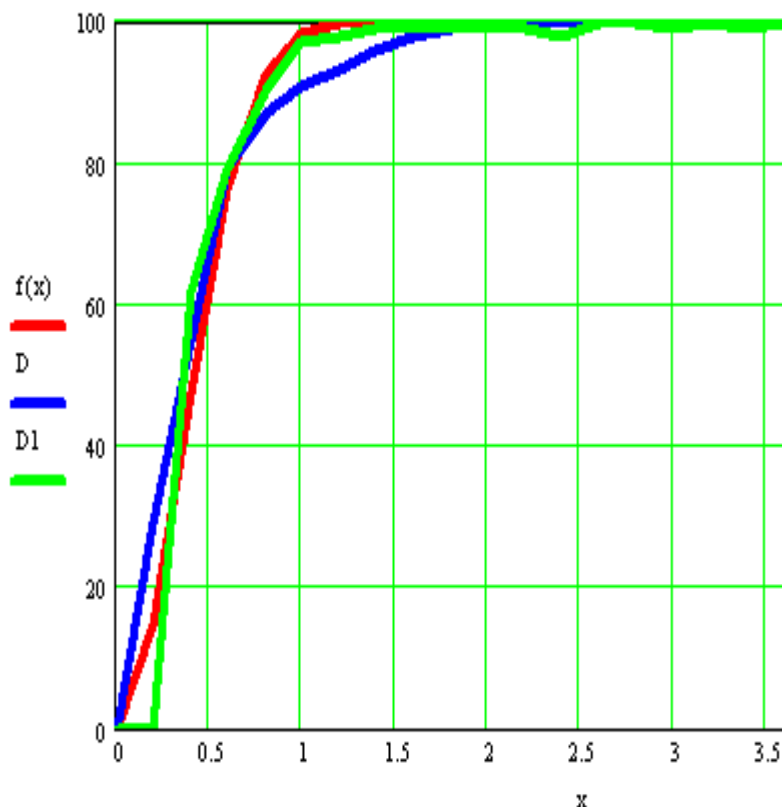


Рис. 2.38 Вид аналитической кривой функции значимости $f(x)$ и экспериментальных зависимостей вероятности существования значимого вектора от уровня модифицированной оценки абсолютной межкадровой разности, полученных для тестовых сюжетов «Большой корабль» (D) и «Катер (новый)» ($D1$).

Определение векторов движения для всех блоков кадра или области поиска является избыточным: если в блоке кадра t нет значимых изменений относительно кадра $t-1$, то с большой

вероятностью вектор движения равен нулю. Искать векторы движения целесообразно только в тех блоках, где произошли какие-либо изменения.

На основе экспериментальных исследований и регрессионного анализа найдена функция значимости, позволяющая определить априорную вероятность существования значимого (ненулевого) вектора движения $P_{z_v}(k,l)$ на основе модифицированной оценки абсолютной межкадровой разности в блоке MAD_M :

$$P_{z_v}(k,l) = 1 - \exp\left[-\beta * MAD_M^2(k,l)\right], \quad (2.40)$$

где $\beta = 4$ – параметр модели, численное значение которого найдено на основе экспериментальных данных методом нелинейного программирования (рис.2.38).

Модифицированная оценка уровня абсолютной межкадровой разности:

$$MAD_M(k,l) = \left| \frac{MAD(k,l) - MAD_{\min}}{MAD_{\min}} \right|,$$

собственно уровень абсолютной межкадровой разности

$$MAD(k,l) = \sum_{j=1}^N \sum_{i=1}^M |L(x_k + j, y_l + i, t) - L(x_k + j, y_l + i, t-1)|,$$

где $L(\)$ – яркости пикселя в текущем t и предыдущем $t-1$ кадрах; MAD_{\min} – оценка абсолютной межкадровой разности блока, обусловленная присутствием шумов:

$$MAD_{\min} = \text{moda}\{MAD(k,l)\} \quad k = 1..KK, l = 1..LL.$$

На основе найденных функций предложен метод определения векторов движения с учетом их априорных оценок достоверности и значимости.

На множестве всех блоков изображения G сформировано два нечетких множества. Первое G_{DH} – множество блоков с «высокой детальностью». В качестве функции принадлежности к этому блоку использована функция достоверности (2.39) $\mu_{G_{DH}} = P_{c_v}$. Второе нечеткое множество G_{MH} – блоки «с высоким уровнем абсолютной межкадровой разности» с функцией значимости (2.40) в качестве функции принадлежности $\mu_{G_{MH}} = P_{z_v}$. Интересующая совокупность блоков - пересечение этих двух множеств – нечеткое множество G_{DMH} .

Пересечение нечетких множеств выполняют в соответствии с t -нормой: «вероятностное пересечение» – норма задана перемножением функций принадлежности $T(G_{DH}, G_{MH}) = \mu_{G_{DH}} * \mu_{G_{MH}}$.

Вектора движения ищут для блоков, принадлежащих α -сечению нечеткого множества G_{DMH} . В результате поиска им присваивают трехкомпонентный вектор движения $(v_{xkl}, v_{ykl}, p_{kl})^T$, где значения p_{kl} – это вероятность верного определения вектора движения, определяемая функцией достоверности (2.39). Блокам, принадлежащим α -сечению только одного нечеткого множества G_{DH} (это блоки с высоким уровнем детальности, но низкой абсолютной межкадровой разностью) без выполнения процедуры поиска присваивают нулевые векторы движения.

В типичных для прикладных ТВ систем сюжетах при установленном уровне достоверности вектора движения 0,95 для дальнейшего анализа оставляют в среднем 20% блоков изображения;

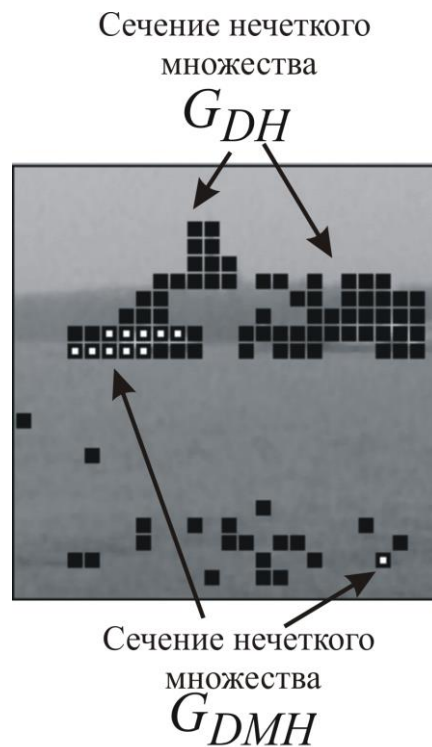


Рис. 2.39 Сечения нечетких множеств G_{DH} и G_{DMH} по уровню 0.95

собственно процедуру поиска вектора движения выполняют в среднем для 5%-10% блоков изображения (рис. 2.39). Найденная для каждого вектора вероятность верного определения (оценка достоверности) p_{kl} представляет самостоятельную ценность: ее учет на дальнейших этапах обработки позволяет реализовать взвешенную оценку признака движения.

Таким образом, предложенный метод позволяет находить трехкомпонентные вектора движения с заданным уровнем достоверности, одновременно, снижая вычислительные затраты по определению поля векторов движения в 10 – 20 раз.

Необходимость обработки изображений объектов, обладающих медленным движением (видимая скорость движения в плоскости кадра менее 1 пикселя), движением со скоростью, не кратной целому

числу пикселей, а также различие объектов с близкими скоростями движения требует субпиксельной оценки векторов движения.

Для субпиксельной оценки векторов движения предложено два метода: интерполяционный метод и метод, основанный на многомасштабной межкадровой разности.

Интерполяционный метод использует несимметрию в области точки оптимума целевой функции, минимизируемой в процессе нахождения векторов движения.

Метод, основанный на многомасштабной межкадровой разности, предполагает использование для определения векторов движения N кадров. Вектор движения находят в $(N-1)$ паре кадров: k -ая пара включает в себя кадр t и кадр $t-k$, $k=1..N$. Результатом является $(N-1)$ оценка вектора движения $\mathbf{V}_k^0, k=1..N-1$. Для обеспечения соизмеримости полученных оценок выполняют пересчет:

$$V_{kx} = \frac{V_{kx}^0}{d}; \quad V_{ky} = \frac{V_{ky}^0}{d},$$

где (V_{kx}, V_{ky}) – k -ый приведенный вектор пакета, (V_{kx}^0, V_{ky}^0) – вектор, найденный методом совмещения блоков по кадрам t и $t-k$, d – расстояние между кадрами t и $t-k$.

Предложено два варианта получения субпиксельной оценки $\mathbf{V}_p = (V_{px}, V_{py})$ на основе логической фильтрации полученного пакета векторов. Минимально отличный от других вектор пакета:

$$L_{V_{kx}} = \sum_{i=1}^N |V_{kx} - V_{ix}|; \quad V_{px} = \arg \min_{V_{kx}} \{L_{V_{kx}}\};$$

$$L_{V_{ky}} = \sum_{i=1}^N |V_{ky} - V_{iy}|; \quad V_{py} = \arg \min_{V_{ky}} \{L_{V_{ky}}\}$$

или медиана пакета:

$$V_{px} = V_{(N/2)x} \text{ при условии } V_{1x} < V_{2x} < \dots < V_{Nx};$$

$$V_{py} = V_{(N/2)y} \text{ при условии } V_{1y} < V_{2y} < \dots < V_{Ny}.$$

Экспериментальное исследование предложенных методов выполнено по двум критериям: уровень достоверности и точность. При оценке уровня достоверности аномальным считают вектор, отличный от корректного на заданную величину $\|\mathbf{V} - \mathbf{V}_{cor}\| > \Delta\xi$. Учитывая субпиксельный характер оценки вектора движения, $\Delta\xi = \pm 0,1$ пикселя. Точность – величина СКО в полученной выборке верных векторов движения.

Согласно полученным данным установлено, наиболее эффективным методом является метод на основе многомасштабной межкадровой разности с минимально отличным вектором в качестве субпиксельной оценки. Его точность 0,07 пикселя. Уровень достоверности $K_\partial = 0,8$. Полученный уровень достоверности поля векторов движения на 20% выше по отношению к полю, найденному методом полного перебора.

Предложенные методы оценки векторов движения в значительной степени устраняют противоречие между жесткими требованиями основного уравнения оптического потока (2.37) и особенностями видеоматериала, полученного в сложных условиях наблюдения. Обеспеченный уровень достоверности и точность определения поля векторов движения снимают ограничения на их применение в прикладных ТВ системах.

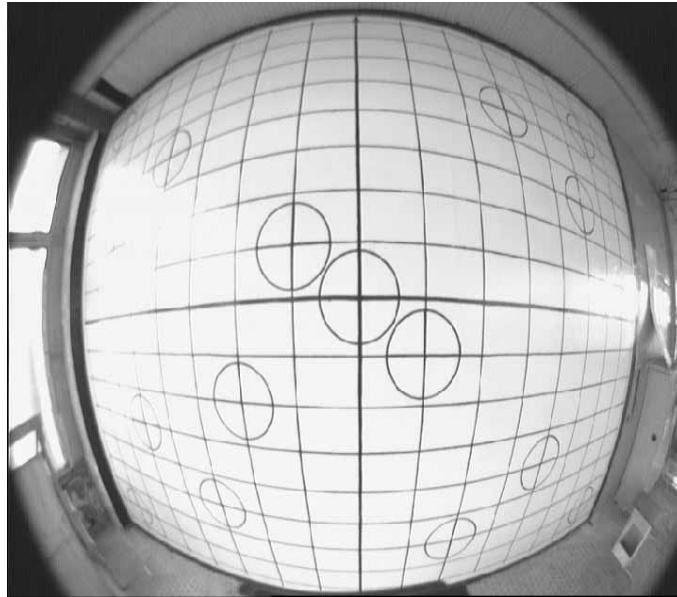


Рис.2.40 Изображение, полученное сверхширокоугольным объективом

При видеонаблюдении протяженных объектов в силу целевого назначения прикладной ТВ системы используют широкоугольные объективы. Их применение обуславливает появление пространственных искажений, приводящих к существенному нарушению геометрического подобия в сформированных изображениях (геометрические искажения). Внесенные искажения составляют: нелинейность кубическая до 120%, искажения типа «бочка/подушка» до 30% (рис.2.40). В этом случае цифровая коррекция пространственных искажений является малоэффективной. Изображения, полученные в результате цифровой коррекции, имеют значительные остаточные искажения и существенную потерю разрешения. Потери разрешения на краях раstra превышают 70%, в центре 40%.

Реализация функциональных особенностей интеллектуальной системы видеонаблюдения требует принципиально нового метода

обработки видеоданных. Метод должен носить комплексный характер - одновременно обеспечивать решение целого ряда задач: коррекцию пространственных искажений; восстановление построчного растра из чересстрочного; возможность просмотра видеоданных с произвольной скоростью; семантическое сжатие; извлечение дополнительной информации об объекте интереса.

Учитывая ключевую особенность объекта интереса – движение, предложено синтезировать его панорамное изображение. Панораму составляют из фрагментов, вырезанных в центральной части кадра. Ширина фрагмента определена скоростью движения объекта. Принцип синтеза панорамного изображения позволяет одновременно решить все перечисленные выше задачи.

Изображение в центральной части кадра имеет наилучшее качество: высокую четкость, минимальные искажения по горизонтали и вертикали. Использование фрагментов, вырезанных в центре кадра, обеспечивает формирование панорамного изображения, у которого искажения в направлении движения объекта соответствуют величине искажений вырезанного фрагмента, а искажения в перпендикулярном направлении могут быть скорректированы до низкого остаточного уровня.

Четкость изображения по площади синтезированного кадра выше, чем у исходного: в центральной части разрешающая способность короткофокусного объектива больше, чем на краях.

При синтезе панорамного изображения постоянно определяют скорость движения протяженного объекта. Эта информация позволяет корректно выполнить операцию совмещения четных и нечетных полей при использовании камер с чересстрочной разверткой. Сформированная панорама имеет разрешение по вертикали равное разрешению полного кадра без эффекта «ступенек».

Предложенный принцип формирования изображения обеспечивает высокий коэффициент сжатия без потери информации об объекте интереса. Коэффициент сжатия

$$k_{compr} = \frac{W}{w},$$

где W – ширина кадра (или высота H в случае движения объекта интереса по вертикали), w – ширина вырезаемого фрагмента по направлению движения.

При практической реализации небольшой размер фрагментов кадров необходимых для построения панорамы позволяет разместить их в буферной памяти ОЗУ и регулировать скорость анимации синтезированного изображения на экране. Это обеспечивает возможность установки удобной для анализа скорости просмотра материала в режиме реального времени.

Найденные оценки скорости при построении панорамы позволяют извлечь дополнительную информацию об объекте: зафиксировать моменты начала и завершения движения, оценить форму объекта на основе пространственного среза скоростей и др.

Главной задачей при формировании панорамного изображения является определение видимой скорости движения объекта или камеры на основе анализа видеоданных. Для получения оценки видимой скорости использованы вектора движения. В этом случае их определение имеет следующие особенности:

- наличие ярко выраженной функциональной зависимости видимой скорости движения различных частей объекта от расстояния между ними и камерой - пространственный срез скоростей;

- зависимость точности оценки скорости от местоположения блока на изображении, обусловленная значимыми пространственными искажениями;

- существенное число аномальных векторов вследствие низкой детальности исходного изображения.

Даны следующие рекомендации по местоположению и размерам блоков при определении векторов движения.

- Блоки изображения, в которых ищут вектора движения, располагают в центре раstra: при движении в горизонтальном направлении – вертикальной полосой, при движении по вертикали – горизонтальной полосой.

- Максимальный размер блока в направлении, совпадающем с направлением движения, определяют в соответствии с уровнем пространственных искажений в кадре и допустимой ошибкой при оценке максимальной скорости.

- Размер блока в направлении, перпендикулярном к направлению движения определен размером зон L пространственного среза скоростей.

Величина зон L обусловлена конкурирующими показателями: качеством построения панорамы $k_1 = f_1(L)$ и точностью оценки векторов движения $k_2 = f_2(L)$. Уменьшение размера зоны L повышает точность формируемого среза скоростей и улучшает качество панорамы. Размер блока $b = L * L$ определяет интервал усреднения при определении векторов движения, его уменьшение снижает точность найденной оценки. Дополнительно, для получения равноточных оценок вектора движения размер всех зон должен быть одинаков.

Показатель k_2 может быть переведен в ограничения: размер зоны L должен обеспечивать потенциальную точность не ниже заданного уровня $\sigma < \sigma_p$.

$$\begin{cases} f_1(L) \rightarrow \min_L \\ f_2(L) \leq \sigma_p \\ L_1 = L_2 = \dots = L_K \end{cases},$$

где K – число зон в пространственном срезе скоростей.

Нахождение L является задачей поиска минимума функции с ограничениями в виде равенств.

Для снижения числа аномальных векторов предложено проводить временную фильтрацию на основе усреднения по мажоритарному принципу. Согласно экспериментальным данным, в результате временной фильтрации число аномальных векторов падает в 2 раза, обеспечиваемый уровень достоверности K_δ составляет 0,8.

Для повышения уровня достоверности и дополнительного сглаживания поступающей информации введена регрессионная предсказывающая полиномиальная модель

$$\hat{U}_j = \sum_{k=0}^K a_k \varphi_k(j), \quad (2.41)$$

где j – номер зоны пространственного среза скоростей $j=0..N$; \hat{U}_j – оценка вектора движения U для j -го блока; a_k – параметры (коэффициенты) регрессионной модели; $\varphi_k(x)$ – базисные функции $k = 0..K$, ортонормированные на системе из N точек с весами

$$w_j = \frac{1}{\sqrt{1-x_j^2}}, \quad \text{где } x_j = \frac{\left(j - \frac{N}{2}\right)}{\frac{N}{2}};$$

Базисные функции получены из последовательности $1, x, x^2, \dots$ методом ортогонализации Грамма-Шмидта, согласно рекуррентным выражениям, которые при симметричной относительно начала координат области изменений аргумента имеют вид:

$$\lambda_k \varphi_{k+1}(x) = x\varphi_k(x) - b_k \varphi_{k-1}(x) ; b_k = \sum_{i=1}^N w_i x_i \varphi_k(x_i) \varphi_{k-1}(x_i).$$

Для расчета полагают $\varphi_{-1}(x) \equiv 0, \varphi_0(x) = \left(\sum_{i=1}^N w_i \right)^{-1/2}$, а

коэффициенты λ_k определяют из условий нормировки.

Коэффициенты a_k определяют из условия минимизации взвешенной квадратической ошибки $\sigma^2 = \sum w_j (\hat{U}(x_j) - U(x_j))^2 = \min$ согласно выражению $\mathbf{a} = \mathbf{\Phi}' \mathbf{W} \mathbf{U}$, где $\mathbf{\Phi}$ – матрица, составленная из отсчетов базисных функций $\varphi_k(x)$; \mathbf{W} – диагональная матрица весовых коэффициентов.

На основе регрессионной модели построен предсказывающий фильтр. Для формирования панорамы используют оценки векторов движения \hat{U} , рассчитанные по модели (2.41).

Алгоритм определения оценок векторов движения для каждого блока на основе предсказывающей регрессионной модели включает в себя три основных шага:

Шаг 1: Масштабирование установленных априорно параметров модели \mathbf{a} по найденной средней скорости движения протяженного объекта интереса.

Шаг 2: Идентификация методом наименьших квадратов параметров модели \mathbf{a} . Определение параметров выполняют по найденным векторам движения \mathbf{U} . При этом вычисляют разницу

между измеренным вектором движения и его оценкой. Чем меньшей погрешностью характеризуются измерения, тем больший вес

$$Weight_j = \exp[koff * (U_j - \hat{U}_j)^2]$$

получают найденные вектора при идентификации параметров модели.

Шаг 3: Расчет по модели значений векторов скорости для каждого из блоков.

Масштабирование модели выполняют один раз после обнаружения начала движения объекта. Обновления параметров модели и расчет по модели оценок векторов движения проводят в течение всего времени формирования панорамы, через заданное число кадров, определяемое средней скоростью движения объекта.

На основе введенного пространственного среза скоростей, временной фильтрации и регрессионной предсказывающей модели разработана процедура определения скорости протяженного движущегося объекта. Согласно экспериментальным данным средняя ошибка при оценке скорости 2,7 % , уровень достоверности $K_D = 0,999985$.

Полученные показатели обеспечивают все заявленные выше характеристики формируемого панорамного изображения и позволяют реализовать основные функциональные особенности интеллектуальной системы видеонаблюдения.

Функциональные особенности интеллектуальной ТВ системы сегментации и сопровождения определяют совокупность требований к применяемым в них методам обработки видеоданных: необходимо обеспечить минимальную ошибку сегментации, сегментацию изображений объектов, находящихся в непосредственной близости

друг к другу и на сложном фоне; на этапе сопровождения разрешить ситуации окклюзии, слияния и разделения объектов интереса.

Для реализации перечисленных задач предложен метод сегментации и сопровождения неточечных объектов с жестким движением (неподвижные объекты рассматриваются как частный случай: движение с нулевой скоростью), основанный на совокупности признаков с использованием методов нечеткой логики.

Использование математического аппарата нечеткой логики обусловлено отсутствием априорной информации об объектах интереса: количество объектов, данные об их форме и размерах неизвестны.

Как наиболее информативные выделены признаки: детальность, движение и форма.

Признаки детальности, движения и пространственной связности используют для первоначальной сегментации объектов интереса, на этапе сопровождения дополнительно применяют признак формы.

На основании признака детальности, под которым понимают уровень высокочастотной энергии в фрагменте изображения заданного размера (2.38) выполняют предварительную классификацию.

Процедура предварительной классификации включает: предобработку изображения с целью подчеркивания высокочастотной составляющей; оценку уровня детальности блоков изображения; пороговую обработку.

Для предобработки изображения применяют многомасштабный морфологический градиент

$$MG(L) = \frac{1}{3} \sum_{i=1}^3 [((L \oplus S_i) - (L \ominus S_i)) \ominus S_{i-1}], \quad (2.42)$$

где \oplus и \ominus – морфологические операции наращивания и эрозии; S_i – квадратная группа структурных элементов. Размер S_i равен $(2i + 1)(2i + 1)$ пикселей для $0 \leq i \leq 3$. В соответствии с выражением (2.41) значения градиентов рассчитывают трижды с использованием структурных элементов различной размерности, а затем результаты складывают.

Данные экспериментов показывают, что применение многомасштабного морфологического градиента обеспечивает максимальную эффективность по сравнению с такими методами предобработки, как пространственное дифференцирование, оператор Превитта и вейвлет фильтр Добеши. Критерии оценки эффективности: K_1 – степень выделения объектов интереса; K_2 – степень выделения фона, K_3 – результативность анализа и K_4 – инвариантность к свойствам изображения.

Распределение оценок детальности блоков изображения в кадре имеет зависимость Релея. Бинарный характер проводимой классификации и известный закон распределения оценок детальности позволяет реализовать пороговое ограничение.

В результате предварительной классификации выделяют множество блоков G_d с высоким уровнем детальности для дальнейшего анализа. Для каждого блока этого множества определяют трехкомпонентный вектор движения $(v_{xk}, v_{yk}, p_k)^T$, где v_{xk}, v_{yk} – составляющие k -го вектора движения по направлениям x и y , p_k – оценка его достоверности.

Вектор признаков $\mathbf{q}_k = (x_k, y_k, v_{kx}, v_{ky}, p_k)$, сопоставленный каждому блоку множества G_d , позволяет выделить объекты

интереса (принять решение к какому объекту принадлежит k -ый блок с вектором признаков \mathbf{q}_k).

Предложено сегментировать объекты, объединяя блоки в группы на основе схожести признаков для блоков одной группы и отличий между группами. Этот подход в аппарате нечеткой логики называется кластеризацией. В терминах нечеткой логики блоки – это элементы, подлежащие кластеризации (объединению), объекты интереса – кластеры.

Задачу сегментации объектов интереса при отсутствии априорной информации следует рассматривать как нечеткую кластеризацию при неизвестном числе кластеров.

Исходной информацией для кластеризации является матрица наблюдений

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1n} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ q_{m1} & q_{m2} & \dots & q_{mn} \end{bmatrix},$$

каждая строка которой представляет собой значения n признаков одного из M элементов кластеризации.

Нечеткие кластеры описывают матрицей возможностного нечеткого разбиения:

$$\mathbf{F} = [\mu_{ki}], \mu_{ki} \in [0,1], k = 1..M, i = 1..c$$

в которой k -ая строка содержит степени принадлежности элемента k к кластерам O_1, O_2, \dots, O_c . При нечетком разбиении степень принадлежности объекта к кластеру μ_{ki} принимает значения из интервала $[0, 1]$.

Матрица возможностного нечеткого разбиения должна обладать следующими свойствами

$$\exists i, \mu_{ki} > 0, \forall k k = 1..M ;$$

$$0 < \sum_{k=1}^M \mu_{ki} < M, i = 1..c .$$

Для определения центров потенциальных кластеров применен алгоритм горной кластеризации.

Потенциал центров кластеров определен выражением:

$$P(\mathbf{Z}_h) = \sum_{k=1}^M \exp(-\alpha \cdot D(\mathbf{Z}_h, \mathbf{q}_k)),$$

где $\mathbf{Z}_h = (z_{1h}, z_{2h}, \dots, z_{nh})$ – вектор признаков потенциального центра h -го кластера, $h = 1..c$; α – положительная константа; $D(\mathbf{Z}_h, \mathbf{q}_k)$ – расстояние между вектором признаков потенциального центра кластера \mathbf{Z}_h и вектором признаков элемента кластеризации \mathbf{q}_k .

Расстояние между вектором признаков потенциального центра и элемента кластеризации:

$$D(\mathbf{Z}_h, \mathbf{q}_k) = \|\mathbf{d}_k^{rv}\|; \quad \mathbf{d}_k^{rv} = (d_r, d_v);$$

$$d_r = \sqrt{(x_k - x_c)^2 + (y_k - y_c)^2}; \quad d_v = p_k p_c \sqrt{(v_{xk} - v_{xc})^2 + (v_{yk} - v_{yc})^2},$$

где x_c, y_c – координаты левого верхнего угла, v_{cx}, v_{cy} и p_c – найденный вектор движения и его достоверность для блока, являющегося потенциальным центром кластера; x_k, y_k – координаты левого верхнего угла, v_{kx}, v_{ky} и p_k – найденный вектор движения и его достоверность для блока, являющегося элементом кластеризации.

В качестве центров кластеров выбирают точки с максимальным потенциалом («горные вершины»). Центром первого кластера

назначают точку с наибольшим потенциалом. При выборе следующего центра кластера исключают влияние найденного кластера. Для этого значения потенциала оставшихся возможных центров кластеров пересчитывают:

$$P_2(\mathbf{Z}_h) = P_1(\mathbf{Z}_h) - P_1(\mathbf{V}_1) \cdot \exp(-\beta \cdot D(\mathbf{Z}_h, \mathbf{V}_1)),$$

где $P_1(\)$ – потенциал на 1-й итерации; $P_2(\)$ – потенциал на 2-й итерации; \mathbf{V}_1 – вектор признаков первого найденного кластера:

$$\arg \max_{Z_1, Z_2, \dots, Z_x} (P_1(\mathbf{Z}_1), P_1(\mathbf{Z}_2), \dots, P_1(\mathbf{Z}_c)); \beta - \text{положительная константа.}$$

Центр второго кластера определяется по максимальному значению обновленного потенциала:

$$\arg \max_{Z_1, Z_2, \dots, Z_x} (P_2(\mathbf{Z}_1), P_2(\mathbf{Z}_2), \dots, P_2(\mathbf{Z}_c))$$

После определения центров кластеров формируют матрицу нечеткого разбиения. Значение элементов матрицы μ_{ki} определены степенью принадлежности элемента (блока) кластеру (объекту), которое задано расстоянием между вектором признаков блока и вектором признаков центра кластера.

При сегментации объектов учитывают признак пространственной связности – блоки должны образовывать связную группу и иметь схожие вектора движения. Первым признаком r_{kc1} принадлежности блока k к кластеру Ω_c является минимальное расстояние $d_{\min} = \min(d_{kl})$, где $l = 1..m$, $k = 1..m$; $l \neq k$ до одного из ранее включенных в кластер блоков (на первом шаге – к центру кластера).

Вторым признаком r_{kc2} является взвешенная норма разностного вектора $p_k \|\mathbf{v}_k - \mathbf{v}_c\|$, где \mathbf{v}_c – наиболее вероятный вектор движения

для данного кластера, а p_k – оценка достоверности вектора движения \mathbf{v}_k .

Для k -го блока формируют обобщенный признак в виде нормы вектора \mathbf{r}_{kc} , составленного из взвешенных значений частных признаков

$$d_{kc} = \|\mathbf{r}_{kc}\| = \mathbf{r}_{kc}^T \mathbf{W} \mathbf{r}_{kc},$$

где матрица \mathbf{W} переменных весовых коэффициентов учитывает динамику свойств объектов интереса и окружающей обстановки.

Для определения степени принадлежности блока кластеру введена экспоненциальная функция на основании обобщенного признака d :

$$\mu = \exp(-d^2 / \beta^2),$$

где β – масштабный коэффициент.

Сопровождение объектов интереса реализовано на основе обобщенной модели формы Гаусса и с использованием теории нечетких множеств. Анализируют соответствие вектора признаков фрагмента изображения (блока) векторам признаков объектов, сегментированных на предыдущем шаге.

Представление в виде эллипсов рассеяния Гаусса является наиболее общим и несет информацию о центре тяжести объекта μ_x, μ_y (центр эллипса), его линейных размерах L_1, L_2 (длины полуосей эллипса), коэффициенте элонгации $elong = \frac{L_1}{L_2}$, угле наклона φ , мере несимметрии объекта E .

Построение модели формы и движения для каждого сегментированного объекта позволяет использовать для решения

задачи сопровождения математический аппарат нечетких множеств. На множестве всех блоков кадра сформированы нечеткие множества объектов O_g $g = 1..G$, где G – число объектов, сегментированных в предыдущем кадре t . Функция принадлежности нечеткого множества O_g получена на основе t -нормы вероятностного пересечения нечетких множеств G_{fog} и G_{vog}

$$\eta_{Og}(b_k \in O_g) = \eta_{vog} \eta_{fog},$$

где η_{fog} – функция принадлежности нечеткого множества G_{fog} блоков, соответствующих модели формы объекта g ; η_{vog} – функция принадлежности нечеткого множества G_{vog} блоков, соответствующих модели движения объекта g

$$\eta_{fog}(b) = \frac{1}{2\pi\sigma_{xg}\sigma_{yg}} \exp \left\{ -\frac{1}{2(1-\rho)} \left[\frac{(x_k - \hat{\mu}_{xg})^2}{\sigma_{xg}^2} - 2\rho \frac{(x_k - \hat{\mu}_{xg})(y_k - \hat{\mu}_{yg})}{\sigma_{xg}\sigma_{yg}} + \frac{(y_k - \hat{\mu}_{yg})^2}{\sigma_{yg}^2} \right] \right\},$$

$$\sigma_{xg} = \sqrt{D_{xg}}; \sigma_{yg} = \sqrt{D_{yg}}; \rho = \sqrt{K_{xyg}},$$

где b – блок с координатами (x_k, y_k) ; D_{xg}, D_{yg}, K_{xyg} – значения дисперсии и ковариации, полученные в кадре t при определении параметров модели формы объекта в виде эллипсов рассеивания Гаусса; $(\hat{\mu}_{xg}, \hat{\mu}_{yg})$ – оценка координат центра тяжести объекта в кадре $t+1$ на основе координат центра тяжести в кадре t .

$$\eta_{vog}(b) = \exp \left(- \left[w_g \|\mathbf{v}_k - \mathbf{v}_{og}\| \right] \right),$$

где \mathbf{v}_{og} – вектор движения объекта, найденный согласно модели движения объекта O_g , w_g – весовой коэффициент, учитывающий

выраженность признака движения и особенности окружающей обстановки.

При сопровождении нескольких близко расположенных объектов функция принадлежности имеет вид

$$\eta(O_g) = \eta^2(O_g) / \sum_{g=1}^G \eta(O_g).$$

Сопровождение объектов выполняют выявлением принадлежности блоков кадра $t + 1$, сформированным нечетким множествам O_g кадра t . Для однозначного определения объектов введены α -сечения каждого из множеств O_g .

В случае длительного исчезновения объекта необходимо отождествить потерянный объект с одним из вновь захваченных объектов. Операцию отождествления выполняют по критерию минимума расстояния между векторами взвешенных параметров потерянного объекта и одного из вновь захваченных объектов

$$d_{gb} = \langle (\mathbf{f}_g - \mathbf{f}_b), \mathbf{W}\mathbf{W}(\mathbf{f}_g - \mathbf{f}_b) \rangle,$$

где \mathbf{f}_g – вектор параметров потерянного объекта $g = 1 \dots G$, G – число объектов, сегментированных в кадре t ; $\mathbf{f}_g = (\boldsymbol{\mu}_g, \mathbf{L}_g, \varphi_g, elong_g, \mathbf{v}_g, \mathbf{a}_g)$, $\boldsymbol{\mu}_g$ – центр тяжести, \mathbf{L}_g – линейные размеры, φ_g – угол поворота, $elong_g$ – коэффициент элонгации, \mathbf{v}_g и \mathbf{a}_g – вектора скорости и ускорения объекта интереса; \mathbf{f}_b – вектор параметров найденного объекта; $b = 1 \dots K$, K – число вновь захваченных объектов в кадре $t+1$; $\mathbf{W}\mathbf{W}$ – весовая матрица, в общем случае отличная от диагональной.

Как показали экспериментальные исследования, предложенный метод позволяет реализовать автоматический захват до 50 объектов интереса на сложном фоне и обеспечивает устойчивое сопровождение

объектов интереса при существенной динамике их свойств и окружающей обстановки, а также при взаимодействии изображений объектов с фоном и друг с другом.



Рис. 2.41. Вид экрана монитора

Многоцелевой телевизионно-компьютерный комплекс видеомониторинга железнодорожных составов предназначен для обеспечения осмотра и выявления негабаритных грузов движущегося товарного состава с последующим сбором, обработкой, хранением и документированием информации о коммерческом состоянии вагонов и грузов, а также передачей ее в автоматизированную систему управления перевозками.

В состав системы входят сенсоры, электронная система ввода видеоизображений в компьютер, персональный компьютер, программное обеспечение. Сенсорами системы являются три видеокамеры, дающие изображения левого и правого по ходу поезда бортов вагонов, а также вид сверху; габаритные электронные ворота для выявления негабаритного груза.

Комплекс позволяет:

- На одном экране представить панорамные изображения всех трех изображений проходящих вагонов и дать дополнительную

информацию о текущем времени, скорости движения состава и порядковом номере вагона (рис. 2.41).

- Предоставить оператору возможность самому устанавливать комфортную скорость просмотра состава с помощью движка, останавливать движение (стоп-кадр) и возвращаться назад.
- Записать синтезированные изображения в архив с высоким коэффициентом сжатия без потери информации об объекте интереса.

Экспериментальное исследование и результаты, полученные в процессе апробации комплекса, показывают:

1. Синтезированное панорамное изображение обеспечивает эффективную коррекцию пространственных искажений (рис. 5):

нелинейные искажения третьего порядка:

исходный уровень 120%; уровень после коррекции менее 5%;

искажения типа «бочка/подушка»:

исходный уровень 20%; уровень после коррекции менее 1%.

2. Полученный размер панорамы одного вагона позволяет разместить ее целиком на экране монитора.

3. При движении протяженного объекта со скоростью от 20 до 60 км/ час и стандартной ширине кадра 768 пикселей по горизонтали достигаемый коэффициент сжатия от 80 до 25 раз, соответственно.

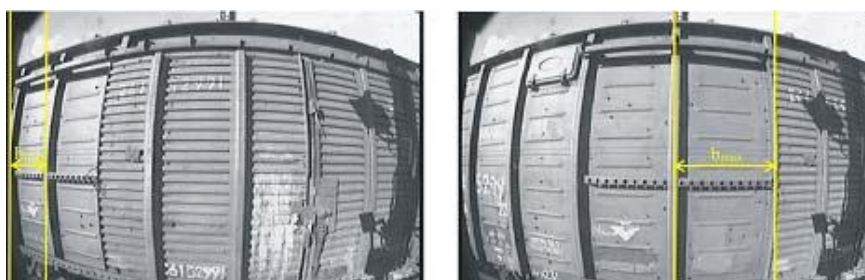
4. Средняя ошибка нарушения геометрического подобия (соответствия с определенным масштабным коэффициентом размеров объекта интереса на созданной панораме размерам реального объекта) составляет 2,7%. Достигнутая точность обеспечивает восприятие оператором синтезированного

изображения, как изображения без искажений. Искажения менее 5% не воспринимаются зрительным анализатором человека.

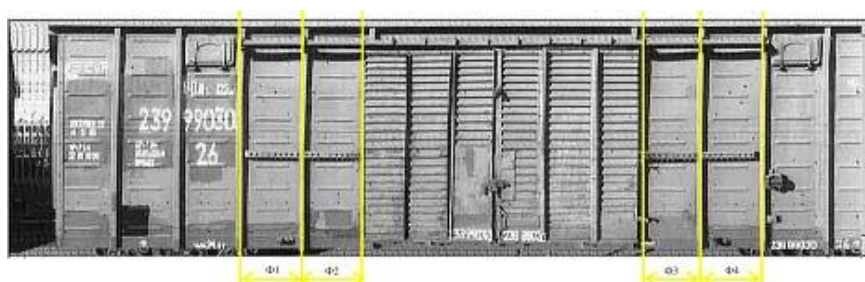
5. Информация о скорости движения протяженного объекта при построении панорамного изображения, одновременно с синтезом изображения объекта интереса, позволяет зафиксировать момент начала и окончания движения железнодорожного состава, определить направление и скорость движения, реализовать счет вагонов и выполнить идентификацию типа вагона.

Полученные результаты показывают, что синтезированное панорамное изображение позволяет реализовать основные функциональные особенности интеллектуальной системы видеонаблюдения протяженных объектов.

В многофункциональном оптико–электронном комплексе решены задачи автоматизированного захвата и автоматического



a)



b)

Рис. 2.42a) Исходное изображение.

Коэффициент нелинейных искажений 109% ($b_{\min} = 56$;
 $b_{\max} = 191$).

сопровождения неточечных объектов интереса. Размеры объектов от 4*4 до 100*100 пикселей. Скорость движения от 0 до 10 пикселей за кадр. Движение объектов произвольное, возможно исчезновение длительностью до 10 сек. Число главных объектов интереса от одного до четырех. Число второстепенных объектов неограниченно. Эффективное отношение сигнал/шум (отношение превышение сигнала цели над фоном к СКО шума) не менее 6 дБ. Автоматизированный захват выполняет оператор.

Система решает следующие задачи:

1. Один из главных объектов (по выбору оператора) всегда удерживают в центре растра с ошибкой, не превышающей 1 пиксель по каждой координате.

2. Остальные объекты сопровождают стробом. Реализуют непрерывную (с частотой кадров) подстройку стробов под изменяющиеся размеры и ракурсы для всех четырех объектов интереса.

3. Выполняют разрешение ситуации окклюзии и временного (до 10 сек) исчезновения объектов интереса.

Экспериментальное исследование показывает:

1. Предложенный метод обеспечивает систематическую ошибку сегментации 16% от площади объекта интереса. Указанное значение в 3 раза ниже средней ошибки при сегментации объекта прямоугольным стробом. Случайная ошибка зависит от размеров объекта интереса: для объектов размером до 5 блоков она составляет 15-20%, для объектов большего размера 3-5%.

2. При сопровождении объектов статичной камерой ошибка при определении центра тяжести не превышает 1 пиксель.

3. Разрешение ситуации окклюзии во всех экспериментах было выполнено корректно и не привело к ложному изменению размеров стробов объектов интереса.

В состав системы входят видеокамера; электронная система ввода видеоизображений в компьютер; поворотный стол; персональный компьютер; программное обеспечение.

В автоматизированном режиме выполняют захват объекта интереса: оператор формирует строб вокруг объекта, подлежащего дальнейшему сопровождению. В автоматическом режиме система реализует удерживание в центре кадра объекта, выделенного оператором. Скорость перемещения объекта интереса в экранной плоскости до 5 пикселей за кадр. Возможно исчезновение объекта интереса длительностью до 3 секунд.

Экспериментальные исследования показывают:

1. Средняя ошибка слежения, определяемая предложенным методом (без влияния инерционности поворотного стола), составляет 1,5 пикселя.

2. Момент начала движения объекта интереса был зафиксирован на всех тестовых сюжетах.

Полученные показатели и качественные характеристики показывают, что предложенные в работе методы представления и обработки видеоданных, позволяют реализовать основные функциональные особенности интеллектуальной системы видеонаблюдения за протяженными объектами и систем сегментации и сопровождения неточечных объектов интереса с жестким движением.

Глава 3. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

3.1. Анализ особенности генетических алгоритмов

Основные принципы работы генетических алгоритмов (ГА) заключены в следующей схеме[5-7]:

1. Генерируем начальную популяцию из n хромосом.
2. Вычисляем для каждой хромосомы ее пригодность.
3. Выбираем пару хромосом-родителей с помощью одного из способов отбора.
4. Проводим кроссинговер двух родителей с вероятностью p_c , производя двух потомков.
5. Проводим мутацию потомков с вероятностью p_m .
6. Повторяем шаги 3–5, пока не будет сгенерировано новое поколение популяции, содержащее n хромосом.
7. Повторяем шаги 2–6, пока не будет достигнут критерий окончания процесса.

Критерием окончания процесса может служить заданное количество поколений или схождение (convergence) популяции.

Схождением называется такое состояние популяции, когда все строки популяции почти одинаковы и находятся в области некоторого экстремума. В такой ситуации кроссинговер практически никак не изменяет популяции, так как создаваемые при нем потомки представляют собой копии родителей с переменными участками хромосом. Вышедшие из этой области за счет мутации особи склонны вымирать, так как чаще имеют меньшую приспособленность, особенно если данный экстремум является глобальным максимумом. Таким образом, схождение популяции обычно означает, что найдено лучшее или близкое к нему решение.

Основными операторами ГА являются кроссинговер, мутация, выбор родителей и селекция (отбор хромосом в новую популяцию). Вид оператора играет важную роль в реализации и эффективности ГА. Существуют основные формы операторов, чистое использование или модернизация которых ведет к получению ГА, пригодного для решения конкретной задачи. Рассмотрим некоторые из них.

3.1.1. Операторы выбора родителей

Существует несколько подходов к выбору родительской пары. Наиболее распространенными операторами выбора родителей являются следующие.

Панмиксия — самый простой оператор отбора. В соответствии с ним каждому члену популяции сопоставляется случайное целое число на отрезке $[1;n]$, где n — количество особей в популяции. Будем рассматривать эти числа как номера особей, которые примут участие в скрещивании. При таком выборе какие-то из членов популяции не будут участвовать в процессе размножения, так как образуют пару сами с собой. Какие-то члены популяции примут участие в процессе воспроизводства неоднократно с различными особями популяции. Несмотря на простоту, такой подход универсален для решения различных классов задач. Однако он достаточно критичен к численности популяции, поскольку эффективность алгоритма, реализующего такой подход, снижается с ростом численности популяции.

Инбридинг представляет собой такой метод, когда первый родитель выбирается случайным образом, а вторым родителем является член популяции ближайший к первому. Здесь «ближайший» может пониматься, например, в смысле минимального расстояния Хемминга (для бинарных строк) или евклидова расстояния между

двумя вещественными векторами. Расстояние Хемминга равно числу различающихся локусов (разрядов) в бинарной строке.

При аутбридинге также используют понятие схожести особей. Однако теперь брачные пары формируют из максимально далеких особей. Последние два способа по разному влияют на поведение генетического алгоритма. Так, инбридинг можно охарактеризовать свойством концентрации поиска в локальных узлах, что фактически приводит к разбиению популяции на отдельные локальные группы вокруг подозрительных на экстремум участков ландшафта. Аутбридинг же направлен на предупреждение сходимости алгоритма к уже найденным решениям, заставляя алгоритм просматривать новые, неисследованные области. Инбридинг и аутбридинг бывает генотипным (когда в качестве расстояния берется разность значений целевой функции для соответствующих особей) и фенотипным (в качестве расстояния берется расстояние Хемминга).

Селекция состоит 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 более быструю сходимость алгоритма. Однако из-за быстрой сходимости селективный выбор родительской пары не подходит тогда, когда ставится задача определения нескольких экстремумов, поскольку для таких задач алгоритм, как правило, быстро сходится к одному из решений.

Кроме того, для некоторых многомерных задач со сложным ландшафтом целевой функции быстрая сходимость может превратиться в преждевременную сходимость к квазиоптимальному решению. Этот недостаток может быть отчасти компенсирован использованием подходящего механизма отбора, который бы «тормозил» слишком быструю сходимость алгоритма.

Пороговая величина в селекции может быть вычислена разными способами. Поэтому в литературе по ГА выделяют различные вариации селекции. Наиболее известные из них — это турнирный и рулеточный (пропорциональный) отборы.

При турнирном отборе (tournamentselection) 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. Вообще же t называют численностью турнира. Преимуществом данного способа является то, что он не требует дополнительных вычислений.

В методе рулетки (roulette-wheelselection) особи отбираются с помощью N «запусков» рулетки, где N — размер популяции. Колесо рулетки содержит по одному сектору для каждого члена популяции. Размер i -госектора пропорционален вероятности попадания в новую популяцию $P(i)$.

Ожидаемое число копий i -ой хромосомы после оператора рулетки определяются по формуле $N_i = P(i)N$.

При таком отборе члены популяции с более высокой приспособленностью с большей вероятностью будут чаще выбираться, чем особи с низкой приспособленностью).

Другие способы отбора можно получить на основе модификации выше приведенных. Так, например, в рулеточном отборе можно изменить формулу для вероятности попадания особи в новую популяцию.

3.1.2. Рекомбинация (воспроизведение)

Оператор рекомбинации применяют сразу же после оператора отбора родителей для получения новых особей-потомков. Смысл рекомбинации заключается в том, что созданные потомки должны наследовать генную информацию от обоих родителей. Различают дискретную рекомбинацию и кроссинговер.

3.1.2.1. Дискретная рекомбинация

Дискретная рекомбинация (Discrete recombination) в основном применяется к хромосомам с вещественными генами. Основными способами дискретной рекомбинации являются собственно дискретная рекомбинация, промежуточная, линейная и расширенно линейная рекомбинации.

Промежуточная рекомбинация (Intermediate recombination) применима только к вещественным переменным, но не к бинарным. В данном методе предварительно определяется числовой интервал значений генов потомков, который должен содержать значения генов родителей. Потомки создаются по следующему правилу:

Потомок = Родитель 1 + $\alpha \cdot (\text{Родитель 2} - \text{Родитель 1})$, где множитель α — случайное число на отрезке $[-d, 1 + d]$. Как отмечают сторонники этого метода, наиболее оптимальное воспроизведение получается при $d = 0,25$.

При промежуточной рекомбинации возникают значения генов, отличные от значения генов особей-родителей. Это приводит к возникновению новых особей, пригодность которых может быть лучше, чем пригодность родителей. В литературе такой оператор рекомбинации иногда называется дифференциальным скрещиванием.

Линейная рекомбинация (Line recombination) отличается от промежуточной тем, что множитель α выбирается для каждого потомка один раз.

3.1.2.2. Кроссинговер (бинарная рекомбинация)

Рекомбинацию бинарных строк принято называть кроссинговером (кроссовером) или скрещиванием.

Одноточечный кроссинговер (Single-point crossover) моделируется следующим образом. Пусть имеются две родительские особи с хромосомами $X = \{x_i, i \in [0, L]\}$ и $Y = \{y_i, i \in [0, L]\}$. Случайным образом определяется точка внутри хромосомы (точка разрыва), в которой обе хромосомы делятся на две части и обмениваются ими. Такой тип кроссинговера называется одноточечным, так как при нем родительские хромосомы разделяются только в одной случайной точке.

Также применяется двух- и N -точечный кроссинговер.

В двухточечном кроссинговере (и многоточечном кроссинговере вообще) хромосомы рассматриваются как циклы, которые формируются соединением концов линейной хромосомы вместе. Для замены сегмента одного цикла сегментом другого цикла требуется выбор двух точек разреза. В этом представлении, одноточечный кроссинговер может быть рассмотрен как кроссинговер с двумя точками, но с одной точкой разреза, зафиксированной в начале строки. Следовательно, двухточечный кроссинговер решает ту же самую задачу, что и одноточечный, но более полно. Хромосома, рассматриваемая как цикл, может содержать большее количество стандартных блоков, так как они могут совершить «циклический возврат» в конце строки. В настоящий момент многие исследователи соглашаются, что двухточечный кроссинговер вообще лучше, чем одноточечный.

Для многоточечного кроссинговера (Multi-point crossover), выбираем m точек разреза. Точки разреза выбираются случайно без повторений и сортируются в порядке возрастания. При кроссинговере

происходит обмен участками хромосом, ограниченными точками разреза и таким образом получают двух потомков. Участок особи с первым геном до первой точки разреза в обмене не участвует.

Применение многоточечного кроссинговера требует введения нескольких переменных (точек разреза), и для воспроизведения выбираются особи с наибольшей приспособленностью.

Одноточечный и многоточечный кроссинговер определяют точки разреза, которые разделяют особи на части. Однородный кроссинговер (Uniform crossover) создает маску (схему) особи, в каждом локусе которой находится потенциальная точка кроссинговера. Маска кроссинговера имеет ту же длину, что и скрещивающиеся особи.

Таким образом, гены маски представляют собой случайные двоичные числа (0 или 1). Согласно этим значениям, геном потомка становится первая (если ген маски = 0) или вторая (если ген маски = 1) особь-родитель.

Однородный кроссинговер очень похож на многоточечный, но строка случайных битовых значений в нем длиннее. Это гарантирует, что в потомках будут чередоваться короткие строки особей-родителей.

Алгоритм однородного кроссинговера для двоичных строк полностью идентичен дискретному воспроизведению для вещественных хромосом. Такой вид кроссинговера еще называют унифицированным.

Триадный кроссинговер (Triadic crossover). Данная разновидность кроссинговера отличается от однородного тем, что после отбора пары родителей из остальных членов популяции случайным образом выбирается особь, которая в дальнейшем используется в качестве маски. Далее 10 % генов маски мутируют.

Затем гены первого родителя сравниваются с генами маски: если гены одинаковы, то они передаются первому потомку, в противном случае на соответствующие позиции хромосомы потомка переходят гены второго родителя. Генотип второго потомка отличается от генотипа первого тем, что на тех позициях, где у первого потомка стоят гены первого родителя, у второго потомка стоят гены второго родителя и наоборот.

Перетасовочный кроссинговер (Shuffler crossover). В данном алгоритме особи, отобранные для кроссинговера, случайным образом обмениваются генами. Затем выбирают точку для одноточечного кроссинговера и проводят обмен частями хромосом. После скрещивания созданные потомки вновь тасуются. Таким образом, при каждом кроссинговере создаются не только новые потомки, но и модифицируются родители (старые родители удаляются), что позволяет сократить число операций по сравнению с однородным кроссинговером.

Кроссинговер с уменьшением замены (Crossover with reduced surrogate). Оператор уменьшения замены ограничивает кроссинговер, чтобы всегда, когда это возможно, создавать новые особи. Это осуществляется за счет ограничения на выбор точки разреза: точки разреза должны появляться только там, где гены различаются.

Как было показано выше, кроссинговер генерирует новое решение (в виде особи-потомка) на основе двух имеющихся, комбинируя их части. Поэтому число различных решений, которые могут быть получены кроссинговером при использовании одной и той же пары готовых решений, ограничено. Соответственно, пространство, которое ГА может покрыть, используя лишь кроссинговер, жестко зависит от генофонда популяции. Чем разнообразнее генотипы решений популяции, тем больше

пространство покрытия. При обнаружении локального оптимума соответствующий ему генотип будет стремиться занять все позиции в популяции, и алгоритм может сойтись к ложному оптимуму. Поэтому в генетическом алгоритме важную роль играют мутации. Существует несколько способов мутирования генов. Вопрос о выборе подходящего оператора мутации решается в рамках поставленной задачи.

3.1.3. Мутация

После процесса воспроизводства происходят мутации (mutation). Данный оператор необходим для «выбивания» популяции из локального экстремума и препятствует преждевременной сходимости. Это достигается за счет того, что изменяется случайно выбранный ген в хромосоме.

Так же как и кроссинговер, мутации могут проводиться не только по одной случайной точке. Можно выбирать для изменения несколько точек в хромосоме, причем их число также может быть случайным. Используют и мутации с изменением сразу некоторой группы подряд идущих точек[5].

3.1.4. Операторы отбора особей в новую популяцию

Для создания новой популяции можно использовать различные методы отбора особей.

Отбор усечением (Truncation selection). При отборе усечением используют популяцию, состоящую как из особей-родителей, так и особей потомков, отсортированную по возрастанию значений функции пригодности особей. Число особей для скрещивания выбирается в соответствии с порогом $T \in [0;1]$. Порог определяет, какая доля особей, начиная с самой первой (самой пригодной), будет принимать участие в отборе. В принципе, порог можно задать и числом больше 1, тогда он будет просто равен числу особей из

текущей популяции, допущенных к отбору. Среди особей, попавших «под порог», случайным образом выбирается одна и записывается в новую популяцию. Процесс повторяется n раз, пока размер новой популяции не станет равен размеру исходной популяции. Новая популяция состоит только из особей с высокой пригодностью, причем одна и та же особь может встречаться несколько раз, а некоторые особи, имеющие пригодность выше пороговой, могут не попасть в новую популяцию. Из-за того, что в этой стратегии используется отсортированная популяция, время ее работы может быть большим для популяции большого размера и зависеть также от алгоритма сортировки.

Элитарный отбор (Elite selection). Создается промежуточная популяция, которая включает в себя как родителей, так и их потомков. Члены этой популяции оцениваются, а за тем из них выбираются N самых лучших (пригодных), которые и войдут в следующее поколение. Зачастую данный метод комбинируют с другими — выбирают в новую популяцию, например, 10 % «элитных» особей, а остальные 90 % — одним из традиционных методов селекции. Иногда эти 90 % особей создают случайно, как при создании начальной популяции перед запуском работы генетического алгоритма. Использование стратегии элитизма оказывается весьма полезным для эффективности ГА, так как не допускает потерю лучших решений. К примеру, если популяция сошлась в локальном максимуме, а мутация вывела одну из строк в область глобального, то при предыдущей стратегии весьма вероятно, что эта особь в результате скрещивания будет потеряна, и решение задачи не будет получено. Если же используется элитизм, то полученное хорошее решение будет оставаться в популяции до тех пор, пока не будет найдено еще лучшее.

Отбор вытеснением (Exclusion selection). В данном отборе выбор особи в новую популяцию зависит не только от величины ее пригодности, но и от того, есть ли уже в формируемой популяции особь с аналогичным хромосомным набором. Отбор проводится из числа родителей и их потомков. Из всех особей с одинаковой приспособленностью предпочтение сначала отдается особям с разными генотипами. Таким образом, достигаются две цели: во-первых, не теряются лучшие найденные решения, обладающие различными хромосомными наборами, во-вторых, в популяции постоянно поддерживается генетическое разнообразие. Вытеснение в данном случае формирует новую популяцию скорее из удаленных особей, вместо особей, группирующихся около текущего найденного решения. Данный метод наиболее пригоден для многоэкстремальных задач, при этом помимо определения глобальных экстремумов появляется возможность выделить и те локальные максимумы, значения которых близки к глобальным.

Метод Больцмана, или метод отжига (Boltzman selection). В данном методе вероятность отбора в новую популяцию зависит от управляющего параметра — температуры T [6].

3.1.5. Параметры ГА

Вероятность кроссинговера обычно выбирается достаточной высокой 80–95 %. Однако, в некоторых задачах наилучший результат достигается при кроссинговере с вероятностью 60 %.

Вероятность мутации должна быть мала: 0,5–1 %.

Как это ни странно, очень большой размер популяции обычно не приводит к хорошим результатам (скорость сходимости алгоритма не увеличивается). Оптимальный размер популяции — 20–30 особей, однако в некоторых задачах, как пишут исследователи, требуется 50–

100 особей. Исследования показывают, что оптимальный размер популяции зависит от размера кодовых строк (хромосом). Так, для алгоритма с 32-битовыми хромосомами размер популяции будет больше, чем для алгоритма с 16-битовыми хромосомами.

В основном используют рулеточный отбор, но и иногда лучше применить отбор с усечением. Существует много других методов, меняющих параметры отбора в течение всей работы ГА. Элитизм применяется, если не используются другие методы, сохраняющие найденное хорошее решение. Выбор способа кодирования обусловлен поставленной задачей и размером объекта поиска.

Операторы жизненного цикла (кроссинговера и мутации) определяются выбранным кодированием.

3.1.6. Модернизация ГА

Одной из серьезных проблем, возникающих при использовании генетических алгоритмов, является преждевременная сходимость. Не рекомендуется использовать классические ГА на маленьких популяциях, поскольку в популяциях с малым размером гены распространяются слишком быстро: все особи становятся похожими (популяция вырождается) еще до того, как найдено решение задачи. То есть новый генотип с лучшей оценкой быстро вытесняет менее хорошие комбинации генов, исключая тем самым возможность получения лучшего решения на их базе. Можно предложить три основных пути устранения преждевременной сходимости: увеличение размера популяции, применение самоадаптирующихся генетических операторов и создание «банка» заменяемых особей.

В первом случае, увеличивая размер популяции, можно надеяться на достижение многообразия генотипа в популяции. Но, с другой стороны увеличение числа особей ведет к увеличению занимаемой памяти и времени работы алгоритма. Данный подход

может быть эффективен или при параллельных вычислениях, или при наличии достаточно простой целевой функции.

Второй, и самый распространенный способ, — использование самоадаптирующихся алгоритмов — является более эффективным. Самоадаптация заключается в применении динамических мутаций. Динамические мутации в зависимости от скрещивающихся особей меняют значение вероятности мутации, тем самым становится возможным самоуправление алгоритма. В таких случаях выбирается малый размер популяции.

В третьем подходе создается массив для сохранения особей, генотип которых был утерян при формировании новых поколений, и временами эти особи добавляются в популяцию.

Модификация стандартного варианта генетического алгоритма (SteadyStateGA) [7] затронула способ формирования *промежуточной популяции* (MatingPool), являющейся результатом отбора (селекции) для формирования наследников с помощью генетических операторов. SSGA не формируют промежуточную популяцию как стандартный генетический алгоритм, а осуществляют последовательно выбор пары наилучших индивидов, применяя к ним генетические операторы с целью формирования наследников, которые заменяют худшие индивиды популяции. Данная модификация генетического алгоритма – SteadyStateGA состоит из следующих этапов:

Шаг 0. Определение генетического представления задачи

Шаг 1. Создание первоначальной популяции $P(0) = x_1^0, \dots, x_N^0, t = 0$.

Шаг 2. Вычисление относительной (нормализованной) степени пригодности $f_H(x_i) = f(x_i) / \sum_{i=1}^N f(x_i) / N$.

Шаг 3. Выбор пары из лучших индивидов. Выбор худшего индивида. Применение скрещивания и мутация к выбранной паре лучших индивидов. Результат замещает худший индивид.

Шаг 4. $t = t + 1$, возврат к шагу 2.

При проектировании генетического алгоритма могут быть выгодно использованы знания, полученные селекционерами в области искусственной селекции. Генетические алгоритмы селекционеров (ГАС) моделируют именно искусственную селекцию ГАС, где под виртуальным селекционером понимается некоторый механизм селекции, который и является основным отличием ГАС от стандартного генетического алгоритма.

Генетический алгоритм селекционеров состоит из следующих этапов:

Шаг 0. Определение генетического представления задачи.

Шаг 1. Создание первоначальной популяции индивидов $P(0)$ размером N , $t = 0$.

Шаг 2. Виртуальный селекционер отбирает $T\%$ популяции для создания потомков. Это дает набор отобранных родителей.

Шаг 3. Формирование случайном образом из данного набора $N/2$ пар. Применение к каждой паре скрещивания и мутация, формируя новой популяции $P(t+1)$.

Шаг 4. $t = t + 1$, возврат к шагу 2.

Шаг 5. Возврат к шагу 3.

Селекция основывается преимущественно на статистических методах, которые позволяют произвести теоретический анализ и прогнозировать эффективность механизмов селекции, мутации и рекомбинации с помощью введенных уравнений селекции, реакции на селекцию и понятия наследственности.

Еще одна модификация генетического алгоритма затрагивает решение многокритериальных задач. Многокритериальный генетический алгоритм (МГА) также является модификацией стандартного генетического алгоритма и отличается способом селекции, поскольку при отборе пар родителей в этом случае используется не один, а несколько критериев. При этом предлагается большое число вариантов схем селекции и соответственно вариантов МГА. Сравнительные оценки показывают, что по эффективности VEGA имеет средние показатели, однако не оценивалась вычислительная сложность для различных вариантов МГА, по которой VEGA может существенно улучшить свои показатели[2].

Многокритериальный генетический алгоритм состоит из следующих этапов:

Шаг 0. Определение генетического представления задачи

Шаг 1. Создание первоначальной популяции $P(0) = x_1^0, \dots, x_N^0, t = 0$.

Шаг 2. Последовательное выполнение шагов 2.1-2.3.

Шаг 2.1. Вычисление значения степени пригодности каждого индивида по критерию $i = 1, \dots, k$.

Шаг 2.2. Для j от 1 до N/k осуществление селекции индивида из популяции в промежуточную популяцию.

Шаг 2.3. Возврат к шагу 2.1, если $l < k$.

Шаг 3. Формирование случайном образом из данного набора $N/2$ пар. Применение к каждой паре скрещивания, а также других генетических операторов, таких как мутация для формирования новой популяции $P(t+1)$.

Шаг 4. $t = t + 1$, возврат к шагу 2.

Стандартный генетический алгоритм представляет собой строго синхронизованный последовательный алгоритм, который в условиях большого пространства поиска или сложного пространства поиска

может быть неэффективен по критерию времени. Эту проблему позволяет решить другой вид генетического алгоритма – параллельный генетический алгоритм (ПГА). Следует отметить, что любая последовательная модификация стандартного генетического алгоритма может быть преобразована в параллельную.

По степени распараллеливания можно выделить следующие типы параллельных генетических алгоритмов:

- ПГА на базе популяции;
- ПГА на базе подпопуляций;
- ПГА на базе индивидов.

ПГА на базе популяции сохраняет стандартную структуру генетического алгоритма, работающего с целой популяцией, распараллеливание реализуется на этапе скрещивания и мутации. По степени распараллеливания процессов можно выделить следующие модели [7]:

- Синхронная модель «ведущий-ведомый», где главный процесс хранит целую популяцию в собственной памяти, выполняет селекцию, скрещивание и мутацию, но оставляет вычисление степени пригодности новых индивидов k подчиненным процессам;
- Полусинхронная модель «ведущий-ведомый», где новый индивид обрабатывается по мере освобождения одного из процессов;
- Асинхронная параллельная модель, где индивиды популяции хранятся в общей памяти, к которой можно обращаться k параллельным процессам. Каждый процесс выполняет оценку степени пригодности, а также генетические операции.

Каждый процесс работает независимо от других. Единственное отличие между этой моделью и стандартным генетическим алгоритмом заключается в механизме селекции [5]. Очевидным в этом случае является вариант использования $N/2$ параллельных

процессоров при популяции в N индивидов. Тогда каждый процессор дважды случайным образом выбирает два индивида из общей памяти и оставляет лучшего. Два выбранных индивида затем подвергаются скрещиванию, мутации и оценке степени пригодности. Возникающие в результате наследники размещаются в общей памяти.

Распределенный параллельный генетический алгоритм состоит из следующих этапов:

Шаг 0. Определение генетического представления задачи

Шаг 1. Создание первоначальной популяции индивидов и разделение на подпопуляции SP_1, \dots, SP_N .

Шаг 2. Формирование структуры подпопуляций.

Шаг 3. Для SP_1, \dots, SP_N - выполнение параллельно шагов 3.1-3.3.

Шаг 3.1. Применение в течение m поколений селекции и генетических операторов.

Шаг 3.2. Перемещение k хромосом в соседние подпопуляции.

Шаг 3.3. Получение хромосом из соседних подпопуляций.

Шаг 4. Возврат к шагу 3.

Особенность ПГА на базе подпопуляций заключается в использовании независимых конкурирующих подпопуляций, которые обмениваются индивидами с заданной частотой. При этом каждый процессорный блок выполняет последовательный генетический алгоритм с собственной подпопуляцией, при условии максимизации одной общей для всех функции степени пригодности. В этом случае для обмена индивидами должна быть определена структура связей подпопуляций. С точки зрения оценки и сравнения эффективности может быть рассмотрен вариант распределенной модели, в которой обмен индивидами не осуществляется. Результаты, представленные в [6], свидетельствуют о большей эффективности распределенного ПГА

по сравнению с этим частным случаем, а также со стандартным генетическим алгоритмом.

Существенным недостатком модели может стать снижение степени разнообразия при интенсивном обмене индивидами. С другой стороны, недостаточно частое перемещение может привести к преждевременной сходимости подпопуляций. При построении такой модели важно определить следующее:

- Связи между процессорами для обмена индивидами;
- Частоту обмена индивидами (оптимальной является частота обмена через 20 поколений [6]);
- Степень перемещения или число обмениваемых индивидов (оптимальным является 20% подпопуляции [6]);
- Способ селекции индивида для обмена;
- Критерий, по которому полученный индивид сможет заменить члена подпопуляции.

С точки зрения времени и даже числа поколений, затрачиваемых на решение задачи, ПГА эффективнее стандартного генетического алгоритма, но при этом некоторые задачи могут быть слишком простыми для ПГА. Параллельный поиск имеет смысл в том случае, если пространство поиска большое и сложное [4]. Увеличение числа процессоров в данной модели улучшает скорость сходимости, но не качество решения.

ПГА на базе индивидов имеют одну строку индивида, постоянно находящуюся в каждом процессорном элементе (ячейке). Индивиды выбирают пары и рекомбинируют с другими индивидами в их непосредственном ближайшем окружении (по вертикали горизонтали). Выбранный индивид затем совмещается с индивидом, постоянно находящимся в ячейке. В результате формируется один наследник, который может или не может заменить индивида в ячейке

в зависимости от выбранной схемы замещения. Таким образом, модель является полностью распределенной и не нуждается в централизованном управлении.

Параллельный генетический алгоритм на базе индивидов состоит из следующих этапов:

Шаг 0. Определение генетического представления задачи

Шаг 1. Создание первоначальной популяции индивидов и формирование структуры популяций.

Шаг 2. Локальное повышение каждым индивидом своей производительности (hill-climbing).

Шаг 3. Выполнение каждым индивидом селекции с целью поиска пары.

Шаг 4. Применение к паре скрещивания а также других генетических операторов, таких как мутация.

Шаг 5. Локальное повышение наследником своей производительности (hill-climbing). Замещение наследником родителя в соответствии с заданным критерием качества.

Шаг 6. Возврат к шагу 3.

При работе с моделью на базе индивидов необходимо задать:

- Структуру связей ячеек;
- Схему селекции;
- Схему замещения.

Исследования этой модели показали, что для сложных задач она способна обеспечить лучшие решения, чем стандартный генетический алгоритм.

В ходе исследований в области генетических алгоритмов и эволюционных алгоритмов в целом появилось большое количество направлений, и их число непрерывно растет.

Исследования в области генетического алгоритма и эволюционных алгоритмов в целом позволяют выделить важную особенность данных методов – эффективность эволюционных алгоритмов существенно зависит от таких взаимосвязанных параметров, как вероятность применения генетических операторов, их тип и размер популяции.

С этой целью сделан анализ особенности генетических алгоритмов и изучены предпосылки для адаптации.

3.1.7. Преимущества ГА

Генетические алгоритмы обладают рядом преимуществ, например, такими как:

- ГА не требуют никакой информации о поведении функции (например, дифференцируемости и непрерывности);
- разрывы, существующие на поверхности ответа, имеют незначительный эффект на полную эффективность оптимизации;
- ГА относительно стойки к попаданию в локальные оптимумы;
- ГА пригодны для решения крупномасштабных проблем оптимизации;
- ГА могут быть использованы для широкого класса задач;
- ГА просты в реализации;
- ГА могут быть использованы в задачах с изменяющейся средой.

На основании проведенного анализа можно сделать выводы о том, что в настоящее время генетические алгоритмы являются мощным вычислительным средством в разнообразных оптимизационных задачах. ГА различаются с большинством обыкновенных оптимизационных и поисковых методов в 4 пунктах:

- ГА оперирует закодированным множеством параметров, а не самими параметрами;

- ГА находит популяцию точек, а не отдельную точку;
- ГА использует значение объектной (целевой) функции, а не ее производную или другие вспомогательные значения;
- в ГА применяется вероятностное правило перехода, а не детерминистическое.

Для применения ГА необходимо прежде всего выбрать функцию пригодности (цели) адекватную задаче. Причем целевая функция должна иметь разнообразный рельеф, так как если на поверхности функции есть большие плоские участки, то ГА неэффективен. Это связано с тем, что многие особи в популяции при различии в генотипе не будут отличаться фенотипом, то есть несмотря на то, что особи различны, они имеют одинаковую пригодность, а значит алгоритм не имеет возможности выбрать лучшее решение и направление дальнейшего развития. Эта проблема известна как «проблема поля для гольфа», где все пространство абсолютно одинаково, за исключением лишь одной точки, которая и является оптимальным решением — в этом случае ГА просто остановится или будет блуждать абсолютно случайно. Другое требование к функции пригодности, состоит в требовании минимума вычислительных ресурсов при ее оценке, так как это влияет на скорость алгоритма.

В настоящее время ГА используются для решения таких задач, как:

- поиск глобального экстремума многопараметрической функции;
- аппроксимация функций;
- задачи о кратчайшем пути;
- настройка искусственной нейронной сети;
- обучение машин.

3.2. Исследование сходимости генетических алгоритмов к глобальному оптимуму

Для того чтобы лучше понять функционирование генетического алгоритма, будем использовать понятие *схема* и сформулируем основную теорему, относящуюся к генетическим алгоритмам и называемую теоремой о схемах [5-7]. Понятие *схема* было введено для определения множества хромосом, обладающих некоторыми общими свойствами, т.е. подобных друг другу. Если биты принимают значения 0 или 1 (рассматриваются хромосомы с двоичным алфавитом), то *схема* представляет собой множество хромосом, содержащих нули и единицы на некоторых заранее определенных позициях. При рассмотрении схем удобно использовать расширенный алфавит $\{0, 1, *\}$, в который помимо 0 и 1 введен дополнительный символ *, обозначающий любое допустимое значение, т.е. 0 или 1; символ * в конкретной позиции означает «все равно».

Считается, что хромосома принадлежит к данной схеме, если для каждой j -й позиции, $j = 1, 2, \dots, L$, где L - длина хромосомы; символ, занимающий j -ю позицию схемы, причем символу * соответствуют как 0, так и 1.

Генетический алгоритм базируется на принципе трансформации наиболее приспособленных хромосом. Пусть $P(0)$ означает исходную популяцию особей, а $P(k)$ - текущую популяцию (на k -й итерации алгоритма). Из каждой популяции $P(k)$, $k = 0, 1, \dots$ методом селекции выбираются хромосомы с наибольшей приспособленностью $M(k)$. Далее, результате объединения хромосом из популяции $M(k)$ в родительские пары и выполнения операции скрещивания с вероятностью p_c , а также операции мутации с вероятностью p_m

формируется новая популяция $P(k+1)$, в которую входят потомки хромосом из популяции $M(k)$.

Следовательно, для любой схемы, представляющей хорошее решение, было бы желательным, чтобы количество хромосом, соответствующих этой схеме, возрастала с увеличением количества итерации k .

На соответствующее преобразование схем в генетическом алгоритме оказывают влияние 3 фактора: селекция хромосом, скрещивание и мутация. Проанализируем воздействие каждого из них с точки зрения увеличения ожидаемого количества представителей отдельно взятой схемы.

Обозначим рассматриваемую схему символом s , а количества хромосом популяции $P(k)$, соответствующих этой схеме – $c(s,k)$. Следовательно, $c(s,k)$ можно считать количеством элементов (т.е. мощностью) множества $P(k) \cap S$.

Начнем с исследования влияния селекции. При выполнении этой операции хромосомы из популяции $P(k)$ копируются в родительскую хромосому $M(k)$ с вероятностью, определяемой выражением $P_s(ch_j) = F(ch_j) / \sum_{i=1}^N F(ch_i)$, где N - численность популяции.

Пусть $F(s,k)$ обозначает среднее значение функции приспособленности хромосом из популяции $P(k)$, которые соответствуют схеме s . Если

$$P(k)S = \{ch_1, \dots, ch_{c(s,k)}\},$$

то

$$F(s,k) = \frac{\sum_{j=1}^{c(s,k)} F(ch_j)}{c(s,k)}. \quad (3.1)$$

Величина $F(S, k)$ также называется приспособленностью схемы S на k -й итерации.

Пусть $\mathfrak{Z}(k)$ обозначает сумму значений функций приспособленности хромосом из популяции $P(k)$ мощностью N , т.е.

$$\mathfrak{Z}(k) = \sum_{i=1}^N F(ch_i^{(k)}).$$

Обозначим через $\bar{F}(k)$ среднее значение функции приспособленности хромосом этой популяции, т.е.

$$\bar{F}(k) = \frac{1}{N} \mathfrak{Z}(k).$$

Пусть $ch_j^{(k)}$ обозначает элемент из родительской хромосомы $M(k)$. Для каждого $ch_j^{(k)} \in M(k)$ и для каждого $j = 1, \dots, c(S, k)$ вероятность того, что $ch_j^{(k)} = ch_j$ определяется отношением $F(ch_j)/F(k)$. Поэтому ожидаемое количество хромосом в популяции $M(k)$, которые равны ch_j составит

$$N \frac{F(ch_j)}{\mathfrak{Z}(k)} = \frac{F(ch_j)}{\bar{F}(k)}.$$

Таким образом, ожидаемое количество хромосом из множества $P(k) \cap S$, отобранных для включения в родительскую хромосому $M(k)$, будет равно

$$\sum_{i=1}^{c(S, k)} \frac{F(ch_i)}{\bar{F}(k)} = c(S, k) \frac{F(S, k)}{\bar{F}(k)},$$

что следует из выражения (3.1).

Поскольку каждая хромосома из родительской популяции $M(k)$ одновременно принадлежит популяции $P(k)$, то хромосомы, составляющие множества $M(k) \cap S$ - это те же самые хромосомы, которые были отобраны из множества $P(k) \cap S$ для включения в популяцию $M(k)$. Если количество хромосом родительской популяции $M(k)$, соответствующих схеме S (т.е. количество элементов

множества $M(k) \cap S$, обозначить $b(S, k)$, то из приведенных рассуждений можно сделать следующий вывод:

Ожидаемое значение $b(S, k)$, т.е. ожидаемое значение количества хромосом из родительской популяции $M(k)$, соответствующих схеме S , определяется выражением

$$E[b(S, k)] = c(S, k) \frac{F(S, k)}{\bar{F}(k)}. \quad (3.2)$$

Из этого следует, что если схема S содержит хромосомы со значением функции приспособленности, превышающим среднее значение (т.е. приспособленности хромосом из популяции $P(k)$, и поэтому $F(S, k)/\bar{F}(k) > 1$), то ожидаемое количество хромосом из родительской популяции $M(k)$, соответствующих схеме S , будет больше количества хромосом из популяции $P(k)$, соответствующих схеме S . Поэтому можно утверждать, что селекция вызывает распространение схем с приспособленностью «лучше средней» и исчезновение схем с «худшей» приспособленностью.

Прежде чем приступить к анализу влияния генетических операторов скрещивания и мутации на хромосомы из родительской популяции, определим необходимые для дальнейших рассуждений понятия порядка и охвата схемы. Пусть L обозначает длину хромосом, соответствующих схеме S .

Порядок схемы S , иначе называемый с четностью схемы и обозначаемый $o(S)$ - это количество постоянных позиций в схеме, т.е. нулей и единиц в случае алфавита $\{0, 1, *\}$.

Порядок схемы $o(S)$ равен длине L за вычетом количества символов $*$. Порядок схемы, состоящей исключительно из символов $*$, равен нулю, а порядок схемы без единого символа $*$ равен L .

Охват схемы S , называемый также *длиной* схемы и обозначаемый $d(S)$ - это расстояние между первым и последним

постоянным символом (т.е. разность между правой и левой крайними позициями, содержащими постоянные символы).

Охват схемы $d(S)$ - это целое число из интервала $[0, L-1]$. Отметим, что охват схемы с постоянными символами на первой и последней позиции равен $L-1$. Охват схемы с единственной постоянной позицией равен нулю. Охват характеризует содержательность информации, заключенной в схеме.

В ходе анализа влияния операции скрещивания на родительский хромосомы $M(k)$ рассмотрим некоторую хромосому из множества $M(k) \cap S$, т.е. хромосому из родительской популяции, соответствующую схеме S . Вероятность того, что эта хромосома будет отобрана для скрещивания, равна p_c . Если ни один из потомков этой хромосомы не будет принадлежать к схеме S , то это означает, что точка скрещивания должна находиться между первым и последним постоянным символом данной схемы. Вероятность этого равна $d(S)/(L-1)$. Из этого можно сделать следующие выводы о влиянии скрещивания:

Для некоторой хромосомы из $M(k) \cap S$ вероятность того, что она будет отобрана для скрещивания и ни один из ее потомков не будет принадлежать к схеме S . Ограничена сверху величиной

$$p_c \frac{d(S)}{L-1}.$$

Эта величина называется *вероятностью уничтожения схемы S* .

Для некоторой хромосомы из $M(k) \cap S$ вероятность того, что она не будет отобрана для скрещивания либо, что хотя бы один из ее потомков после скрещивания будет принадлежать к схеме S , ограничена снизу величиной

$$1 - p_c \frac{d(S)}{L-1}.$$

Эта величина называется *вероятностью выживания схемы S* .

Легко показать, что если данная хромосома принадлежит к схеме S и отбирается для скрещивания, а вторая родительская хромосома также принадлежит к схеме S , то оба их потомка тоже будут принадлежать к схеме S . Выводы подтверждают значимость показателя охвата схемы $d(S)$ для оценки вероятности уничтожения или выживания схемы.

Рассмотрим теперь влияние мутации на родительскую хромосому $M(k)$. Оператор мутации с вероятностью p_m случайным образом изменяет значение в конкретной позиции с 0 на 1 и обратно. Очевидно, что схема переживет мутацию только в том случае, когда все ее постоянные позиции останутся после выполнения этой операции неизменными.

Популяция родительской хромосомы, принадлежащая к схеме S (т.е. хромосома из множества $M(k) \cap S$) останется в этой схеме тогда и только тогда, когда ни один символ этой хромосомы, соответствующий постоянным символам схемы S , не изменится в процессе мутации. Вероятность такого события равна $(1 - p_m)^{o(S)}$. Этот результат можно представить в форме следующего вывода о влиянии мутации:

Вероятность того, что некоторая хромосома из $M(k) \cap S$ будет принадлежать к схеме S после операции мутации, определяется выражением

$$(1 - p_m)^{o(S)}.$$

Эта величина называется *вероятностью выживания схемы S после мутации*.

Если вероятность мутации p_m мала ($p_m \ll 1$), то можно считать, что вероятность выживания схемы S после мутации приближенно равна

$$1 - p_m o(S).$$

Эффект совместного воздействия селекции, скрещивания и мутации с учетом факта, что если хромосома из множества $M(k) \cap S$ дает потомка, соответствующего схеме S , то он будет принадлежать к $P(k+1) \cap S$, ведет к построению следующей схемы репродукции [13]:

$$E[c(S, k+1)] \geq c(S, k) \frac{F(S, k)}{\bar{F}(k)} \left(1 - p_c \frac{d(S)}{L-1} \right) (1 - p_m)^{o(S)}. \quad (3.3)$$

Зависимость (3.3) показывает, как изменяется от популяции к популяции количество хромосом, соответствующих данной схеме. Это изменение вызывается тремя факторами, представленными в правой части выражения (3.3), в частности: $F(S, k)/\bar{F}(k)$ отражает роль среднего значения функции приспособленности, $1 - p_c d(S)/(L-1)$ показывает влияние скрещивания и $(1 - p_m)^{o(S)}$ - влияние мутации. Чем больше значение каждого из этих факторов, тем большим оказывается ожидаемое количество соответствий схеме S в следующей популяции. Это позволяет представить зависимость (3.3) в виде

$$E[c(S, k+1)] \geq c(S, k) \frac{F(S, k)}{\bar{F}(k)} \left(1 - p_c \frac{d(S)}{L-1} - p_m o(S) \right). \quad (3.4)$$

Для больших популяций зависимость (3.3) можно аппроксимировать выражением

$$c(S, k+1) \geq c(S, k) \frac{F(S, k)}{\bar{F}(k)} \left(1 - p_c \frac{d(S)}{L-1} - p_m o(S) \right). \quad (3.5)$$

Из формул (3.3) и (3.4) следует, что ожидаемое количество хромосом, соответствующих схеме S в следующем поколении, можно считать функций от фактического количества хромосом, принадлежащих этой схеме, относительной приспособленности схемы, а также порядка и охвата схемы. Заметно, что схемы с приспособленностью выше средней и с малым порядком и охватом характеризуется возрастанием количества своих представителей в

последующих популяциях. Подобный рост имеет показательный характер, что следует из выражения (3.2). Для больших популяций эту формулу можно заменить рекуррентной зависимостью вида [16]

$$c(S, k+1) = c(S, k) \frac{F(S, k)}{\bar{F}(k)}. \quad (3.6)$$

Если допустить, что схема S имеет приспособленность на $\varepsilon\%$ выше средней, т.е.

$$F(S, k) = \bar{F}(k) + \varepsilon \bar{F}(k).$$

то при подстановке выражения (3.5) в неравенство (3.4) в предположении, что ε не изменяется во времени, при старте от $k = 0$ получаем

$$\begin{aligned} c(S, k) &= c(S, 0)(1 + \varepsilon)^k, \\ \varepsilon &= (F(S, k) - \bar{F}(k)) / \bar{F}(k), \end{aligned} \quad (3.7)$$

т.е. $\varepsilon > 0$ для схемы с приспособленностью выше средней и $\varepsilon < 0$ - в противном случае.

Равенство (3.7) описывает геометрическую прогрессию. Из этого следует, что в процессе репродукции схемы, оказавшиеся лучше (хуже) средних, выбираются на очередных итерациях генетического алгоритма в показательно возрастающих (убывающих) количествах. Обратим внимание, что зависимости (3.2)-(3.6) основаны на предположении, что функция приспособленности F принимает только положительные значения. При использовании генетических алгоритмов для решения оптимизационных задач, в которых целевая функция может принимать и отрицательные значения, необходимы некоторые дополнительные соотношения между оптимизируемой функцией и функцией приспособленности. Конечный результат, получаемый из выражений (3.3)-(3.5), можно сформулировать в форме теоремы. Это основная теорема генетических алгоритмов, иначе называемая теоремой о схемах [6].

Схемы малого порядка, с малым охватом и с приспособленностью выше средней формирует показательно возрастающее количество своих представителей в последующих поколениях генетического алгоритма.

В соответствии с приведенной теоремой важным вопросом становится кодирование, которое должно обеспечивать построение схем малого порядка, с малым охватом и с приспособленностью выше средней.

Для эволюционных алгоритмов известно[5], что сходимость к глобальному оптимуму в задаче оптимизации достигается в том случае, если есть уверенность в том, что алгоритм находит решения за конечное число шагов, и если такое решение будет оставаться в дальнейшем в популяции. Поскольку состояния переходов эволюционных алгоритмов имеют стохастический характер, детерминированная концепция сходимости не может быть использована для определения срока действия таких алгоритмов. Существуют две широко используемые методы стохастической сходимости эволюционных алгоритмов – это полное совпадения и совпадение по значению [7].

Определение 3.1. Пусть x есть случайная переменная и $x_t, t > 0$ - последовательность случайных переменных. Тогда последовательность x_t сходиться полностью в x для любых $\varepsilon > 0$, если

$$\lim_{t \rightarrow \infty} \sum_{i=0}^t P(|x_i - x| > \varepsilon) < \infty.$$

Определение 3.2. Пусть x есть случайная переменная и $x_t, t > 0$ - последовательность случайных переменных. Тогда последовательность x_t будет совпадать по значению в x , если

$$\lim_{t \rightarrow \infty} E[|x_t - x|] = 0.$$

Приведем дополнительно определения для сходимости генетического алгоритма:

Определение 3.3. Пусть $x_t : t \geq 0$ является последовательностью популяций хромосом, сгенерированной генетическим алгоритмом, и пусть F_t является соответствующим значением лучшей хромосомы в популяции в момент времени t . Генетический алгоритм имеет полную сходимость к глобальному оптимуму f^* задачи оптимизации, определенной функцией $f : x \rightarrow R$, если неотрицательная случайная последовательность $D_t = f^* - F_t$ сходится полностью в 0.

Определение 3.4. Пусть $x_t : t \geq 0$ является последовательностью популяций хромосом, сгенерированной генетическим алгоритмом и пусть F_t является соответствующим значением лучшей хромосомы в популяции в момент времени t . Генетические алгоритмы имеют сходимость по значению к глобальному оптимуму f^* задачи оптимизации, определенной функцией $f : x \rightarrow R$, если неотрицательная случайная последовательность $D_t = f^* - F_t$ сходится полностью в 0.

В общем виде один шаг работы генетического алгоритма можно представить следующим образом:

$$\begin{aligned}(x'_1, \dots, x'_n) &= sel(x_1, \dots, x_n); \\(x''_1, \dots, x''_n) &= cross(x'_1, \dots, x'_n); \\(x'''_1, \dots, x'''_n) &= mut(x'_1, \dots, x'_n);\end{aligned}$$

где $(x_1, \dots, x_n) \in x^n$ - текущая популяция хромосом; (x'_1, \dots, x'_n) - популяция хромосом, возникающая в результате селекции; (x''_1, \dots, x''_n) - популяция хромосом, возникающая в результате скрещивания; (x'''_1, \dots, x'''_n) - популяция хромосом, возникающая в результате мутации.

В [6] доказано, что эволюционные алгоритмы сходятся как в среднем, так и полностью к глобальному оптимуму при выполнении следующих условий:

- Каждая хромосома в популяции может быть изменена на произвольную другую хромосому в одной единственной мутации с вероятностью $p > 0$.
- Лучшая хромосома в популяции выживает в каждом поколении с вероятностью $p = 1$.

Формально эти условия можно представить в виде:

$$\forall x, y \in X \quad p\{y = mut(x)\} \geq \delta_m > 0;$$

$$p\{V_n^*(sel(x_1, \dots, x_k)) = V_k^*(x_1, \dots, x_k)\} = 1$$

где V_i^* оператор возврата лучшей хромосомы из i хромосом популяции.

Если условие 1 действительно, можно показать, что эволюционный алгоритм сходиться к глобальному оптимуму за конечное число шагов с вероятностью $p = 1$, независимо от его инициализации, но может сходиться, если нельзя гарантировать, что оптимум действительно останется в популяции после того, как он был найден. Если условие 2 тоже верно, то можно показать, что эволюционный алгоритм сходиться к глобальному оптимуму.

В генетическом алгоритме оператор скрещивания тоже принимает участие в механизме выбора эволюционного процесса. Следовательно, условие 1 может быть применено к генетическому алгоритму в отношении мутации и скрещивания, а оператор селекции нужно рассматривать в формальном описании условия 2.

Тогда выражения

$$p\{V_n^*(sel(x_1, \dots, x_n)) = V_k^*(x'_1, \dots, x'_n)\} = 1$$

$$p\{V_n^*(cross(x'_1, \dots, x'_n)) = V_k^*(x''_1, \dots, x''_n)\} = 1$$

правильно описывают условие 2 по отношению к ГА.

Покажем, что генетический алгоритм сходиться полностью и в среднем к глобальному оптимуму. Для этого покажем, что условия 1 и 2 удовлетворяются. Операторы мутации и скрещивания могут вносить изменения в хромосомы, использования которых способствует поиску оптимума.

Пусть битовые строки хромосом длиной L представлены вектором $\{0,1\}^L$. Если при сравнении цепочка бит длиной $(L-c)$ в хромосоме представляет оптимум, следовательно, в c битах хромосом и не совпадет с оптимум. Тогда вероятность оператора мутации и скрещивания для достижения глобального оптимума за один шаг равна:

$$P_c^{(L)} = \frac{c!}{L_c} \cdot \frac{1}{L}.$$

Здесь благоприятное число выборов $c!$ является числом перестановок с элементов из возможного числа выборов L_c . Эта вероятность должна быть умножена на вероятность того, что оператор мутации случайным образом изменяет биты, т.е. на $1/L$, которая является вероятностью того, что $c=r$, где r -случайно выбранная число битов, которые подвергаются мутации.

Если хромосома представлено вектором $\{0,1,\dots,k-1\}^L$, каждый элемент которого взят из алфавита из k элементов, то вероятность того, что каждая из цифр, которые будут подвержены мутации и скрещиванию, будет являться цифрой оптимальной цепочки, равна

$$P_c^{(L)} = \frac{c!}{L_c} \cdot \frac{1}{(k-1)^c} \cdot \frac{1}{L}.$$

Если $P_c^{(L)}$ всегда положительно, то условие 1 выполняется. При этом следует отметить, что существует также вероятность $p_s > 0$ того, что оптимум случайно введен в популяцию оператором селекции,

который нужно рассматривать, хотя рассмотрена вероятностном $P_c^{(L)}$ достаточно, чтобы показать выполнения условия 1.

Чтобы показать, что условие 2 также выполняется, необходимо учесть все операторы, действующие на популяцию, а также показать, что ни один из них не приводит к потере оптимального решения, если оно будет найдено. Проанализируем операторы генетического алгоритма.

Оператор селекции, после избавления от наименее пригодных хромосом, пополняет популяцию новыми хромосомами, поэтому оптимум не может быть потерян.

Оператор мутации и скрещивания действуют только на хромосомы популяции, не трогая при этом лучшую хромосому в популяции. Следовательно, эти операторы также не теряют оптимум.

Таким образом, приведенные рассуждения показывают выполнения условия 2. Следовательно, имеет место сходимости генетического алгоритма, если операторы мутации, скрещивания и селекции удовлетворяют условию 1, и условие 2 тоже выполняется.

Покажем, что генетический алгоритм может не сходиться полностью к глобальному оптимуму независимо от его инициализации, если оператор мутации меняет отдельные биты и используется не элитный вариант оператора скрещивания.

Выше было показано выполнение условия 1, в соответствии с которым генетический алгоритм достигает оптимума при условии, что он использует операторы мутации, скрещивания и (или) селекции. Необходимо показать выполнение условия 2, которое является необходимым, но не достаточным условием сходимости.

Чтобы показать, что генетический алгоритм не сходиться полностью к глобальному оптимуму, достаточно показать, что каждый раз, когда найден оптимум, будет существовать следующие

поколение хромосом, котором генетический алгоритм не имеет оптимума в популяции. Если есть вероятность, даже очень малая, того, что генетический алгоритм выйдет из оптимума, найдя в другой момент времени другое оптимальное решение, то это значит, что генетический алгоритм не сойдется полностью с вероятностью 1. Это вероятность гарантируется также неэлитизмом оператора селекции в том смысле, что хромосома, представляющее оптимизм при достижении возраста через t поколений, будет изменены в популяции.

Оператор селекции создаст популяцию, состоящий из копий оптимальных решений x_1 и из неоптимальных решений x_2 . Так как оператор мутации всегда изменяет, по крайней мере, один бит, хромосомы оптимальных решений не будут достигать глобальных оптимумов.

С другой стороны, каждое неоптимальное решение может стать оптимальным в следующем шаге с вероятностью p такой, что $p \leq p_{d=1}$, где $p_{d=1}$ является вероятностью достижения оптимума из наиболее вероятного состояния с шагом $d = 1$.

Эти рассуждения доказывают следующее утверждение 3.1.

Утверждение 3.1. Анализ сходимости генетического алгоритма основан на выполнении двух условий:

- 1) В результате мутации и скрещивания можно достичь оптимальное состояние из неоптимального за один шаг;
- 2) Как только оптимальное состояние будет найдено, оно сохранится в популяции и не будет утеряно.

3.3. Решение многокритериальной оптимизации с использованием генетических алгоритмов с искусственным отбором

Пусть задана выборка нечетких экспериментальных данных (X_r, y_r) , $r = \overline{1, M}$; здесь $X_r = (x_{r1}, x_{r2}, \dots, x_{rn})$ - входной n -мерный вектор и $y_r = (y_1, y_2, \dots, y_M)$ - соответствующий ему выходной вектор.

В общем виде требуется построить модель, основанную на нечетких правилах вывода:

$$\bigcup_{p=1}^{k_j} \left(\bigcap_{i=1}^n x_i = a_{i,jp} \wedge \bigwedge_{jp} w_{jp} \right) \rightarrow y_j = b_{m0} + b_{m1}x_1^j + \dots + b_{mn}x_n^j.$$

В процессе построения модели нужно найти такие значения коэффициентов правил

при которых достигается минимум следующего выражения:

$$\sum_{r=1}^M (y_r - y_r^f) \rightarrow \min, \quad (3.8)$$

где y_r^f - результат нечетких правил вывода с параметром B в r -й строке выборки (X_r) .

Входному матрицу X_r соответствует следующий результат нечеткого вывода:

$$y_r^f = \frac{\sum_{i=1}^m \mu_{d_i}(X_r) \cdot d_i}{\sum_{i=1}^m \mu_{d_i}(X_r)}, \quad (3.9)$$

здесь $d_i = b_{i0} + b_{i1}x_{r1} + b_{i2}x_{r2} + \dots + b_{in}x_{rn}$ - выход i -правила; $\mu_{d_i}(X_r)$ - функция принадлежности, соответствующая каждой экспериментальной информации:

В общем случае процедура оптимизации на основе обычного последовательного комплекс-метода выглядит следующим образом: требуется отыскать минимум некоторой функции

$$E(x) = \sum_{r=1}^M (y_r - y_r^f)^2 \rightarrow \min$$

достаточно общего вида, при этом о характере этой функции не делается практически никаких априорных предложений. Работа алгоритма начинается с формирования начального комплекса

$$x_i(0) = \begin{pmatrix} x_1(0) \\ x_2(0) \\ \vdots \\ x_N(0) \end{pmatrix} = \begin{pmatrix} x_{11}(0) & x_{12}(0) & \dots & x_{1n}(0) \\ x_{21}(0) & x_{22}(0) & \dots & x_{2n}(0) \\ \vdots & \vdots & \dots & \vdots \\ x_{N1}(0) & x_{N2}(0) & \dots & x_{Nn}(0) \end{pmatrix} \quad i = \overline{1, N} \geq n + 1$$

представляющего собой популяцию хромосом, достаточно произвольно расположенных в n -мерном пространстве поиска. С начала выполняется операции селекции, затем скрещивания и мутация. От этого получается новая популяция хромосом $x_i(1)$.

После этого производится операция выбора. На этом этапе вычисляется значение функции во всех хромосом и находится средняя годность популяции

$$E_{\text{сред}} = \frac{1}{N} \sum_{i=1}^N E(x_i(k)).$$

Затем хромосомы с годностью меньше средней по всей популяции заменяется на «наилучшей» хромосому.

Если $E_{\text{сред}} < E(x_i(k))$ то будет $x_i(k+1) = x_i(k)$, который дает $\min_{i=1, N} (E(x_i(k)))$.

Среди множества этих хромосом находится «наихудшая» $x_i(1)$, в которой значение функции $E(x_H(1))$ максимально, после чего эта точка отражается через центр тяжести всех остальных вершин-точек, формируя новый комплекс $x_i(1), i = \overline{1, N}$. Такое отражение вместе с

растяжением и сжатием обеспечивают движение комплекса к экстремуму функции $E(x)$, при этом, благодаря достаточно случайному распределению хромосом в популяции, поиск имеет глобальный характер.

С формальной точки зрения рассмотрим процесс оптимизации на k -й итерации поиска, когда сформирован комплекс $x_i(k)$, $i = 1, 2, \dots, N$.

Среди множества $x_i(k)$ находится «наихудшая» такая, что

$$E(x_H(k)) = \min_i \{E(x_1(k)), \dots, E(x_H(k))\},$$

после чего определяется центр тяжести популяции без наихудшей точки:

$$\begin{aligned} x_{c1}(k) &= (x_{11}(k) + x_{21}(k) + \dots + x_{N1}(k) - x_{H1}(k)) / (N - 1) \\ x_{c2}(k) &= (x_{12}(k) + x_{22}(k) + \dots + x_{N2}(k) - x_{H2}(k)) / (N - 1) \\ &\dots \\ x_{cn}(k) &= (x_{1n}(k) + x_{2n}(k) + \dots + x_{Nn}(k) - x_{Hn}(k)) / (N - 1) \end{aligned}$$

Далее $x_H(k)$ отражается через центр тяжести $x_c(k)$, формируя новую вершину комплекса $x_R(k)$, которая теоретически расположена ближе к экстремуму, чем $x_H(k)$ и $x_c(k)$, т.е.

$$E(x_R(k)) < E(x_c(k)) < E(x_H(k)).$$

Операция отражения формально имеет следующий вид:

$$\begin{aligned} x_R(k) &= x_c(k) + \eta_R(x_c(k) - x_H(k)) = \\ &= \frac{1}{N-1}x_1(k) + \dots + \frac{1}{N-1}x_{N-1}(k) + \frac{\eta_R}{N-1}x_1(k) + \dots \\ &\quad + \frac{\eta_R}{N-1}x_{N-1}(k) - \eta_R x_H(k) = X(k)R, \end{aligned}$$

где η_R - параметр шага отражения, часто полагаемый равным единице,

$X(k) = (x_H(k), x_1(k), \dots, x_{N-1}(k))$ - $(n \times N)$ -матрица координат вершин

комплекса, $R = \left(-\eta_R, \frac{1+\eta_R}{N-1}, \dots, \frac{1+\eta_R}{N-1}\right)^T$ - $(N \times 1)$ -вектор.

В случае, если отражения вершины $x_R(k)$ окажется «наилучшей» среди всех остальных популяции хромосом, т.е.

$$E(x_R(k)) < E(x_i(k)) < E(x_H(k)), \quad i = 1, 2, \dots, N-1,$$

то производится операция растяжения комплекса в направлении от центра тяжести $x_C(k)$ до $x_R(k)$ согласно выражению

$$x_E(k) = x_C(k) + \eta_E(x_R(k) - x_C(k)) = X(k)E,$$

где η_E – параметр шага растяжения, часто полагаемый равным двум,

$$E = \left(-\eta_E, \eta_R, \frac{1 - \eta_E(1 - \eta_R)}{N-1}, \dots, \frac{1 - \eta_E(1 - \eta_R)}{N-1} \right)^T.$$

Если же $x_R(k)$ окажется наихудшей среди всех $x_i(k)$, комплекс сжимается согласно соотношению

$$x_S(k) = x_C(k) + \eta_S(x_R(k) - x_C(k)) = X(k)S,$$

где η_S – параметр шага сжатия, обычно полагаемый равным 0,5,

$$S = \left(-\eta_S, \eta_R, \frac{1 - \eta_S(1 - \eta_R)}{N-1}, \dots, \frac{1 - \eta_S(1 - \eta_R)}{N-1} \right)^T.$$

При $\eta_R = 1$, $\eta_E = 2$, $\eta_S = 0,5$ приходим к простым выражениям

$$R = \left(-1, \frac{2}{N-1}, \dots, \frac{2}{N-1} \right)^T, \quad E = \left(-2, \frac{1}{N-1}, \dots, \frac{1}{N-1} \right)^T$$

$$S = \left(-0,5, \frac{1}{N-1}, \dots, \frac{1}{N-1} \right)^T.$$

Таким образом, в процессе своего движения к экстремуму оптимизируемой функции комплекс на каждой итерации теряет одну наихудшую вершину и приобретает одну новую точку так, что на $k+1$ -й итерации новый комплекс также имеет N точек-вершин.

В отличие от комплекс-метода, в генетических алгоритмах в результате селекции из популяции одновременно исключаются несколько особей с наихудшими (максимальными) значениями функции приспособленности. В связи с этим представляется целесообразным ввести алгоритм комплекс-метода с отражением, растяжением и сжатием сразу нескольких вершин.

Итак, пусть на k -й итерации процесса оптимизации имеется комплекс $x_i(k)$, $i=1,2,\dots,N_c$ с $P < N$ наихудшими вершинами $x_{H_p}(k)$, $p=1,2,\dots,P$. Тогда координаты центра тяжести комплекса без вершин $x_{H_p}(k)$ задаются выражением

$$x_c(k) = \frac{1}{N-P} \left(\sum_{i=1}^N x_i(k) - \sum_{p=1}^P x_{H_p}(k) \right),$$

а процедура отражения описывается системой уравнений

$$\begin{cases} x_{R_1}(k) = x_c(k) + \eta_R(x_c(k) - x_{H_1}(k)) \\ \vdots \\ x_{R_p}(k) = x_c(k) + \eta_R(x_c(k) - x_{H_p}(k)) \end{cases}$$

или

$$\begin{cases} x_{R_1}(k) = (1 + \eta_R)x_c(k) - \eta_R x_{H_1}(k) \\ \vdots \\ x_{R_p}(k) = (1 + \eta_R)x_c(k) - \eta_R x_{H_p}(k) \end{cases}$$

В матричной форме эти системы уравнений могут быть записаны более компактно

$$X_R(k) = X(k)R_p,$$

где $X(k) = \left(\underbrace{x_{H_1}(k), \dots, x_{H_p}(k)}_{(n \times P)}, \underbrace{x_1(k), \dots, x_{N-P}(k)}_{(n \times (N-P))} \right)$ - $(n \times N)$ -матрица,

$X_R(k) = (x_{R_1}(k), \dots, x_{R_p}(k))$ - $(n \times P)$ -матрица, $R_p = \begin{matrix} P \\ \left\{ \begin{matrix} -\eta_R I_P \\ \hline \frac{1 + \eta_R}{N - P} I_{N-P, P} \end{matrix} \right\} \\ N - P \end{matrix}$ -

$(N \times P)$ - матрица, $I_P - (P \times P)$ -единичная матрица, $I_{N-P, P} - ((N - P) \times P)$ -единичная матрица, образованная единицами. В случае, если среди отраженных вершин оказывается $Q \leq P$ наилучших, комплекс растягивается в их направлении согласно уравнениям

$$\begin{cases} x_{E_1}(k) = x_C(k) + \eta_E(x_{R_1}(k) - x_C(k)), \\ \vdots \\ x_{E_Q}(k) = x_C(k) + \eta_E(x_{R_Q}(k) - x_C(k)), \end{cases}$$

или

$$\begin{cases} x_{E_1}(k) = (1 - \eta_E)x_C(k) + \eta_E x_{R_1}(k), \\ \vdots \\ x_{E_Q}(k) = (1 - \eta_E)x_C(k) + \eta_E x_{R_Q}(k), \end{cases}$$

или

$$X_E(k) = X(k)E_Q,$$

где $X_E(k) = (x_{E_1}(k), \dots, x_{E_Q}(k))$ - $(n \times Q)$ -матрица,

$$E_Q = \begin{matrix} Q \\ N-Q \end{matrix} \left\{ \begin{matrix} \left(\begin{array}{c} -\eta_E \eta_R I_Q \\ \hline 1 - \frac{\eta_E(1-\eta_R)}{N-P} I_{N-Q,Q} \end{array} \right) \end{matrix} \right. - (N \times Q)\text{-матрица.}$$

Если, далее, среди отражаемых вершин окажется $U \leq P$ наихудших, комплекс сжимается в их направлении согласно уравнениям

$$\begin{cases} x_{S_1}(k) = x_C(k) + \eta_S(x_{R_1}(k) - x_C(k)), \\ \vdots \\ x_{S_U}(k) = x_C(k) + \eta_S(x_{R_U}(k) - x_C(k)), \end{cases}$$

или

$$\begin{cases} x_{S_1}(k) = (1 + \eta_S)x_C(k) + \eta_S x_{R_1}(k), \\ \vdots \\ x_{S_U}(k) = (1 + \eta_S)x_C(k) + \eta_S x_{R_U}(k), \end{cases}$$

или $X_S(k) = X(k)S_U$,

где $X_S(k) = (x_{S_1}(k), \dots, x_{S_U}(k))$ - $(n \times U)$ - матрица,

$$S_U = \begin{matrix} U \\ N-U \end{matrix} \left\{ \begin{matrix} \left(\begin{array}{c} -\eta_S \eta_R I_U \\ \hline 1 - \frac{\eta_S(1-\eta_R)}{N-U} I_{N-U,U} \end{array} \right) \end{matrix} \right. - (N \times U)\text{-матрица.}$$

Таким образом, комплекс-метод приобретает черты генетического алгоритма, у которого в результате селекции на каждой итерации из популяции удаляется несколько наихудших особей.

Объединяя введенную модификацию комплекс-метода с Холландовской генетической процедурой, приходим к алгоритму, реализующему идею искусственного отбора, состоящую в данном случае в том, что из популяции не только удаляются наихудшие особи, но и одновременно создаются их «антиподы», обладающие улучшенными свойствами.

Работа такого алгоритма образована последовательностью следующих шагов:

- создание начальной популяции, образованной $P(0)$ особями хромосомами – вершинами комплекса;
- операция скрещивания с увеличением популяции $P_{CR}(0) > P(0)$;
- операция мутации $P_M(0) > P_{CR}(0)$;
- первая селекция (определение наихудших особей) без сокращения популяции $P_{SELI}(0) = P_M(0)$;
- операция выбора заменяем значения наилучшему во всей популяции;
- операция отражения с удалением P наихудших особей $P_M(0) < P_{SELI}(0)$;
- операция растяжения без увеличения популяции $P_E(0) = P_R(0)$;
- операция сжатия без увеличения популяции $P_I(0) = P_E(0)$;
- вторая селекция с удалением $P_W(0)$ наихудших особей $P_{SEL2}(0) = P_I(0) - P_W(0) = P(1)$ и формирование популяции $P(1)$ следующей итерации алгоритма.

3.4. Решения практических задач оптимизации в области телекоммуникаций и обработки изображений на основе использования генетического алгоритма с искусственным отбором

3.4.1 Интеграция нечетко-множественного подхода для комплексного исследования устойчивости функционирования системы телекоммуникации

Средства и сети телекоммуникации (в дальнейшем системы телекоммуникации (СТ)) являются главными артериями для устойчивого развития всех секторов экономики.

Под системой телекоммуникации понимается организационно-технологическая система, включающая в себя средства и сети телекоммуникации (линии связи, линии передачи, физические цепи, линейно-кабельные сооружения связи), системы энергообеспечения, обеспечения информационной безопасности, защиты информации, технической эксплуатации и технического обслуживания, абонентские терминалы, подготовки и переподготовки кадров.

Известно, что создание цифровой системы телекоммуникации является сложной и капиталоемкой задачей, в тоже время является достаточно консервативной. Консервативность заключается в том, что вложенные капитальные средства на создание цифровой системы телекоммуникации возвращаются годами, путем создания и продажи услуг телекоммуникации.

В этих условиях актуальной является решение проблемы обеспечения устойчивого функционирования цифровой системы телекоммуникации [1].

Перерыв в работе системы телекоммуникации даже на короткий промежуток времени функционирования вызывает потерю большого объема передаваемой информации, что являясь нарушением устойчивости функционирования системы, может привести к серьезным техническим и финансовым последствиям [5].

Другой важной задачей, определяющей устойчивость функционирования системы телекоммуникации в целом является требование к доставке абоненту (потребителям услуг) достоверной информации, передаваемой по каналам связи. При современных сверхвысокоскоростных волоконно-оптических системах передачи информации даже, кратковременные внешние и внутренние факторы влияния могут вызвать, так называемые «потери пакетов данных». Обнаружение таких пакетов обуславливает необходимость их повторной передачи. Это прежде всего вызывает запаздывание доставляемой абоненту сообщения на время обнаружения и повторной передачи «потерянного пакета данных», ухудшает качество предоставляемых услуг и следовательно отрицательно влияет на рыночный имидж оператора телекоммуникации, особенно международного оператора.

Для оперативной оценки состояния функционирования СТ предлагается концепция оценки на основе комплексной модели, основанной на системном рассмотрении элементов и составляющих СТ.

Комплексная модель включает в себя следующие базовые составляющие:

- идентификация (оценка) технического состояния СТ с учетом внешних и внутренних воздействующих факторов;
- принятие решений по регулированию устранения влияния воздействующих факторов.

Из-за того, что факторы оказывающие влияние на устойчивость системы телекоммуникации характеризуются разнородностью, т.е. разными предметными шкалами измерений и характера информации (количественная, качественная, смешанная), то для реализации такой концепции целесообразным представляется применение принципов теории нечетких множеств (ТНИ), позволяющей принимать решения в условиях, когда цели, ограничения и последствия возможных воздействий на СТ считаются точно не известными [1].

Допустим, что S_1, S_2, S_3, S_4 – состояния СТ, определяемые лингвистически в следующем порядке:

S_1 - <устойчивое состояние СТ>;

S_2 - <более менее устойчивое состояние СТ>;

S_3 - <неустойчивое состояние СТ>;

S_4 - <абсолютно устойчивое состояние СТ>.

Сформулированные состояния S_1, S_2, S_3, S_4 считаются видами оценки состояния СТ, подлежащих определению на основе результатов их исследования.

В комплексной модели приняты следующие обозначения:

S - <состояние СТ>;

X - <влияние внешних факторов>;

Y - <влияние внутренних факторов>;

Z - <влияние энергетических факторов>;

P - <обслуживание>;

$$S = f_s(X, Y, Z, P),$$

$$S_1 = f_{s_1}(X_1, X_2),$$

$$S_2 = f_{s_2}(Y_1, Y_2),$$

$$S_3 = f_{s_3}(Z_1, Z_2, Z_3),$$

$$S_4 = f_{s_4}(P_1, P_2, P_3).$$

X_1 – влияние факторов физического происхождения;

X_2 – влияние преднамеренных факторов;

$$X_1 = f_{X1}(X_{11}, X_{12}, X_{13}, X_{14}, X_{15}),$$

$$X_2 = f_{X2}(X_{21}, X_{22}, X_{23}, X_{24}, X_{25}),$$

Y_1 – воздействие в виде параметрического отказа;

Y_2 – воздействие в виде катастрофического отказа;

$$Y_1 = f_{Y1}(Y_{11}, Y_{12}, Y_{13}),$$

$$Y_2 = f_{Y2}(Y_{21}, Y_{22}).$$

Z_1 – воздействие в виде кратковременных отключений;

Z_2 – аварийное отключение;

Z_3 – использование возобновляемых видов энергии;

P_1 – оперативное устранение неустойчивости СТ;

P_2 – программное устранение неустойчивости СТ;

P_3 – восстановление работы;

X_{11} – ухудшение соотношения с/ш;

X_{12} – старение кабеля;

X_{13} – обрыв кабеля;

X_{14} – электромагнитные влияния;

X_{15} – энергетические проблемы;

Y_{11} – <откл. параметров от нормы>;

Y_{12} - < перегрузка трафика >;

Y_{13} - <нарушение синхронизации>;

Y_{21} - <выход из строя части СТ>;

Y_{22} - < выход из строя СТ>;

Для оценки значений лингвистических переменных X , Y , Z , P и S и их составляющих вводится единая шкала качественных терминов: Н - низкий; НС - ниже среднего; С - средний; ВС - выше среднего; В-высокий.

Для реализации комплексной модели оценки устойчивости СТ проектируется база знаний (таблицы 3.1-3.5):

Таблица 3.1

Знания о соотношении $S = f_s(S_1, S_2, S_3, S_4)$

S	S ₁			S ₂			S ₃			S ₄	
X	Н	НС	С	НС	НС	НС	С	ВС	В	В	В
Y	Н	Н	НС	НС	НС	НС	С	ВС	В	С	ВС
Z	Н	Н	Н	НС	Н	НС	С	ВС	В	ВС	ВС
P	В	В	В	НС	Н	НС	Н	ВС	В	Н	НС

Таблица 3.2

- Знания о соотношении $S_1 = f_{s_1}(X_1, X_2)$

S ₁	Н			НС			С			ВС			В		
X ₁	Н	НС	НС	Н	НС	С	С	С	С	С	ВС	С	ВС	ВС	В
X ₂	Н	НС	Н	С	С	С	Н	НС	С	С	С	ВС	ВС	В	В

Таблица 3.3

Знания о соотношении $S_2 = f_{s_2}(Y_1, Y_2)$

S ₂	Н			НС			С			ВС			В		
Y ₁	Н	Н	Н	Н	С	Н	Н	Н	С	С	С	В	В	В	В
Y ₂	Н	С	С	Н	Н	С	С	Н	Н	В	В	В	В	В	В

Таблица 3.4

Знания о соотношении $S_3 = f_{s_3}(Z_1, Z_2, Z_3)$

S ₃	Н			НС			С			ВС			В		
Z ₁	Н	Н	Н	Н	Н	С	Н	НС	С	С	С	С	С	С	С
Z ₂	Н	НС	С	НС	С	С	С	НС	С	Н	Н	Н	Н	НС	Н
Z ₃	Н	НС	Н	С	С	Н	С	С	НС	ВС	С	ВС	ВС	В	В

Таблица 3.5

Знания о соотношении $S_4 = f_{s_4}(P_1, P_2, P_3)$

S	H			HC			C			BC			B		
4															
1	H	H	H	H	H	H	C	C	C	C	C	B	B	B	B
				C	C	C						C	C	C	
2	H	H	H	H	H	H	H	C	C	B	C	C	B	B	B
		C	C	C		C	C			C			C	C	
3	H	H	H	H	H	H	H	H	C	C	B	C	C	B	B
			C		C	C	C	C			C			C	C

На основе таблиц 3.1-3.5 и используя принципы ТНМ получается система нечетких логических уравнений, позволяющие определить состояния телекоммуникационной системы:

$$\mu^{S_1}(s) = [\mu^B(x) \times \mu^C(y) \times \mu^{BC}(z) \times \mu^H(p)] \vee [\mu^B(x) \times \mu^{BC}(y) \times \mu^{BC}(z) \times \mu^{HC}(p)];$$

$$\mu^{S_2}(s) = [\mu^{HC}(x) \times \mu^C(y) \times \mu^{HC}(z) \times \mu^{HC}(p)] \vee [\mu^{HC}(x) \times \mu^{HC}(y) \times \mu^H(z) \times \mu^H(p)] \vee [\mu^C(x) \times \mu^{HC}(y) \times \mu^{HC}(z) \times \mu^{HC}(p)];$$

$$\mu^{S_3}(s) = [\mu^C(x) \times \mu^C(y) \times \mu^C(z) \times \mu^H(p)] \vee [\mu^{BC}(x) \times \mu^{BC}(y) \times \mu^{BC}(z) \times \mu^{BC}(p)] \vee [\mu^B(x) \times \mu^B(y) \times \mu^B(z) \times \mu^B(p)];$$

$$\mu^{S_4}(s) = [\mu^H(x) \times \mu^H(y) \times \mu^H(z) \times \mu^B(p)] \vee [\mu^{HC}(x) \times \mu^H(y) \times \mu^H(z) \times \mu^B(p)] \vee [\mu^C(x) \times \mu^{HC}(y) \times \mu^H(z) \times \mu^B(p)];$$

$$\mu^H(S_1) = [\mu^H(x_1) \times \mu^H(x_2)] \vee [\mu^{HC}(x_1) \times \mu^{HC}(x_2)] \vee [\mu^{HC}(x_1) \times \mu^H(x_2)];$$

$$\mu^{HC}(S_1) = [\mu^H(x_1) \times \mu^C(x_2)] \vee [\mu^{HC}(x_1) \times \mu^C(x_2)] \vee [\mu^C(x_1) \times \mu^C(x_2)];$$

$$\mu^C(S_1) = [\mu^C(x_1) \times \mu^H(x_2)] \vee [\mu^C(x_1) \times \mu^{HC}(x_2)] \vee [\mu^C(x_1) \times \mu^C(x_2)];$$

$$\mu^{BC}(S_1) = [\mu^C(x_1) \times \mu^C(x_2)] \vee [\mu^{BC}(x_1) \times \mu^C(x_2)] \vee [\mu^C(x_1) \times \mu^{BC}(x_2)];$$

$$\mu^B(S_1) = [\mu^{BC}(x_1) \times \mu^{BC}(x_2)] \vee [\mu^{BC}(x_1) \times \mu^B(x_2)] \vee [\mu^B(x_1) \times \mu^B(x_2)];$$

$$\mu^H(S_2) = [\mu^H(y_1) \times \mu^H(y_2)] \vee [\mu^H(y_1) \times \mu^C(y_2)] \vee [\mu^{HC}(y_1) \times \mu^C(y_2)];$$

$$\mu^{HC}(S_2) = [\mu^{HC}(y_1) \times \mu^{HC}(y_2)] \vee [\mu^C(y_1) \times \mu^{HC}(y_2)] \vee [\mu^{HC}(y_1) \times \mu^C(y_2)];$$

$$\mu^C(S_2) = [\mu^{HC}(y_1) \times \mu^C(y_2)] \vee [\mu^{HC}(y_1) \times \mu^{HC}(y_2)] \vee [\mu^C(y_1) \times \mu^{HC}(y_2)];$$

$$\mu^{BC}(S_2) = [\mu^C(y_1) \times \mu^{BC}(y_2)] \vee [\mu^C(y_1) \times \mu^{BC}(y_2)] \vee [\mu^{BC}(y_1) \times \mu^{BC}(y_2)];$$

$$\mu^B(S_2) = [\mu^{BC}(y_1) \times \mu^{BC}(y_2)] \vee [\mu^B(y_1) \times \mu^{BC}(y_2)] \vee [\mu^B(y_1) \times \mu^B(y_2)];$$

$$\mu^H(S_3) = [\mu^H(z_1) \times \mu^H(z_2) \times \mu^H(z_3)] \vee [\mu^H(z_1) \times \mu^{HC}(z_2) \times \mu^{HC}(z_3)] \vee [\mu^H(z_1) \times \mu^C(z_2) \times \mu^H(z_3)];$$

$$\mu^{HC}(S_3) = [\mu^H(z_1) \times \mu^{HC}(z_2) \times \mu^C(z_3)] \vee [\mu^H(z_1) \times \mu^C(z_2) \times \mu^C(z_3)] \vee [\mu^C(z_1) \times \mu^C(z_2) \times \mu^H(z_3)];$$

$$\mu^C(S_3) = [\mu^H(z_1) \times \mu^{HC}(z_2) \times \mu^C(z_3)] \vee [\mu^{HC}(z_1) \times \mu^{HC}(z_2) \times \mu^C(z_3)] \vee [\mu^C(z_1) \times \mu^C(z_2) \times \mu^{HC}(z_3)];$$

$$\mu^{BC}(S_3) = [\mu^C(z_1) \times \mu^H(z_2) \times \mu^{BC}(z_3)] \vee [\mu^C(z_1) \times \mu^H(z_2) \times \mu^C(z_3)] \vee [\mu^C(z_1) \times \mu^{HC}(z_2) \times \mu^{BC}(z_3)];$$

$$\mu^B(S_3) = [\mu^C(z_1) \times \mu^H(z_2) \times \mu^{BC}(z_3)] \vee [\mu^C(z_1) \times \mu^{HC}(z_2) \times \mu^B(z_3)] \vee [\mu^C(z_1) \times \mu^H(z_2) \times \mu^B(z_3)];$$

$$\mu^H(S_4) = [\mu^H(p_1) \times \mu^H(p_2) \times \mu^H(p_3)] \vee [\mu^H(p_1) \times \mu^{HC}(p_2) \times \mu^H(p_3)] \vee [\mu^H(p_1) \times \mu^{HC}(p_2) \times \mu^{HC}(p_3)];$$

$$\mu^{HC}(S_4) = [\mu^{HC}(p_1) \times \mu^{HC}(p_2) \times \mu^H(p_3)] \vee [\mu^{HC}(p_1) \times \mu^H(p_2) \times \mu^{HC}(p_3)] \vee [\mu^{HC}(p_1) \times \mu^{HC}(p_2) \times \mu^{HC}(p_3)];$$

$$\mu^C(S_4) = [\mu^C(p_1) \times \mu^{HC}(p_2) \times \mu^{HC}(p_3)] \vee [\mu^C(p_1) \times \mu^C(p_2) \times \mu^{HC}(p_3)] \vee [\mu^C(p_1) \times \mu^C(p_2) \times \mu^C(p_3)];$$

$$\begin{aligned} \mu^{BC}(S_4) &= [\mu^C(p_1) \times \mu^{BC}(p_2) \times \mu^C(p_3)] \vee [\mu^C(p_1) \times \mu^C(p_2) \times \\ &\mu^C(p_3)] \vee [\mu^{BC}(p_1) \times \mu^C(p_2) \times \mu^C(p_3)]; \\ \mu^B(S_4) &= [\mu^{BC}(p_1) \times \mu^{BC}(p_2) \times \mu^C(p_3)] \vee [\mu^{BC}(p_1) \times \mu^{BC}(p_2) \times \\ &\mu^{BC}(p_3)] \vee [\mu^B(p_1) \times \mu^B(p_2) \times \mu^{BC}(p_3)]. \end{aligned}$$

Для упрощения расчетов для всех лингвистических переменных комплексной модели используется единая форма функции принадлежности нечетким термам (Н, НС, С, ВС, В) с помощью универсальной подстановки [27]

$$\mu^j(u) : u = 4 \frac{x_i - \underline{x}_i}{\bar{x} - \underline{x}_i}; j = \text{Н; НС; С; ВС; В}; u \rightarrow [0,4].$$

Аналитическая функция принадлежности имеет вид;

$$\mu^j(u) = \frac{1}{1 - \left(\frac{u - b}{c} \right)^2}.$$

Интуитивно, исходя из опыта, для параметра b для термов Н, НС, С, ВС, В принимаются значения 0,1,2,3,4 соответственно и параметр c для всех термов принимается равным 0,9 [17]. Предлагаемые нечетко-логические модели позволяют оценить устойчивость СТ и реализуются по следующему алгоритму:

1. Фиксируются значения параметров состояния:

$$U^* = \{x_1^*, x_2^*, y_1^*, y_2^*, z_1^*, z_2^*, z_3^*, p_1^*, p_2^*, p_3^*\}.$$

2. Определяются значения функции принадлежности (ФП) при фиксированных значениях U^* .

3. На основе системы нечетко-логических уравнений вычисляются значения ФП для состояний S_1, S_2, S_3, S_4 .

4. Ранжировав значения ФП по состояниям в порядке убывания определяются конкретные состояния из множеств возможных $S = \{S_1, S_2, S_3, S_4\}$.

Интеграция нечетко-множественного подхода в процесс оценки устойчивости функционирования СТ представляется перспективной в плане представления СТ в целом, в рамках единой комплексной модели СТ.

В общем виде требуется построить модель, основанную на нечетких правилах вывода:

$$\bigcup_{p=1}^{k_j} \left(\bigcap_{i=1}^n x_i = a_{i,jp} - \text{с весом } w_{jp} \right) \rightarrow f_s \equiv y_j = b_{m0} + b_{m1}x_1^j + \dots + b_{mn}x_n^j.$$

В процессе построения модели нужно найти такие значения коэффициентов правил

$$B = (b_{ij}), i = \overline{1, m}, j = \overline{0, n}$$

при которых достигается минимум следующего выражения (3.1). Как находится минимум выражения (3.1) с использованием генетического алгоритма с искусственным отбором подробно рассмотрены в главе 3.

3.4.2. Решение задачи маршрутизации с использованием генетического алгоритма

Маршрутизация является одной из важных задач для телекоммуникационных сетей различного назначения. Задачи, связанные с выбором маршрута, планированием работы средств связи и т. п., относятся к классу сложных комбинаторно-оптимизационных задач, как правило, не имеющих простых аналитических решений. Кроме того, сложность необходимых вычислений экспоненциально возрастает при увеличении количества узлов в сети. Поэтому в настоящее время широко применяют различные эвристические алгоритмы и процедуры, полученные путем творческого поиска, интуиции и опыта исследователя. Альтернативой существующим методам решения задач маршрутизации является использование

генетических алгоритмов, которые позволяют при значительном снижении временных затрат получить хорошие субоптимальные решения.

Задача маршрутизации относится к классу *NP*-трудных задач.

В теории алгоритмов *NP*-полные задачи — это класс задач, лежащих в классе *NP* (то есть для которых пока не найдено быстрых алгоритмов решения, но проверка того, является ли данное решение правильным, проходит быстро), к которым сводятся все задачи класса *NP*.

Телекоммуникационная сеть состоит из узлового сетевого оборудования и каналов связи. В качестве сетевого оборудования могут выступать маршрутизаторы, антенны, коммутаторы и другие устройства. Каналы связи – это кабели или, в случае радиорелейной сети, направление в пространстве [28].

При выборе маршрута трафика учитываются следующие параметры: стоимость пропускания трафика по каналу, величина полосы пропускания, величина временной задержки, коэффициент надёжности, а также количество промежуточных сетевых узлов. Для обеспечения требуемого качества обслуживания может применяться технология разделения трафика. Идея метода состоит в том, чтобы расщепить трафик на отдельные составляющие (в зависимости от типа сервиса) и осуществить его транспортировку по сетям связи, удовлетворяющим требованиям сервиса к ресурсам. Этот метод часто применяется для маршрутизации голосовых пакетов на магистральных сетях.

Метод разделения трафика является одним из лучших методов решения задачи маршрутизации с технической точки зрения.

Заметим, однако, что при наличии в сети нескольких альтернативных маршрутов равной стоимости (метрики), трафик

делится между ними, и нагрузка на маршрутизаторы и каналы связи распределяется более сбалансировано. Но когда стоимость альтернативных маршрутов даже незначительно хуже, чем у кратчайшего маршрута, этот инструмент не работает.

Вторым недостатком маршрутизации IP является то, что маршруты рассчитываются с учетом локальной оптимизации. В рамках всей сети такой выбор может оказаться неоптимальным. Чтобы оптимизировать использование ресурсов в масштабах всей сети, решение о маршрутизации должно приниматься с учетом назначения всей сети и общего представления о ней.

Третьим недостатком традиционных методов маршрутизации трафика в сетях IP является то, что пути выбираются без учета текущей загрузки ресурсов сети. Если кратчайший путь уже перегружен, то пакеты все равно будут посылаться по нему. Современные протоколы маршрутизации при расчёте метрики учитывают такие параметры сети, как:

- Число маршрутизаторов, входящих в маршрут,
- Пропускная ширина канала
- Время задержек
- Процент потерянных пакетов и др.

Например, протокол IGRP (Interior Gateway Routing Protocol) использует следующую формулу для подсчёта метрики, на основании которой выбирается маршрут:

$$m = \left[\frac{K_1}{b \cdot (1 - o)} + (K_2 \cdot d) \right] \cdot r,$$

где

d – коэффициент, характеризующий временные задержки при движении пути по маршруту пакетов,

b – ширина пропускания канала в самом узком сегменте пути,

α – коэффициент, характеризующий загрузку канала

r – коэффициент надежности маршрута

K_1, K_2 – константы

Вид функции и значения коэффициентов K_1, K_2 заданы априори. Этот протокол учитывает множество параметров сети, необходимые для расчёта пути: полоса пропускания, задержки, текущая загрузка. Но он неэффективен для трафика разных типа сервисов, когда сам вид функции для расчёта метрики и константы зависят от конкретного типа сервиса. Помимо технической стороны вопроса, этот протокол никак не учитывает экономическую составляющую при выборе маршрута. Как уже говорилось выше, каждое соединение имеет эксплуатационную стоимость и без учета этого параметра при выборе маршрута нельзя построить экономически оптимальную сеть связи.

В настоящей работе предлагается многокритериальный подход, который решает задачу маршрутизации в мультисервисных сетях связи и который учитывает не только технические требования к ресурсам, но и экономическую составляющую.

Многокритериальный анализ имеет ряд преимуществ перед однопараметрической оптимизацией и, поэтому, в последнее время широко применяется в разных технических задачах (см., например [28,29]). На сегодняшний день существуют лишь несколько попыток применения многокритериального анализа в области телекоммуникации. С появлением всё большего числа сервисов (услуг) число аспектов или критериев, которые должны учитываться при решении задач в современных телекоммуникационных сетях, увеличивается и, тем самым, возникает необходимость применения многокритериального анализа [29,30].

Потребность количественного ранжирования частных критериев и неопределенность при их описании в задачах

многокритериальной оптимизации объективно являются источниками субъективизма, неопределенности. Необходимость использования информации качественного характера признается многими исследователями, и предложены различные пути формализации и решения этой проблемы

Теория нечетких множеств, особенно ее концептуальная основа и математический аппарат для работы с объектами лингвистической природы, оказались плодотворными, эффективными средствами постановки и решения задач многокритериальной оптимизации при наличии неопределенностей нестатистического характера. При этом следует отметить, что существует чрезвычайно большое многообразие такого рода задач, и поэтому не существует единой универсальной методики их решения [30].

Задача многокритериальной оптимизации имеет следующий вид:

$$f(x) = [f_1(x), f_2(x), \dots, f_q(x)]^T \rightarrow \min, \quad x \in X, \quad (3.11)$$

где
$$f_k(x) = \sum_{j=1}^n c_{kj} x_j,$$

$$k \in Q = \{1, 2, \dots, q\}, \quad X = \{x \in R^n \mid Ax \geq b, x \geq 0\}, \quad \left(\sum_{j=1}^n a_{ij} x_j \leq b_i, i = \overline{1, m} \right).$$

Задача маршрутизации может быть сформулирована как целочисленная введением булевых переменных $x_{ij} = 1$, если маршрут включает переезд из точки i непосредственно в точку j и $x_{ij} = 0$ в противном случае. Тогда можно задать математическую модель задачи, то есть записать целевую функцию и систему ограничений

$$f(x) = [f_1(x), f_2(x), \dots, f_q(x)]^T \rightarrow \min, \quad x \in X, \quad (3.12)$$

где
$$f_k(x) = \sum_{j=1}^n c_{kj} x_{ij},$$

$$\sum_{i=1}^n x_{ij} = 1 \quad j = \overline{1, n}, \quad (3.13)$$

$$\sum_{j=1}^n x_{ij} = 1 \quad i = \overline{1, n}, \quad (3.14)$$

$$x_{ij} \geq 0 \quad i, j = \overline{1, n}. \quad (3.15)$$

Условия (3.13)–(3.15) в совокупности обеспечивают, что каждая переменная x_{ij} равна или нулю, или единице. Однако этих ограничений не достаточно для постановки задачи маршрутизации, так как они не исключают решения, где вместо простого цикла, проходящего через n вершин, отыскиваются 2 и более отдельных цикла (подцикла), проходящего через меньшее число вершин. Поэтому задача, описанная уравнениями (3.13)–(3.15) должна быть дополнена ограничениями, обеспечивающими связность искомого цикла.

Для того, чтобы исключить при постановке задачи все возможные подциклы в систему ограничений задачи включают следующее ограничение:

$$U_i - U_j + n \cdot X_{ij} \leq n - 1, \text{ где } U_i \equiv \sum_{j=1}^n x_{ij}, U_j = \sum_{i=1}^n x_{ij}, i = \overline{2, n}, j = \overline{2, n} \text{ и } i \neq j.$$

Методы решения задачи маршрутизации различны.

К эвристическим методам решения задачи маршрутизации следует отнести «жадный» алгоритм, на каждом шаге выбирающий ребро наименьшей стоимости из множества рёбер, не нарушающих корректности решения. Эти методы имеют большую погрешность. Метод перебора прост, но только лишь при небольшом количестве итераций.

Решить задачу маршрутизации можно с помощью алгоритма Крускала. Также нам могут помочь алгоритмы Борувки и Прима, так называемые «жадные алгоритмы» Эти методы ускоряют разработку алгоритма по сравнению с методом полного перебора, однако не

всегда дают оптимальное решение.

Генетический алгоритм — это алгоритм, который позволяет найти удовлетворительное решение к аналитически неразрешимым проблемам через последовательный подбор и комбинирование искомым параметров с использованием механизмов, напоминающих биологическую эволюцию. Является разновидностью эволюционных вычислений. Отличительной особенностью генетического алгоритма является акцент на использование оператора «кроссовера», который производит операцию, роль которой аналогична роли скрещивания в живой природе.

Задача кодируется таким образом, чтобы её решение могло быть представлено в виде вектора («хромосома»). Случайным образом создаётся некоторое количество начальных векторов («начальная популяция»). Они оцениваются с использованием «функции приспособленности», в результате чего каждому вектору присваивается определённое значение («приспособленность»), которое определяет вероятность выживания организма, представленного данным вектором. После этого с использованием полученных значений приспособленности выбираются вектора (селекция), допущенные к «скрещиванию». К этим векторам применяются «генетические операторы» (в большинстве случаев «скрещивание» - crossover и «мутация» - mutation), создавая таким образом следующее «поколение». Хромосомы следующего поколения также оцениваются, затем производится селекция, применяются генетические операторы и т.д. Так моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (поколений), пока не будет выполнен критерий останова алгоритма. Таким критерием может быть: нахождение глобального, либо субоптимального решения; исчерпание числа поколений, отпущенных

на эволюцию; исчерпание времени, отпущенного на эволюцию. Генетические алгоритмы служат, главным образом, для поиска решений в очень больших, сложных пространствах поиска.

Таким образом, можно выделить следующие этапы генетического алгоритма:

- создание начальной популяции;
- вычисление функций полезности для особей популяции (оценивание);
- выбор индивидов из текущей популяции (селекция);
- скрещивание и/или мутация;
- вычисление функций полезности для всех особей;
- формирование нового поколения;
- если условия совпали, то решение найдено (конец цикла), если нет, то цикл повторяется.

Основная идея генетического алгоритма состоит в преобразовании задачу (3.12) – (3.15) в последовательность задач безусловной минимизации

$$f_k(x) = f(x) + \frac{1}{k} \left(\sum_{i=1}^n x_{ij} + \sum_{j=1}^m x_{ij} - 2 \right) \rightarrow \min, \quad k = 1, 2, \dots$$

Переформулируем задачу оптимизации как задачу нахождения минимума некоторой функции $f(x_1, x_2, \dots, x_n)$, называемой *функцией приспособленности* (fitness function). Она должна принимать неотрицательные значения на ограниченной области определения (для того, чтобы мы могли для каждой особи считать её приспособленность, которая не может быть отрицательной), при этом совершенно не требуются непрерывность и дифференцируемость.

Каждый параметр функции приспособленности кодируется строкой битов.

Особью будет называться строка, являющаяся конкатенацией строк упорядоченного набора параметров:

$$\begin{array}{ccccccc} 1010 & 10110 & 101 & \dots & 10101 \\ | x_1 | & | x_2 | & | x_3 | & | \dots | & | x_n | \end{array}$$

Универсальность генетического алгоритма заключается в том, что от конкретной задачи зависят только такие параметры, как функция приспособленности и кодирование решений. Остальные шаги для всех задач производятся одинаково.

Генетические алгоритмы оперируют совокупностью особей (популяцией), которые представляют собой строки, кодирующие одно из решений задачи. Этим генетический алгоритм отличается от большинства других алгоритмов оптимизации, которые оперируют лишь с одним решением, улучшая его.

С помощью функции приспособленности среди всех особей популяции выделяют:

- наиболее приспособленные (более подходящие решения), которые получают возможность скрещиваться и давать потомство
- наихудшие (плохие решения), которые удаляются из популяции и не дают потомства

Таким образом, приспособленность нового поколения в среднем выше предыдущего.

В классическом генетическом алгоритме:

- начальная популяция формируется случайным образом
- размер популяции (количество особей N) фиксируется и не изменяется в течение работы всего алгоритма
- каждая особь генерируется как случайная L -битная строка, где L -длина кодировки особи
- длина кодировки для всех особей одинакова

Разработана программа реализации генетического алгоритма в соответствии с выданным вариантом.

В интерфейсе программы предусмотреть возможность ввода параметров:

- объём популяции;
- число поколений;
- коэффициент скрещивания;
- коэффициент мутации;
- для дифференциального кроссовера коэффициенты k, c ;
- для задачи маршрутизации ввод [3..2000] числа вершин и их расстановку вручную и автоматически;
- для биологической задачи возможность ввода названий характеристик [10.15], их значений [4. .40], значимости и веса [0.1] каждой характеристики.

Результаты работы программы должны включать:

- на каждом шаге отображать номер поколения и лучшее значение фитнес-функции в этом поколении;
- лучшее значение фитнес-функции за все поколения и соответствующую ей структуру особи;
- для биологической задачи и задачи оптимизации функции график зависимости значения целевой функции от номера поколения;
- для задачи маршрутизации на каждом поколении графически отображать лучший маршрут.

Интерфейс программы включает характеристики генетического алгоритма в соответствии с вариантом, сведения о разработчике, краткую справку.

- Тип задачи – маршрутизация.
- Выбор пары – панмиксия.
- Кроссовер – двухточечный.

- Мутация – перестановка.
- Отбор – элитный.
- Критерий останова - количество поколений.

Анализируя полученные результаты моделирования приходим к выводу, что оптимальным количеством маршрутов можно считать 200, число поколений, нет необходимости повторять алгоритм больше 500 раз (поколений), чтобы получить хороший результат. Также на значение фитнес-функции влияет коэффициент скрещивания: оптимальный коэффициент скрещивания - 1, коэффициент мутации также играет большую роль в моделировании генетического алгоритма, оптимальный коэффициент мутации - 0,1.

Для того, чтобы открыть программу необходимо мышью дважды кликнуть по файлу “Marshrut. exe”. На экране монитора появится главное окно программы, как показано на рис. 3.1-3.2.

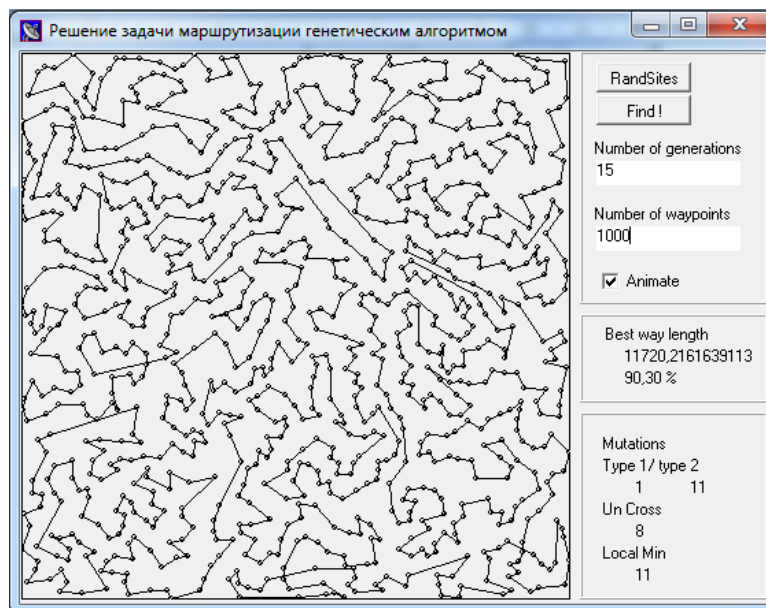


Рис.3.1. Решение задачи маршрутизации для 1000вершин

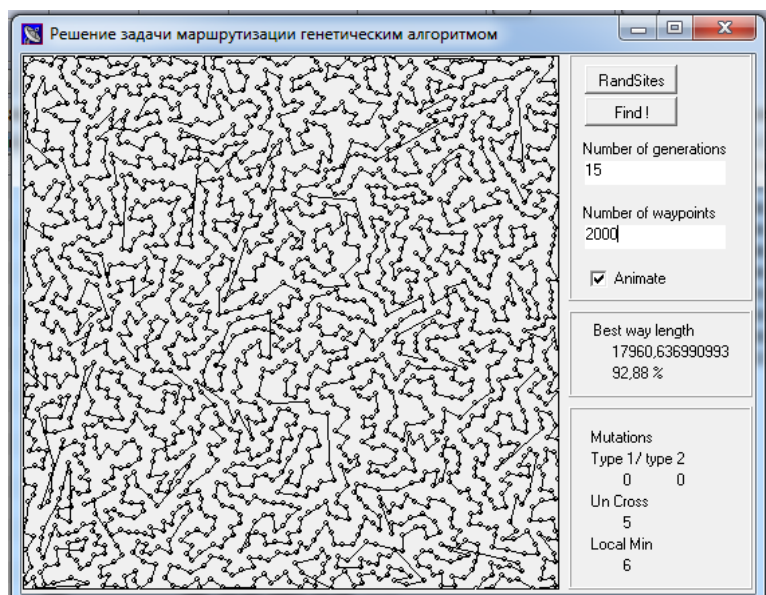


Рис.3.2. Решение задачи маршрутизации для 2000вершин

4.3. Распознавания текстовой информации

Задача распознавания текстовой информации при переводе печатного и рукописного текста в машинные коды является одной из важнейших составляющих проектов, имеющих целью автоматизацию документооборота. Вместе с тем эта задача является одной из наиболее сложных и наукоемких в области автоматического анализа изображений. Даже человек, читающий рукописный текст, в отрыве от контекста делает в среднем 4% ошибок. Что касается систем считывания печатных документов, то здесь сложность заключается в том, что в ответственных приложениях, таких как, например, автоматизация ввода паспортно-визовой информации, необходимо обеспечить высокую надежность распознавания (более 98-99%) даже при плохом качестве печати и оцифровки исходного текста.

В последние десятилетия, благодаря использованию современных достижений компьютерных технологий, были развиты новые методы обработки изображений и распознавания образов, благодаря чему стало возможным создание таких систем распознавания печатного текста, которые удовлетворяли бы основным

требованиям систем автоматизации документооборота. Тем не менее, создание каждого нового приложения в данной области по-прежнему остается творческой задачей и требует дополнительных исследований в связи со специфическими требованиями по разрешению, быстродействию, надежности распознавания и объему памяти, которыми характеризуется каждая конкретная задача разработки проблемно-ориентированной системы автоматического ввода в компьютер бумажной документации.

Различные технологии, объединенные под общим термином "распознавание символов", подразделяются на распознавание в реальном режиме времени и распознавание в пакетном режиме, каждый из которых характеризуется собственной аппаратной частью и собственными алгоритмами распознавания.

В типичной системе оптического распознавания текстов (OCR) вводимые символы читаются и оцифровываются оптическим сканером. После этого каждый символ подвергается локализации и выделению, и получившаяся матрица подвергается предобработке, т. е. сглаживанию, фильтрации и нормализации. В результате предобработки выделяются характерные признаки, после чего производится классификация. В работе описываются базовые идеи и методы, позволяющие решить указанные подзадачи, а также их модификации, используемые в современных системах распознавания.

Теория машинного зрения существует не первый день, по этому в литературе можно найти достаточно подходов и решений. Для начала перечислим некоторые из них:

1. Алгоритм скелетизации.

Это метод распознавания одинарных бинарных образов, основанный на построение скелетов этих образов и выделения из скелетов ребер и узлов. Далее по соотношению ребер, их числу и

числу узлов строится таблица соответствия образам. Так, например, скелетом круга будет один узел, скелетом буквы П - три ребра и два узла, причем ребра относятся как 2:2:1. В программировании данный метод имеет несколько возможных реализаций, подробнее информацию по методу скелетизации можно найти ниже в разделе ссылки.

2. Нейросетевые структуры.

Направление было очень популярным в 60е-70е годы, в последствии интерес к ним немного поубавился, т.к солидное число нейронов требует солидные вычислительные мощности, которые обычно отсутствуют на простеньких мобильных платформах. Однако надо иметь ввиду, что нейросети иногда дают весьма интересные результаты, за счет своей нелинейной структуры, более того некоторые нейросети способны распознавать образы инвариантные относительно поворота без какой либо внешней предобработки. Так, например, сети на основе неоконейтронов способны выделять некоторые характерные черты образов, и распознавать их как бы образы не были повернуты.

3. Инвариантные числа.

Из геометрии образов можно выделить некоторые числа, инвариантные относительно размера и поворота образов, далее можно составить таблицу соответствия этих чисел конкретному образу (почти как в алгоритме скелетизации). Примеры инвариантных чисел - число Эйлера, эксцентриситет, ориентация (в смысле расположения главной оси инерции относительно чего-нибудь тоже инвариантного).

4. Поточечное процентное сравнение с эталоном.

Здесь должна быть некоторая предобработка, для получения инвариантности относительно размера и положения, затем осуществляется сравнение с заготовленной базой эталонов

изображений - если совпадение больше чем какая-то отметка, то считаем образ распознанным.

Существует ряд существенных проблем, связанных с распознаванием рукописных и печатных символов. Наиболее важные из них следующие:

- разнообразие форм начертания символов;
- искажения изображений;
- вариации размеров и масштаба символов.

Каждый отдельный символ может быть написан различными стандартными шрифтами, например (Gothic, Elite, Courier, Orator), специальными шрифтами, использующимися в системах OCR, а также множеством нестандартных шрифтов. Кроме того, различные символы могут обладать сходными очертаниями. Например, 'U и 'V, 'S' и '5', 'Z' и '2', 'G' и '6'.

Искажения цифровых изображений символов могут быть следующих видов:

Искажения формы: разорванность строк, непропечатанность символов, изолированность отдельных точек, неплоский характер информационного носителя (например, эффект коробления), смещения символов или их частей относительно местоположения в строке; вращения с изменением наклона символов; грубым дискретом оцифровке изображений;

Кроме того, необходимо выделить радиометрические искажения: дефекты освещения, тени, блики, неравномерный фон, ошибки при сканировании или при съемке видеокамерой.

Существенным является и влияние исходного масштаба печати. В принятой терминологии масштаб 10, 12 или 17 означает, что в дюйме строки помещаются 10, 12 или 17 символов. При этом,

например, символы масштаба 10 обычно крупнее и шире символа масштаба 12.

Помимо указанных проблем, система оптического распознавания текста (OCR), должна выделять на изображении текстовые области, в них выделять отдельные символы, распознавать эти символы и быть нечувствительной к способу печати (верстки) и расстоянию между строками.

Как правило, системы OCR состоят из нескольких блоков, предполагающих аппаратную или программную реализацию:

- оптический сканер;
- блок локализации и выделения элементов текста;
- блок предобработки изображений;
- блок выделения признаков;
- блок распознавания;
- блок постобработки результатов распознавания.

В результате работы оптического сканера исходный текст вводится в компьютер в виде полутонового или бинарного изображения.

В целях экономии памяти и уменьшения затрат времени на обработку информации, в системах OCR, как правило, применяется преобразование полутонового изображения в черно-белое. Такую операцию называют бинаризацией. Однако необходимо иметь в виду, что операция бинаризации может привести к ухудшению эффективности распознавания.

Программное обеспечение в системах OCR отвечает за представление данных в цифровом виде и разбиение связного текста на отдельные символы.

После разбиения символы, представленные в виде бинарных матриц, подвергаются сглаживанию, фильтрации с целью устранения

шумов, нормализации размера, а также другим преобразованиям с целью выделения признаков, используемых впоследствии для распознавания.

Распознавание символов происходит в процессе сравнения выделенных характерных признаков с эталонными признаками, отбираемыми в ходе статистического анализа результатов, полученных в процессе обучения системы.

Таким образом, смысловая или контекстная информация может быть использована как для разрешения неопределенностей, возникающих при распознавании символов, обладающих идентичными размерами, так и для корректировки слов и фраз в целом.

Для классификации символов необходимо в первую очередь создать библиотеку эталонных векторов признаков. Для этого на стадии обучения оператор или разработчик вводит в систему OCR большое количество образцов начертания символов. Для каждого образца система выделяет признаки и сохраняет их в виде соответствующего вектора признаков. Набор векторов признаков, описывающих символ, называется классом или кластером.

В процессе эксплуатации системы OCR может появиться необходимость расширить базу знаний. Для осуществления данной цели некоторые системы обладают возможностью дообучения в реальном режиме времени. Однако процесс обучения требует участия оператора и существенных затрат времени, хотя и проводятся исследования, направленные на автоматизацию процесса обучения, что в будущем позволит свести к минимуму участие в нем человека-оператора.

Задачей классификации является определение класса, которому принадлежит вектор признаков, полученный для данного символа.

Алгоритмы классификации основаны на определении степени близости набора признаков рассматриваемого символа каждому из классов. Правдоподобие получаемого результата зависит от выбранной метрики пространства признаков. К наиболее известным метрикам относится Евклидово расстояние:

$$D_j^E = \sqrt{\sum_{i=1}^N (F_{ij}^L - F_i^L)^2},$$

где F_{ij}^L - i -й признак из j -го эталонного вектора; F_i^L - i -й признак тестируемого изображения символа.

Одна из методик, позволяющих улучшить метрику сходства, основана на использовании генетического алгоритма.

При этом в процессе классификации более надежным признакам отдается больший приоритет:

$$D_j^E = \sqrt{\sum_{i=1}^N w_i (F_{ij}^L - F_i^L)^2} \rightarrow \min,$$

где w_i , - вес i -го признака.

Переформулируем задачу оптимизации как задачу нахождения минимума некоторой функции D_j^E , называемой *функцией приспособленности* (fitness function). Она должна принимать неотрицательные значения на ограниченной области определения (для того, чтобы мы могли для каждой особи считать её приспособленность, которая не может быть отрицательной), при этом совершенно не требуются непрерывность и дифференцируемость.

Каждый параметр функции приспособленности кодируется строкой битов.

Особью будет называться строка, являющаяся конкатенацией строк упорядоченного набора параметров:

$$\begin{array}{ccccccc} 1010 & 10110 & 101 & \dots & 10101 \\ | F1 | & F2 | & F3 | & \dots | & F_n | \end{array}$$

Шаг алгоритма состоит из трех стадий:

1. Генерация промежуточной популяции (*intermediate generation*) путем отбора (*selection*) текущего поколения.

2. Скрещивание (*recombination*) особей промежуточной популяции путем *кроссовера* (*crossover*), что приводит к формированию нового поколения.

Предварительно задается величина P_k – вероятность кроссинговера и вводится флажок FG с двумя состояниями «выполнять», «не выполнять». Исходное состояние FG «не выполнять». При выполнении кроссинговера последовательно просматриваются локусы выбранной пары хромосом. С вероятностью P_k «флажок» FG переходит в состояние «выполнять». Если FG перешел в состояние «выполнять», то производится обмен генами между парой хромосом в текущем локусе, далее «флажок» переходит в состояние «не выполнять», а затем осуществляется переход к следующему локусу[5].

3. Мутация нового поколения.

Алгоритм мутации реализуется следующим образом.

Последовательно выбираются хромосомы из текущей популяции. В пределах выбранной хромосомы последовательно просматриваются гены. После перехода к очередному гену, FG с вероятностью P_M переходит в состояние «выполнять». Если FG перешел в состояние «выполнять», то случайным образом ген g_n принимает одно из значений в заданном диапазоне, за исключением значения, которое ген имеет перед мутацией. Далее FG переходит в состояние «не выполнять» и выбирается следующий ген хромосомы, или следующая хромосома[7].

Такой процесс эволюции, вообще говоря, может продолжаться до бесконечности. Критерием останова может служить заданное количество поколений или *схождение (convergence)* популяции.

Схождением называется состояние популяции, когда все строки популяции находятся в области некоторого экстремума и почти одинаковы. То есть кроссовер практически никак не изменяет популяции, а мутирующие особи склонны вымирать, так как менее приспособлены. Таким образом, схождение популяции означает, что достигнуто решение близкое к оптимальному.

Была написана программа частично реализующая основные блоки автоматического распознавания букв.

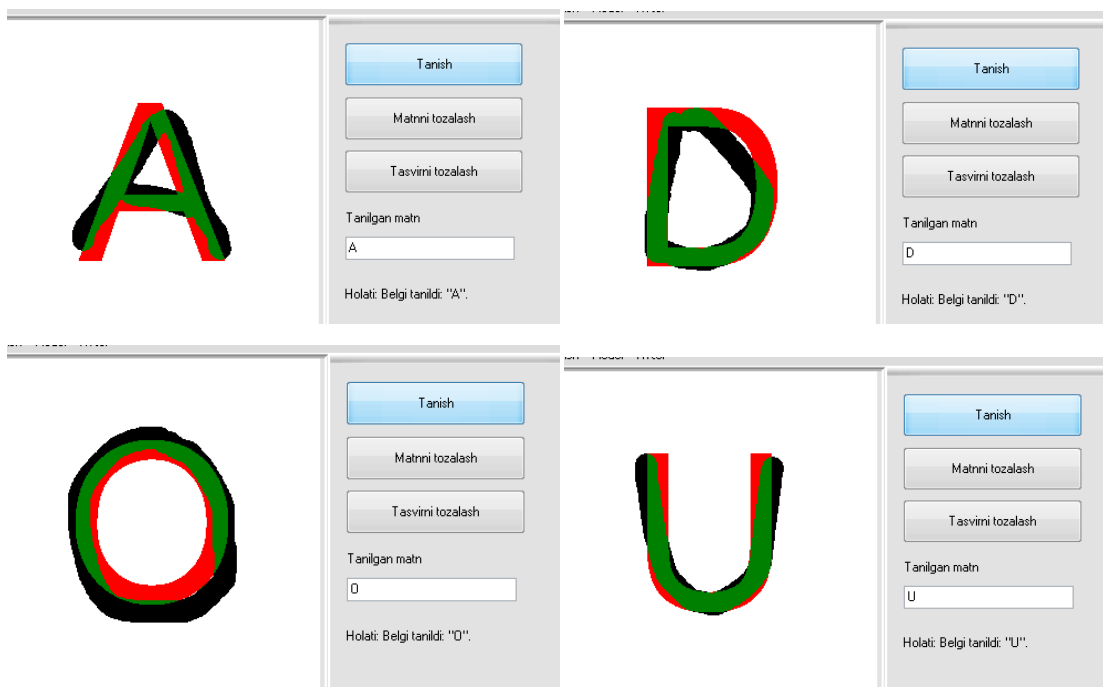


Рис.3.3. Распознавание рукописных букв

Таким образом, в настоящее время разработка и применение генетических алгоритмов является интенсивно развивающимся направлением. Благодаря универсальности вычислительной схемы, возможностям параллельной реализации, устойчивости к шуму, генетические алгоритмы находят успешное практическое применение при решении многих сложных нелинейных многомерных задач

оптимизации. В отличие от традиционных методов многокритериальной оптимизации, многие из которых часто характеризуются резким ростом вычислительных затрат при увеличении числа варьируемых параметров, генетические алгоритмы хорошо зарекомендовали себя на задачах большой размерности. В настоящее время они широко используются для решения большого круга практических задач, и область их применения постоянно расширяется.

3.5. Применение генетического алгоритма для фрактального сжатия изображения

В последнее время изображения и иллюстрации стали использоваться повсеместно. Проблема, связанная с большим объемом для их обработки и хранения, появилась при работе и на рабочих станциях, и на персональных компьютерах. Разработано большое количество различных алгоритмов архивации графики.

Майкл Барнсли и Алан Слоун нашли новый метод решения данной задачи. В методе используется принципиально новая идея - близость цветов в локальной области, а подобие разных по размеру областей изображения. Так с помощью стандартных приемов обработки изображений, таких, как выделение краев и анализ текстурных вариаций, изображение делится на сегменты и кодируется с помощью некоторого сжимающего аффинного преобразования. Восстановление изображения происходит с помощью многократного применения этого аффинного преобразования.

Целью этого параграфа является построение метода фрактального сжатия для статических изображений с использованием генетических алгоритмов (ГА). В работе рассматриваются основные

принципы метода, его обоснование, несколько алгоритмов его реализации и модификация генетического алгоритма в применении к задаче поиска кодирующего преобразования.

В начале работы излагаются общие принципы и понятия, необходимые для описания некоторых методов сжатия изображений, приводятся несколько методов кодирования цифровой информации и схемы их реализации. Далее, описывается более подробно метод фрактального сжатия и его математическое обоснование. В заключении изложены результаты вычислительного эксперимента для задачи фрактального сжатия с помощью ГА.

3.5.1. Представление изображений конечным объемом данных

Компьютерное изображение в его цифровом представлении является набором значений интенсивностей светового потока, распределенных по конечной площади, имеющей обычно прямоугольную форму.

Для простоты рассмотрим сначала монохромные изображения. Тогда интенсивность излучаемой световой энергии с единицы поверхности в точке с координатами (ξ, η) изображения можно представить некоторым числом $B(\xi, \eta)$. Единичный элемент изображения, характеризуемый определенным значением (ξ, η) , называется пикселем, а величина $z = f(\xi, \eta)$ - яркостью.

Статистическая и визуальная избыточность изображений

Поток данных об изображении имеет существенное количество излишней информации, которая может быть устранена практически без заметных для глаза искажений.

Существует два типа избыточности.

- Статистическая избыточность, связанная с корреляцией и предсказуемостью данных. Эта избыточность может быть устранена

без потери информации, исходные данные при этом могут быть полностью восстановлены.

- Визуальная (субъективная) избыточность, которую можно устранить с частичной потерей данных, мало влияющих на качество воспроизводимых изображений; это - информация, которую можно изъять из изображения, не нарушая визуально воспринимаемое качество изображений.

3.5.2. Статистическая избыточность изображений

Пусть имеется дискретизированное $M \times N$ пикселей и квантованное с точностью K бит на пиксель монохромное изображение. Следовательно, для хранения этого изображения необходимо $M \times N \times K$ бит информации.

Если предположить, что квантованные значения яркости не равновероятны, то уменьшение информации возможно путем изменения количества бит информации для кодирования пикселей: более вероятные кодируются словами с меньшим количеством бит, менее вероятные - с большим. Этот метод называется кодированием словами переменной длины или энтропийным кодированием.

Пусть квантованный уровень яркости z имеет вероятность $P(z)$ и ему присваивается слово - код длины $L(z)$ бит. Тогда средняя длина кода для всего изображения составит $\hat{L} = \sum_b L(z) \cdot P(z)$ бит на пиксель.

Нижняя граница для \hat{L} определяется информационной теоремой и называется энтропией случайной величины:

$$H(f) = - \sum_z P(z) \cdot \log_2 P(z) \leq \hat{L}.$$

Таким образом, энтропия - это мера количества информации, которую несет случайная величина уровня яркости z .

$H(z) \geq 0$, поскольку $P(z) \in [0,1]$. Из формулы $H(f)$ вытекает, что чем более неравномерно распределение $P(z)$, тем меньше энтропия и тем эффективнее может быть энтропийное кодирование.

Наиболее известные методы эффективного кодирования символов основаны на знании частоты каждого символа присутствующего в сообщении. Зная эти частоты, строят таблицу кодов, обладающую следующими свойствами:

- различные коды могут иметь различное количество бит
- коды символов с большей частотой встречаемости, имеют больше бит, чем коды символов с меньшей частотой
- хотя коды имеют различную битовую длину, они могут быть восстановлены единственным образом.

Этими свойствами обладает известный алгоритм Хаффмана [6].

3.5.3. Визуальная избыточность изображений

Устранение визуальной избыточности изображений является основным резервом сокращения передаваемой информации. Для оптимизации процесса кодирования с точки зрения обеспечения передачи наименьшего объема информации необходимо, с одной стороны, не передавать избыточную информацию, а, с другой, - не допустить чрезмерной потери качества изображения.

До сих пор не существует простой и адекватной модели визуального восприятия изображений, пригодной для оптимизации их кодирования [5].

3.5.4. О кодировании цветных изображений

Большинство цветных растровых изображений для сохранения деталей цвета пикселя применяют систему цветов *RGB*, прежде всего потому, что эта система используется для изображения цветов монитором компьютера. Для точного сжатия изображения с цветами *RGB* необходимо все три компоненты сжимать с одинаковым уровнем

точности. Комбинация этих трех компонент определяет относительную яркость пикселя, а также его цвет. Поэтому изменение любого из трех значений скажется и на яркости, и на цвете пикселя.

Так как глаз человека более чувствителен к изменению яркости, чем к изменению цвета, часто возникает желание преобразовать систему цветов *RGB* в такую систему, где информация о яркости пикселях запоминалась бы отдельно от информации о цвете. Общеупотребительной системой цветов, которая хранит яркость пикселя в виде отдельной компоненты данных о его цвете, является система *HSB* (тон, насыщенность, яркость) и *YUV* (*Y* - компонента яркости; *U* и *V* хранят характеристики цвета).

3.5.5. Обзор некоторых алгоритмов сжатия с потерями

Метод усеченного блочного кодирования (УБК)

Название метода отражает тот факт, что изображение разбивается на небольшие прямоугольные куски одинакового размера, называемые блоками. Этот метод в отличие от большинства других подстраивает параметры кодирования не под некоторую усредненную характеристику всего изображения, а под локальные особенности в пределах каждого блока. Это позволяет сохранить мелкие детали изображений. Метод не приводит к размыванию границ, что характерно для некоторых других алгоритмов. Метод УБК сопоставим с большинством других методов по эффективности сжатия данных и по объему вычислений, требуемых для кодирования, но не имеет конкурентов по простоте декодирования.

Базовый алгоритм УБК строится следующим образом. Изображение, представленное $M \times N$ - матрицей $\|b_{ij}\|$ яркостей пикселей, разбивается на небольшие прямоугольные блоки $m \times n$ элементов. Каждый такой блок обрабатывается независимо от других, поэтому опишем алгоритм обработки одного блока.

Обработка блока начинается с вычисления порога и двух уровней квантования (описанных ниже), затем проводится квантование блока на два уровня, после чего следует упаковка проквантованного блока. Для определения уровней квантования сначала вычисляются два первых выборочных момента - среднее значение C и средний квадрат E :

$$C = \frac{1}{m \times n} \sum_i \sum_j b_{ij}, \quad E = \frac{1}{m \times n} \sum_i \sum_j b_{ij}^2,$$

(где суммируются элементы изображения в пределах блока) и дисперсия

$$\sigma^2 = E - C^2.$$

Пороговая величина квантователя d полагается равной среднему C . Верхний a и нижний b уровни квантования вычисляются по следующим формулам:

$$a = C - \sigma \sqrt{q/(p-q)}, \quad b = C + \sigma \sqrt{(p-q)/q},$$

где $p = m \times n$ - число элементов блока, q - число элементов блока, превышающих порог d .

Квантование проводится по правилу:

$$s_{ij} = \begin{cases} a, & \text{если } b_{ij} < d \\ b, & \text{если } b_{ij} \geq d \end{cases},$$

где s_{ij} - элементы изображения после квантования.

После квантования получается блок, содержащий только уровни a и b . Нетрудно показать, что среднее значение и средний квадрат исходного и проквантованного блоков совпадают. Практически для удобства последующей упаковки вместо a записывается нуль, вместо b - единица. Уровни a и b записываются отдельно.

Упаковка состоит в том, что блок, содержащий только нули и единицы, интерпретируется как двоичное число, имеющее $m \times n$

разрядов. Восстановление закодированного изображения также проводится поблочно и состоит в распаковке и обратной подстановке.

Из алгоритма видно, что степень сжатия непосредственно зависит от размеров блока. Наиболее удовлетворительные результаты, как по степени сжатия, так и по качеству восстановленного изображения были получены при использовании блоков размером 4×4 (см. [3]).

Описанный выше способ определения порога и уровней квантования не является единственным. Существует ряд других критериев. Важно, чтобы критерий соответствовал целям последующей обработки изображения и ее конкретным особенностям.

3.5.6. JPEG

JPEG - один из самых распространенных и достаточно мощных алгоритмов, представляет собой метод сжатия изображений, реализуемый различными способами. Работает он как на черно-белых, так и на полноцветных изображениях.

Коэффициент архивации в *JPEG* может изменяться в пределах от 2 до 200 раз. Как и у любого другого алгоритма сжатия с потерями, у *JPEG* свои особенности. Наиболее известны 'эффект Гиббса' и дробление изображения на квадраты. Первый проявляется около резких границ предмета, образуя вокруг своеобразный 'ореол'. Разбиение на квадраты происходит, когда задается слишком большой коэффициент сжатия для данной картинке. Тем не менее, не смотря на эти недостатки, для архивации изображений, предназначенных для просмотра человеком, он на данный момент является лучшим.

Широкое применение *JPEG* сдерживается, пожалуй, лишь тем, что он оперирует 24-битными изображениями. Поэтому для того, чтобы с приемлемым качеством посмотреть картинку на обычном мониторе в 256-цветной палитре, требуется применение

соответствующих алгоритмов и, следовательно, определенное время. В приложениях, ориентированных на придирчивого пользователя, таких, например, как игры, подобные задержки неприемлемы. Кроме того, если имеющиеся изображения, допустим, в 8-битном формате *GIF* перевести в 24-битный *JPEG*, а потом обратно в *GIF* для просмотра, то потеря качества произойдет дважды при обоих преобразованиях. Тем не менее, выигрыш в размерах архивов зачастую настолько велик (3-20 раз), а потери качества настолько малы, что хранение изображений в *JPEG* оказывается очень эффективным.

3.5.7.Преобразование цветов RGB в цвета YUV

В сжатии *JPEG* применяется система цветов *YUV*. Разделение данных *RGB* на данные *YUV* позволяет программе сжатия уделять больше внимание данным о яркости (*Y*), чем данным о цвете (*UV*). Этот процесс называется подвыборкой, так как три компонента выбираются с различной частотой. Так метод подвыборки, который называется *YUV411*, на каждую выборку цвета делает четыре выборки данных о яркости. Например, если рисунок не подвергался подвыборке, то величины *YUV* встречаются в нем с одинаковой частотой. При использовании подвыборки *YUV411* на шесть значений выборки обработанного файла приходится двенадцать значений выборки исходного файла. Таким образом, подвыборка данных сразу уменьшает размер файла изображения.

3.5.8.Метод сжатия JPEG

Процесс сжатия по схеме *JPEG* состоит из трех шагов (не считая подвыборку). Первый шаг - это запись изменения значений пикселей в виде изменения частот: как быстро меняются яркость и цвет пикселей. Второй шаг - группировка этих отдельных изменений частот по средним значениям (первый этап сжатия). И третий этап - сжатие этих

усредненных данных с помощью модифицированного алгоритма кодирования Хаффмана.

3.5.9.Изменение частоты

JPEG определяет изменение частоты данных с помощью дискретного преобразования Фурье (*ДПФ*), которое применяется для каждой пиксельной компоненты в выбранной области пикселей. Например, выбирается группа из 8×8 пикселей, и к ней применяется преобразование *ДПФ*: сначала к величинам красных компонент во всей группе, затем зеленых, и, наконец, синих.

Вместо действительных значений пикселей величины *ДПФ* хранят скорость изменения интенсивности от пикселя к пикселю. На самом деле данных о частоте получается больше, чем исходных пиксельных данных, но следующие два шага это устраняют.

3.5.10. Усреднение

После вычисления с помощью *ДПФ* значений изменения частоты эти величины усредняются в соответствии с плавающей шкалой относительной важности. Это значит, что изменения частоты, которые меньше влияют на общий вид изображения (например, быстрые изменения частоты), усредняются больше других значений. Именно на этой стадии сжатия изображения происходят потери, так как величина применяемого усреднения может регулироваться.

3.5.11.Сжатие методом Хаффмана

Окончательно усредненные данные о частоте сжимаются с помощью модифицированного алгоритма кодирования Хаффмана, который особенно эффективен для этого типа данных, так как строит таблицы кодов, базирующиеся на частоте повторения величин.

3.5.12.Фрактальное сжатие

Эта группа алгоритмов является самой перспективной и развивается наиболее бурно. Основные идеи фрактального сжатия

принадлежат Barnsley. Первые практические результаты были получены Jacquin в 1992 году. Алгоритм основывается на идее подобия между элементами изображения. Коэффициенты сжатия у фрактальных алгоритмов варьируются в пределах 2-2000 раз. Причем большие коэффициенты достигаются на реальных изображениях, что нетипично для предшествующих алгоритмов. Кроме того, при разархивации изображение можно масштабировать. Уникальная особенность этого алгоритма заключается в том, что увеличенное изображение не дробится на квадраты.

Недостатком этого алгоритма является потребность в больших вычислительных мощностях при архивации. При этом распаковка требует меньше вычислений, чем у *JPEG*. Фактически это первый существенно несимметричный алгоритм. Причем, если у всех предшествующих алгоритмов коэффициент симметричности (отношение времени архивации ко времени разархивации) не превышает 3, то у фрактального алгоритма он колеблется от 1000 до 10000.

3.5.13. Метод фрактального сжатия изображений

Пусть задано изображение $M \times N$ пикселей. Обозначим через $\Omega = (0, M] \times (0, N] \in R^2$ множество точек изображения. При этом каждому пикселю (i, j) сопоставляется площадка:

$$\{(\xi, \eta) \in \Omega, \xi \in (j-1, j], \eta \in (i-1, i]\},$$

точка $(i, j) \in \Omega$ - правый верхний угол площадки пикселя.

Основная идея фрактального сжатия состоит в представлении исходного изображения, заданного функцией $z = f(\xi, \eta)$, с помощью некоторого сжимающего отображения, неподвижная точка которого соответствует исходному изображению.

Пусть X - множество векторов из R^n , где $n = M \times N$ и i - я координата вектора соответствует одному пикселю изображения:

$x_i = f(1 + (i-1) \bmod N, \left\lfloor \frac{i}{N} \right\rfloor + 1)$. Здесь $i=1, \dots, n$, $\lfloor \cdot \rfloor$ - символ округления вниз.

Нужное нам отображение будем искать в виде: $A: X \rightarrow X$.

Для задания такого отображения разобьем Ω на множество R -блоков R_1, R_2, \dots, R_r , где $R_i \subseteq \Omega, i=1, \dots, r$ есть квадратный $B \times B$ пиксельный фрагмент изображения и $\bigcup_i R_i = \Omega$. Рассмотрим также множество D -блоков: D_1, D_2, \dots, D_d , где $D_i \subseteq \Omega$ представляющие квадратные (возможно пересекающиеся) $2B \times 2B$ пиксельные фрагменты и $\bigcup_i D_i = \Omega$.

Процесс кодирования изображения заключается в поиске двух совокупностей аффинных преобразований W_i и F_i .

$W_i: D_{j(i)} \rightarrow R_i, i=1, \dots, r$ - биекция, действие которой описывается следующим образом: пусть $(\xi', \eta') = W_i(\xi, \eta)$ имеет вид

$$\begin{bmatrix} \xi' \\ \eta' \end{bmatrix} = \begin{bmatrix} \alpha_i & \beta_i \\ \gamma_i & \delta_i \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix} + \begin{bmatrix} \varphi_i \\ \psi_i \end{bmatrix}.$$

Очевидно, существует всего восемь способов такого преобразования квадрата в квадрат.

Функционал $F_i: R_i \rightarrow R, i=1, \dots, r$ задает цветовые характеристики каждого пикселя (ξ, η) блока R_i и определяется как усредненные значения яркости пикселей прообраза (ξ, η) :

$$F_i(\xi, \eta) = s_i \cdot \frac{1}{4} \sum \left(f(\tilde{\xi}, \tilde{\eta}) : (\tilde{\xi}, \tilde{\eta}) \in W^{-1}((\xi-1, \xi] \times (\eta-1, \eta]) \cap Z^2 \right) + o_i, \quad (3.16)$$

где s_i отвечает за контрастность, а o_i за яркость.

Представим искомое отображение A , как результат действия совокупностей W_i и F_i следующим образом: пусть координаты вектора $x' = A(x)$ имеют вид $x'_k = F_{i(k)}(\xi(k), \eta(k))$, где $\xi(k) = 1 + (k-1) \bmod N$,

$\eta(k) = \left\lfloor \frac{k}{N} \right\rfloor + 1$ - координаты правого верхнего угла площадки k - го пикселя, а $i(k)$ - номер R блока, содержащего точку $(\xi(k), \eta(k))$.

3.5.14. Обоснование метода фрактального сжатия

При фрактальном кодировании отображение A подбирается таким образом, чтобы минимизировать расстояние между вектором x и его образом $A(x)$ в некоторой метрике.

В соответствии с теоремой Банаха, если A - сжимающее отображение и (X, ρ) - полное метрическое пространство с метрикой ρ , тогда последовательность $\{x_k\}$, построенная по правилу

$$x_{k+1} = A(x_k), \quad (3.17)$$

сходится для произвольной точки $x_0 \in X$ к единственной неподвижной точке $x_A \in X$ преобразования A :

$$x_A = A(x_A). \quad (3.18).$$

Поэтому для восстановления изображения необходимо запомнить лишь отображение A .

Нетрудно видеть, что если семейство отображений F_i такое, что

$$L = \max_i |s_i| < 1, \text{ то } \rho(x, y) \leq L \cdot \rho(A(x), A(y)), \quad (3.19)$$

где ρ метрика, порожденная нормой l^∞ . Таким образом, для сходимости итерационного процесса (3.17) достаточно контролировать параметр s_i , чтобы $s_i < 1 \forall i = 1, \dots, r$. Тем не менее, необходимо отметить, что условие $L < 1$ не является необходимым для сходимости процесса восстановления.

Вообще говоря x_A не всегда будет точно совпадать с образом $A(x)$, но, при выполнении некоторых условий на оператор A можно гарантировать, что x_A будет «почти» также близко к x , как и $A(x)$.

Соответствующую оценку расстояния между x и $A(x)$ дает следующая теорема (Collage theorem) [9].

Теорема: $\rho(x, x_A) \leq \frac{1}{1-L} \rho(x, A(x))$, где $L < 1$ - константа Липшица сжимающего оператора A в некоторой метрике ρ .

В случае использования евклидовой метрики константа Липшица оператора A может быть вычислена следующим образом [10]

$$L = \max_{1 \leq j \leq d} \sqrt{\frac{1}{4} \sum (s_i^2 : j(i) = j)} . \quad (3.20)$$

В предлагаемом далее алгоритме в качестве критерия близости $A(x)$ к исходному вектору изображения x используется евклидова метрика. А именно, требуется найти такое отображение A , что $\rho(A(x), x)$ минимально. Это равносильно задаче: найти для каждого R -блока $R_i, i = 1, \dots, r$ такую тройку $(W_i, D_{j(i)}, F_i)$, что

$$\sigma_{R_i}^2(W_i, D_{j(i)}, F_i) = \sum ([f(\xi, \eta) - F_i(\xi, \eta)]^2 : (\xi, \eta) \in R_i \cap Z^2) \rightarrow \min , \quad (3.21)$$

где суммирование ведется по всем пикселям блока R_i .

При решении этой задачи для любых вариантов $D_{j(i)}$ и W_i можно подобрать оптимальные параметры s_i и o_i , используя метод наименьших квадратов для линейной регрессии.

Пусть даны две последовательности из B^2 значений цветов пикселей до и после применения W_i :

$$z_1 = f(\xi_1, \eta_1), z_2 = f(\xi_2, \eta_2), \dots, z_k = f(\xi_{B^2}, \eta_{B^2}) \text{ и}$$

$$z'_1 = \frac{1}{4} \sum \left(f(\tilde{\xi}, \tilde{\eta}) : (\tilde{\xi}, \tilde{\eta}) \in W^{-1}((\xi_1 - 1, \xi_1] \times (\eta_1 - 1, \eta_1]) \cap Z^2 \right),$$

...

$$z'_k = \frac{1}{4} \sum \left(f(\tilde{\xi}, \tilde{\eta}) : (\tilde{\xi}, \tilde{\eta}) \in W^{-1}((\xi_k - 1, \xi_k] \times (\eta_k - 1, \eta_k]) \cap Z^2 \right),$$

где $(\xi_k, \eta_k) \in R_i$ и z'_k усредненные значения яркости пикселей прообраза $(\xi_k, \eta_k) \forall k = 1, \dots, B^2$.

Мы можем найти s_i и o_i для (3.16) минимизировав сумму

$$\sigma_{R_i}^2 = \frac{1}{B^2} \sum_{k=1}^{B^2} (s_i z'_k + o_i - z_k)^2. \quad (3.22)$$

Функция (3.22) имеет минимум в тех точках, в которых частные производные от $\sigma_{R_i}^2$ по параметрам s_i и o_i обращаются в нуль. В результате дифференцирования и элементарных преобразований для определения параметров получаем систему двух линейных уравнений с двумя неизвестными s_i и o_i откуда получаем, что

$$s_i = \frac{\left[B^2 \cdot \left(\sum_{k=1}^{B^2} z'_k z_k \right) - \left(\sum_{k=1}^{B^2} z'_k \right) \cdot \left(\sum_{k=1}^{B^2} z_k \right) \right]}{B^2 \cdot \sum_{k=1}^{B^2} z_k'^2 - \left(\sum_{k=1}^{B^2} z'_k \right)^2}, \quad o_i = \frac{1}{B^2} \left(\sum_{k=1}^{B^2} z_k - s_i \cdot \sum_{k=1}^{B^2} z'_k \right).$$

В частности, если $s_i = 0$, то $o_i = \frac{1}{B^2} \sum_{k=1}^{B^2} z_k$. Таким образом

среднеквадратичная ошибка преобразованного блока от его исходного состояния

$$\sigma_{R_i}^2 = \frac{1}{B^2} \left[\sum_{k=1}^{B^2} z_k^2 + s \cdot \left(s \cdot \sum_{k=1}^{B^2} z_k'^2 - 2 \cdot \sum_{k=1}^{B^2} z'_k z_k + 2 \cdot o \sum_{k=1}^{B^2} z'_k \right) + o \cdot \left(o \cdot B^2 - 2 \cdot \sum_{k=1}^{B^2} z_k \right) \right].$$

Задача поиска D-блока $D_{j(i)}$ и его аффинного отображения W_i может быть решена различными способами. Например с помощью ГА, алгоритм Фишера [10], полный перебор и т.д.

3.5.15. Алгоритмы поиска фрактальной модели изображения

Алгоритм Фишера

Идея алгоритма заключается в том, чтобы некоторым образом классифицировать D-блоки и R-блоки, а поиск близкого D-блока

производить в том же классе, к которому относится ранговая область. Делается это следующим образом.

Исходные блоки разбиваются на четыре части. Для каждой из частей подсчитывается среднее значение A_i и дисперсия V_i пикселей. Далее блоки классифицируются по следующему принципу. Определим три базовых типа блоков

$$\text{тип 1: } A_1 \geq A_2 \geq A_3 \leq A_4,$$

$$\text{тип 2: } A_1 \geq A_2 \geq A_4 \geq A_3,$$

$$\text{тип 3: } A_1 \geq A_4 \geq A_2 \geq A_3.$$

Понятно, что любой блок при помощи соответствующего аффинного преобразования квадрата в квадрат можно привести к виду, соответствующему одному из указанных типов. После того, как мы зафиксировали три основных класса, блоки классифицируются по дисперсии. Таким образом, в каждом из трех классов появляются 24 подкласса, итого 72 класса. Поиск близкого к R-блоку D-блока производится перебором в соответствующем классе.

При использовании ГА для поиска оптимальных решений каждый элемент $x \in X$ пространства оптимизации должен быть представлен как вектор $b \in B$ из N символов двоичного алфавита $A = \{0,1\}$, где $B = A^N$. Необходимо также, чтобы пространство оптимизации X состояло из конечного числа элементов.

Популяцией $\Pi = (\chi^1, \chi^2, \dots, \chi^M)$ численности M считается вектор пространства B^M , координаты которого называются генотипами особей данной популяции.

Шагом ГА является переход от текущего поколения к следующему, т.е. получение новой популяции Π_{t+1} из Π_t . В построении очередной особи новой популяции участвуют операторы кроссинговера, мутации и случайный оператор отбора, $Select: B^M \rightarrow$

$\{1, \dots, M\}$ действие которого состоит в выборе номера особи родителя при порождении очередного потомка.

Для определения ГА в необходимо задать оператор кроссинговера (скрещивания) $Cross: V \times V \rightarrow V \times V$ и оператор мутации $Mut: V \rightarrow V$.

Действие кроссинговера $(\chi', \tau') = Cross(\chi, \tau)$ заключается в выборе случайным образом некоторой позиции j , равномерно распределенной от 1 до $N-1$, после чего результат формируется в виде

$$\chi' = (\chi_1, \chi_2, \dots, \chi_j, \tau_{j+1}, \dots, \tau_N), \quad \tau' = (\tau_1, \tau_2, \dots, \tau_j, \chi_{j+1}, \dots, \chi_N).$$

Влияние кроссинговера регулируют с помощью вероятности P_{Cross} срабатывания этого оператора (в противном случае все остается без изменений).

Оператор мутации в каждой позиции аргумента с заданной вероятностью P_{mut} заменяет ее содержимое случайным элементом двоичного алфавита A , выбранный в соответствии с равномерным распределением (в противном случае все остается без изменений).

Целевая функция исходной задачи, заменяется в ГА на неотрицательную функцию пригодности генотипа $\Phi(\chi)$, где $\chi \in V$.

Процесс работы алгоритма представляет собой последовательную смену поколений, на каждом шаге которой популяция Π_{t+1} наполняется парами потомков от особей популяции Π_t по формуле

$$(\chi_k^{t+1}, \chi_{k+1}^{t+1}) = Mut(Cross(\chi_{Select(\Pi_t)}^t, \chi_{Select(\Pi_t)}^t)),$$

где $(\chi_k^{t+1}, \chi_{k+1}^{t+1})$ - особи с наименьшей пригодностью популяции Π_t . То есть индивиды извлекаются попарно из Π_t и после кроссинговера и мутации помещаются в Π_{t+1} . Изменение вероятностей мутации и кроссинговера позволяет регулировать работу ГА и настраивать его на конкретные задачи.

Модификация генетического алгоритма для задачи фрактального сжатия

Опишем схему ГА в применении к задаче фрактального сжатия. В качестве генотипа ГА удобно взять вектор, компонентами которого будут пиксельные координаты области $D_{j(i)}$ исходного изображения, определенного на тороидальной поверхности, и число кодирующее аффинное преобразование W_i . Имеется восемь способов аффинного преобразования квадрата в квадрат: поворот на четыре стороны или зеркальное отражение и поворот на четыре стороны. Следовательно, на кодировку этого преобразования достаточно трех бит. Функцию пригодности положим равной

$$\Phi = \frac{1}{1 + \sum \left([f(\xi, \eta) - F_i(\xi, \eta)]^2 : (\xi, \eta) \in R_i \cap Z^2 \right)},$$

где в нижней части под знаком суммы – евклидово расстояние между исходным и преобразованным блоком. Данная функция удовлетворяет требованиям ГА (неотрицательна) и адекватна для оператора рулеточной селекции, при которой каждый индивид $\chi^{i,t}$ популяции Π_t оказывается родителем при формировании очередной особи $\chi^{i,t+1}$ популяции Π_{t+1} с вероятностью

$$P_{select}(\chi^{i,t}) = \frac{\Phi(\chi^{i,t})}{\sum_j \Phi(\chi^{j,t})}.$$

При таком представлении хромосом, определяющих данный генотип, любой вектор пространства решений всегда допустим и имеет ненулевую пригодность.

Оператор мутации для данного алгоритма – стандартный, а оператор кроссинговера был модифицирован следующим образом:

- позиция кроссинговера может располагаться только в местах стыковки двоичного представления координат.

- выполняется только для близкородственных особей, т.е., если расстояние $\rho(\chi, \tau)$ мало, где ρ - некоторая метрика.

3.5.16. Увеличение быстродействия метода

Как уже было сказано выше, недостатком данного метода сжатия является очень большой объем вычислений для кодирования изображений. Для программ реализующих данный метод, применяются как обычные методы увеличения скорости программ: целочисленная арифметика, ассемблерная реализация и другие, так и некоторые специфические приемы: использование аффинных преобразований только одного типа, различные варианты разбиения изображения на области R_i .

Приведем схемы двух алгоритмов, которые для некоторых классов изображений могут значительно уменьшить объем вычислений. Параметрами первого алгоритма служат уровень потерь при кодировании и минимальный размер областей R_i . Этот алгоритм обеспечивает равномерное качество кодирования всего изображения. Параметром второго алгоритма является количество областей R_i , используемых для кодирования изображения, что прямо влияет на объем вычислений, но он не обеспечивает достаточной точности кодирования отдельных фрагментов изображения.

Алгоритм 1.

- Выберем допустимый уровень потерь при кодировании ϵ .
- $R_1 = \Omega$ и пометим его как необработанный фрагмент.
- Пока есть необработанный фрагмент R_i выполнять:
 1. Найти $D_{j(i)}$, W_i и F_i , которые наилучшим образом приближают R_i (на которых достигается минимум $\sigma_{R_i}^2(W_i, D_{j(i)}, F_i)$).

2. Если $\sigma_{R_i}^2(W_i, D_{j(i)}, F_i) < \epsilon$ или размер $R_i < \min$, то пометить R_i как обработанное. Иначе, разбить R_i на более мелкие фрагменты и пометить их как необработанные.

Алгоритм 2.

- Выберем максимальное число N фрагментов R_i .
- Добавим фрагмент $R_1 = \Omega$ в список преобразований и пометим его как необработанный.
- Пока есть необработанные фрагменты в списке выполнять:
 1. Для каждого необработанного фрагмента найти соответствующие $D_{j(i)}$, W_i и F_i .
 2. Найти в списке фрагмент R_i наибольшего размера и наибольшей метрикой $\sigma_{R_i}^2(W_i, D_{j(i)}, F_i)$.
 3. Если число фрагментов в списке меньше N , тогда разбить фрагмент R_i на более мелкие и занести их в список как необработанные. Вычеркнуть из списка фрагмент R_i .

Вычислительный эксперимент

Описанный метод фрактального сжатия статических изображений был запрограммирован на языке C++ с применением объектно-ориентированного подхода. Программа работает как с черно-белыми, так и с полноцветными изображениями формата *WindowsBitmap*.

Для оценки качества закодированного 8-битного изображения или одного канала цветного использовалось соотношение сигнал-шум, измеряемое в децибелах (dB) и определенное как

$$PSNR = 10 \cdot \lg \left(\frac{255^2}{\sum (f(\xi, \eta) - \tilde{f}(\xi, \eta))^2} \right),$$

где суммирование ведется по всем пикселям исходного f и приближенного \tilde{f} изображения.

Алгоритм кодирования реализован в двух вариантах.

1. Алгоритм кодирования с фиксированным размером блоков.

Алгоритм оперирует с прямоугольными областями одинакового размера. Размер областей фиксирован от начала работы алгоритма до конца.

Преимуществом данного алгоритма состоит в том, что при соответствующем выборе размеров обрабатываемых областей обеспечивается равномерное качество кодирования всего изображения. Недостатком алгоритма является малый коэффициент сжатия.

2. Алгоритм кодирования с разбиением изображения на дерево подблоков.

За основу был взят *алгоритм 1* из предыдущего параграфа. При недостаточной точности кодирования обрабатываемый фрагмент разбивается на четыре части, каждая из которых обрабатывается так, как и все остальные.

Данное разбиение эффективно с точки зрения хранения его в файле. Так любое такое разбиение можно представить в виде двоичной последовательности символов, где 1 - означает то, что блок разбит, 0 - иначе. При тестировании алгоритм показал лучшие результаты, чем его предшественник, по степени сжатия, но не всегда обеспечивает достаточной точности кодирования некоторых однотонных областей.

Степень сжатия изображения при фрактальном кодировании зависит от размеров обрабатываемых R-блоков, параметров s_i и o_i

преобразования F_i . Для черно-белых изображений она может быть вычислена по формуле:

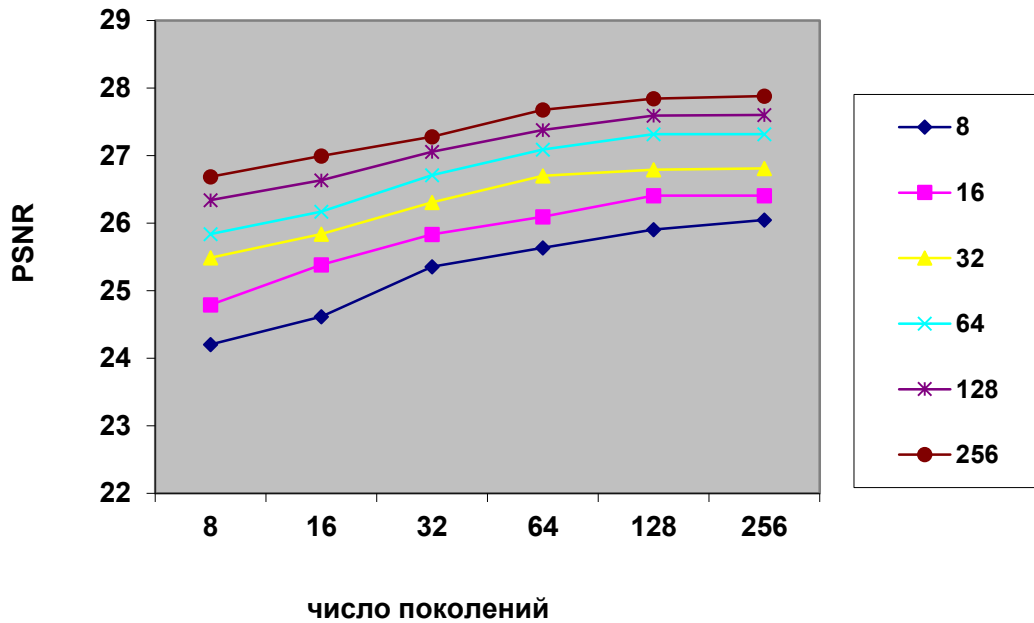
$$\frac{2^8 * \text{размер блока в пикселях}}{\text{число бит для } s + \text{число бит для } o + \lceil \log_2(8 \times \text{размер кодовой книги}) \rceil},$$

где $\lceil \rceil$ - символ округления вверх. Таким образом, мы можем получить дополнительную степень сжатия, если квантовать параметры s_i и o_i . Очевидно, что после квантования этих параметров среднеквадратическое отклонение между исходным и приближенным изображением возрастет на величину не большую, чем

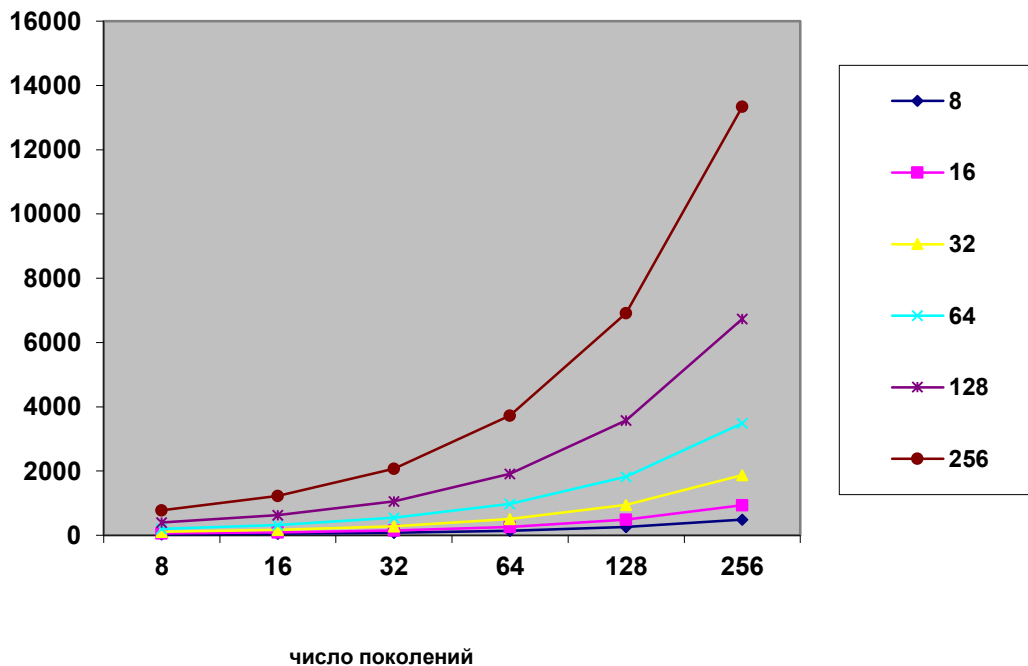
$$\left(\frac{\max_i(s_i) - \min_i(s_i)}{2^{\text{число бит для } s}} * 2^8 + \frac{\max_i(o_i) - \min_i(o_i)}{2^{\text{число бит для } o}} \right)^2.$$

Поиск фрактальной модели (для каждого R -блока соответствующий D -блок и преобразование W) изображения осуществлялся с помощью модифицированного генетического алгоритма, описанного ранее. При проведении вычислительного эксперимента ГА прекращал свою работу по истечении заданного числа поколений или при достижении определенного качества кодирования, т.е если степень приспособленности наилучшей особи будет выше некоторого заранее заданного числа. Лучшие результаты были получены при запуске ГА со следующими параметрами $P_{Mut} = 0.1$, $P_{Cross} = 0.75$. Иллюстрации полученных результатов для различных вариантов алгоритма кодирования приведены в приложении. Представленные ниже статистические данные о работе ГА были получены при проведении эксперимента для изображения и алгоритма кодирования с фиксированным размером блоков 8×8 пикселей.

Зависимость качества кодирования от числа поколений и размера популяции



Следует отметить, что использование генетических алгоритмов для задачи фрактального сжатия является одним из вариантов ухода от полного перебора. Так для изображения размером 256×256 пикселей, для каждого R-блока нужно перебрать $2^8 * 2^8 * 2^3$ комбинаций D-блоков с аффинным преобразованием W . Следующая диаграмма показывает среднее число просчитанных вариантов при работе ГА в зависимости от числа поколений и размера популяции.



- Из этих графиков видно, что предложенный ГА имеет значительные преимущества по сравнению с алгоритмом полного перебора по времени работы.

- Представляется целесообразным использование данного алгоритма в сочетании с другими эвристиками такими, как алгоритм классификации Фишера, поиск с учетом фрактальной размерности и т.д.

- Приведенный вычислительный эксперимент показывает, что ГА может быть использован для сжатия графических изображений. В дальнейшем было бы интересно сравнить эффективность ГА с другими алгоритмами поиска кодирующего преобразования.

Глава 4. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ РАЗРАБОТАННЫХ ПОДХОДОВ И ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ИХ РАБОТЫ

4.1. Улучшение качества изображений на основе применения эволюционирующей нейронной сети, вейвлет-преобразования и генетического алгоритма

Разработан двухэтапный метод улучшения качества полноцветных изображений на основе вейвлет-преобразования и генетического алгоритма. На первом этапе происходит удаление шумов. Для этой цели было выбрано вейвлет-преобразование, поскольку оно позволяет легко удалять высокочастотные компоненты. В течение второго этапа обработки происходит автоматическая настройка яркости и контраста на основе применения генетического алгоритма.

1. Применение нейроэволюционного алгоритма

В общем случае проблема попиксельной обработки изображений может быть представлена в виде проблемы поиска следующей функции преобразования $I^* = T(I, \Omega)$. Здесь I и I^* – интенсивность пикселей до и после обработки соответственно; Ω – вектор параметров, определяющих локальные и глобальные характеристики каждого пикселя обрабатываемого изображения. В данной работе предлагается трехэтапный метод улучшения качества изображений посредством аппроксимации функции T на основании ее свойств [1]. Функция T аппроксимируется с помощью искусственной нейронной сети, которая обучается попиксельной обработке изображений с использованием разработанного эволюционного алгоритма (нейроэволюционный (НЭ) подход).

Разработанный НЭ алгоритм NEvA [2] позволяет одновременно настраивать веса и структуру ИНС. Каждая ИНС закодирована в

генотипе в виде списка всех имеющихся связей. Веса связей представлены 19-битными числами, равномерно распределенными в интервале $[-26,2144; +26,2143]$ с шагом 0,0001. Родительская подпопуляция формируется с использованием отбора усечением. Используются оригинальные операторы скрещивания и мутации, учитывающие структуру ИНС. Рассматриваются нейроны с лог-сигмоидной функцией активации. Размер популяции динамически настраивается в процессе эволюционного поиска.

При обработке изображений применяется локально-адаптивный подход. Особенность подхода заключается в независимой обработке каждого пикселя изображения, исходя из имеющегося набора его локальных и глобальных характеристик. Таким образом, ИНС обучается обработке одного пикселя. При этом уменьшаются требования к объему оперативной памяти, необходимой для хранения информации об ИНС, и появляется возможность обрабатывать изображения произвольных размеров [3]. Проведенный анализ показал, что целесообразно рассматривать функцию преобразования интенсивности пикселей изображения в следующем виде $L^*(x, y) = T(L(x, y), D_{(x, y)}, m_{(x, y)})$. Здесь $L^*(x, y)$ и $L(x, y)$ – соответственно обработанное и исходное значение яркости пикселя (x, y) , $m_{(x, y)}$ и $D_{(x, y)}$ – соответственно средняя яркость и дисперсия яркости в локальной окрестности обрабатываемого пикселя. Таким образом, для рассматриваемого преобразования ИНС, аппроксимирующая функцию T , должна иметь 3 входа и 1 выход [3].

При обработке цветных изображений, сначала происходит преобразование изображений в полутоновые, затем осуществляется обработка с использованием ИНС, а после этого информация о цвете восстанавливается.

Во время обучения ИНС оцениваются в зависимости от

визуального качества обработанных изображений. Используется приближительная оценка качества изображений на основе модифицированной оценки из [4] следующего вида [3]:

$$f = \frac{N * M - \mu}{N * M} + \frac{256 - \exp(H)}{192}, \quad H = - \sum_{i=1}^{256} l_i \log l_i.$$

Здесь N и M – соответственно ширина и высота изображения, μ – количество пикселей на границах перепадов уровней яркости, l_i – доля пикселей с i -м уровнем яркости. Значение функции f рассматривается в качестве ошибки ИНС, и, таким образом, задача эволюционной настройки ИНС заключается в минимизации функции f .

Первое слагаемое в выражении для функции f необходимо для максимизации числа пикселей на границах (краях) областей перепадов уровней яркости, для обеспечения детализированности обработанного изображения. Чем больше пикселей присутствует на краях изображения, тем более контрастным оно является. Второе слагаемое в указанном выражении препятствует «вырождению» обработанного изображения в бинарное, на котором присутствуют только черные и белые пиксели.

Оценивание функционирования ИНС осуществляется по последовательности $N * M$ ее выходных сигналов. Анализ получаемых нейросетевых решений и результаты экспериментов показали необходимость использования пред- и постобработки изображений [3]. В результате проведенного исследования, для предварительной обработки была выбрана мультипликативная подстройка яркости исходного изображения [5].

ИНС обрабатывает пиксели с использованием локальных характеристик, поэтому для более эффективной обработки представляется разумным применение, в качестве алгоритма третьего этапа обработки, «глобального» алгоритма улучшения качества

изображений. Для этой цели применяется алгоритм автоматической настройки уровней яркости, реализованный во многих графических пакетах. Таким образом, предлагаемый способ трехэтапной обработки включает в себя следующие этапы:

1. Предобработка изображений с помощью мультипликативной подстройки яркости.
2. Обработка на локальном уровне с использованием ИНС.
3. Глобальная обработка с применением алгоритма автонастройки уровней яркости.

В соответствии с предлагаемым подходом, применение обученных ИНС подразумевает использование локальных средней и дисперсии яркости, поэтому время обработки изображений существенно зависит от скорости вычисления этих характеристик. Ясно, что с увеличением размера окрестности вычислительная сложность также увеличивается, т.к. увеличивается количество обрабатываемых пикселей. В предположении скореллированности распределения яркости в соседних строках (т.е. рассматриваем случай достаточно «гладкого» распределения яркости на изображении) и достаточно малого размера окрестности, получены формулы для приближенных среднего $\tilde{m}_{(x,y)}$ и дисперсии $\tilde{D}_{(x,y)}$ в прямоугольной окрестности пикселя с координатами (x, y) [5].

Для подсчета числа пикселей на границах областей различной яркости, необходимого для оценивания изображения, обработанного ИНС, применяется алгоритм детектора края Собеля, описанный в [6]. В качестве критерия прекращения работы программы используется значение $f_0 = 1,5$. Длительность эволюции составляет 25 поколений. Начальный размер популяции равен 50 особям и адаптивно изменяется во время запуска алгоритма NEvA. При обучении ИНС с целью повышения скорости обучения используется изображение с

небольшими размерами (128x128 пикселей). По окончании процесса обучения в популяции выбирается лучшая ИНС.

Время обучения ИНС составляет около 70 секунд на процессоре Интел Пентиум – 4 с частотой 3 ГГц. Полученная ИНС, как правило, содержит три входных нейрона, один или несколько скрытых нейронов и один выходной нейрон. Проведена трехэтапная обработка тестовых изображений и осуществлено сравнение с алгоритмом автоматической настройки уровней яркости и результатами обработки алгоритмом Multi-Scale Retinex (NASA) [7]. Время обработки цветного изображения размером 512x512 пикселей составляет около 1 секунды.

Ниже приведен результат улучшения качества исходного изображения (рис. 4.1) после применения трехэтапной обработки на основе алгоритма NEvA (рис. 4.2). Для сравнения на рис. 3 приведен результат улучшения исходного изображения на основе применения алгоритма Multi-Scale Retinex [7]. Следует отметить сопоставимое качество улучшенных изображений (рис. 4.2, 4.3), однако предложенный в данной работе трехэтапный метод обладает меньшей вычислительной сложностью по сравнению с алгоритмом Multi-Scale Retinex.

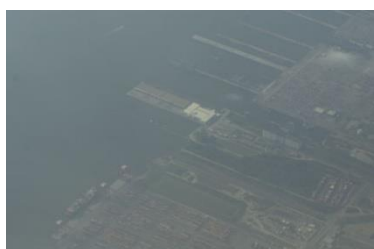


Рис. 4.1.
Исходное
изображение [7]



Рис. 4.2.
Изображение после
трехэтапной обработки
на основе алгоритма
NEvA



Рис. 4.3.
Изображение после
обработки алгоритмом
Multi-Scale
Retinex [7]

2. Применение вейвлет-преобразования и генетического алгоритма.

Дискретное вейвлет-преобразование (Discrete Wavelet Transform, DWT) разработано для быстрого вычисления вейвлет-преобразования. Оно просто в выполнении и позволяет уменьшить время вычисления и количество требуемых ресурсов. В непрерывном вейвлет-преобразовании сигналы анализируются с использованием набора базисных функций, получающихся друг из друга путем сдвига и масштабирования. В случае DWT-преобразования масштабное временное представление сигнала получается путем цифровой фильтрации. Анализируемый сигнал проходит через фильтры, на которых происходит «обрезание» определенных частот с различными масштабами.

Первоначально изображение, представленное в формате RGB переводится в цветовое пространство YUV по соответствующим формулам [9]. Компоненты U и V не подвергаются изменению во время фильтрации. Входным сигналом для системы является яркостная составляющая изображения (Y).

Вейвлет-преобразование в данном случае – это двумерное дискретное вейвлет-преобразование, основанное на материнском вейвлете «3-5», имеющем 3 высокочастотных коэффициента и 5 низкочастотных [9]. Фильтрация заключается в удалении из полученного набора вейвлет-коэффициентов высокочастотных компонент, которые и представляют собой шумы. Пользователь сам может определить порог при выполнении этой операции. После обратного вейвлет-преобразования изображение переводится в цветовое пространство RGB [9].

Для коррекции яркости и контраста применяется генетический алгоритм, позволяющий решить задачу оптимизации значений четырех

параметров ядра улучшения. В разработанном алгоритме каждая особь имеет 4 хромосомы, которые представлены целыми числами (32 бита) [9]. Методы локального улучшения основаны на применении функции, зависящей от распределения яркости в окрестности каждой точки изображения. Одним из примеров таких методов является адаптивное выравнивание гистограмм, показавшее хорошие результаты в обработке медицинских изображений. Однако этот метод использует ядро, требующее больших вычислительных затрат. Ядро улучшения представляет собой преобразование, применяемое к каждой точке изображения с координатами (x, y) . Входным параметром для этого преобразования является исходная яркость точки полутонового изображения, выходное значение – итоговая яркость точки [9].

Для автоматизации процесса улучшения качества изображения необходимо определить объективный критерий для оценки качества улучшения изображения. Этот критерий используется для построения целевой функции генетического алгоритма. Критерий качества в данном алгоритме включает 3 составляющих:

1. Количество и интенсивность краевых пикселей. Для выявления этих характеристик применяется детектор края Собеля [6].
2. Мера энтропии изображения. Необходимо оценить какое количество пикселей с различным уровнем яркости присутствуют на изображении.
3. Уровень адаптации к зрению человека по яркости. Человеческое зрение наиболее восприимчиво к изображениям, среднее значение яркости на которых соответствует середине всего диапазона возможных яркостей. Поэтому в целевую функцию был введен соответствующий множитель [9].

Разработана программа, реализующая данный метод улучшения изображений. Программа была применена для обработки набора тестовых изображений. Оказалось, что данный метод хорошо проявляет себя на двух классах изображений:

1. Слишком светлые или темные изображения. Данный метод позволяет добиться осветления слишком темных участков и затемнения слишком светлых. Кроме того, он позволяет проявить объекты, которые на исходном изображении были практически неразличимы.

2. Затуманенные изображения. Применение данного метода к затуманенным или нечетким изображениям позволяет повысить их четкость, а также снять затуманенность.

Реализованный генетический алгоритм при размере популяции 20 особей находит достаточно хорошее решение за 5 поколений. При этом обработка изображения в среднем занимает 10-20 секунд (например, для изображения размером 498x326 пикселей обработка заняла 16 секунд на компьютере с процессором Pentium 4 2,4 ГГц), что существенно превосходит скорость обработки изображений, достигнутую в работе [4].

Таким образом, в данной работе предложены трехэтапный и двухэтапный методы улучшения качества цифровых изображений. В основе трехэтапного метода лежит применение нейроэволюционного алгоритма, позволяющего определять оптимальные топологию и веса связей ИНС для решения задачи улучшения качества изображений. Данный метод позволяет, используя обученную ИНС, с приемлемой скоростью обрабатывать пакеты изображений.

4.2. Построение моделей интеллектуального анализа состояний слабоформализуемых процессов

При решении практических задач для построения системы классификации, оценки и прогнозирования в условиях неопределенности, необходимую нечеткую информацию, обладающую стохастическими характеристиками, можно разделить на две части: численную (количественную) и полученную от эксперта лингвистическую (качественную) части. Большинство нечетких систем используют знания второго типа – чаще всего данные, описываемые в виде баз правил нечеткого вывода, которые объединяются в системы нечетких выводов. Алгоритмы построения нечетких логических моделей, основанных на правилах нечеткого вывода, играют основную роль при решении задач классификации, оценки и прогнозирования в условиях неопределенности входных данных.

Формирование правил нечеткого вывода при построении моделей классификации, оценки и прогнозирования состояния слабоформализуемых процессов определяют важность оптимального сокращения количества правил.

Корректное использование текущей информации об объекте в процессе моделирования, то есть определение адекватности модели, имеет важное значение. В этом плане сформулированы основные проблемы разработки моделей слабоформализуемых процессов.

Традиционные нечеткие системы обладают некоторыми недостатками, поэтому необходимо привлекать экспертов той или иной области для формирования правил и функций принадлежности. Это в свою очередь является фактором возникновения целого ряда неудобств. Адаптивные нечеткие системы (adaptive fuzzy systems)

решают данную проблему. В подобных системах в процессе обучения осуществляется настройка их параметров на основе экспериментальных данных. Процесс адаптации нечетких систем состоит из двух этапов: 1) создание лингвистических правил; 2) настройка параметров модели. Для создания нечетких правил необходимы соответствующие функции, а для принятия нечеткого вывода нужны правила.

Вопросы построения нечетких моделей классификации, оценки и прогнозирования можно выразить в качестве задачи многокритериальной оптимизации с четырьмя целевыми функциями

$$f_1(S) \rightarrow \max, f_2(S) \rightarrow \min, f_3(S) \rightarrow \min, f_4(S) = \frac{1}{2} \sum_{j=1}^M (y_j - \hat{y}_j)^2 \rightarrow \min$$

Здесь $f_1(S)$ – количество правильно классифицированных объектов с использованием множества правил S ; $f_2(S)$ – количество нечетких правил во множестве правил S ; $f_3(S)$ – общее количество элементов множества S и $f_4(S)$ – среднеквадратическая погрешность между полученными и ожидаемыми результатами модели. Таким образом, задача сводится к решению задачи многокритериальной оптимизации.

Основная задача, требующая решения, преодолевается построением логической модели, основанной на правилах нечеткого вывода, с помощью метода нечеткой кластеризации.

Отличие предложенного подхода от традиционных подходов состоит в использовании современных технологий интеллектуального анализа данных (база знаний, компоненты Soft Computing – нейронные сети, алгоритмы роя пчел) для разработки алгоритмических и программных средств построения логических моделей на основе метода нечеткой кластеризации задач классификации, оценки и прогнозирования.

Модели классификации, оценки и прогнозирования состояний слабоформализуемых процессопределяются посредством следующих правил нечеткого вывода:

$$\bigcup_{p=1}^{k_j} \left(\bigcap_{i=1}^n x_i = a_{i,jp}, \text{ с весом } w_{jp} \right) \rightarrow y_j = f_j(x_1, x_2, \dots, x_n). \quad (4.1)$$

Здесь $a_{i,jp}$ - выражает лингвистический терм переменной x_i строки jp ; w_{jp} - коэффициент веса правила jp ; $y_j = f_j(x_1, x_2, \dots, x_n)$ - нечеткое правило вывода.

Разработаны три вида нечетких моделей интеллектуального анализа состояния слабоформализуемых процессов, описываемых с помощью правил нечеткого вывода.

1. Нечеткая модель классификации, оценки и прогнозирования состояний слабоформализуемых процессов в виде вывода нелинейной связи

$$\bigcup_{p=1}^{k_j} \left(\bigcap_{i=1}^n x_i = a_{i,jp}, \text{ с весом } w_{jp} \right) \rightarrow y_j = b_{j_0} + \sum_{h=1}^H [b_{j_{(h-1)n+1}} (x_1)^h + \dots + b_{j_{hn}} (x_n)^h] \quad (4.2)$$

2. Нечеткая модель классификации, оценки и прогнозирования состояний процессов в виде вывода линейной связи

$$\bigcup_{p=1}^{k_j} \left(\bigcap_{i=1}^n x_i = a_{i,jp}, \text{ с весом } w_{jp} \right) \rightarrow y_j = b_{j_0} + b_{j_1} x_1 + \dots + b_{j_n} x_n. \quad (4.3)$$

3. Нечеткая модель классификации, оценки и прогнозирования состояний процессов в виде вывода нечетких терм

$$\bigcup_{p=1}^{k_j} \left(\bigcap_{i=1}^n x_i = a_{i,jp}, \text{ с весом } w_{jp} \right) \rightarrow y_j = r_j, j = \overline{1, M}. \quad (4.4)$$

При построении логической модели классификации, оценки и прогнозирования состояний слабоформализуемых процессов разработан алгоритм нечеткой кластеризации, состоящий из семи шагов.

При построении нечеткой модели в случае различных видов функций принадлежности осуществляется настройка параметров

модели на основе нейронных сетей и пчелиного роя, то есть решена задача обучения логической нечеткой модели. Суть обучения состоит в решении оптимизационной задачи минимизации отличий между настоящими свойствами объекта и результатами нечеткой аппроксимации.

Настройка параметров нечеткой логической модели состоит из двух этапов. На первом этапе определяются значения модели (y). На втором этапе определяется значение погрешности (E) и вычисляются значения параметров функций принадлежности.

В этом процессе с помощью функций принадлежности, дающие в результате самые высокие значения, создается модель, которая состоит из нечетких правил вывода (4.2)-(4.4). Здесь требуется найти значения коэффициентов $b_{ji}(i=0,1,2, \dots, t; j=1, \dots, m)$.

Здесь в случае, если модель линейного типа, то $t = n$, а при нелинейности модели $t = 2n$.

Значения полученных коэффициентов (4.3) считаются значениями, минимизирующими квадратическое отклонение.

Входной вектор $X_r = (x_{r,1}, x_{r,2}, \dots, x_{r,n})$ имеет следующий нечеткий выход:

$$y_r = \frac{\sum_{j=1}^m \mu_{d_j} \cdot (x_r) \cdot y_j}{\sum_{j=1}^m \mu_{d_j} \cdot (x_r)}.$$

Уровень выполнения нечеткого правила вывода j определяется с помощью выражения

$$\mu_{y_j}(x_r) = \mu_j^{k_j}(x_{r1}) \cdot \mu_j^{k_j}(x_{r2}) \cdot \dots \cdot \mu_j^{k_j}(x_{rm}).$$

Посредством выражения $\beta_{jr} = \frac{\mu_{y_j}(x_r)}{\sum_{k=1}^m \mu_{y_k}(x_r)}$ для входного вектора X_r

определяется относительный уровень выполнения нечеткого правила

вывода j .

Тогда:

а) при линейной зависимости выхода:

$$y_r = \sum_{j=1}^m \beta_r y_j = \sum_{j=1}^m (\beta_{r_j} b_{j_0} + \beta_{r_j} b_{j_1} \cdot x_{r_1} + \dots + \beta_{r_j} \cdot b_{j_n} \cdot x_{r_n}).$$

б) при нелинейной зависимости выхода:

$$y_r = \sum_{j=1}^m \beta_r y_j = \sum_{j=1}^m (\beta_{r_j} b_{j_0} + \beta_{r_j} b_{j_1} \cdot x_{r_1} + \dots + \beta_{r_j} \cdot b_{j_n} \cdot x_{r_n} + \dots + \beta_{r_j} \cdot b_{j_{(h-1)n+1}} \cdot x_{r_{(h-1)n+1}} + \dots + \beta_{r_j} \cdot b_{j_n} \cdot x_{r_n}).$$

Значения параметра β_{r_j} определяются в соответствии с видом функции принадлежности (таблица 4.1):

Таблица 4.1

Определение параметра β_{r_j} при различных видах функций принадлежности

Функции принадлежности	Значение параметра β_{r_j}
<p>Форма Гаусса:</p> $\mu(x) = \exp\left(-\left(\frac{x-c}{\sigma}\right)^2\right)$	$\beta_{r_j} = \exp\left[-\frac{1}{2} \sum_{i=1}^t \left(\frac{x_{r_i} - c_{ij}}{\sigma_{ij}}\right)^2\right] / \sum_{k=1}^m \exp\left[-\frac{1}{2} \sum_{i=1}^t \left(\frac{x_{r_i} - c_{ik}}{\sigma_{ik}}\right)^2\right].$
<p>Колоколообразная форма:</p> $\mu(x) = \frac{1}{1 + \left(\frac{x-c}{\sigma}\right)^2}$	$\beta_{r_j} = \prod_{i=1}^t \frac{1}{1 + \left(\frac{x_{r_i} - c_{ij}}{\sigma_{ij}}\right)^2} / \sum_{k=1}^m \prod_{i=1}^t \frac{1}{1 + \left(\frac{x_{r_i} - c_{ik}}{\sigma_{ik}}\right)^2}.$
<p>В форме параболы:</p> $\mu(x) = 1 - \left(\frac{x-c}{\sigma}\right)^2$	$\beta_{r_j} = \prod_{i=1}^t \left[1 - \left(\frac{x_{r_i} - c_{ij}}{\sigma_{ij}}\right)^2\right] / \sum_{k=1}^m \prod_{i=1}^t \left[1 - \left(\frac{x_{r_i} - c_{ik}}{\sigma_{ik}}\right)^2\right].$

<p>В форме треугольника</p> $\mu(x) = \begin{cases} \frac{x-a}{b-a}, & a \leq x \leq b, \\ \frac{x-c}{b-c}, & b \leq x \leq c, \\ 0, & \text{в других случаях.} \end{cases}$	$\beta_{r_j} = \begin{cases} \prod_{i=1}^t \left[\frac{x_{r_i} - a_{ij}}{b_{ij} - a_{ij}} \right] / \sum_{k=1}^m \prod_{i=1}^t \left[\frac{x_{r_i} - a_{ik}}{b_{ik} - a_{ik}} \right], & \text{если } a \leq x \leq b, \\ \prod_{i=1}^t \left[\frac{x_{r_i} - c_{ij}}{b_{ij} - c_{ij}} \right] / \sum_{k=1}^m \prod_{i=1}^t \left[\frac{x_{r_i} - c_{ik}}{b_{ik} - c_{ik}} \right], & \text{если } b \leq x \leq c. \end{cases}$
--	--

Введем следующие обозначения:

$$Y = (y_1, y_2, \dots, y_M)^T,$$

$$\hat{Y} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_M)^T,$$

$$A = \begin{bmatrix} \beta_{1,1}, \dots, \beta_{1,m}, & x_{1,1} \cdot \beta_{1,1}, \dots, x_{1,1} \cdot \beta_{1,m}, & \dots, & x_{1,n} \cdot \beta_{1,1}, \dots, x_{1,t} \cdot \beta_{1,m} \\ \vdots \\ \beta_{M,1}, \dots, \beta_{M,m}, & x_{M,1} \cdot \beta_{1,1}, \dots, x_{M,1} \cdot \beta_{1,m}, & \dots, & x_{M,n} \cdot \beta_{M,1}, \dots, x_{M,t} \cdot \beta_{M,m} \end{bmatrix}.$$

Тогда задача (4.3) сводится к матричному виду: необходимо найти вектор B , отвечающий следующим требованиям:

$$E = (Y - \hat{Y})^T \cdot (Y - \hat{Y}) \rightarrow \min. \quad (4.5)$$

Для решения задачи (4.5) используем генетический алгоритм с искусственным отбором [5-7].

Вычислительный эксперимент

Предложенный подход был апробирован при решении задачи оценивания и прогнозирования с использованием реальных данных.

Разработаны три вида модели оценки риска недополучения урожая, основанные на нечетких правилах вывода.

1. Модель оценки риска, выход которой выражается линейной зависимостью.

$$\text{Если } x_1^1 = H \text{ и } x_2^1 = H \text{ и } x_3^1 = H \text{ и } x_4^1 = C$$

То

$$r_1 = 0,33 - 0,05 \frac{\sum_{j=1}^n \mu(x_1^{1j})x_1^{1j}}{\sum_{j=1}^n \mu(x_1^{1j})} - 0,02 \frac{\sum_{j=1}^n \mu(x_2^{1j})x_2^{1j}}{\sum_{j=1}^n \mu(x_2^{1j})} - 0,21 \frac{\sum_{j=1}^n \mu(x_3^{1j})x_3^{1j}}{\sum_{j=1}^n \mu(x_3^{1j})} - 0,1 \frac{\sum_{j=1}^n \mu(x_4^{1j})x_4^{1j}}{\sum_{j=1}^n \mu(x_4^{1j})}$$

Если $x_1^2 = H$ и $x_2^2 = H$ и $x_3^2 = C$ и $x_4^2 = C$

$$\text{То } r_2 = 0,257 - 0,0393 \frac{\sum_{j=1}^n \mu(x_1^{2j})x_1^{2j}}{\sum_{j=1}^n \mu(x_1^{2j})} - 0,112 \frac{\sum_{j=1}^n \mu(x_4^{2j})x_4^{2j}}{\sum_{j=1}^n \mu(x_4^{2j})}.$$

Если $x_1^3 = H$ и $x_2^3 = C$ и $x_3^3 = H$ и $x_4^3 = C$

То

$$r_3 = 0,18 - 0,01 \frac{\sum_{j=1}^n \mu(x_1^{3j})x_1^{3j}}{\sum_{j=1}^n \mu(x_1^{3j})} - 0,07 \frac{\sum_{j=1}^n \mu(x_2^{3j})x_2^{3j}}{\sum_{j=1}^n \mu(x_2^{3j})} - 0,05 \frac{\sum_{j=1}^n \mu(x_3^{3j})x_3^{3j}}{\sum_{j=1}^n \mu(x_3^{3j})} - 0,111 \frac{\sum_{j=1}^n \mu(x_4^{3j})x_4^{3j}}{\sum_{j=1}^n \mu(x_4^{3j})}.$$

Если $x_1^4 = H$ и $x_2^4 = C$ и $x_3^4 = C$ и $x_4^4 = C$

То

$$r_4 = 0,26 - 0,02 \frac{\sum_{j=1}^n \mu(x_1^{4j})x_1^{4j}}{\sum_{j=1}^n \mu(x_1^{4j})} - 0,05 \frac{\sum_{j=1}^n \mu(x_2^{4j})x_2^{4j}}{\sum_{j=1}^n \mu(x_2^{4j})} - 0,03 \frac{\sum_{j=1}^n \mu(x_3^{4j})x_3^{4j}}{\sum_{j=1}^n \mu(x_3^{4j})} - 0,134 \frac{\sum_{j=1}^n \mu(x_4^{4j})x_4^{4j}}{\sum_{j=1}^n \mu(x_4^{4j})}.$$

Если $x_1^5 = H$ и $x_2^5 = C$ и $x_3^5 = B$ и $x_4^5 = C$

То

$$r_5 = 0,202 - 0,10 \frac{\sum_{j=1}^n \mu(x_1^{5j})x_1^{5j}}{\sum_{j=1}^n \mu(x_1^{5j})} - 0,08 \frac{\sum_{j=1}^n \mu(x_2^{5j})x_2^{5j}}{\sum_{j=1}^n \mu(x_2^{5j})} - 0,04 \frac{\sum_{j=1}^n \mu(x_3^{5j})x_3^{5j}}{\sum_{j=1}^n \mu(x_3^{5j})} - 0,12 \frac{\sum_{j=1}^n \mu(x_4^{5j})x_4^{5j}}{\sum_{j=1}^n \mu(x_4^{5j})}.$$

2. Модель оценки риска, выход которой выражается нелинейной зависимостью.

Если $x_1^1=H$ и $x_2^1=H$ и $x_3^1=B$ и $x_4^1=C$

То

$$r_1 = 0,33 - 0,05 \frac{\sum_{j=1}^n \mu(x_1^{1j})x_1^{1j}}{\sum_{j=1}^n \mu(x_1^{1j})} - 0,02 \frac{\sum_{j=1}^n \mu(x_2^{1j})x_2^{1j}}{\sum_{j=1}^n \mu(x_2^{1j})} - 0,21 \frac{\sum_{j=1}^n \mu(x_3^{1j})x_3^{1j}}{\sum_{j=1}^n \mu(x_3^{1j})} - 0,1 \frac{\sum_{j=1}^n \mu(x_4^{1j})x_4^{1j}}{\sum_{j=1}^n \mu(x_4^{1j})} +$$

$$+ 0,003 \left[\frac{\sum_{j=1}^n \mu(x_1^{1j})x_1^{1j}}{\sum_{j=1}^n \mu(x_1^{1j})} \right]^2 - 0,004 \left[\frac{\sum_{j=1}^n \mu(x_2^{1j})x_2^{1j}}{\sum_{j=1}^n \mu(x_2^{1j})} \right]^2 + 0,007 \left[\frac{\sum_{j=1}^n \mu(x_3^{1j})x_3^{1j}}{\sum_{j=1}^n \mu(x_3^{1j})} \right]^2 + 0,0011 \left[\frac{\sum_{j=1}^n \mu(x_4^{1j})x_4^{1j}}{\sum_{j=1}^n \mu(x_4^{1j})} \right]^2$$

Если $x_1^{16}=C$ и $x_2^{16}=B$ и $x_3^{16}=H$ и $x_4^{16}=C$

То

$$r_{16} = 0,184 - 0,007 \frac{\sum_{j=1}^n \mu(x_1^{16j})x_1^{16j}}{\sum_{j=1}^n \mu(x_1^{16j})} - 0,005 \frac{\sum_{j=1}^n \mu(x_2^{16j})x_2^{16j}}{\sum_{j=1}^n \mu(x_2^{16j})} - 0,003 \frac{\sum_{j=1}^n \mu(x_3^{16j})x_3^{16j}}{\sum_{j=1}^n \mu(x_3^{16j})} - 0,09 \frac{\sum_{j=1}^n \mu(x_4^{16j})x_4^{16j}}{\sum_{j=1}^n \mu(x_4^{16j})} +$$

$$+ 0,002 \left[\frac{\sum_{j=1}^n \mu(x_1^{16j})x_1^{16j}}{\sum_{j=1}^n \mu(x_1^{16j})} \right]^2 - 0,0009 \left[\frac{\sum_{j=1}^n \mu(x_2^{16j})x_2^{16j}}{\sum_{j=1}^n \mu(x_2^{16j})} \right]^2 + 0,0005 \left[\frac{\sum_{j=1}^n \mu(x_3^{16j})x_3^{16j}}{\sum_{j=1}^n \mu(x_3^{16j})} \right]^2 + 0,0015 \left[\frac{\sum_{j=1}^n \mu(x_4^{16j})x_4^{16j}}{\sum_{j=1}^n \mu(x_4^{16j})} \right]^2.$$

Если $x_1^{27}=B$ и $x_2^{27}=B$ и $x_3^{27}=C$ и $x_4^{27}=C$

То

$$r_{27} = 0,17 - 0,003 \frac{\sum_{j=1}^n \mu(x_1^{27j})x_1^{27j}}{\sum_{j=1}^n \mu(x_1^{27j})} - 0,001 \frac{\sum_{j=1}^n \mu(x_2^{27j})x_2^{27j}}{\sum_{j=1}^n \mu(x_2^{27j})} - 0,07 \frac{\sum_{j=1}^n \mu(x_3^{27j})x_3^{27j}}{\sum_{j=1}^n \mu(x_3^{27j})} - 0,09 \frac{\sum_{j=1}^n \mu(x_4^{27j})x_4^{27j}}{\sum_{j=1}^n \mu(x_4^{27j})} +$$

$$+ 0,01 \left[\frac{\sum_{j=1}^n \mu(x_1^{27j})x_1^{27j}}{\sum_{j=1}^n \mu(x_1^{27j})} \right]^2 - 0,0005 \left[\frac{\sum_{j=1}^n \mu(x_2^{27j})x_2^{27j}}{\sum_{j=1}^n \mu(x_2^{27j})} \right]^2 + 0,0002 \left[\frac{\sum_{j=1}^n \mu(x_3^{27j})x_3^{27j}}{\sum_{j=1}^n \mu(x_3^{27j})} \right]^2 + 0,0024 \left[\frac{\sum_{j=1}^n \mu(x_4^{27j})x_4^{27j}}{\sum_{j=1}^n \mu(x_4^{27j})} \right]^2.$$

3. Модель оценки риска, выход которой выражается нечетким термом.

Если $x_1^1=H$ и $x_2^1=H$ и $x_3^1=H$ и $x_4^1=H$ с весом 0.5

или $x_1^1=C$ и $x_2^1=H$ и $x_3^1=H$ и $x_4^1=H$ с весом 0.5

То $r_1=B$.

Если $x_1^2=H$ и $x_2^2=H$ и $x_3^2=H$ и $x_4^2=C$ с весом 0.33

или $x_1^2=H$ и $x_2^2=H$ и $x_3^2=H$ и $x_4^2=B$ с весом 0.33

или $x_1^2=H$ и $x_2^2=H$ и $x_3^2=C$ и $x_4^2=H$ с весом 0.33

То $r_2=BC$.

Если $x_1^3=H$ и $x_2^3=H$ и $x_3^3=H$ и $x_4^3=B$ с весом 0.33

или $x_1^3=H$ и $x_2^3=H$ и $x_3^3=C$ и $x_4^3=C$ с весом 0.33

или $x_1^3=H$ и $x_2^3=H$ и $x_3^3=C$ и $x_4^3=B$ с весом 0.33

То $r_3=C$.

Если $x_1^4=H$ и $x_2^4=B$ и $x_3^4=C$ и $x_4^4=C$ с весом 0.5

или $x_1^4=H$ и $x_2^4=B$ и $x_3^4=C$ и $x_4^4=B$ с весом 0.5

То $r_4=HC$.

Если $x_1^5=C$ и $x_2^5=B$ и $x_3^5=C$ и $x_4^5=B$ с весом 0.33

или $x_1^5=B$ и $x_2^5=B$ и $x_3^5=C$ и $x_4^5=B$ с весом 0.33

или $x_1^5=B$ и $x_2^5=B$ и $x_3^5=B$ и $x_4^5=B$ с весом 0.33

То $r_5=H$.

Функции принадлежности имеет следующий вид [1]:

$$\mu^k(x_i^j) = \frac{1}{1 + \left(\frac{x_i^j - c_k^j}{\sigma_k^j} \right)^2}$$

В работе рассматривались проблемы, связанные с обработкой неполной, неточной и недостоверной информации в системах искусственного интеллекта. Применение новых подходов на основе эволюционных методов для оптимизации нечетких баз правил позволит значительно снизить время формирования решения и повысить достоверность его принятия в интеллектуальных системах.

4.3. Методы и алгоритмы автоматического формирования нечетких систем управления

Для упрощения работы с БП при оптимизации нами был предложен следующий способ преобразования БП из вида $ifx = Atheny = B$ к целочисленному виду. Каждому терму лингвистических переменных присваивался номер, соответствующий порядку его расположения на числовой оси. Например, в случае описания переменной 5-ю термами, им будут присвоены следующие номера – «Отрицательное большое» = 0, «Отрицательное малое» = 1, «Ноль» = 2, «Положительное малое» = 3 и «Положительное большое» = 4 (см. рис. 4.4).

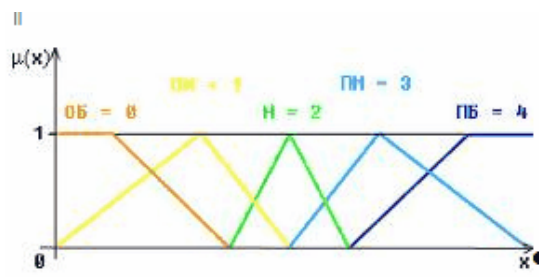


Рис. 4.4 Порядок нумерации терм лингвистических переменных

Тогда, в случае n входов и одного выхода, БП будет представлять собой n -мерный массив. Порядковые номера терм входных лингвистических переменных будут служить индексами данного массива, а элементами массива будут являться номера терм выходной переменной.

4.3.1 Генетический подход к формированию БП для нечеткой системы управления

Каждый индивид в ГА представляет собой решение – БП, содержащую все правила, необходимые для управления системой, и принадлежащую множеству всех возможных БП.

В ГА БП кодируется бинарной строкой фиксированной длины. Поскольку БП представляет собой n-мерный целочисленный массив, то ее кодировка не составляет особой сложности (см. рис.4.5). Если учесть тот факт, что БП симметрична относительно точки равновесия (точка в БП, где все входные параметры принимают нулевые значения), то длину хромосомы можно уменьшить в два раза. Тем самым мы уменьшаем время, затрачиваемое на поиск лучшего решения, и одновременно увеличиваем “качество” найденных решений.

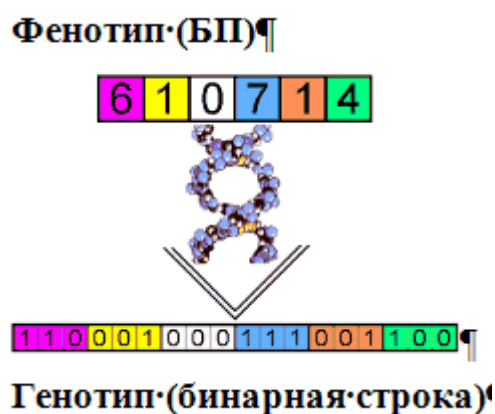


Рис. 4.5. Кодирование решений в ГА

После кодирования БП в хромосому запускается механизм обычного ГА, параметры которого были подобраны наиболее эффективным образом на основании проведенных нами исследований.

На первом этапе работы ГА происходит случайная генерация решений - БП (начальной популяции). Затем, эта популяция проходит процесс эволюции в форме естественного отбора. В каждом поколении относительно хорошие решения дают потомков, которые заменяют относительно плохие решения, исчезающие из популяции. Функция пригодности играет роль внешней среды, которая определяет хорошие и плохие решения, т.о. функция пригодности назначает каждому индивиду число, пропорциональное мере адаптации индивида к “внешней среде”.

Полученная таким образом БП достаточно хорошо справляется с управлением системой, однако, поскольку поисковое пространство дискретно и содержит порядка 7^{875} точек, из которых надо выбрать наилучшую, то оптимизационному алгоритму сложно справиться с такой задачей. Вследствие чего, в сформированной БП существуют “пробелы”, попадая в которые система теряет управляемость. Возникает необходимость более эффективного подхода к формированию базы правил.

4.3.2 Нейросетевой подход к формированию БП для нечеткой системы управления

Другим подходом к формированию БП является нейросетевой. Он заключается в интегрировании ИНС в НСУ с последующим ее обучением. Данное интегрирование осуществляется заменой базы нечетких правил в НСУ нейронной сетью. Таким образом, число входов нейронной сети равно числу входных лингвистических переменных, число выходов сети равно единице. Принцип работы такой системы изображен на рисунке 4.6.

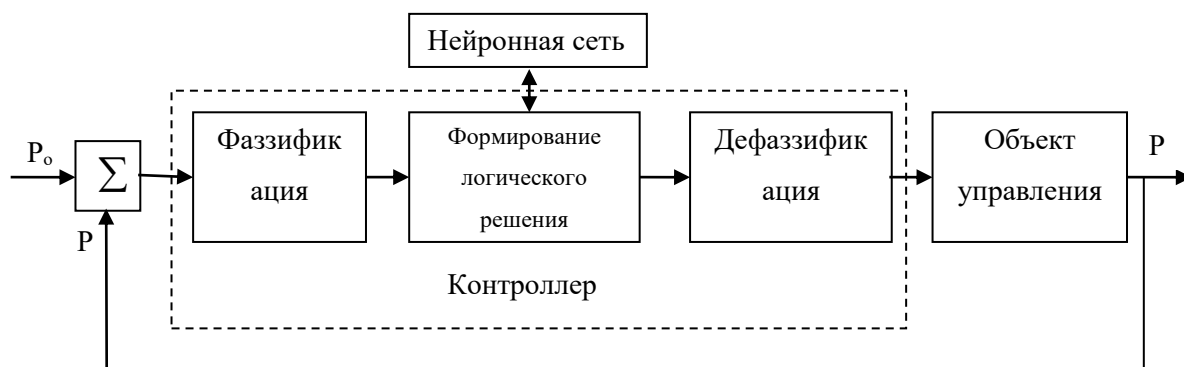


Рис. 4.6. Блок-схема работы нечеткой системы при замене Базы Правил нейронной сетью

Интегрирование ИНС в НСУ позволяет уменьшить размерность поискового пространства, поскольку, в этом случае, происходит подмена задачи оптимизации БП на задачу подбора коэффициентов синаптических связей (весов) нейронной сети, аппроксимирующей данную БП. Размер поискового пространства второй задачи в несколько раз ниже, чем первой, что соответственно уменьшает время на поиск наиболее эффективной БП. Свойство обобщения ИНС позволяет получать решения даже на те задачи, на которые система не обучалась.

Структура нейронной сети и параметры оптимизационного алгоритма были подобраны наиболее эффективным образом по результатам проведенных экспериментов. В нашей работе ИНС обучается без учителя с помощью генетического алгоритма.

По окончании обучения ИНС происходит извлечение БП из «черного ящика» сети и представление ее (БП) в понятном пользователю формате. В дальнейшем, управление системой осуществляется по БП, извлеченной из обученной ИНС, без участия самой ИНС.

4.3.3 Настройка семантики лингвистических переменных

Настройка семантики лингвистических переменных осуществляется генетическим алгоритмом. При этом оптимизационному алгоритму необходимо найти массив точек, которые представляют собой вершины термов лингвистических переменных.

Основную сложность при оптимизации термов лингвистических переменных составляет кодирование терм в бинарную строку. Это связано со следующими фактами (проблемами):

1. каждая лингвистическая переменная имеет свой диапазон изменения, соответственно, разные переменные изменяются в разных диапазонах, а для использования оптимизационного алгоритма необходимо, чтобы все переменные имели одинаковый диапазон варьирования;

2. в каждой лингвистической переменной должен соблюдаться строгий порядок расположения ее термов (к примеру, ... «отрицательное среднее», «отрицательное малое», «ноль», ...), который в ходе оптимизации не должен нарушаться;

3. термы должны покрывать весь интервал изменения переменной, т.е. между термами не должно существовать разрывов.

Данные проблемы можно обойти, если наложить на решение множество ограничений, но тогда вместо задачи безусловной оптимизации пришлось бы решать задачу условной оптимизации, что гораздо сложнее. Следовательно, необходимо разработать такой способ кодирования и декодирования термов, который позволил бы решить обозначенные выше проблемы. Первая проблема решается достаточно просто - для каждой лингвистической переменной составляется своя масштабная функция вида $f_M(x) = k \cdot (x - l_1) + l_{M1}$, в

результате применения которой диапазоны изменения переменных становятся одинаковыми. Здесь, x – исходная координата вершины терма по оси абсцисс, $k = d/D$ – коэффициент масштаба, $d = (l_2 - l_1)$ – ширина интервала изменения переменной до масштабирования, $D = (l_{M2} - l_{M1})$ – ширина интервала изменения переменной после масштабирования, l_1 и l_2 – левая и правая границы изменения переменной до масштабирования, l_{M1} и l_{M2} – левая и правая границы изменения переменной после масштабирования, $f_M(x)$ – промасштабированная абсцисса вершины. При декодировании полученного решения используется обратная к масштабной функция:

$$f_M^{-1}(x_M) = k_M \cdot (x_M - l_{M1}) + l_1,$$

где $k_M = 1/k = D/d$ – коэффициент обратного масштаба и $x_M = f_M(x)$.

Решение второй проблемы также, на первый взгляд, сравнительно просто. Необходимо лишь применить сортировку к массиву точек, представляющих собой вершины термов, найденных оптимизационным алгоритмом, и затем присваивать отсортированные значения вершинам термов. Порядок следования вершин (нумерация) устанавливается заранее (см. рис. 4.7).

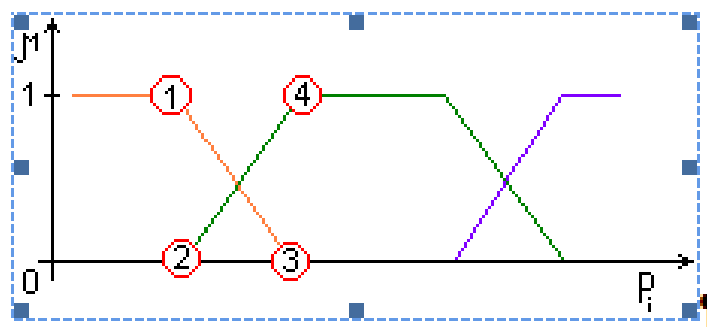


Рис.4.7. Порядок кодирования точек при настройке семантики

Однако сразу возникает вопрос, а какая из точек (1) и (2), (3) и (4) должна находиться левее, т.е. (1)>(2) или (2)>(1), (3)>(4) или (4)>(3)? Конечно, если предварительно определить очередность их расположения, например, (1)>(2), (3)>(4), то можно будет избежать

данной проблемы. Однако, убрав из рассмотрения остальные варианты расположения точек, мы искусственно сократим поисковое пространство, что не допустимо при оптимизации.

Для решения данной проблемы авторами был разработан алгоритм перевода массива точек в вершины термов. Для каждой пары таких точек ((1) и (2), (3) и (4), ...; рис. 4.7), назовем их спорными, определяется одна приоритетная точка, к примеру, первая точка пары. Затем найденный массив точек термов сортируется в порядке возрастания их значений. Каждой паре спорных точек в сортированном массиве соответствует своя двойка значений. Первой спорной паре первого термина соответствуют первый и второй элементы сортированного массива, второй паре второго термина – (3) и (4) элементы массива и т.д. Когда возникает вопрос о том, какая из точек будет лежать левее, из двух значений массива приоритетной точке приравнивается то число, которое в исходной строке (не сортированной) встречается раньше. Таким образом, отпадает какая-либо необходимость ввода ограничений на решение или искусственного сокращения пространство поиска.

Рассмотрим этот алгоритм на примере. Пусть некоторая лингвистическая переменная описывается тремя терминами и пусть в ходе оптимизации было получено решение в виде массива {1,2,0,7,5,4,8,6}. Для каждой пары спорных точек (1 и 2, 3 и 4, 5 и 6, ...; рис. 3.3.1) назначаем приоритетной первую точку, т.е. для 1 и 2 приоритетной является точка 1, для 3 и 4 – это 3 и т.д. После сортировки массив точек примет вид {0,1,2,4,5,6,7,8}. Далее, решается вопрос о расположении точек. Так, точка (2) термина будет лежать левее точки (1), поскольку элемент приоритетной точки {1} в исходном массиве находится левее точки {0}, т.е. точка (2) = {0} термина будет

лежать левее точки (1) = {1}. Схематически данная процедура отображена на рисунке 4.8.

Для предупреждения разрывов между термами порядок расположения таких точек как (2) и (3) (рис. 3.3.1) оговаривается заранее и не претерпевает изменений в процессе оптимизации. В данном случае, точка (2) должна лежать левее, либо совпадать с точкой (3), в противном случае возникнет разрыв функции, что не допустимо.

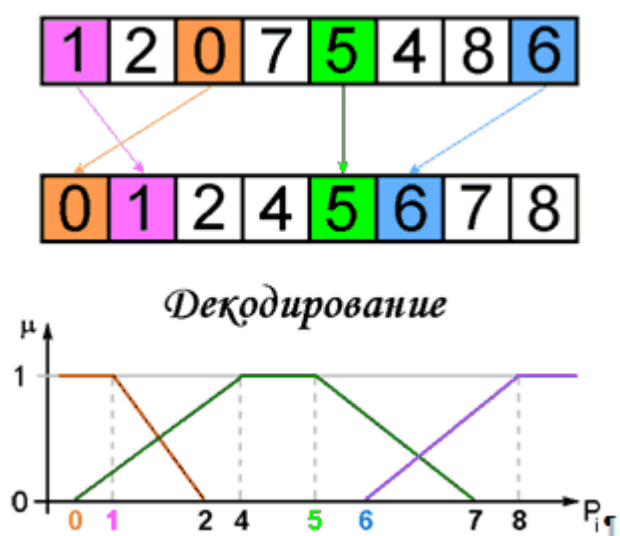


Рис.4.8. Алгоритм преобразования массива точек в вершинк термов

Оптимизация термов производится генетическим алгоритмом. Пригодность каждого набора термов определяется аналогично тому, как и в случае оценивания БП.

4.3.4 Коэволюционный алгоритм для формирования НСУ в целом

В ходе экспериментов было установлено, что оптимизация только БП при фиксированных значениях термов, либо только

настройка лингвистических переменных при фиксированной БП не позволяет в достаточной степени настроить нечеткую систему управления на решаемую задачу. Возникает необходимость одновременной настройки термов лингвистических переменных и БП.

Нами был разработан такой алгоритм настройки термов лингвистических переменных и БП, суть которого заключается в попеременной оптимизации БП и семантики лингвистических переменных, причем каждый последующий алгоритм запускается при фиксированном значении решения, полученном в ходе работы предыдущего алгоритма.

Алгоритм состоит из следующих шагов:

1. Инициализация БП и термов случайными значениями;
2. Фиксация значений термов лингвистических переменных, и оптимизация по БП;
3. Фиксация БП, найденной на предыдущем этапе, и оптимизация термов;
4. Если пройдено заданное число этапов, то завершение (найденные на последних этапах оптимизации БП и термы являются решением), иначе возвращаемся на п.2.

Оптимизация БП происходит с использованием генетического подхода. Данный подход был включен в коэволюцию, поскольку он, как настройка семантики лингвистических переменных, основан на ГА, что обеспечивает единообразность коэволюционного алгоритма. Авторы данной работы отдают себе отчет в том, что генетический подход является не самым лучшим алгоритмом оптимизации БП.

4.3.5. Формирование задачника

Для более объективной оценки качества управления, был введен тестовый задачник (ТЗ). ТЗ состоит из набора начальных параметров системы. Система ставится в условия каждой тестовой задачи, после

чего запускается контроллер. Качество управления оценивается по критерию (2.1). Общая оценка находится как сумма оценок качества по каждой задаче.

Задачник может создаваться случайным образом или составляться экспертом.

Если есть возможность создать задачник на основе заключений экспертов, то предпочтительнее использовать его. В таком случае задачник должен отвечать двум основным и, в то же время, противоречивым требованиям. Во-первых, ТЗ не должен быть слишком большого объема (не должен содержать слишком много тестовых задач), иначе оценка одной БП займет неприемлемо много времени. Во-вторых, ТЗ должен отражать основные (наиболее общие) ситуации, для более объективной оценки качества управления.

4.3.6. Общая постановка задачи

Необходимо разработать методы, позволяющие автоматически настраивать базу правил или базу правил и семантику лингвистических переменных контроллера решающего задачи из ТЗ.

$$\sum_{i=1}^M f_i(s) = \sum_{i=1}^M (FitS_i + [A]_i + [B]_i + [C]_i) \rightarrow \min_{BP}, \quad (4.6)$$

где $f_i(s)$ - значение критерия (1.2.1) на i -ой тестовой задаче из задачника, $FitS_i, [A]_i, [B]_i, [C]_i$ - значение соответствующего компонента критерия на i -ой тестовой задаче, M - количество задач в ТЗ.

4.3.7. Тестовая задача

Оценка качества рассматриваемых подходов производилась на следующей тестовой задаче: имеется система «Тележка - Перевернутый маятник», изображенная на рисунке 4.9.

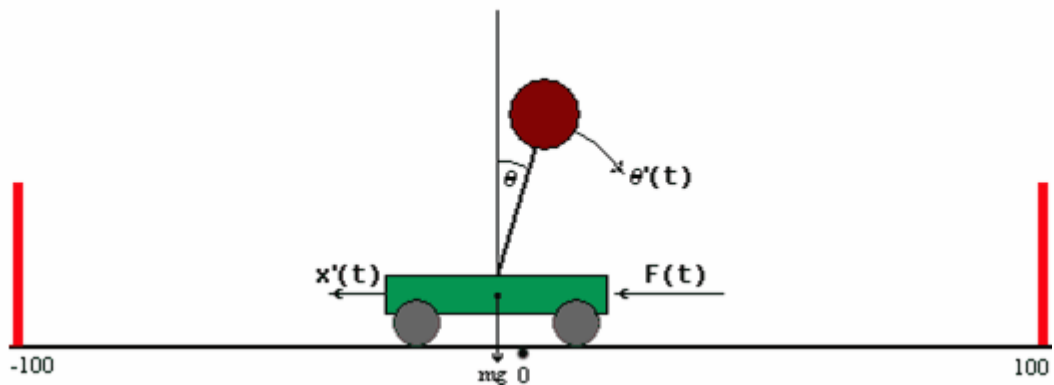


Рис. 4.9 Система «Тележка – Перевернутый маятник»

Динамическое поведение перевернутого маятника описывается двумя дифференциальными уравнениями:

$$x'' = \frac{1}{m} \cdot \left\{ F(t) + m_b \cdot l \cdot \left[(\theta'(t))^2 \cdot \sin(\theta(t)) - \theta''(t) \cdot \cos(\theta(t)) \right] \right\}, \quad (4.7)$$

$$\theta'' = \frac{m \cdot g \cdot \sin(\theta(t)) - \cos(\theta(t)) \cdot \left[F(t) + m_b \cdot l \cdot (\theta'(t))^2 \cdot \sin(\theta(t)) \right]}{\left(\frac{4}{3}\right) \cdot m \cdot l - m_b \cdot l \cdot \cos^2(\theta(t))}, \quad (4.8)$$

где $\theta(t)$ – угол наклона маятника от вертикали в момент времени t ,

$\theta''(t)$ – угловая скорость,

$\theta'''(t)$ – угловое ускорение,

$x(t)$ – позиция тележки в момент времени t ,

$x'(t)$ – скорость тележки,

$x''(t)$ – ускорение движения тележки,

l – длина маятника (от точки вращения до центра масс),

m_b – масса маятника,

m – суммарная масса тележки и маятника,

g – гравитационная постоянная,

$F(t)$ – сила, прилагаемая к тележке в момент времени t .

Данные уравнения не учитывают силу трения при движении.

Состояние системы в каждый момент времени характеризуется значением 4-х ее параметров:

- положение системы – $x(t)$;
- линейная скорость системы – $x'(t)$;
- угол отклонения маятника – $\Theta(t)$;
- угловая скорость маятника – $\Theta'(t)$.

Управляющим воздействием является сила $F(t)$, прикладываемая к системе «Тележка - Перевернутый маятник».

Система может перемещаться вдоль одной оси в интервале от минус 100 до 100 м. Маятник может совершать колебания в интервале от минус 12 до 12 градусов. Если значения положения системы или угла маятника превышают указанные интервалы, то считается, что система управления потерпела неудачу.

Необходимо построить нечеткий логический контроллер, управляющий системой «Тележка - Перевернутый маятник». Целью управления является приведение системы в состояние равновесия, которое характеризуется нулевым значением отклонения маятника от вертикальной оси и нулевым значением позиция тележки, за счет передвижения системы вдоль оси X , при любом начальном допустимом положении тележки и маятника.

4.3.8. Функция пригодности

Оценка эффективности работы БП для задачи «Тележка - Перевернутый маятник» производилась по следующему критерию, который необходимо было минимизировать в процессе оптимизации:

$$f(s) = FitS + [A] + [B] + [C] \rightarrow \underset{БП}{min}, \quad (4.9)$$

$$FitS = \left(\frac{0.5 \cdot a}{S_q} \right), \quad (4.10)$$

$$A = \sum_{i=1}^N K_i \cdot \frac{P_{iCur}}{P_{imax}}, \quad (4.11)$$

$$B = \begin{cases} 200, & \text{if } S_q < a; \\ 0, & \text{otherwise} \end{cases}, \quad (4.12)$$

$$C = \frac{1}{a} \left(0.3 \cdot K_1 \cdot \sum_{j=1}^{S_q} P_{1j} + K_3 \cdot \sum_{j=1}^{S_q} P_{3j} \right), \quad (4.13)$$

где a – максимально допустимое число шагов (тактов) работы системы при тестировании,

S_q – число шагов, сделанных системой, до момента прекращения тестирования,

K_i – уровень значимости i -го параметра системы, $i = 1, 2, 3, 4$,

P_{iCur} – текущее значение i -го параметра системы, $i = 1, 2, 3, 4$,

P_{imax} – максимальное реально достигаемое значение i -го параметра системы, $i = 1, 2, 3, 4$,

N – количество параметров системы (в нашем случае $N = 4$),

P_{1j} – позиция тележки на j -м шаге,

P_{2j} – линейная скорость тележки на j -м шаге,

P_{3j} – угол отклонения маятника от вертикали на j -м шаге,

P_{4j} – угловая скорость отклонения маятника на j -м шаге.

Значения 0.5 в формуле **FitS**, 200 в формуле **B** и 0.3 в формуле **C** были подобраны экспериментально.

Функции, указанные в квадратных скобках, могут быть включены в общий критерий по желанию пользователя (включение/исключение соответствующей функции производится в настройках алгоритма оптимизации).

Данная функция качества позволяет на первых этапах работы ГА производить отбор решений, которые достаточно

продолжительное время пытаются привести систему в равновесное состояние (доминирование слагаемого ошибки FitS). Включение же дополнительных критериев дает следующие преимущества:

[A] – когда почти все индивиды начнут более или менее хорошо “ловить” маятник, их пригодность будет уже определяться тем, насколько хорошо они “ловят” (FitS устремится к 0.5, а [A] начнет доминировать над FitS, определяя качество);

[B] – если индивид не смог продержаться заданное число тактов при тестировании, т.е. потерял управление над системой за число шагов, меньшее чем a , то он считается не способным решить данную тестовую задачу за что и получает «штрафные баллы»;

[C] – данный критерий обеспечивает быстроту и плавность управления системой, отбрасывая индивиды, совершающие большие колебательные движения и находящиеся далеко от нулевого положения.

В конечном счете, этот критерий поможет отобрать решение, которое за счет приведения системы в равновесие сможет достаточно долго не потерять управление, при этом достижение равновесного состояния будет происходить достаточно быстро и плавно.

4.3.9. Программная реализация

Приведенные выше алгоритмы были реализованы в виде программной системы (ПС). Стартовое окно данной системы отображено на рисунке 4.1.1.

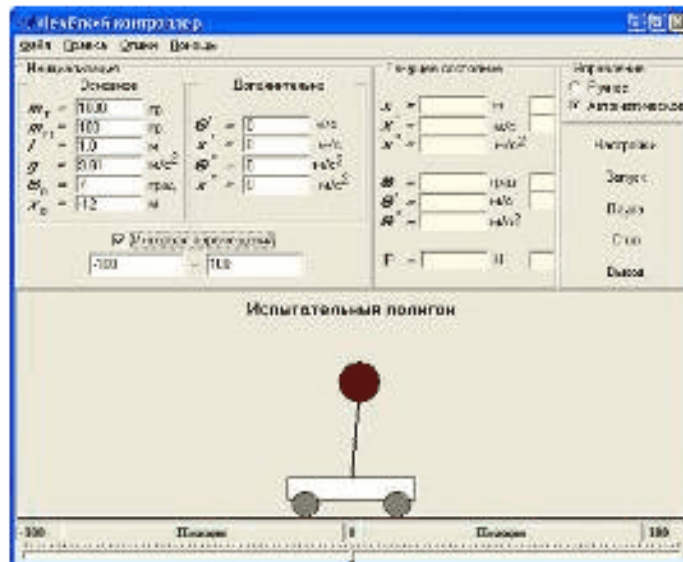


Рис. 4.10. Стартовое окно программной системы

Для упрощения работы с программой главная форма разбита на 4 основных блока: «Инициализация», «Текущее состояние», «Управление» (включая кнопки) и «Испытательный полигон». Рассмотрим каждый из этих элементов.

В блоке «Инициализация» задаются начальные параметры системы.

В блоке «Текущее состояние» отображаются текущие значения параметров системы и вектор силы, прикладываемый к тележке для управления системой.

Блок «Испытательный полигон» предназначен для визуального отображения работы системы.

Блок «Управление» состоит из 2-х частей: флага ручного либо автоматического управления и кнопок управления.

- Нажатие кнопки «Запуск» запускает механизм расчета физических параметров системы.
- Нажатие кнопки «Пауза/Продолжить» временно останавливает (замораживает) движение системы с сохранением всех параметров / запускает «физику» системы.

- При нажатии кнопки «Стоп» происходит остановка работы системы без запоминания последних параметров.
- Кнопка «Выход» - выход из программы, при необходимости выдается запрос о сохранении изменений БП и термов.
- Нажатие кнопки «Настройки» вызывает форму, на которой находятся редактор терм, редактор БП и панель работы с алгоритмами автоматического формирования БП (см. рис. 4.11, 4.12, 4.13).

Программная система предоставляет широкие возможности по настройке функции качества, параметров алгоритмов, а так же позволяет регулировать термы и БП вручную (используя соответствующие редакторы). Формы панели редакторов термов и БП приведены на рисунках 4.12 и 4.13, соответственно. Для настройки параметров или запуска какого-либо алгоритма необходимо нажать соответствующую кнопку.

После запуска оптимизационного алгоритма открывается окно Diagram, отображающее ход работы программы (см. рис. 4.14).

После того, как оптимизационный алгоритм отработал заданное число итераций, можно проверить полученную БП на испытательном полигоне (см. стартовое окно программы), для чего нужно просто закрыть окно настроек. Так же можно просмотреть найденные правила (используя редактор БП) или сохранить БП в файл (стартовая форма, меню файл, сохранить БП).

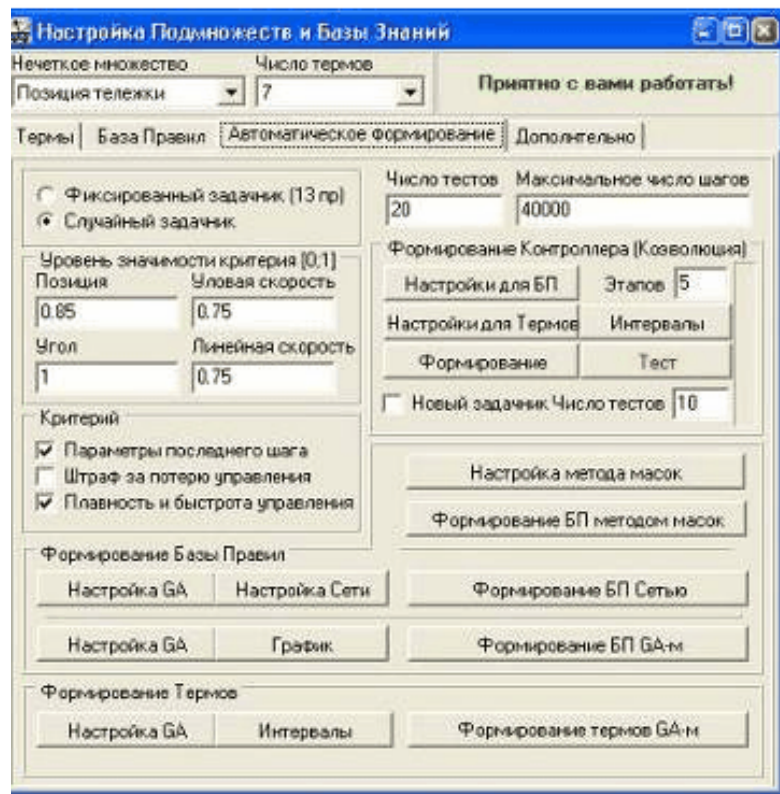


Рис. 4.11. Форма панели настроек

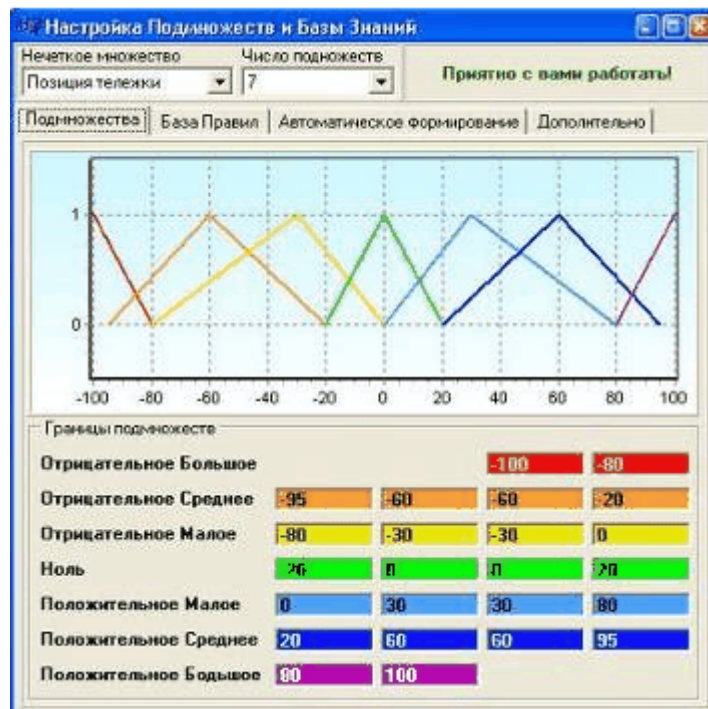


Рис. 4.12. Форма панели редактора термов

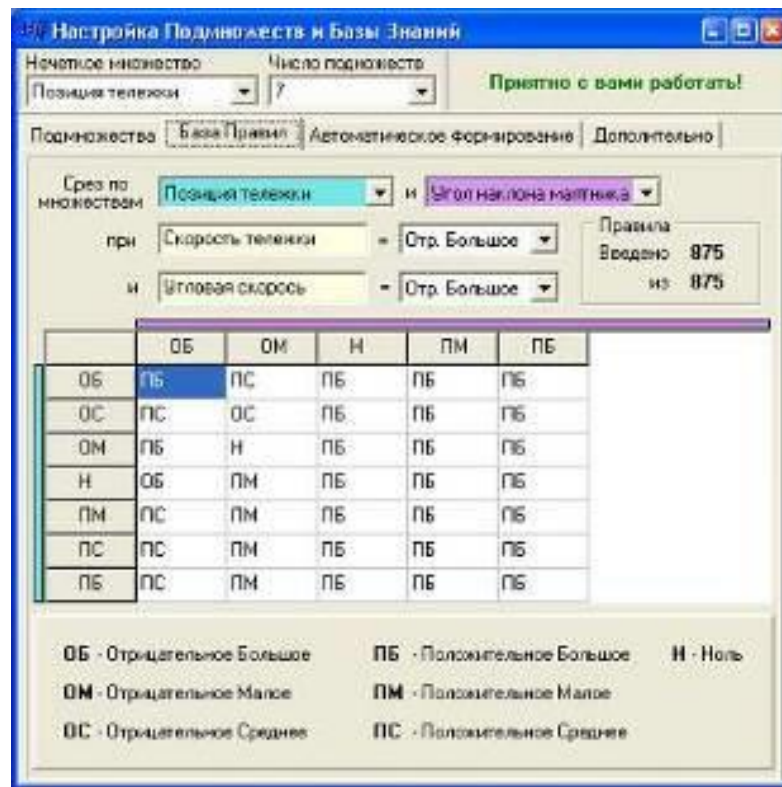


Рис. 4.13. Форма панели редактора БП

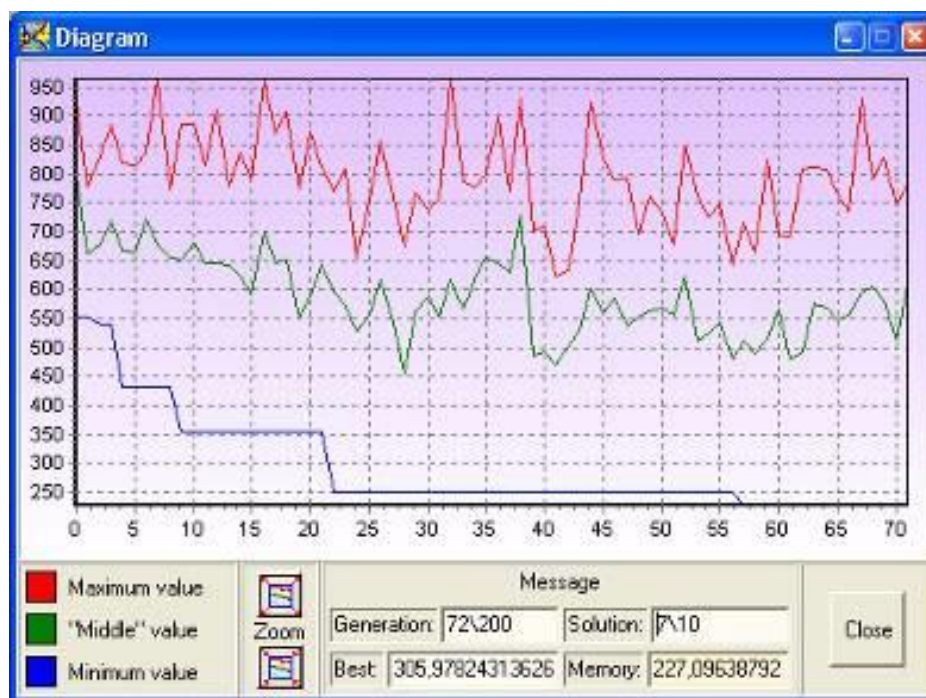


Рис. 4.14. График работы оптимизационного алгоритма

Разработанная нами ПС позволяет производить анализ работы реализованных алгоритмов и методов и получать статистику по

процессу оптимизации. Это позволило провести исследования эффективности работы двух алгоритмов автоматического формирования БП и алгоритма автоматического формирования нечеткого контроллера, а также подобрать оптимальную структуру для каждого подхода.

4.3.10. Подбор оптимальной структуры оптимизационного алгоритма

Генетический подход. Структура ГА состояла из параметров двух видов: фиксированных и переменных. Фиксированными параметрами являлись размер турнира и элиты (для турнирной и элитарной селекций соответственно), которые были равны 5 и 6 индивидам, вероятность мутации – средняя. Переменными параметрами были вид селекции, тип рекомбинации, объем популяции, число поколений. Число вычислений целевой функции было ограничено 2000. Для сравнения брались статистически усредненные значения за 50 прогонов. Сравнение алгоритмов осуществлялось по нескольким критериям, среди которых найденный минимум, скорость сходимости алгоритма к наилучшему решению, период активного спуска алгоритма (в процентах). Данные экспериментов приведены в таблице А.1 в приложении. Наиболее эффективной среди всех структур оказалась следующая: Элитарная + равновероятная селекция + Двухточечная рекомбинация + 20 индивидов x 100 поколений [5].

Нейросетевой подход. Фиксированными параметрами являлись размер турнира и элиты (для турнирной и элитарной селекций соответственно), которые были равны 2 и 3 индивидам, вероятность мутации – средняя. Также фиксированной была структура нейронной сети, которая имела следующие параметры: один скрытый слой, четыре нейрона на скрытом слое, параметр α активационной функции

был равен 0,5. Переменными параметрами были вид селекции, тип рекомбинации, объем популяции, число поколений. Число вычислений целевой функции было ограничено 2000. Для сравнения брались статистически усредненные значения за 50 прогонов. Сравнение алгоритмов осуществлялось также по нескольким критериям. Данные экспериментов приведены в таблице А.2 в приложении. Наиболее эффективной среди всех структур оказалась следующая: Элитарная + пропорциональная селекция + Одноточечная рекомбинация + 10 индивидов x 200 поколений.

4.3.11. Сравнение эффективности подходов

Перед алгоритмами ставилась задача поиска наилучшего решения за 2000 вычислений целевой функции. Для каждого алгоритма использовалась оптимальная структура ГА. Для сравнения брались статистически усредненные значения за 50 прогонов. Сравнение осуществлялось по скорости сходимости алгоритмов к наилучшему решению. Графики работы оптимизационных алгоритмов (статистически усредненные по 50 прогонам) приведены на рисунках 4.15 и 4.16, соответственно, для нейросетевого и генетического подходов формирования БП и график коэволюционного алгоритма - на рисунке 4.17. На графиках отображен разброс решений (ось ординат) за каждые 20 вычислений целевой функции (ось абсцисс).

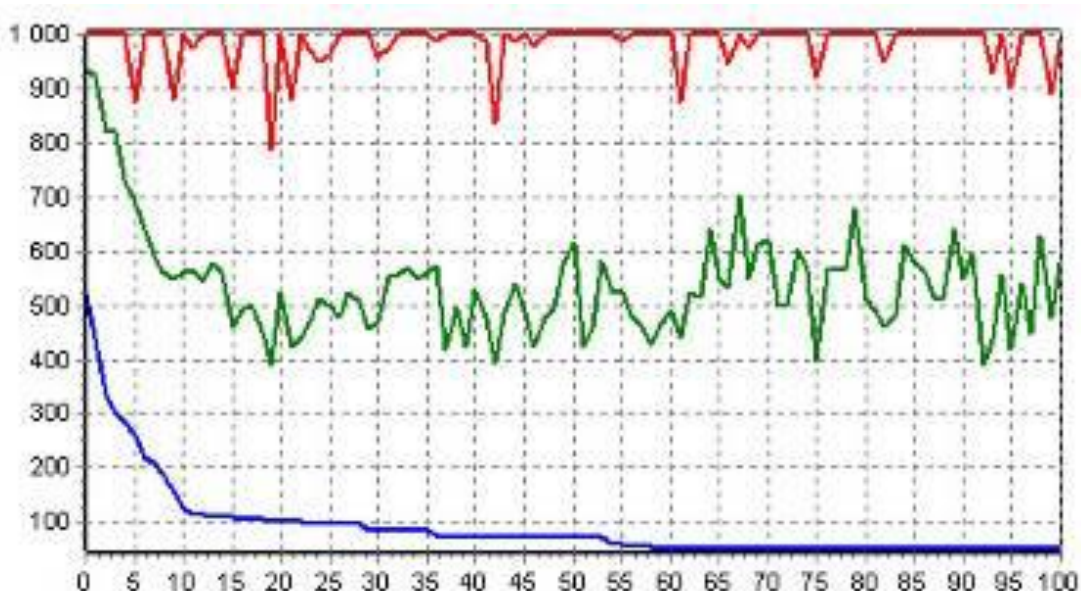


Рис. 4.15 График работы оптимизационного алгоритма при нейросетевом подходе (Получено решение с пригодностью $f = 33,2290$)

В результате проведенных исследований было установлено, что при наиболее эффективных параметрах оптимизационного алгоритма качество управления объектом (система «тележка - перевернутый маятник») по БП, полученной с помощью ИНС выше, чем по БП, полученной с помощью ГА. Козволюционный алгоритм занимает промежуточное положение между нейросетевым и генетическим подходами. Это доказывает, что коэволюционная процедура улучшает качество работы используемого алгоритма формирования БП, в нашем случае генетического. Включение же нейросетевого подхода формирования БП в коэволюционный алгоритм позволит ускорить процедуру оптимизации и повысить качество формируемой НСУ.

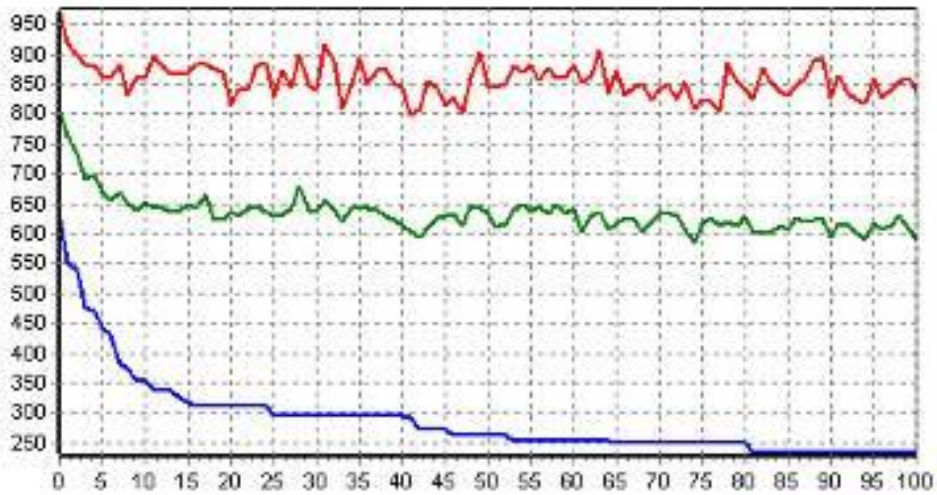


Рис. 4.16. График работы оптимизационного алгоритма при генетическом подходе (Получено решение с пригодностью $f = 232,8956$)

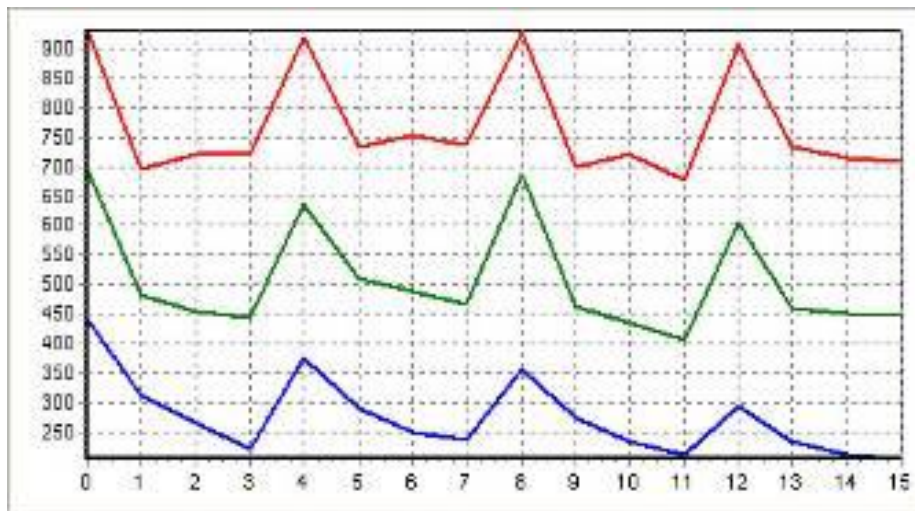


Рис. 4.17. График работы оптимизационного алгоритма при коэволюционном алгоритме (Получено решение с пригодностью $f = 205.068$)

Резкие скачки значений Min, Max и Mid в коэволюционном алгоритме обусловлены переходами от формирования БП к формированию термов лингвистических переменных и наоборот, а также изменением примеров в задачнике.

4.4. Применение алгоритмов природных вычислений к решению задач оптимизации

Для нахождения приближенных решений оптимизации применяется множество алгоритмов, семь из которых рассматриваются в данной работе: нечеткие множества, искусственные нейронные сети, генетический алгоритм, муравьиный алгоритм, алгоритм роя частиц, ДНК-вычисления и новый подход, основанный на искусственных иммунных системах (ИИС). Все эти методы относятся к направлению «природных вычислений», т.е. моделируют те или иные биологические процессы, алгоритмы которых природа создавала миллионы лет. Стоит отметить, что эффективность того или иного алгоритма зависит от характеристик исходных данных задачи, поэтому нельзя однозначно определить, какой из алгоритмов наиболее эффективен.

Рассмотрим тестовую задачу комбинаторной оптимизации о торговых агентах и задачу маршрутизации. Торговая фирма продает товары в n различных городах, покупательная способность жителей которых оценивается в b_j усл. ед., $j = \overline{1, n}$. Для реализации товаров фирма располагает m торговыми агентами, каждого из которых она направляет в один из городов. Профессиональный уровень агентов различен; доля реализуемых i -м торговым агентом покупательных способностей составляет a_i , $i = \overline{1, m}$.

Введем параметр $r_{ij} = a_i b_j$, характеризующий величину покупательных способностей, реализуемых i -м торговым агентом в j -м городе.

Управляющие переменные z_{ij} , $i = \overline{1, m}$; $j = \overline{1, n}$ определяются по формуле

$$r_{ij} = \begin{cases} 1, & \text{если } i\text{-й агент направлен в } j\text{-й город;} \\ 0, & \text{в противном случае.} \end{cases}$$

Математическая модель запишется в следующей форме:

$$R = \sum_{i=1}^n \sum_{j=1}^n r_{ij} z_{ij} \rightarrow \max. \quad (4.14)$$

$$\begin{cases} \sum_{j=1}^n z_{ij} = 1, & i = \overline{1, n}; \\ \sum_{i=1}^n z_{ij} = 1, & j = \overline{1, n}; \\ z_{ij} \in \{0; 1\}, & i = \overline{1, n} \quad j = \overline{1, n}. \end{cases} \quad (4.15)$$

Первое и второе ограничения формализуют соответственно условия о том, что в каждый город направляется один торговый агент, и один торговый агент не может работать в двух городах. Целевая функция $R(z)$ - это сумма реализованных покупательных способностей всеми торговыми агентами во всех городах. Она должна быть максимальна.

Решить задачу о торговых агентах – значит найти z_{ij} , удовлетворяющие (4.15) и доставляющие минимум функции (4.14).

Существуют три наиболее эффективных алгоритма нахождения кратчайшего пути:

- алгоритм Дейкстры (используется для нахождения оптимального маршрута между двумя вершинами);
- алгоритм Флойда (для нахождения оптимального маршрута между всеми парами вершин);
- алгоритм Йена (для нахождения k -оптимальных маршрутов между двумя вершинами).

Указанные алгоритмы легко выполняются при малом количестве вершин в графе. При увеличении их количества задача поиска кратчайшего пути усложняется.

В связи с этим исследуются современные методы к решению задач оптимизации.

4.4.1. Нечеткие множества

Рассмотрим формирование маршрутной матрицы на основе теории F-множеств.

Модель F-маршрутизации

Пусть априорно известны источник И и получатель П, множество допустимых планов маршрутов $L = \{l_j\}$, множество параметров маршрутов $\pi_n = \Pi$ (длина маршрута, задержка передачи, безопасность, достоверность, пропускная способность, коэффициенты старения информации и готовности элементов сети и простоя и т.д.), возможные ситуации повреждений сети $C_u = \{c_k\}$ [1].

В условиях реальной эксплуатации, как правило, отсутствует статистика распределения потоков информации в экстремальных ситуациях в момент времени T_i . Однако в этих условиях необходимо выбирать приемлемый маршрут l^* для установления требуемой связи с удовлетворительным качеством приёма информации.

В связи с этим на каждом узле коммутации априорно строятся нечёткие рефлексивные матрицы предпочтений маршрутов M_n^l , на основе стандартных функций принадлежности по условию ситуации $c_k \in C_u$, хранящиеся в базе данных B_{mk} . Предварительно база данных B_k формируется на основе мнений экспертов по телекоммуникационной сети в виде логических нечётких правил.

Если учесть, что менеджер в системе управления сетью является доминирующим на своем уровне распределенного управления (РУ),

то достаточно сформировать F-матрицу маршрутизации только для него.

Таким образом, с учётом адреса (кода) агента, параметров маршрутов $\pi_n = \Pi$ и возможных ситуаций $c_k \in C_u$ в управляющей системе M определяется подмножество приемлемых маршрутов $L^* \subset L = \{l_j\}$. Из множества L^* строится нечёткая матрица предпочтений, которая имеет вид:

$$M_n^l = \begin{matrix} & l_1 & l_2 & l_3 & \dots & l_j \\ \begin{matrix} l_1 \\ l_2 \\ l_3 \\ \dots \\ l_q \end{matrix} & \left| \begin{array}{cccccc} 1 & \alpha_{12} & \alpha_{13} & \dots & \alpha_{1j} \\ \alpha_{21} & 1 & \alpha_{23} & \dots & \alpha_{2j} \\ \alpha_{31} & \alpha_{32} & 1 & \dots & \alpha_{3j} \\ \dots & \dots & \dots & \dots & \dots \\ \alpha_{q1} & \alpha_{q2} & \alpha_{q3} & \dots & 1 \end{array} \right. \end{matrix}, \quad (4.16)$$

где M_n^l - матрица отношения маршрутов; n - индекс параметра π , относительно которого эксперт определяет степень отношения предпочтения маршрутов в сети $l_j \in L$; α_{qj} - числовое значение функции принадлежности

$$\mu_R(l_q, l_j) = \langle \text{не хуже } l_j \rangle, l_q, l_j \in L, \alpha_{qj} \in [0,1]. \quad (4.17)$$

Также строится рефлексивная матрица предпочтений для параметров маршрутов $\pi_n = \Pi$, на основе априорно определенной базы данных $B_{\pi k}$ при условии ситуации $c_k \in C_u$:

$$M_k^\pi = \begin{matrix} & \pi_1 & \pi_2 & \pi_3 & \dots & \pi_j \\ \begin{matrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \dots \\ \pi_q \end{matrix} & \left| \begin{array}{cccccc} 1 & b_{12} & b_{13} & \dots & b_{1j} \\ b_{21} & 1 & b_{23} & \dots & b_{2j} \\ b_{31} & b_{32} & 1 & \dots & b_{3j} \\ \dots & \dots & \dots & \dots & \dots \\ b_{q1} & b_{q2} & b_{q3} & \dots & 1 \end{array} \right. \end{matrix}, \quad (4.18)$$

где π_n - параметр маршрута при ситуации c_k ; b_{ji} - числовое значение функции принадлежности

$$\mu_R(\pi_i, \pi_j) = \langle \pi_i \text{ не хуже } \pi_j \rangle, i, j = \overline{1, n}, b_{ij} \in [0, 1].$$

Далее необходимо соотнести матрицы (4.16) и (4.18) и принять решение о наиболее приемлемом маршруте для передачи информации к получателю, т.е. получить матрицу

$$M_k^l = \begin{matrix} & l_1 & l_2 & l_3 & \dots & l_j \\ \begin{matrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \dots \\ \pi_q \end{matrix} & \left| \begin{matrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \dots & \alpha_{1j} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \dots & \alpha_{2j} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \dots & \alpha_{3j} \\ \dots & \dots & \dots & \dots & \dots \\ \alpha_{q1} & \alpha_{q2} & \alpha_{q3} & \dots & \alpha_{nj} \end{matrix} \right. \end{matrix} \quad (4.19)$$

Каждая строка матрицы (4.19) есть нечёткое множество $L_{kn} = \langle \text{ВОЗМОЖНАЯ ОБЛАСТЬ МАРШРУТОВ ПО УСЛОВИЮ ПАРАМЕТРА } \pi_n \rangle$ с функцией принадлежности $\mu(l_j) \in [0, 1], \alpha_{nj} \in [0, 1]$, где n, j - текущие индексы, k - индекс ситуации c_k .

Каждая строка матрицы (4.19) является сжатием матриц типа (4.16) по алгоритму [1].

Шаг1. На основе матриц типа (4.14) строятся матрицы строгого предпочтения по выражению

$$R^S = R^p - (R^p)^{-1}$$

с функцией принадлежности

$$\mu_R(x, y) = \max_{x, y \in L} (0, \mu_R(x, y) - \mu_R(y, x)), \quad (4.20)$$

где R^p - нечёткое отношение предпочтения $R^p = M^d$; $(R^p)^{-1}$ - обратное отношение; R^S - нечёткое отношение строго предпочтения; x, y - значения переменных тождественных l (маршрутам), $l_j \in L$.

Шаг2. Берётся дополнение от (4.20)

$$\overline{R} = 1 - R^S$$

с функцией принадлежности

$$\mu_{\bar{R}}(x, y) = 1 - \mu_{R^s}(x, y). \quad (4.21)$$

Шаг 3. Определяется нечёткое подмножество недоминируемых (н.д.) альтернатив, т.е. производится свертка матрицы (4.163) M_n^l в виде строки

$$\mu_{R_i}^{HD}(y) = \min_{x \in L} [1 - \mu_R(x, y)], \quad y \in L, \quad (4.22)$$

или

$$\mu_{R_i}^{HD}(y) = 1 - \max_{x \in L} \mu_R(x, y), \quad y \in L, \quad i = \overline{1, n}. \quad (4.23)$$

В результате проведения операций (4.20) - (4.23) образуется матрица нечётких отношений (4.19).

Этап 2. Далее формируется матрица нечётких отношений на основе матриц (4.19) и (4.17) по алгоритму.

Шаг 4. Берётся max-min композиция матриц (4.19) и (4.18)

$$R_k = M_k^l \circ M_k^\pi \quad (4.24)$$

с функцией принадлежности

$$\mu_R(l_i, l_j) = \max_{x, y \in L} \min [\mu_{M^l}(l_i, \pi_j), \mu_{M^\pi}(\pi_i, \pi_j)]. \quad (4.25)$$

Шаг 5. Из матрицы R_k (4.24) определяется нечёткое подмножество недоминирующих альтернатив аналогично (4.22) или (4.23). Таким образом находится подмножество $L_k \subset L$ возможных маршрутов системы при k -й сетевой ситуации с функцией принадлежности

$$\mu_{L_k}(l_j / \pi_1, \pi_2, \dots, \pi_n), \quad (4.26)$$

по условию параметров $\pi_i, i = 1, \dots, n$.

Шаг 6. В соответствие с выражением (4.18) определяется наиболее возможный маршрут в сети управления.

Далее матрица (4.19) по алгоритму из [1] сжимается в матрицу-строку и определяется наиболее приемлемый маршрут по правилу

$$l_k^* = \arg \max_{l_j \in L} \mu_L(l_j / \pi_1, \pi_2, \dots, \pi_i), \quad (4.27)$$

где $\mu_L(l_j / \pi_1, \pi_2, \dots, \pi_i)$ - функция принадлежности <НАИБОЛЕЕ ВОЗМОЖНАЯ ОБЛАСТЬ МАРШРУТОВ ЛПО УСЛОВИЮ ПАРАМЕТРОВ $\pi_1, \pi_2, \dots, \pi_i$ >; l_k^* - наиболее возможный маршрут при k -й ситуации в сети.

4.4.2. Нейронные сети

Хопфилд и Танк показали подход к ее приближенному решению на основе сетей Хопфилда. Рассмотрим вкратце этот подход. Для описания возможных маршрутов авторы предложили специальный тип матрицы. В ней города образуют строки, а столбцы отображают последовательность городов в маршруте. В позиции (x, i) матрицы стоит 1 в том случае, когда город x занимает i -е место в маршруте [2].

В случае n городов существует $\frac{n!}{2^n}$ различных маршрутов, среди которых необходимо найти кратчайший. Для получения решения задача отображается сетью Хопфилда.

В ней каждый нейрон обозначается двумя индексами x и i , причем x отражает город, а i -ю позицию в маршруте, т.е. z_{xi} - это выход нейрона, в котором город x размещен на i -й позиции маршрута.

К энергетической функции E сети Хопфилда предъявляются следующие условия:

- должна быть минимальна только для допустимых решений, которые содержат одну единицу в каждой строке и в каждом столбце матрицы описания маршрутов;
- для решений с более короткими маршрутами должна принимать меньшие значения.

Энергетическая функция, удовлетворяющая этим условиям, может иметь вид:

$$\begin{aligned}
E = & \frac{A}{2} \sum_x \sum_i \sum_{j \neq i} z_{xi} z_{xj} + \frac{b}{2} \sum_x \sum_i \sum_{y \neq x} z_{yi} z_{xi} + \\
& + \frac{C}{2} \left(\left(\sum_x \sum_i z_{xi} \right) - n \right)^2 + \\
& + \frac{D}{2} \sum_x \sum_i \sum_y r_{xy} z_{xi} (z_{yi+1} + z_{yi-1})
\end{aligned}$$

При ее выборе учтены следующие соображения:

- первое слагаемое равно нулю только в тех случаях, когда каждая строка матрицы описания маршрутов содержит лишь одну единицу;
- второе слагаемое равно нулю только тогда, когда каждый столбец матрицы содержит лишь одну единицу;
- третье слагаемое равно нулю лишь в тех случаях, когда в матрице описания маршрутов имеется n единиц, что означает: каждый город посещается лишь один раз;
- четвертое слагаемое отражает длину маршрута. Обратим внимание, что для каждого города x , расположенного на i -й позиции, определяется расстояние r_{xy} до его последователя y на позиции $i+1$ и его предшественника y на позиции $i-1$.

Применение сетей Хопфилда ограничивается высокой вычислительной сложностью (n^4 , где n – размерность задачи) и необходимостью тщательного подбора параметров функции энергии и функции активации нейрона. Ситуацию можно существенно упростить, если применить следующий подход.

Для решения задачи (4.14)-(4.15) предложена рекуррентная нейронная сеть [3], которая описывается дифференциальным уравнением

$$\frac{\partial u_{ij}(t)}{\partial t} = -\eta \left(\sum_{k=1}^n z_{ik}(t) + \sum_{l=1}^n z_{lj}(t) - 2 \right) + \lambda r_{ij} \exp\left(-\frac{t}{\tau}\right), \quad (4.28)$$

где $z_{ij} = f(u_{ij}(t))$, $f(u) = \frac{1}{1 + \exp(-\beta u)}$. Как и в сети Хопфилда, здесь используется матрица нейронов размером $n \times n$, но нейроны взаимодействуют не по принципу «каждый с каждым», а по строкам и столбцам.

Разностный вариант этого уравнения имеет вид

$$u_{ij}^{t+1} = u_{ij}^t - \Delta t \cdot \left[\eta \left(\sum_{k=1}^n z_{ik}(t) + \sum_{l=1}^n z_{lj}(t) - 2 \right) - \lambda r_{ij} \exp\left(-\frac{t}{\tau}\right) \right], \quad (4.29)$$

где Δt - шаг по времени. Параметры $\Delta t, \eta, \lambda, \tau, \beta$ подбираются экспериментально и существенно влияют на скорость достижения решения задачи и качество этого решения.

Для ускорения решения системы уравнений (4.29) предложен принцип «Winnertakesall» [3]:

1. Порождается матрица $\|z_{ij}^0\|$ случайных значений $z_{ij}^0 \in [0,1]$. Итерация (4.29) продолжаются до тех пор, пока не выполнится неравенство

$$\sum_{k=1}^n z_{ik}(t) + \sum_{l=1}^n z_{lj}(t) - 2 \leq \varepsilon,$$

где ε - заданная точность выполнения ограничений (4.26).

2. Выполняется преобразование поученной матрицы решения $\|z_{ij}\|$:

2.1. $i = 1$.

2.2. В i -й строке матрицы отыскивается максимальный элемент $z_{i, j_{\max}}$, j_{\max} - номер столбца с максимальным элементом.

2.3. Выполняется преобразование $z_{i, j_{\max}} = 1$. Все остальные элементы первой строки и столбца обращаются в нуль:

$$z_{ij} = 0, \quad j \neq j_{\max},$$

$$z_{k, j_{\max}} = 0, \quad k \neq i.$$

Далее происходит переход к строке j_{\max} .

Действия 2.2. и 2.3 повторяются, пока не произойдет возврат к первой строке, что будет означать завершение построения цикла

3. Если возврат к строке 1 произошел раньше, чем в матрице $\|z_{ij}\|$ значение 1 получили n элементов, то это означает, что длина построенного цикла меньше n . В этом случае шаги 1 и 2 повторяются.

При таком подходе вычислительная сложность алгоритма решения задач маршрутизации снижается с $O(n^4)$ до $O(n^2)$.

4.4.3. Генетический алгоритм

Генетические алгоритмы являются одним из видов эволюционных вычислений. Как правило, они применяются для поиска решений в задачах с большой комбинаторной сложностью, для которых трудно найти решение аналитическим путем. Принцип работы генетического алгоритма основан на теории эволюции Чарльза Дарвина. Основателем генетических алгоритмов считается Джон Холланд (John Holland), опубликовавший в 1975 году первую книгу в этой области исследований под названием “Адаптация в естественных и искусственных системах” (“Adaptation in Natural and Artificial Systems”) [4].

В классическом генетическом алгоритме решение кодируется двоичной последовательностью. Такой способ подходит для случаев, когда изменение одного бита в решении в большинстве случаев приводит к незначительному изменению целевой функции.

В задачах на графах решением является маршрут - последовательность вершин. Изменение одной вершины в маршруте может кардинально отразиться на значении целевой функции. Поэтому в подобных задачах операции производятся над последовательностями целых чисел, обозначающих номера вершин. Функцией приспособленности является длина маршрута [5].

Основная идея генетического алгоритма состоит в преобразовании задачу (4.14) – (4.15) в последовательность задач безусловной минимизации

$$f_k(z) = f(x) + \frac{1}{k} \left(\sum_{i=1}^n z_{ij} + \sum_{j=1}^m z_{ij} - 2 \right) \rightarrow \min, \quad k = 1, 2, \dots$$

Переформулируем задачу оптимизации как задачу нахождения минимума некоторой функции $f^k(z_1, z_2, \dots, z_n)$, называемой *функцией приспособленности* (fitness function). Она должна принимать неотрицательные значения на ограниченной области определения (для того, чтобы мы могли для каждой особи считать её приспособленность, которая не может быть отрицательной), при этом совершенно не требуются непрерывность и дифференцируемость.

Шаг алгоритма состоит из трех стадий [6,7]:

1. Генерация промежуточной популяции (*intermediate generation*) путем отбора (*selection*) текущего поколения.
2. Скрещивание (*recombination*) особей промежуточной популяции путем *кроссовера* (*crossover*), что приводит к формированию нового поколения.

Кроссинговер заключается во взаимном обмене генами между «родителями» - хромосомами предварительно выбранной пары.

Предварительно задается величина P_k – вероятность кроссинговера и вводится флажок FG с двумя состояниями «выполнять», «не выполнять». Исходное состояние FG «не выполнять». При выполнении кроссинговера последовательно просматриваются локусы выбранной пары хромосом. С вероятностью P_k «флажок» FG переходит в состояние «выполнять». Если FG перешел в состояние «выполнять», то производится обмен генами между парой хромосом в текущем локусе, далее «флажок» переходит

в состояние «не выполнять», а затем осуществляется переход к следующему локусу.

3. Мутация нового поколения.

Операция мутации заключается в изменении значения гена. Алгоритм мутации реализуется следующим образом.

Последовательно выбираются хромосомы из текущей популяции. В пределах выбранной хромосомы последовательно просматриваются гены. После перехода к очередному гену, FG с вероятностью P_M переходит в состояние «выполнять». Если FG перешел в состояние «выполнять», то случайным образом ген g_n принимает одно из значений в заданном диапазоне, за исключением значения, которое ген имеет перед мутацией. Далее FG переходит в состояние «не выполнять» и выбирается следующий ген хромосомы, или следующая хромосома.

4.4.4. Муравьиный алгоритм

Муравьиные алгоритмы основаны на моделировании взаимодействия муравьев в муравьиной колонии. Муравьиные алгоритмы эффективны при решении различных комбинаторных задач на графах, включая задачу о коммивояжере. Их исследование началось в начале 90-х годов Марко Дориго в Университете Брюсселя [8].

Муравьиная колония представляет собой сложную распределенную не централизованную систему. Каждый муравей (по сути является агентом в мультиагентной системе) выполняет простые однообразные действия, обладая минимумом информации и взаимодействуя с небольшим количеством других муравьев и небольшими участками окружающей среды. Однако вся система в целом решает сложные задачи оптимизации маршрутов, находя очень

близкие к оптимальным решения, адаптируясь к внешним изменяющимся условиям.

Взаимодействие между муравьями происходит по средствам прямого и непрямого обмена информацией. Прямой обмен заключается в непосредственном контакте, а непрямой в изменении окружающей среды одним муравьем и распознавание этих изменений другими. На основе непрямого обмена и моделируются муравьиные алгоритмы. Для изменения окружающей среды муравьи распыляют феромон, запах которого остается в почве некоторое время.

На начальном этапе работы алгоритма муравьи выбирают направления случайным образом, пытаясь достичь цели (найти пищу), пометая феромонами свой путь. Те муравьи, которые раньше всех достигнут цели, соответственно вернуться обратно раньше всех тем же путем, увеличив содержание феромонов на этом маршруте. Через несколько итераций кратчайший маршрут будет сильно отличаться от всех остальных вариантов пути [8]. Адаптивность к внешним изменениям осуществляется за счет испарения феромонов. Если на кратчайшем пути встречается препятствие, муравей случайным образом выбирает другой путь, таким образом через некоторое время находится новый кратчайший маршрут. Муравьиный алгоритм, в отличие от генетического, гораздо быстрее адаптируется к изменению внешних условий.

4.4.5. Алгоритм роя частиц

Стая птиц представляет собой прекрасный пример коллективного поведения животных. Летая большими группами, они почти никогда не сталкиваются в воздухе. Стая двигается плавно и скоординировано, словно ей кто-то управляет. А любой, кто вешал в

своем дворе кормушку, знает, что спустя несколько часов его найдут все птицы в округе.

Наблюдение за птицами вдохновило Крейга Рейнольдса (Craig Reynolds) на создание в 1986 году компьютерной модели, которую он назвал *Voids*. Для имитации поведения стаи птиц, Рейнольдс запрограммировал поведение каждой из них в отдельности, а также их взаимодействие. При этом он использовал три простых принципа. Во-первых, каждая птица в его модели стремилась избежать столкновений с другими птицами. Во-вторых, каждая птица двигалась в том же направлении, что и находящиеся неподалеку птицы. В-третьих, птицы стремились двигаться на одинаковом расстоянии друг от друга.

Результаты первых же симуляций удивили самого создателя: несмотря на простоту лежащих в основе программы алгоритмов, стая на экране выглядела крайне правдоподобно. Птицы сбивались в группы, уходили от столкновений и даже хаотично металась точь-в-точь как настоящие. Как специалист в области компьютерной графики, Крейг Рейнольдс был в первую очередь заинтересован визуальной стороной результатов созданной им имитации. Однако, в посвященной *Voids* статье он также отметил, что разработанная им поведенческая модель может быть расширена введением дополнительных факторов – таких, как поиск пищи или боязнь хищников [9].

Классический алгоритм роя частиц. В 1995 году Джеймс Кеннеди (James Kennedy) и Рассел Эберхарт (Russel Eberhart) предложили метод для оптимизации непрерывных нелинейных функций, названный ими алгоритмом роя частиц [10]. Вдохновением для них послужила имитационная модель Рейнольдса, а также работа Хеппнера (Heppner) и Гренадера (Grenader) на схожую тему [11].

Кеннеди и Эберхарт отметили, что обе модели основаны на управлении дистанциями между птицами – а, следовательно, синхронность стаи является в них функцией от усилий, которые птицы прилагают для сохранения оптимальной дистанции.

Разработанный ими алгоритм довольно прост и может быть реализован буквально в нескольких десятках строчек кода на любом высокоуровневом языке программирования. Он моделирует многоагентную систему, где агенты-частицы двигаются к оптимальным решениям, обмениваясь при этом информацией с соседями.

Текущее состояние частицы характеризуется координатами в пространстве решений (то есть, собственно, связанным с ними решением), а также вектором скорости перемещения. Оба этих параметра выбираются случайным образом на этапе инициализации. Кроме того, каждая частица хранит координаты лучшего из найденных ей решений, а также лучшее из пройденных всеми частицами решений – этим имитируется мгновенный обмен информацией между птицами.

На каждой итерации алгоритма направление и длина вектора скорости каждой из частиц изменяются в соответствии со сведениями о найденных оптимумах:

$$v_i = v_i + a_1 \cdot \text{rnd}() \cdot (pbest_i - x_i) + a_2 \cdot \text{rnd}() \cdot (gbest - x_i),$$

где v – вектор скорости частицы (v_i – его i -ая компонента), a_1, a_2 – постоянные ускорения, $pbest$ – лучшая найденная частицей точка, $gbest$ – лучшая точка из пройденных всеми частицами системы, x – текущее положение частицы, а функция $\text{rnd}()$ возвращает случайное число от 0 до 1 включительно.

После вычисления направления вектора v , частица перемещается в точку $x = x + v$. В случае необходимости, обновляются

значения лучших точек для каждой частицы и для всех частиц в целом. После этого цикл повторяется.

Модификации классического алгоритма. Алгоритм роя частиц появился относительно недавно, однако различными исследователями уже был предложен целый ряд его модификаций, и новые работы на эту тему не перестают публиковаться. Можно выделить несколько путей улучшения классического алгоритма, реализованных в большинстве из них. Это соединение алгоритма с другими алгоритмами оптимизации, уменьшение вероятности преждевременной сходимости путем изменения характеристик движения частиц, а также динамическое изменение параметров алгоритма во время оптимизации. Ниже рассмотрены наиболее примечательные из модификаций.

Позже в том же 1995 году была опубликована статья Кеннеди и Эберхарта, в которой они назвали оригинальный алгоритм “GBEST”, поскольку он использует глобальное лучшее решение (global best) для формирования векторов скоростей, а также предложили его модификацию, названную ими “LBEST”. При обновлении направления и скорости движения частицы в LBEST используют информацию о решениях соседних с ними частиц[12]:

$$v_i = v_i + a_1 \cdot rnd() \cdot (pbest_i - x_i) + a_2 \cdot rnd() \cdot (lbest_i - x_i),$$

где $lbest$ – лучший результат среди частицы и ее соседей. Соседними считаются либо частицы, отличающиеся от данной индексом не более чем на некоторое заданное значение, либо частицы, расстояние до которых не превышает заданного порога.

Данный алгоритм более тщательно исследует пространство поиска, однако является более медленным, чем оригинальный. При этом, чем меньше число соседей учитывается при формировании

вектора скорости, тем ниже скорость сходимости алгоритма но тем эффективней он избегает субоптимальных решений.

В 1998 году Юхи Ши (Yuhui Shi) и Рассел Эберхарт предложили модификацию, на первый взгляд совсем незначительно отличающуюся от классического алгоритма [13]. В своей статье Ши и Эберхарт заметили, что одной из главных проблем при решении задач оптимизации является баланс между тщательностью исследованием пространства поиска и скоростью сходимости алгоритма. В зависимости от задачи и характеристик поискового пространства в ней, этот баланс должен быть различным.

С учетом этого, Ши и Эберхарт предложили изменить правило обновления векторов скоростей частиц:

$$v_i = w \cdot v_i + a_1 \cdot \text{rnd}() \cdot (pbest_i - x_i) + a_2 \cdot \text{rnd}() \cdot (gbest_i - x_i),$$

Коэффициент w , названный ими коэффициентом инерции, определяет упомянутый баланс между широтой исследования и вниманием к найденным субоптимальным решениям. В случае, когда $w > 1$, скорости частиц увеличиваются, они разлетаются в стороны и исследуют пространство более тщательно. В противном случае, скорости частиц со временем уменьшаются, и скорость сходимости в таком случае зависит от выбора параметров a_1 и a_2 .

В своей работе 1998 года, Ши и Эберхарт отметили, что инерция не обязательно должна быть положительной константой: она может изменяться во время работы алгоритма по линейному и даже нелинейному закону [13]. В статье 1999 года и более поздних работах они наиболее часто использовали линейный закон убывания, как достаточно эффективный и вместе с тем простой [14]. Тем не менее, разрабатывались и успешно применялись и другие законы изменения инерции.

Значение коэффициента инерции может как убывать, так и расти. При его убывании, частицы сначала исследуют область поиска экстенсивно, находя множество субоптимальных решений, и со временем все более концентрируются на исследовании их окрестностей. Возрастание инерции способствует сходимости алгоритма на поздних стадиях работы.

В 2002 году Марис Клер (Maurice Clerc) и Джеймс Кеннеди предложили свою модификацию алгоритма роя частиц, которая стала настолько популярной, что теперь ее принято называть каноническим алгоритмом роя частиц [15]. Он позволяет избавиться от необходимости «угадывать» подходящие значения регулируемых параметров алгоритма, контролируя сходимость частиц.

Клер и Кеннеди изменили способ вычисления векторов скоростей частиц, введя в него дополнительный множитель:

$$v_i = \chi \cdot [v_i + a_1 \cdot \text{rnd}() \cdot (pbest_i - x_i) + a_2 \cdot \text{rnd}() \cdot (gbest_i - x_i)],$$

где $a = a_1 + a_2 > 4$, а коэффициент сжатия χ равен:

$$\chi = \frac{2k}{|2 - a - \sqrt{a^2 - 4a}|}.$$

Такой подход гарантирует сходимость алгоритма без необходимости явно контролировать скорость частиц.

В своей работе 2004 года Руи Мендес (Rui Mendes), Джеймс Кеннеди и Жозе Невес (José Neves) заметили, что принятое в каноническом алгоритме роя частиц допущение о том, что на каждую из частиц влияет только наиболее успешная, не соответствует лежащим в его основе природным механизмам и, возможно, ведет к снижению эффективности алгоритма [16]. Они предположили, что из-за чрезмерного внимания алгоритма к единственному решению может быть потеряна важная информация о структуре пространства поиска.

Исходя из этого, они решили сделать все частицы «полностью информированными», то есть получающими информацию от всех соседних частиц. Для этого они изменили в каноническом алгоритме закон изменения скорости:

$$v_i = \chi \cdot \left[v_i + \sum_{k \in N} a_k \cdot W(k) \cdot \text{rnd}() \cdot (pbest_{k,i} - x_i) \right],$$

где N – множество соседей частицы, $pbest_k$ – лучшая из пройденных k -ым соседом точек. $W(k)$ – весовая функция, которая может отражать любую характеристику k -ой частицы, которая считается важной: значение целевой функции в точке, в которой она находится, дистанцию от нее до данной частицы и так далее.

4.4.6. ДНК – вычисления (DNA-computing)

(A – acid - кислота): позволяют сразу сгенерировать все возможные варианты решений путем синтеза соответствующих структур молекул с помощью известных биохимических реакций. Эти структуры отображают закодированные значения решений. Затем быстро находится (отфильтровывается) та молекула, в которой закодировано нужное решение.

Впервые ДНК – вычисления продемонстрировал Леонард Адлеман (профессор университета Южной Калифорнии) в 1994г. на примере «задачи о коммивояжере» с семью пунктами маршрута обхода (вершинами графа)[18-20].

Проблемы:

- трудоемкость проводимых реакций;
- трудность масштабирования задач: катастрофическое увеличение числа необходимых для представления всех возможных решений ДНК при росте числа вершин графа (так, при числе пунктов обхода около 200 масса необходимого количества ДНК превысит массу нашей планеты).

4.4.7. Реализация алгоритма ИИС для решения задач маршрутизации

Для решения задачи маршрутизации с помощью алгоритмов искусственной иммунной системы необходимо сопоставить биологические объекты и процессы их математическим аналогам [17].

Антигены, обозначающие в терминах иммунной системы вещества из внешней среды, соответствуют условиям задачи – набору вершин графа. В-лимфоциты соответствуют агентам, перемещающимся по вершинам графа, клонирующие и уничтожающие себя, используя алгоритмы положительного и отрицательного отбора. Агент начинает свой путь в начальной вершине, при каждой итерации алгоритма он имеет возможность клонировать себя. Попадая в вершину, которую он уже посещал, агент уничтожает себя согласно правилу положительного отбора. После завершения обхода по правилу отрицательного отбора выбирается агент с наименьшей длиной пути.

Очевидно, что популяция агентов будет расти экспоненциально. Чтобы этого избежать, введем параметр, ограничивающий максимальное количество агентов: клонирование новых агентов будет происходить только в том случае, если в популяции есть свободные места. Для сокращения количества бесполезных агентов, длина маршрутов которых превысили текущий наилучший результат до завершения пути, будем записывать текущую минимальную длину пути при создании нового агента. Если путь агента превышает это значение, он самоуничтожается.

Как правило, оптимальный путь в задаче маршрутизации состоит из ребер, соединяющих ближайшие вершины, поэтому

логичнее будет выбирать новые вершины не с равной вероятностью, а с зависящей от удаления от текущей (чем дальше, тем меньше вероятность). Агент, запущенный для прохождения пути повторно, содержит свой предыдущий маршрут.

Алгоритм действий одного агента в псевдокоде выглядит следующим образом:

1. Если маршрут уже проходил ранее - перейти в очередную вершину, в противном случае перейти в одну из вершин графа (вероятность выбора зависит от удаленности вершины от текущей);
2. Если агент уже был в этой вершине – самоуничтожиться;
3. Увеличить длину на размер пройденного пути;
4. Если длина превышает известный агенту минимум – самоуничтожиться;
5. Если в популяции есть свободные места - клонировать себя, изменив последнюю вершину у клона на случайную;
6. Перейти к шагу 1.

Анализ алгоритмов природных вычислений к решению задач оптимизации показал, что среди рассмотренных в данной работе алгоритмов, муравьиный алгоритм и алгоритм на искусственных иммунных системах, применительно к классической задачи маршрутизации показали хорошие результаты.

4.5. Решение задачи нечеткой многокритериальной оптимизации в условиях риска

Существует большое количество публикаций по принятию решений в условиях риска. Широко используются минимаксный подход, оптимизация ожидаемой полезности, минимизация среднего ущерба или вероятности неблагоприятного события, модели стохастического программирования и др. [1-5]. Вопросы формализации рисков в экономическом плане (оценка риска инвестиции, принятия проектов) с использованием нечетких множеств как средства преодоления статистической и лингвистической неопределенности рассматриваются в [6].

Однако этих постановок недостаточно для принятия решений в нечетких условиях, когда невозможно ориентироваться на средние показатели эффективности решений. Это оправдано в случае многократно повторяющихся ситуаций. Рисковые ситуации уникальны, они могут случиться и завтра, и не случиться никогда. Последние характеризуются возможностью крайне маловероятных, но исключительно больших потерь, граничащих с выживанием рассматриваемой системы. Ясно, что такие традиционные показатели риска, как дисперсия, в данном случае неадекватны.

Поэтому для оценки рисков в нечетких условиях систему ограничений стандартной задачи принятия решений предлагается дополнить набором ограничений по возможным потерям. При этом для избранных сценариев следует построить модель их последствий (ущербов) как функций управляющих параметров и накладывать экспертные ограничения по приемлемому уровню относительного ущерба для каждого сценария.

В практике разработки интеллектуальных систем имеется тенденция использования гибридных моделей для решения различных практических задач. Так, применительно к обучению нечетких моделей некоторые авторы, наряду с традиционными методами оптимизации, используют иммунный алгоритм [17]. Экспериментальные исследования, проведенные авторами, показывают, что использование данного подхода позволяет добиться лучших результатов по повышению скорости и точности решаемых задач. Таким образом, применительно к проблеме формирования баз знаний экспертных систем актуальной задачей является разработка эффективного алгоритма обучения нечёткой модели на основе интеллектуальных методов и эффективных эвристических алгоритмов.

4.5.1. Подходы к решению оптимизационной задачи

Рассматривается задача нечеткой многокритериальной оптимизации с критерием “максимальная доходность - минимальный риск”. Очевидно, что найти идеальный вариант решения в этом случае удаётся лишь в очень редких случаях. Поэтому предлагаются следующие подходы к решению такой оптимизационной задачи (табл.1).

Здесь $K = \{y \in R^m, y \leq g\}$ – заданное выпуклое подмножество пространства R^m :

$$\sum_j XK_j = \left\{ y \in R^m, y = \sum_{i=1}^n x_i a_{ij}, x_i \in X, a_{ij} \in K_j \subset R^m, i = 1, \dots, n; j = 1, \dots, m \right\}.$$

Таблица 4.2

Подходы к решению оптимизационной задачи

№	Подходы	Модель
1	Подход “максимум выигрыша” заключается в том, что из всех вариантов выбирается тот, который приводит к максимальному значению выигрыша (F) при приемлемом для лиц, принимающих решений (ЛПР) риске ($R_{np.don}$).	$F \rightarrow \max ,$ $R = R_{np.don} ,$ $\sum_j XK_j \subset K .$
2	Подход “оптимальная вероятность” состоит в том, что из возможных решений выбирается тот, при котором обеспечивается максимум математического ожидания выигрыша ($M(F)$) при приемлемом для ЛПР риске (R).	$M(F) \rightarrow \max ,$ $\sum_j XK_j \subset K .$
3	Сочетание подходов “оптимальная вероятность” и “оптимальная колеблемость” заключается в выборе варианта, обеспечивающего минимум коэффициент вариации ($CV(F)$) при приемлемом для ЛПР риске (R).	$CV(F) \rightarrow \min ,$ $\sum_j XK_j \subset K .$
4	Подход “минимум риска” состоит в выборе варианта, который позволяет получить ожидаемый выигрыш, т.е. предельно допустимое значение F при минимальном риске.	$F = F_{np.don} ,$ $R \rightarrow \min ,$ $\sum_j XK_j \subset K .$
5	Подход “максимальная доходность - минимальный риск” состоит в выборе варианта, обеспечивающего максимум F при минимуме R при удовлетворении заданным ограничениям.	$F \rightarrow \max ,$ $R \rightarrow \min ,$ $\sum_j XK_j \subset K .$

4.5.2. Решение задачи нечеткой многокритериальной оптимизации

В многокритериальных задачах сложно оценить решение задачи в комплексе всех критериев. Наиболее распространенным является метод аддитивной свертки и оценка лицом, принимающим решение (ЛПР) [7-9].

Предлагается применение нечетких методов для оценки альтернативных решений. Решение многокритериальной задачи оптимизации содержит следующие этапы[8-11]:

- Формирование целевой функции в нечеткой постановке.
- Определение значений критериев оценки в нечетком виде.
- Разработка функций принадлежности для критериев.
- Определение базы правил и/или базы предпочтений для критериев.
- Вычисление значений целевой функции.
- Деффазификация (приведение к четкому виду) целевой функции.

Обозначим через $F(X, \Lambda)$ операцию свертки частных критериев оптимальности, где $\Lambda \in D_\Lambda \subset R^s$ - вектор весовых множителей ($\Lambda = \{\lambda_i, i = \overline{1, s}\}$); $D_\Lambda = \{\lambda_i | \lambda_i \geq 0, \sum \lambda_i = 1, i \in [1: s]\}$ - множество допустимых значений этого вектора.

Задача параметрического программирования с s независимыми параметрами $\Lambda = \{\lambda_i, i = \overline{1, s}\}$, или S -задача параметрического программирования в матричном виде записывается следующим образом:

$$F(X, \Lambda) = (\bar{a}_0 + \Lambda \bar{b})X + \bar{e}\Lambda \rightarrow extr,$$

$$\sum_j XK_j \subset K,$$

$$\lambda \in R^s.$$

Здесь $X = \{x_j\}, j = \overline{1, n}$ – решение S -задачи параметрического программирования, $\bar{a}_0, \bar{b}, \bar{e}$ – коэффициенты, являющиеся нечеткими величинами, представляемыми обычно в виде нечетких множеств с заданными функциями принадлежности $\mu_{\bar{a}_0}(a_0) (\bar{a}_0 \subset A_0), \mu_{\bar{b}}(b) (\bar{b} \subset B)$ и $\mu_{\bar{e}}(e) (\bar{e} \subset E)$.

Для решения задачи параметрического программирования при нечетких исходных данных (коэффициентов $\bar{a}_0, \bar{b}, \bar{e}$) предлагается три подхода:

1. Используя различные операции дефаззификации над нечеткими множествами $\bar{a}_0, \bar{b}, \bar{e}$ (интегрирования, суммирования, осреднения и др.), можно получить нечеткие оценки значения коэффициентов a_0, b, e [8]. Тогда, введя их в S -задачу параметрического программирования вместо нечетких коэффициентов и записав ограничения в виде соответствующих неравенств, исходную задачу сведем к виду:

$$F(X, \Lambda) = (a_0 + \Lambda b)X + e\Lambda \rightarrow \text{extr}(\min),$$

$$\sum_j X K_j \leq g, \quad (4.30)$$

$$\lambda \in R^s.$$

Заметим, что в силу нечеткости описания коэффициентов \bar{a}_0 и \bar{b} оценка любого решения $x(\lambda) \in X$ (и, соответственно, значения функции $F(X, \Lambda)$ при $x = x(\lambda)$) представляет собой нечеткое подмножество числовой оси базового множества X .

2. Сведение решения исходной задачи к решению задач линейного программирования для каждого дискретного α - уровня [9].

В результате нечёткие ограничения запишутся в следующем интервальном виде:

$$P = \begin{cases} \sigma_\alpha(a_{i1})x_1 + \sigma_\alpha(a_{i2})x_2 + \dots + \sigma_\alpha(a_{in})x_n \subseteq \sigma_\alpha(b_i), & i = \overline{1, m}, \alpha = \overline{1, p}, \\ x_j \geq 0, & j = \overline{1, n} \end{cases}$$

Здесь $X = \{x_j\}, j = \overline{1, n}$ – решение многокритериальной задачи параметрического программирования на каждом дискретном α - уровне, $\sigma_\alpha(a_{i,j})$ и $\sigma_\alpha(b_i)$ - интервальные значения коэффициентов $a_{i,j}$ и b_i на каждом дискретном α - уровне.

3. Решение задачи многокритериальной оптимизации адаптивным методом [7]. Каждая итерация этих методов включает в себя фазу анализа, выполняемую ЛПР, и фазу расчетов, выполняемую системой многокритериальной оптимизации.

Прямой адаптивный метод решения многокритериальной задачи, который исследуется в данной работе, основан на предположении существования «функции предпочтений ЛПР» $F(X, \Lambda) = (a_0 + \Lambda b)X + e\Lambda$, которая определена на множестве D_X допустимых значений вектора варьируемых параметров X и выполняет отображение этого множества на множество действительных чисел R . При этом задача многокритериальной оптимизации сводится к задаче выбора вектора $X^* \in D_X$ ($X^* = \{x_j^*\}, j = \overline{1, n}$) такого, что

$$\min_X F(X, \Lambda) = F(X^*, \Lambda) \quad X \in D_X. \quad (4.31)$$

При каждом фиксированном векторе $\Lambda \in D_\Lambda$ метод скалярной свертки сводит решение задачи (4.30) к решению однокритериальной задачи глобальной условной оптимизации (4.31)

Отметим, что в случае аддитивной свертки $F(X, \Lambda)$ вектор X^* принадлежит множеству эффективных по Парето векторов [9].

Данное обстоятельство позволяет полагать, что в этом случае функция предпочтений ЛПР определена не на множестве D_X , а на множестве D_A :

$$F : \Lambda \rightarrow R .$$

В результате многокритериальная задача сводится к задаче выбора вектора $\Lambda^* \in D_\Lambda$ такого, что

$$\min_{\Lambda} F(X, \Lambda) = F(X, \Lambda^*), \quad \Lambda \in D_\Lambda . \quad (4.32)$$

Поскольку обычно $s \ll n$, переход от задачи (4.30) к задаче (4.32) важен с точки зрения уменьшения вычислительных затрат.

Вектор $\Lambda^* \in D_\Lambda$ находится с помощью нечетких правил вывода:

$$\bigcup_{p=1}^{k_j} \left(\bigcap_{i=1}^s \lambda_i = \Psi_{i,jp} - c \text{ весом } w_{jp} \right) \rightarrow F(X, \Lambda) = F(X, \Lambda^*) .$$

Здесь $\Psi_{i,jp}$ - лингвистический терм, которым оценивается переменная λ_i в строчке с номером jp ;

w_{jp} - весовой коэффициент правила с порядковым номером jp ;

$F(X, \Lambda) = F(X, \Lambda^*)$ - выход нечеткого правила.

Величину Ψ будем считать лингвистической переменной со значениями от «Очень-очень плохо» до «Отлично». Ядро нечеткой переменной Ψ обозначим $\dot{\Psi}$ [8], так что значению переменной Ψ «Очень-очень плохо» соответствует $\dot{\Psi} = 1$, а значению «Отлично» - $\dot{\Psi} = l$.

В результате многокритериальная задача сводится к задаче отыскания вектора $\Lambda^* \in D_\Lambda$, обеспечивающего максимальное значение дискретной функции $\dot{\Psi}(\Lambda)$:

$$\max_{\Lambda} \dot{\Psi}(\Lambda) = \dot{\Psi}(\Lambda^*) = \dot{\Psi}^* \quad \Lambda \in D_\Lambda . \quad (4.33)$$

Каждая входная переменная имеет свои собственные функции принадлежности нечетким термам Ψ_{jp} .

Функции принадлежности элемента λ_i терму Ψ_{jp} имеет следующий вид:

$$\mu^{jp}(\lambda_i) = \frac{1}{1 + \left(\frac{\lambda_i - b_i^{jp}}{c_i^{jp}} \right)^2},$$

где b_i^{jp}, c_i^{jp} - параметры функции принадлежности.

Общая схема рассматриваемого метода является итерационной и состоит из следующих основных этапов.

Этап 1. Случайно последовательно генерируется s векторов $\lambda_1, \lambda_2, \dots, \lambda_s$ и для каждого из этих векторов выполняются следующие действия:

1) решается многокритериальная задача

$$\min_X F(X, \Lambda) = F(X^*, \Lambda) \quad X \in D_X. \quad (4.34)$$

2) ЛПР предъявляет найденное решение X^* , а также соответствующие значения всех частных критериев оптимальности $f_1(X^*), f_2(X^*), \dots, f_s(X^*)$;

3) ЛПР оценивает эти данные и вводит в задачу соответствующее значение своей функции предпочтений $\dot{\Psi}(\Lambda_i)$.

Этап 2. На основе всех имеющихся значений $\lambda_1, \lambda_2, \dots, \lambda_s$ вектора Λ и соответствующих оценок функции предпочтений выполняются следующие действия:

1) строится функция $\tilde{F}_1(X, \Lambda)$, аппроксимирующая функцию $F(X, \Lambda)$ в окрестности точек $\lambda_1, \lambda_2, \dots, \lambda_s$;

2) отыскивается минимум функции $\tilde{F}_1(X, \Lambda)$

$$\min_{\Lambda} \tilde{\Psi}_1(\Lambda) = \tilde{\Psi}(\Lambda_1^*), \quad \Lambda \in D_{\Lambda};$$

3) с найденным вектором Λ_1^* решается задача вида (4.34) – находится вектор параметров и соответствующие значения частных критериев оптимальности, а затем предъявляется ЛПР; ЛПР оценивает

указанные данные и вводит в систему соответствующее значение своей функции предпочтений $F(\Lambda_1^*)$.

Этап 3. На основе всех имеющихся в системе значений $\lambda_1, \lambda_2, \dots, \lambda_s$ вектора Λ и соответствующих оценок функции предпочтений $F(X, \Lambda_1^*)$ выполняется аппроксимация функции $F(X, \Lambda)$ в окрестности точек $\Lambda_1, \Lambda_2, \dots, \Lambda_k, \Lambda_1^*$, строится функция $\tilde{F}_2(X, \Lambda)$ по схеме первого этапа до тех пор, пока ЛПР не примет решение о прекращении вычислений.

Входами системы нечеткого вывода являются значения весов частных критериев оптимальности – нечеткие термы $\lambda_i, i = 1, 2, \dots, k \in [1: s]$. Выходной переменной системы нечеткого вывода является лингвистическая переменная Ψ , ядро которой $\dot{\Psi}$ принимает значения $1, 2, \dots, l$.

Совокупность значений указанных нечетких входных переменных, выходных лингвистических переменных, а также правил нечетких продукций образуют нечеткую базу знаний.

4.5.3. Настройка нечетких баз знаний с применением иммунных алгоритмов оптимизации

Иммунные алгоритмы (ИА) оптимизации содержат следующие операторы: клонирование, мутация, старение и селекция. Рассмотрим их подробнее.

Алгоритм настройки параметров функций принадлежности $B = (b_1, b_2, \dots, b_q)$ и $C = (c_1, c_2, \dots, c_q)$ и веса правил $W = (w_1, w_2, \dots, w_N)$ состоит из следующих этапов.

1. Формирование исходной популяции *Оператор клонирования* генерирует новое поколение копий антител в будущей популяции. Известны следующие основные операторы клонирования: а) статический оператор клонирования, который просто копирует каждую В-клетку, производя переходную популяцию; б)

пропорциональный оператор клонирования, который клонирует В-клетки пропорционально их антигенной схожести; в) оператор вероятностного клонирования, в соответствии с которым В-клетки выбираются из текущего поколения в зависимости от вероятности клональной селекции.

Для реализации ИА следует задать способ кодирования нечетких моделей. Сведем неизвестные параметры W, B, C в один вектор:

$$S = (W, B, C) = (w_1, w_2, \dots, w_N, b_{11}, c_{11}, \dots, b_{1l_1}, c_{1l_1}, b_{n1}, c_{n1}, \dots, b_{nl_1}, c_{nl_1}),$$

где N - общее число строк в нечеткой базе знаний;

l_i - количество термов-оценок входной переменной λ_i ,
 $l_1 + l_2 + \dots + l_n = q, i = \overline{1, n}$;

q - общее число термов;

2. *Оператор мутации* действует в зависимости от имеющейся популяции клонов, применяя к каждому антители определенное количество одиночных мутаций, осуществляемых случайным образом.

Каждый элемент вектора S может подвергнуться операции мутации с вероятностью p_m . Обозначим мутацию элемента s через $Mu(s)$:

$$Mu(w_j) = RANDOM([\underline{w}, \overline{w}]), j = \overline{1, N},$$

$$Mu(b_{ip}) = RANDOM([\underline{x}_i, \overline{x}_i]),$$

$$Mu(c_{ip}) = RANDOM([\underline{c}_i, \overline{c}_i]),$$

где $\underline{w}, (\overline{w})$ - нижняя (верхняя) граница интервала возможных значений весов правил, $[\underline{w}, \overline{w}] \subset [0, 1]$; $[\underline{\tilde{n}}_i, (\overline{c}_i)]$ - интервал возможных значений коэффициента концентрации-растяжения функций принадлежности термов-оценок входной переменной x_i , $[\underline{\tilde{n}}_i, (\overline{c}_i)] \subset (0, +\infty]$, $i = \overline{1, n}$;

$RANDOM([\underline{\xi}, \bar{\xi}])$, обозначает операцию нахождения равномерно распределенного на интервале $[\underline{\xi}, \bar{\xi}]$ случайного числа.

3. *Оператор старения* устраняет старые особи. Статический оператор старения использует возрастной параметр для максимального количества поколений антител, которым разрешено оставаться в популяции. Когда антитело старше, оно удаляется из системы, даже если оно может оказаться вполне пригодным на последующих итерациях.

При клональной экспансии клонированное антитело наследует возраст его родителя. После этапа мутации только те антитела, которые получили высшее значение аффинности, получают возраст, равный 0. Элитный вариант этого оператора получается путем взятия наилучших антител популяции в поколение с возрастом, равным 0.

Элитный вариант этого оператора получается взятием лучшего антитела из популяции в поколение.

4. *Оператор селекции* заменяет наихудшие антитела в популяции новыми случайными антителами.

На основе описанного ИА разработано программное обеспечение для решения практических задач многокритериальной оптимизации и получены результаты оптимизационной задачи [10,11].

Таким образом, показана целесообразность объединения метода нечеткого вывода и иммунных алгоритмов в задачах с параметрами, содержащими неопределенности различного типа, а также в задачах, для которых характерны интуитивные решения. Предложенный метод позволяет существенно улучшить качество решения многокритериальных задач оптимизации с нечетко заданными параметрами и критериями.

4.6. Численное решение некорректно поставленных задач с применением природных алгоритмов

Типичным примером некорректно поставленной задачи является линейное операторное уравнение [30]

$$Az = u, \quad z \in Z, \quad u \in U,$$

в случае, когда оператор A вполне непрерывен. В этом случае, как известно, нарушаются оба условия корректности задачи по Адамару (здесь Z, U – метрическое пространство). Если Z – бесконечномерное пространство, то, во-первых, оператор A^{-1} определен не на всем U ($AZ \neq U$) и, во-вторых, A^{-1} (определенный на $AZ \subset U$) не является непрерывным.

К некорректно поставленным задачам относятся многие задачи теории оптимального управления, линейной алгебры, задача суммирования рядов Фурье с неточно заданными коэффициентами, задача минимизации функционалов и многие другие [30-34].

4.6.1. Решение интегральных уравнений Фредгольма 1-го рода рекуррентными нейронными сетями

Рассмотрим интегральное уравнение Фредгольма первого рода [30]:

$$Az = \int_a^b K(x, s)z(s)ds = u(x), \quad c \leq x \leq d. \quad (4.35)$$

Будем предполагать, что $K(x, s)$ – действительная функция, непрерывная в прямоугольнике $\Pi = \{a \leq s \leq b, c \leq x \leq d\}$. Будем также считать для простоты, что ядро K невырождено. Уравнение (4.35) запишем в операторном виде

$$Az = u, \quad z \in Z, \quad u \in U,$$

где Z, U – метрическое пространство. Дано отображение $A^{-1}: U \rightarrow Z$, заданное на подмножестве $D_{A^{-1}} = AZ \subset U$. Если отображение A^{-1} определено на всех U и непрерывно, то рассматриваемая задача

корректна по Адамару. В этом случае если вместо элемента $\bar{u} \in D_{A^{-1}}$ известно его приближенное значение – элемент $u_\delta \in U$ такой, что $\rho_U(u_\delta, \bar{u}) \leq \delta$, то в качестве приближенного значения элемента $z = A^{-1}(\bar{u})$ можно взять элемент $A^{-1}(u_\delta) \in Z$, причем $\rho_Z(A^{-1}(u_\delta), \bar{z}) \rightarrow 0$ при $\delta \rightarrow 0$.

Если задача некорректно поставлена, то элемент $A^{-1}(u_\delta)$ может вовсе не существовать, так как u_δ не обязательно принадлежит $D_{A^{-1}}$, а если и принадлежит, то $\rho_Z(A^{-1}(u_\delta), \bar{z})$, вообще говоря, не стремится к нулю при $\delta \rightarrow 0$.

Таким образом, рассматриваемую проблему можно трактовать как задачу о приближенном вычислении значений функции $A^{-1}(u)$ при неточно заданном аргументе u . Под приближенными данными \bar{u} будем понимать пару (u_δ, δ) такую, что $\rho_U(u_\delta, \bar{u}) \leq \delta$, причем элемент u_δ не обязательно принадлежит $D_{A^{-1}}$. Пусть заданы элементы u_δ и линейный оператор A_h такие, что $\|u_\delta - \bar{u}\|_U \leq \delta$, $\|A_h - A\| \leq h$. Таким образом, входной информацией является набор $\{u_\delta, A_h, \delta, h\}$. По этим данным требуется построить элемент $z_\eta \in Z$, $\eta = \{\delta, h\}$ такой, что $z_\eta \rightarrow \bar{z}$ при $\eta \rightarrow 0$.

Пусть z_η^α – экстремаль функционала $M^\alpha[z] = \|A_h z - u_\delta\|_U^2 + \alpha \|z\|_Z^2$, т.е. элемент, минимизирующий $M^\alpha[z]$ на Z . Если параметр регуляризации $\alpha = \alpha(\eta)$ согласован определенным образом $\eta = \{\delta, h\}$, то элемент $z_\eta^{\alpha(\eta)}$ и будет в определенном смысле решением задачи (4.35).

В описанной постановке функционал $M^\alpha[z]$ для задачи (4.35) имеет вид

$$\begin{aligned} \hat{M}^\alpha[z] &= \sum_{i=1}^m \left[\sum_{j=1}^n a_{ij} z_j h_s - u_i \right]^2 h_x + \\ &+ \alpha \sum_{j=1}^m z_j^2 h_s + \alpha \sum_{j=1}^n (z_j - z_{j-1})^2 / h_s. \end{aligned} \quad (4.36)$$

Для решения этой задачи построим рекуррентную нейронную сеть, которая описывается дифференциальным уравнением

$$\begin{aligned} \frac{\partial \hat{M}^\alpha[z]}{\partial z_1} &= h_x h_s \sum_{k=1}^n \left[\sum_{i=1}^m a_{ik} a_{i1} \right] z_k + \alpha h_s z_1 - \alpha \frac{z_2 - z_1}{h_s} - \sum_{i=1}^m a_{i1} u_i h_x h_s, \\ \frac{\partial \hat{M}^\alpha[z]}{\partial z_j} &= h_x h_s \sum_{k=1}^n \left[\sum_{i=1}^m a_{ik} a_{ij} \right] z_k + \alpha h_s z_j - \alpha \frac{z_{j-1} - 2z_j + z_{j-2}}{h_s} - \sum_{i=1}^m a_{ij} u_i h_x h_s, \\ &, \\ & j=2, 3, \dots, n-1, \\ \frac{\partial \hat{M}^\alpha[z]}{\partial z_n} &= h_x h_s \sum_{k=1}^n \left[\sum_{i=1}^m a_{ik} a_{in} \right] z_k + \alpha h_s z_n - \alpha \frac{z_{n-1} - z_n}{h_s} - \sum_{i=1}^m a_{in} u_i h_x h_s. \end{aligned}$$

Уровни активности входных нейронов зависят от получаемого входного вектора

$$y_j = f(z_j), \quad f(z) = \frac{1}{1 + \exp(-\beta z)},$$

где f – непрерывная гладкая функция активации. Пусть мы имеем множество обучающих пар векторов $\{Z^n, Y^m\}$ и задача заключается в построении по возможности более корректных отображений частных входных векторов Z_i^n на соответствующие выходные Y_i^m . Для проведения обучения нам надо определить меру корректности работы сети. Такой мерой может служить функционал $M^\alpha[z]$.

Таким образом, задачу обучения можно свести к задаче минимизации соотношения (4.36).

4.6.2. Решение интегральных уравнений Фредгольма 1-го рода с использованием генетических алгоритмов

При решении некорректных задач обычно необходимо аппроксимировать исходную, чаще всего бесконечномерную задачу некоторой конечномерной задачей, для которой и будут разрабатываться вычислительные алгоритмы.

Используя стандартную схему генетического алгоритма, мы получим приближенное решение $z_\eta^{\alpha(\eta)} \in Z = W_2^1[a, b]$, которое при $\eta \rightarrow 0$ сходится к \bar{z} в норме пространства $W_2^1[a, b]$.

Для решения этой задачи широко используется следующая конструкция. Рассмотрим функционал

$$M^\alpha[z] = \|A_h z - u_\delta\|_U^2 + \alpha \|z\|_Z^2.$$

Пусть z_η^α - экстремаль функционала $M^\alpha[z]$, т.е. элемент, минимизирующий $M^\alpha[z]$ на Z . Если параметр регуляризации $\alpha = \alpha(\eta)$ согласован определенным образом $\eta = \{\delta, h\}$, то элемент $z_\eta^{\alpha(\eta)}$ и будет в определенном смысле решением задачи (1).

В описанной постановке функционал $M^\alpha[z]$ для задачи (4.35) имеет вид

$$\begin{aligned} M^\alpha[z] &= \|A_h z - u_\delta\|_{L_2}^2 + \alpha \|z\|_{W_2^1}^2 = \\ &= \int_c^d \left[\int_a^b K_h(x, s) z(s) ds - u_\delta(x) \right]^2 dx + \alpha \int_a^b \{z^2(s) + [z'(s)]^2\} ds. \end{aligned} \quad (4.38)$$

При построении конечно-разностной аппроксимации прежде всего выберем сетки $\{s_j\}_{j=1}^n$ и $\{x_i\}_{i=1}^m$ соответственно на отрезках $[a, b]$ и $[c, d]$. После этого, пользуясь какой-либо квадратурной формулой, можем построить конечно-разностный аналог оператора A интегрального уравнения (4.35). Конечно-разностный оператор

является линейным оператором с матрицей $A = \{a_{ij}\}$. Простейший вариант аппроксимации, которым мы будем пользоваться, дается формулами

$$\begin{aligned} a_{ij} &= K_h(x_i, s_j), \quad j = 2, \dots, n-1; \\ a_{ij} &= K_h(x_i, s_j), \quad j = 1, n; \quad i = 1, 2, \dots, m. \end{aligned}$$

Теперь легко можно выписать функционал, аппроксимирующий (4.36):

$$\begin{aligned} \hat{M}^\alpha[z] &= \sum_{i=1}^m \left[\sum_{j=1}^n a_{ij} z_j h_s - u_i \right]^2 h_x + \\ &+ \alpha \sum_{j=1}^m z_j^2 h_s + \alpha \sum_{j=1}^n (z_j - z_{j-1})^2 / h_s. \end{aligned}$$

Для нахождения экстремума функционала $\hat{M}^\alpha[z]$, т.е. элемента z_η^α , минимизирующего $\hat{M}^\alpha[z]$ на Z , использован генетический алгоритм. Генетический алгоритм представляет собой метод эволюционного моделирования, применяемый для решения задач с большой размерностью. Задача состоит в минимизации функции $\hat{M}^\alpha[z]$ на области допустимых решений D в пространстве решений Z . Поиск решений задачи в генетическом алгоритме ведется с помощью популяции особей, причем каждая особь рассматривается как пара генотипа g и отображения z (фенотипа $(z(g))$), которое соответствует некоторой точке пространства Z . Работа алгоритма представляет собой последовательную смену популяций, состоящих из фиксированного числа особей, соответствующих точкам пространства решений. При создании очередной популяции более пригодные особи оставляют больше потомков, причем часть потомков будет идентична родителям, а часть - претерпевает изменения в результате мутации. При реализации генетического алгоритма для поставленной выше задачи очень важно знать значение целевой функции на лучшем генотипе. Если оно в течение определенного количества операций

остается неизменным, то это будет свидетельствовать о достижении оптимального решения [34].

4.6.3. Решение интегральных уравнений типа свертки с использованием генетических алгоритмов

Рассмотрим двумерное интегральное уравнение типа свертки

$$Az = \int_a^A \int_b^B K(x-s, y-t)z(s,t)dsdt = u(x, y). \quad (4.39)$$

Пусть точное решение $\bar{z}(s,t)$ и ядро $K(u,w)$ имеют локальные носители:

$$\begin{aligned} \text{supp } K(u, w) &\subseteq [l_1, L_1] \times [l_2, L_2], \\ \text{supp } \bar{z}(s, t) &\subseteq [a, A] \times [b, B]. \end{aligned}$$

Тогда для $\bar{u}(x, y)$ имеем $\text{supp } p\bar{u}(x, y) \subseteq [c, C] \times [d, D]$, где $c = a + l_1$, $C = A + L_1$, $d = b + l_2$, $D = B + L_2$. Будем считать, что приближенная правая часть уравнения (4.35) - $u_\delta(x, y)$ имеет локальный носитель $[c, C] \times [d, D]$, а приближенное решение этого уравнения имеет локальный носитель $[a, A] \times [b, B]$.

Рассмотрим функционал

$$M^\alpha[z] = \|A_h z - u_\delta\|_{L_2}^2 + \alpha \|z\|_{W_2^2}^2.$$

Для любого $\alpha > 0$ существует единственная экстремаль функционала z_η^α реализующая минимум $M^\alpha[z]$.

Запишем конечно-разностную аппроксимацию функционала $M^\alpha[z]$ для уравнения (1):

$$\begin{aligned} \hat{M}^\alpha[z] &= \sum_{k=0}^{n_1-1} \sum_{l=0}^{n_2-1} \left(\sum_{p=0}^{n_1-1} \sum_{j=0}^{n_2-1} K_{k-p, l-j} z_{pj} \Delta x \Delta y - u_{kl} \right)^2 \Delta x \Delta y + \\ &+ \alpha \sum_{k=0}^{n_1-1} \sum_{l=0}^{n_2-1} \left\{ z_{kl}^2 + \left[\frac{\partial^2 z(s_k, t_j)}{\partial s^2} \right]^2 + 2 \left[\frac{\partial^2 z(s_k, t_j)}{\partial s \partial t} \right]^2 + \left[\frac{\partial^2 z(s_k, t_j)}{\partial t^2} \right]^2 \right\} \Delta x \Delta y. \end{aligned}$$

Задача состоит в минимизации функции $\hat{M}^\alpha[z]$ на области допустимых решений D в пространстве решений Z . Поиск решений

задачи в генетическом алгоритме ведется с помощью популяции особей. Каждая особь рассматривается как пара генотипа g и отображения z ($z(g)$), которое соответствует некоторой точке пространства Z .

Генотип содержит всю необходимую информацию о соответствующем решении из Z . На основе генотипа в природе формируются все внешние и внутренние признаки особи – фенотипы. В генетическом алгоритме совокупности всех таких признаков соответствует некоторое решение из Z . В предлагаемом алгоритме генерация потомка начинается с выбора родительской пары $\langle g_z(t), g_m(t) \rangle \in \Pi^t \times \Pi^e$ при помощи оператора Sel . Этот оператор имеет вероятностную природу, отдавая предпочтение более пригодным особям.

К выбранным генотипам с фиксированной вероятностью P_c применяется оператор кроссинговера, заменяющий часть генов одного родителя генами другого. К полученному генотипу g или h применяется оператор мутации. Этот оператор случайным образом заменяет гены выбранного генотипа на некоторые случайные символы, т.е. мутация представляется в виде случайной величины $Mut(g)$, с распределением, зависящим от g . Полученный после этого генотип заменяет генотип с наихудшим значением $\hat{M}^\alpha[z]$, и процесс повторяется. Критерием остановки процесса служит достижение требуемого уровня значения $\hat{M}^\alpha[z]$ или заданного количества итераций, а результатом алгоритма служит лучшая особь популяции. Для определения генетического алгоритма необходимы следующие компоненты: 1) Алфавит, из которого будет строиться генотип. 2) Отображение пространства точек решений в пространство генотипов. 3) Функция пригодности. 4) Операторы ГА.

Огромное влияние на скорость и качество решений оказывают параметры генетического алгоритма: мощность популяции и параметры операторов. Обычно они подбираются эмпирически для данного класса задач.

4.7. Принятие слабоструктурированных решений в задачах селекции на основе нечеткого интеграла

4.7.1. Вычисление нечеткой меры

Известно, что мера, в классическом математическом понимании, обладает фундаментальным свойством, называемым аддитивностью: число, которое служит мерой нескольких объединенных характеристик, должно равняться сумме чисел, являющихся мерами соответствующих характеристик. Очевидно, что использование лингвистических переменных непосредственно в математических моделях нарушает предположение об аддитивности мер, то есть возникает проблема изучения и применения мер, свободных от требования аддитивности[1].

Пусть Γ множество элементов x_1, x_2, \dots, x_n , $\Gamma = \{x_1, x_2, \dots, x_n\}$. Конструкция Γ зависит от конкретной задачи. Например, x_1, x_2, \dots, x_n - урожайность хлопчатника при различных погодных условиях $\Gamma = \{25, 30, 35\}$. На Γ построим все возможные подмножества, которые обозначим через $\beta(\Gamma)$. Для данного примера элементами множества $\beta(\Gamma)$ являются: $A_1 = \{25\}$; $A_2 = \{30\}$; $A_3 = \{35\}$; $A_4 = \{25; 30\}$; $A_5 = \{25; 35\}$; $A_6 = \{30; 35\}$; $A_7 = \Gamma = \{25; 30; 35\}$; $A_8 = \{\emptyset\}$;

Основное свойство $\beta(\Gamma)$ состоит в том, что оно замкнуто относительно операций объединения, дополнения и пересечения: $A_i \cup A_j \in \beta(\Gamma)$, $\bar{A}_i \in \beta(\Gamma)$, $A_i \cap A_j \in \beta(\Gamma)$. Например, $A_1 \cup A_2 = A_4 = \{25; 30\} \in \beta(\Gamma)$.

Пусть на множестве $\beta(\Gamma)$ вводится нечеткая мера, которая является функцией множества.

Определение 1. Нечеткой мерой называется функция множества g , заданная на множестве $\beta(\Gamma)$ и удовлетворяющая следующим условиям [9]:

1. Ограниченность. $g(\emptyset) = 0, g(\Gamma) = 1; \Gamma = \{x_1, x_2, \dots, x_n\}$
2. Монотонность. Если $A, B \in \beta(\Gamma), A \subseteq B$, то $g(A) \leq g(B)$;
3. Непрерывность. Пусть последовательность $\{A_i\} \in \beta(\Gamma), 1 \leq i \leq \infty$, тогда если $A_i \supseteq A_z \supseteq \dots \supseteq A_n \supseteq \dots$, то $g(\bigcap_{i=1}^{\infty} A_i) = \lim_{i \rightarrow \infty} g(A_i)$.

Тройка $(\Gamma, \beta(\Gamma), g)$ называется пространством с нечеткой мерой.

Sugeno предложил сконструировать нечеткую меру объединения $g(A \cup B)$ следующим образом [9,10]:

$$g(A \cup B) = g(A) + g(B) + \lambda g(A)g(B).$$

Здесь параметр λ принимает значения в интервале $-1 < \lambda < \infty$, при этом $A \cap B = \emptyset$. Приведенное выражение называется λ -правилом, а нечеткая мера g_λ , соответственно, λ -мерой Sugeno.

Рассмотрим подробнее специальный случай, когда в качестве Γ использоваться конечное множество $\Gamma = \{x_1, x_2, \dots, x_n\}$. Нечеткая мера g_λ в этом случае будет строиться с использованием λ -меры нечеткой плотности, обозначаемой как $g(x_i) = g_\lambda(\{x_i\}), i = 1, 2, \dots, n$. Далее для нечеткой плотности будем использовать обозначение $g_i = g(x_i)$.

При условии, что заданы нечеткие плотности $0 \leq g_i \leq 1$, мера g_λ будет строиться согласно λ -правилу: $g_\lambda(\{x_i, x_j\}) = g_i + g_j + \lambda g_i g_j$.

Обобщая, можно записать, что: $g_\lambda(\{x_1 \dots x_k\}) = \sum_j g_j + \lambda \sum_{i=1}^{k-1} \sum_{i_2=i_1+1}^k g_{i_1} g_{i_2} + \dots + \lambda^{k-1} g_1 g_2 \dots g_k$.

Или, в эквивалентной форме:

$$g_\lambda(\{x_1, \dots, x_k\}) = \begin{cases} \frac{1}{\lambda} \left(\prod_{i=1}^k (1 + \lambda g_i) - 1 \right), & \lambda \neq 0, \\ \sum_{i=1}^k g_i, & \lambda = 0. \end{cases}$$

Если $-1 < \lambda < 0$, то имеет место $\sum_{i=1}^n g_i > 1$, а в случае $0 < \lambda < \infty$,

всегда имеет место $\sum_{i=1}^n g_i < 1$.

Кроме того, если значение какой-нибудь одной из нечетких плотностей равно единице, то значения остальных нечетких плотностей всегда равны нулю, т.е. если существует такое x_j , что $g_j = 1$, то для каждого $x_i \neq x_j$, обязательно, $g_i = 0$.

С учетом вышеизложенного, опишем алгоритм построения нечеткой меры:

Шаг 1. Из условия нормировки $\frac{1}{\lambda} \left(\prod_{i=1}^n (1 + \lambda g_i) - 1 \right) = 1$ вычисляем параметр λ ;

Шаг 2. Мету любого множества $A \in \beta(\Gamma)$ определяем из соотношения $g_\lambda(A) = \frac{1}{\lambda} \left(\prod_{x \in A} (1 + \lambda g_x) - 1 \right)$, которое удовлетворяет λ - правилу Sugeno.

Определение 2. Нечеткий интеграл Sugeno от функции h на множестве U по нечеткой мере g определяется по выражению [9]

$$\int h \circ g = \sup_{\alpha} (\alpha \wedge g(F_\alpha)), \quad F_\alpha = \{x \in U : h(x) \geq \alpha\}, \quad \alpha \in (0,1].$$

Здесь символ \int означает нечеткий интеграл, маленький кружок

o - знак композиции.

Отметим, что если интегрирование производится на множестве $A \subseteq U$, тогда нечеткий интеграл определяется по выражению

$$\int_A h(x) \circ g = \sup_{\alpha} (\alpha \wedge g(A \cap F_{\alpha})), \alpha \in (0,1]$$

Введем определение нечеткого интеграла для случая $U = \{x_1, x_2, \dots, x_n\}$. Если $h(x_1) \leq \dots \leq h(x_n)$, тогда нечеткий интеграл определяется по выражению

$$\int h(x) \circ g = \bigcup_{i=1}^n (h(x_i) \wedge g(E_i)),$$

$$E_i = \{x_i, \dots, x_n\}, \bigcup_{i=1}^n a_i = \max\{a_i\}$$

Если $h(x_1) \geq \dots \geq h(x_n)$, тогда нечеткий интеграл определяется по выражению

$$\int h(x) \circ g = \bigcup_{i=1}^n h(x_i) \wedge g(E_i), \quad E_i = \{x_1, \dots, x_i\}$$

Таким образом, нечеткий интеграл от функции h по нечеткой мере g представляет собой общую оценку в виде нелинейной сверткичастных оценок информационных элементов, и при этом не исключается возможность взаимосвязи информационных элементов [10].

4.7.2. Вычисление нечеткого интеграла

Вычисление нечеткого интеграла состоит из двух частей: определение параметра λ и вычисление собственно самого интеграла по заданной нечеткой плотности.

1. Параметр λ , по определению [9], принадлежит области $(-1, \infty)$. Равенство $\lambda = 0$ означает, что выполняется условие аддитивности - вероятностной мере $\sum_j g(d_j) = 1$.

В случае, когда арифметическая сумма нечетких плотностей $\sum_j g(d_j) \in (0,1)$, либо > 1 , то $\lambda \neq 0$.

Числовое значение параметра λ находится из решения уравнения

$$\frac{1}{\lambda} \left[\prod_{i=1}^m (1 + \lambda \cdot (\omega_i)) - 1 \right] = 1, \quad m = 2^N.$$

2. Вычисление самого интеграла по заданной нечеткой плотности

$$\int h(x) \circ g = \bigcup_{i=1}^n (h(x_i) \wedge g(E_i)),$$

$$E_i = \{x_i, x_{i+1}, \dots, x_n\}, E_1 \supset E_2 \supset \dots \supset E_n, E_1 = U$$

Важность каждого признака задана и выражается через нечеткие плотности

$$g_1 = 0,66, \quad g_2 = 0,89, \quad g_3 = 0,96, \quad g_4 = 0,93 \quad h_1 = 0,19, \quad h_2 = 0,21, \quad h_3 = 0,22, \quad h_4 = 0,24$$

При условии, что заданы нечеткие плотности $0 \leq g_i \leq 1$, мера g_λ строится согласно λ -правилу:

$$g_\lambda(\{x_1, \dots, x_k\}) = \begin{cases} \frac{1}{\lambda} \left(\prod_{i=1}^k (1 + \lambda g_i) - 1 \right), & \lambda \neq 0 \\ \sum_{i=1}^k g_i, & \lambda = 0 \end{cases}$$

$$g_\lambda(x_1, x_2, x_3, x_4) = 1.$$

Из условия нормировки $\frac{1}{\lambda} \left(\prod_{i=1}^n (1 + \lambda g_i) - 1 \right) = 1$ вычисляем параметр λ .

$$g_1 g_2 g_3 g_4 \lambda^3 + (g_1 g_2 g_3 + g_1 g_2 g_4 + g_1 g_3 g_4 + g_2 g_3 g_4) \lambda^2 + (g_1 g_2 + g_1 g_3 + g_1 g_4 + g_2 g_3 + g_2 g_4 + g_3 g_4) \lambda + g_1 + g_2 + g_3 + g_4 = 1.$$

$$0,524 \lambda^3 + 2,49 \lambda^2 + 4,409 \lambda + 2,44 = 0.$$

$$\lambda^3 + 4,75 \lambda^2 + 8,41 \lambda + 4,66 = 0.$$

$$\lambda = -0,96.$$

Исходя из λ -правила Sugeno и учитывая заданные плотности вычисляются меры всех подмножеств $\beta(\Gamma)$ и этим завершается построение меры Sugeno:

$$g_\lambda(x_1, x_2, x_3) = g_1 g_2 g_3 \lambda^2 + (g_1 g_2 + g_1 g_3 + g_2 g_3) \lambda + g_1 + g_2 + g_3 = -0,96^2 \times 0,66 \times 0,89 \times 0,96 - (0,66 \times 0,89 + 0,66 \times 0,96 + 0,89 \times 0,96) \times 0,96 + 0,66 + 0,89 + 0,96 = 1,03$$

$$g_\lambda(x_1, x_2, x_4) = g_1 g_2 g_4 \lambda^2 + (g_1 g_2 + g_1 g_4 + g_2 g_4) \lambda + g_1 + g_2 + g_4 = -0,96^2 \times 0,66 \times 0,89 \times 0,93 - (0,66 \times 0,89 + 0,66 \times 0,93 + 0,89 \times 0,93) \times 0,96 + 0,66 + 0,89 + 0,93 = 1,04$$

$$g_\lambda(x_1, x_3, x_4) = g_1 g_3 g_4 \lambda^2 + (g_1 g_3 + g_1 g_4 + g_3 g_4) \lambda + g_1 + g_3 + g_4 = -0,96^2 \times 0,66 \times 0,96 \times 0,93 - (0,66 \times 0,96 + 0,66 \times 0,93 + 0,96 \times 0,93) \times 0,96 + 0,66 + 0,96 + 0,93 = 1,05$$

$$g_\lambda(x_2, x_3, x_4) = g_2 g_3 g_4 \lambda^2 + (g_2 g_3 + g_2 g_4 + g_3 g_4) \lambda + g_2 + g_3 + g_4 = -0,96^2 \times 0,89 \times 0,96 \times 0,93 - (0,89 \times 0,96 + 0,89 \times 0,93 + 0,96 \times 0,93) \times 0,96 + 0,89 + 0,96 + 0,93 = 1,042$$

$$g_\lambda(x_1, x_2) = g_1 g_2 \lambda + g_1 + g_2 = -0,96 \times 0,66 \times 0,89 + 0,66 + 0,89 = 0,99'$$

$$g_\lambda(x_1, x_3) = g_1 g_3 \lambda + g_1 + g_3 = -0,96 \times 0,66 \times 0,96 + 0,66 + 0,96 = 1,02'$$

$$g_\lambda(x_1, x_4) = g_1 g_4 \lambda + g_1 + g_4 = -0,96 \times 0,66 \times 0,93 + 0,66 + 0,93 = 1,01'$$

$$g_\lambda(x_2, x_3) = g_2 g_3 \lambda + g_2 + g_3 = -0,96 \times 0,89 \times 0,96 + 0,89 + 0,96 = 1,03'$$

$$g_\lambda(x_2, x_4) = g_2 g_4 \lambda + g_2 + g_4 = -0,96 \times 0,89 \times 0,93 + 0,89 + 0,93 = 1,02'$$

$$g_\lambda(x_3, x_4) = g_3 g_4 \lambda + g_3 + g_4 = -0,96 \times 0,96 \times 0,93 + 0,96 + 0,93 = 1,05'$$

Ставится задача получения интегральной оценки выбранной стратегии с помощью нечеткого интеграла $\int h \circ g$.

$$\begin{aligned} h_1 = 0,19, \quad h_2 = 0,21, \\ h_3 = 0,22, \quad h_4 = 0,24 \end{aligned}$$

$$i=1: h(x_1) \wedge g(x_1, x_2, x_3, x_4) = 0,19 \wedge 1,0 = 0,19,$$

$$i=2: h(x_2) \wedge g(x_2, x_3, x_4) = 0,21 \wedge 1,042 = 0,21,$$

$$i=3: h(x_3) \wedge g(x_3, x_4) = 0,22 \wedge 1,05 = 0,22,$$

$$i=4: h(x_4) \wedge g(x_4) = 0,24 \wedge 0,93 = 0,24,$$

$$\begin{aligned} \int h \circ g &= \bigcup_{i=1}^4 (h(x_i) \wedge g(E_i)) = \\ &= \max(0,19; 0,21; 0,22; 0,24) = 0,24 \end{aligned}$$

$$x_4 = 0,24.$$

Результаты ранжирования всех селекционных сортов показали, что сорт 108-Ф является наилучшим среди предложенных селекционных сортов хлопчатника, поскольку результирующее значение степени принадлежности этого сорта нечеткому множеству является наибольшим (0,24).

Таким образом, рассмотрели алгоритм решения задачи на основе нечеткого интеграла на примере выбора наилучшего сорта хлопчатника, обеспечивающего оптимальные значения агротехнологических параметров в различных условиях: посева, выращивания, вегетации и уборки.

4.8. Как рисовать с помощью нейросетей

Рисование в нейросетях сегодня набирает все большую популярность и количество людей, которые следят за развитием искусственного интеллекта неуклонно растет.

О Midjourney говорит весь интернет: искусство, которое творит эта нейросеть, легко спутать с произведениями профессиональных художников. При этом стоит труд искусственного интеллекта в тысячи раз дешевле, а время создания картин измеряется секундами.

Как устроена эта дьявольская машинка изнутри, кто ее придумал и зачем миру живые дизайнеры и художники, если есть Midjourney?

4.8.1. Midjourney

Midjourney — разработанная одноименной американской компанией нейросеть, которая поразила пользователей по всему миру картинами (иначе назвать это невозможно), созданными на основе текстовых запросов. В феврале 2022 года проект основал ученый и предприниматель Дэвид Хольц — 33-летний выпускник факультета прикладной математики Университета Северной Каролины в Чапел-Хилл. В студенчестве Хольц успел поработать в Институте Макса Планка, где изучал алгоритмы нейровизуализации и разрабатывал карту мозга крысы на клеточном уровне, и даже в исследовательском центре NASA, где занимался технологией LiDAR (обнаружение и определение дальности с помощью света, используется, в том числе, в камерах iPhone для лучшей фокусировки на объекте).

В 2011 году молодой ученый ушел из аспирантуры и переехал в Сан-Франциско, где основал первую собственную компанию — Leap Motion (сейчас Ultraleap), которая разрабатывает датчики движения и другие системы распознавания человеческих жестов. Хольц руководил фирмой вплоть до 2021 года, но решил выйти

из бизнеса: ему, по собственным словам, не хотелось управлять крупной компанией, — в интервью The Verge Хольц признался, что ему интересна другая, молодая и быстроразвивающаяся среда.

В начале 2022 года Хольц вышел из состава учредителей Ultraleap и основал Midjourney. По словам предпринимателя, штат компании даже сейчас, когда ее настигла мировая слава, не превышает 10 человек, у проекта нет инвесторов, а деньги не являются главной мотивацией основателя. «Главное, чего я хочу, — чтобы на ближайшие 10 лет у нас был дом, в котором мы могли бы экспериментировать с технологиями и создавать продукты, которые будут иметь значение не только для меня, но и для всего мира. Ну и получать удовольствие в процессе», — говорит он.

Как это работает

Работу Midjourney обеспечивают два технологических прорыва в области искусственного интеллекта, произошедших относительно недавно: умение нейросетей понимать человеческую речь и создавать образы. Чтобы преобразовать два этих скилла в стройную систему, которая по запросу выдает произведения искусства, нейронную сеть обучают выстраивать соответствие между текстовыми описаниями и визуальными образами на сотнях миллионов примеров. Результаты такого обучения позволяют решать различные кросс-модальные задачи — генерацию картинок по текстовому описанию, генерацию текстовых описаний по картинкам, дорисовку частей изображения, и так далее, говорит руководитель управления экспериментальных систем машинного обучения SberDevices Сергей Марков. «Midjourney — это диффузионная нейронная сеть и состоит как бы из двух нейросетей: первая отвечает за обработку и понимание текста, вторая — за генерацию изображения», — объясняет Марков.

В середине июля Midjourney вышла в фазу бета-тестирования и стала доступна пользователям по всему миру. Правда, чтобы дать задание Midjourney, нужно быть зарегистрированным в Discord — кросс-платформенном мессенджере, популярном у геймеров, разработчиков игр и дизайнеров. Сначала нужно зайти на официальный сайт Midjourney и авторизоваться через Discord, затем оплатить подписку или воспользоваться бесплатной версией. Бесплатная версия позволяет сгенерировать и скачать 12 изображений, но не дает доступ к личному кабинету (это мешает отслеживать судьбу своих запросов в общем чате), за \$10 можно создавать до 200 изображений в месяц, за \$30 — генерировать бесконечное количество картинок. Доступна также корпоративная подписка стоимостью \$600, которая дает сотрудникам компании возможность создавать картинки в команде и просматривать индивидуальные работы друг друга.

По словам Хольца, способ доступа к системе через Discord он выбрал из-за группового принципа работы платформы: люди охотнее фантазируют, когда собираются в группы, считает Хольц. Присоединившись к сервису, можно отправлять текстовые команды для создания изображений вместе с другими пользователями или поодиночке на любом из множества каналов Discord.

Чтобы создать изображение, достаточно внести в чат с ботом Midjourney слова, описывающие картину, которую в итоге хочется получить. Система сгенерирует четыре изображения на выбор, а дальше наиболее подходящую картинку можно масштабировать, изменять и дорабатывать до идеала.

Полученные изображения появляются в общем канале Discord примерно через минуту после отправки запроса. Обладатели платной подписки могут отправлять боту команды в формате личных

сообщений, а не через общедоступный канал. Но сгенерированные нейросетью изображения по умолчанию остаются общедоступными для просмотра.

Ничего оригинального

Midjourney не первая и не единственная нейросеть, способная генерировать изображения на основе текстовых запросов, — до этого мир потрясли DALL-E от OpenAI и Imagen от Google.

Разработку DALL-E обеспечил грант в \$1 млрд от Microsoft, который корпорация выделила проекту OpenAI, специализирующемуся на технологиях искусственного интеллекта. На эти средства OpenAI разработал нейросеть на базе третьего поколения алгоритма обработки естественного языка GPT-3, который компания создала годом ранее. Для широкой аудитории нейросеть стала доступна через два года — в январе 2021-го. А в апреле 2022 года вышла ее улучшенная версия — DALL-E 2.

В ноябре 2021 года команда разработчиков Sber AI создала аналогичную нейросеть, которая обрабатывает запросы на русском языке. Проект получил название ruDALL-E, но отношения к оригинальному DALL-E не имеет: разработки засекречены, и российские специалисты самостоятельно обучали нейросеть с нуля. Разработчики заявляют, что процесс обучения ruDALL-E стал самой большой вычислительной задачей в России.

В мае 2022 года нейросеть Imagen, генерирующую изображения, выпустил Google. На официальном сайте продукта разработчик утверждает, что Imagen превосходит DALL-E по точности изображений и соответствию картинки техническому заданию.

Никаких технологических «киллер-фичей» у Midjourney нет, уверен директор по науке и технологиям консалтинговой компании «Агентство Искусственного Интеллекта» Роман Душкин. По его

словам, все перечисленные системы работают по общему принципу: человек дает как можно более конкретное текстовое описание той картинки, которую он хочет получить. Благодаря технологии глубокого обучения, нейросеть понимает смысл текста и преобразует его в картинку. Общая модель работы нейросетей такого типа открыта и понятна: о ней пишут подробные статьи в научных журналах. Разница и конкурентное преимущество — в индивидуальных технологических особенностях, которые влияют на пользовательский опыт и которые разработчики не раскрывают. Например, Midjourney обеспечивает более высокое разрешение итоговых картинок и большую детализацию в соответствии с текстовым запросом, отмечает руководитель лаборатории искусственного интеллекта в новых материалах центра в сфере ИИ на базе университета «Иннополис» Руслан Лукин.

Но все же главный «изюм» Midjourney в том, что ею может воспользоваться любой желающий, в отличие от других подобных нейросетей, доступ к которым возможен только по приглашениям или платно, считает исследователь разработчика ИИ-продуктов DeepPavlov.ai Александр Чернявский. «Среди всех генерирующих картинки из текста нейросетей это первая широкодоступная и частично бесплатная сеть, которая благодаря этой особенности „завирусилась“ в интернете», — говорит он. DALL-E 2 тоже можно использовать свободно, но доступ к ней открыли совсем недавно, а Midjourney изначально пошла по пути массового использования и не прогадала, добавляет Сергей Марков из Sber AI:

«Пути хайпа не всегда поддаются рациональному анализу, но, безусловно, Midjourney сегодня — это одна из лучших моделей для создания изображений по текстовым описаниям среди доступных широкой публике».

По его прогнозу, несмотря на преимущества, хайп вокруг Midjourney продлится недолго: через какое-то время ей на смену

придут еще более совершенные модели, так как прогресс в этой области сейчас очень быстрый, считает эксперт.

Как на этом заработать

Зачем нам нужна Midjourney и ее аналоги? Исключительно для развлечения или для чего-то еще?

По мнению Романа Душкина из «Агентства Искусственного Интеллекта», у функции визуализации нейросетей большой спектр практического применения, начиная от маркетплейсов, где нужно генерировать множество однотипных картинок для представления товаров на сайте, до создания художественных произведений и генерации NFT. Руководитель группы исследований компании по разработке технологий искусственного интеллекта Just AI Мурат Апишев к этому списку добавляет генерацию маркетингового и внутриигрового контента, дизайна, анимации и даже кино. Сергей Марков убежден, что подобные Midjourney системы найдут применение в интерьерном дизайне и дизайне упаковок, генерации прототипов веб-страниц, элементов интерфейсов, эмодзи и стикеров, в гейм-индустрии для создания персонажей или целых миров, в архитектуре, разработке логотипов и фирменного стиля, дизайне одежды, боди-арте, ивент-индустрии, иммерсивных театрах — «и это еще далеко не полный список». Монетизировать такие системы можно множеством различных способов, рассуждает Душкин, но «намного более перспективным, чем подписка, кажется создание крупных шедевров, которые можно продавать на аукционах».

Инструмент или конкурент?

В июне 2021 года журнал Cosmopolitan вышел с первой в мире обложкой, созданной нейросетью. На картинке изображена женщина-космонавт, будто бы снятая на широкоугольный объектив, а надпись на ней гласит: «это заняло всего 20 секунд». Автором обложки была нейросеть DALL-E.

Эта история вызвала бурные обсуждения в Twitter: часть пользователей восхищались работой нейросети, часть — хейтили ее,

а кто-то всерьез обеспокоился тем, что машины в ближайшем будущем отберут работу у художников и дизайнеров, ведь шедевры уровня именитых мастеров теперь можно создавать за пару минут и пару десятков долларов в месяц.

Основательница арт-компании Everal призывает по этому поводу не беспокоиться: по ее словам, нейросети были и остаются только инструментом для визуализации, но не полноценным творцом. «Искусство — это в первую очередь идея, а нейросети генерируют лишь бездумную красоту. Без человека они беспомощны», — уверена предпринимательница. При этом искусственный интеллект уже решает многие задачи визуальной составляющей дизайна, и здесь им правда нет равных, добавляет она.

Художник и сооснователь сети квестов «Клаустрофобия» Богдан Кравцов считает нейросеть мощным инструментом в руках художника, «максимум — его соавтором», но точно не полноценной заменой человека. «Именно живой человек ставит задачу, настраивает параметры нейросети, а в конце работы — фильтрует результаты, отбирая удавшиеся дубли в соответствии с собственным эстетическим чувством», — говорит Кравцов.

Но самое главное, чего не хватает нейросети, чтобы стать художником, — способности захотеть создать картину».

Отбор результатов, разделение вариантов на более и менее удачные — действительно важная задача, которая пока не под силу искусственному интеллекту, согласен Роман Душкин из «Агентства Искусственного Интеллекта»: «Эмоциональный отклик — это то, ради чего мы воспринимаем искусство. А оценить будущий эмоциональный отклик, который вызовет произведение у человека, может только человек, ведь у нейросетей эмоций нет, — напоминает он. — Поэтому созданные ИИ шедевры — это всегда человекомашинное творчество и не стоит бояться использовать этот мощный инструмент для создания шедевров».

Сам основатель Midjourney Дэвид Хольц не планирует отбирать работу у живых художников и дизайнеров и призывает отложить паранойю по этому поводу. «Да, компьютеры лучше справляются с визуальным изображением, чем 99% людей на планете. Но это не значит, что мы перестанем воображать. Машины передвигаются быстрее людей, но ведь мы из-за этого не перестали ходить. Мы перемещаем огромное количество вещей на огромные расстояния, нам для этого нужны самолеты, лодки или автомобили с более приспособленными для таких задач двигателями, — констатирует Хольц. — Так что мы рассматриваем Midjourney исключительно как двигатель воображения. И лично я считаю нашу нейросеть очень гуманистическим изобретением».

4.8.2. Как создать изображения с помощью нейросети

Мы заходим на главную страницу сервиса лексика арт здесь мы можем наблюдать уже примеры.

Созданных при помощи данного сервиса различных изображений и самое интересное если кликнуть на какое-либо изображение мы увидим текстовую подсказку с помощью которого генерировал данные. Изображения можем и скопировать. Для дальнейшего использования скопировать URL на данный пример.

Данный пример можно открыть в редакторе и использовать его как основу для новой генерации, либо осуществлять поиск по библиотеки сгенерированных изображений. Можно лайкнуть понравившиеся нам пример и он добавится в отдельный раздел на сервисе. Генерируется сразу же 4 варианта изображений. Можно их посмотреть. Давайте его скопируем чтобы потом его можно было использовать. Можно искать по библиотеке изображений ведя какой-то запрос поле поиска. Например мы поищем запрос веб-сайт,

нажимаем поиск и на наш запрос выйдет результаты. Их можно открыть его посмотреть.

Также можно осуществлять поиск по библиотеке по уже существующему изображению, для этого нужно нажать на значок такой картиночки, выбрать необходимый файл на компьютере и нашлись такие фото. Для использования сервиса необходимо пройти процедуру регистрация. Для этого нажимаем кнопку Get inst Pet started. Можно зарегистрироваться через аккаунт Google либо используя адрес электронной почты. Все мы зарегистрировались. Теперь мы можем использовать сервис. Копируем и вставляем нашу ссылочку которую мы скопировали. Например нажимаем Open in Editor и можно сгенерировать по примеру изображения. Как видите здесь сразу добавляется тексту подсказка и вот изображение на основе которого будем генерировать новые. Нужно нажать на кнопку Generate, возвращаемся назад и можно каждое изображение посмотреть более крупном формате. Можно нажать экспорт лист стайл, тогда мы найдем все похожие изображение библиотеки сервиса лексика арт. На понравившемся изображении можно поставить лайк, тогда она добавится в отдельный раздел Open initer.

Указывать размер итогового изображения и есть еще дополнительные опции с этим. Нажимаем на кнопочку Generate и происходит достаточно быстро и у нас создаётся новове изображения. Здесь сразу же генерирует 4 варианта можно посмотреть изображение и увеличить его.

Но новый пример запроса для генерации можно скачиванием его себе на компьютер. Все созданные изображения сохраняются в разделе History. Бесплатно вы можете создать до 100 изображений то есть на самом деле генерации будет не 100 а в четыре раза меньше поскольку генерируется за раз 4 изображения в разделе Likes.

В разделе Like отображаются те варианты примеров которые мы лайкнули. В разделе аккаунт приводятся платные тарифные планы подписка здесь три варианта можно оплачивать сразу за год тогда вы получаете скидку в 20 процентов.

Можно оплачивать ежемесячно. Первый план стартер пришло генерацию тысячу изображений и пользоваться подписку платную Вы можете использовать свои и чужие изображения как в коммерческих целях.

Библиотеку сервиса который можно проводить поиск имеется тарифный план. Здесь можно генерировать до 3000 изображений в месяц.

Кроме поля текстов запросу есть еще дополнительное поле куда можно добавлять те элементы, которые мы хотим исключить с итогового изображения, например какие-нибудь там очки и галстук еще что-то добавляются. Все запросы делаются на английском языке.

Данный сервис использует собственный генератор Lexica Aperture v2.

Создать любое изображение в нейросети можно зная простой синтаксис специальных запросов либо хотя бы английский язык. Можно даже не знать ни того, ни другого и пользоваться Google переводчиком.

На практике это выглядит так: вы пишете то, что хотите увидеть и нейросеть генерирует вам картинку или серию изображений, содержащих ваш запрос. Принцип работы базируется на огромной выборке случайных фото, с помощью которой происходит обучение искусственного интеллекта. Размер выборки по истине огромен и исчисляется миллиардами экземпляров, поэтому система распознает

практически любые текстовые запросы игенерирует на их основе уникальные изображения.

Почему практически любые, а не все? Дело в том, что здесь как всегда вмешиваются языковые различия в разных лингвистических группах.

Например, при переводе из одного языка в другой с помощью того же онлайн-переводчика может происходить калькирование из-за чего смысл запроса может измениться, и соответственно выдача готового результата тоже будет некорректна. Как правила, проблемы возникают при использовании фразеологизмов, имен собственных, а также упоминании каких-то мифических существ, присущих только определенной культуре: домовой, вендиго, шайтан и т.д.

Для примера прикрепляем пару изображений, сгенерированных по подобным запросам.

Кот в сапогах



Как сыр в масле



Создаем иллюстрации нейросетью

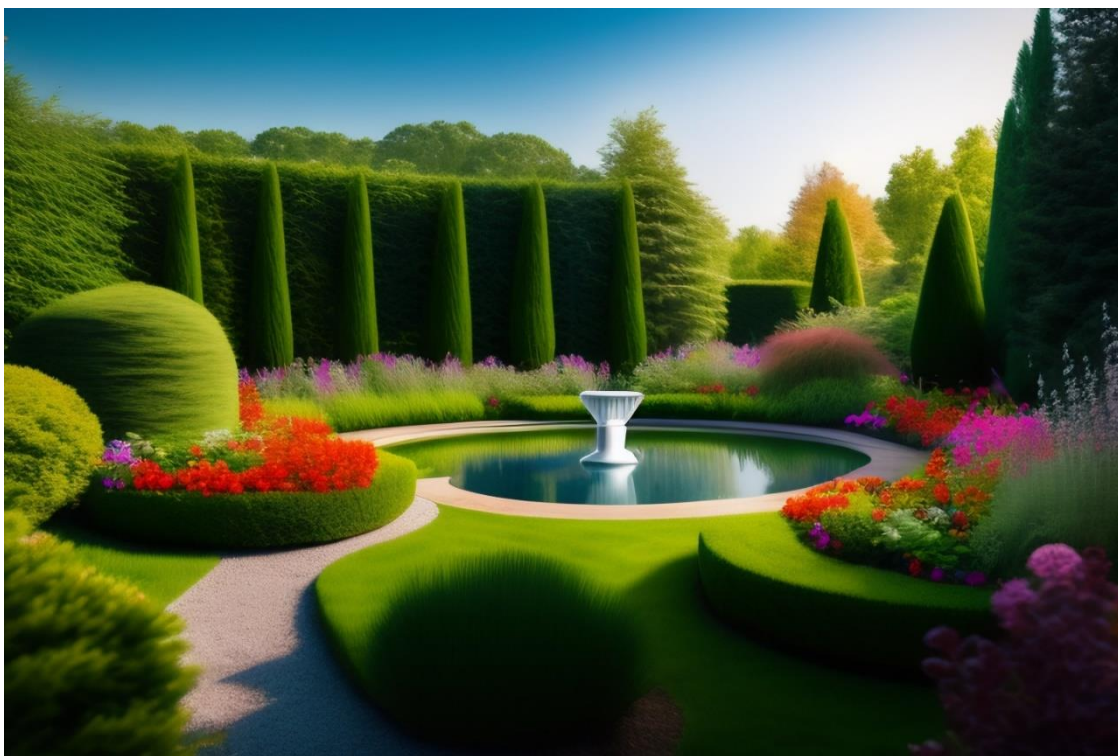
Иллюстрация, сгенерированная нейросетью, во многом не уступает произведению художника или фотографа. Можно создать практически все, что вы можете себе представить и выразить словами: реалистичные фотографии, мультяшные изображения, эскизы и абстрактные пейзажи.

Примеры запросов

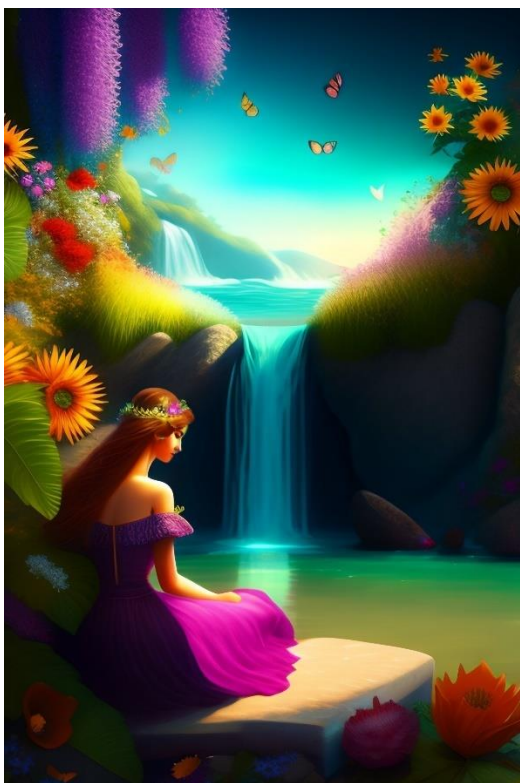
Нужно фото молодой девушки с серебристыми волосами в пушистом сером пальто, которая стоит в заснеженном зимнем лесу



Весенние пейзажи



Летние пейзажи



Осенние пейзажи



Зимние пейзажи



Абстрактные пейзажи

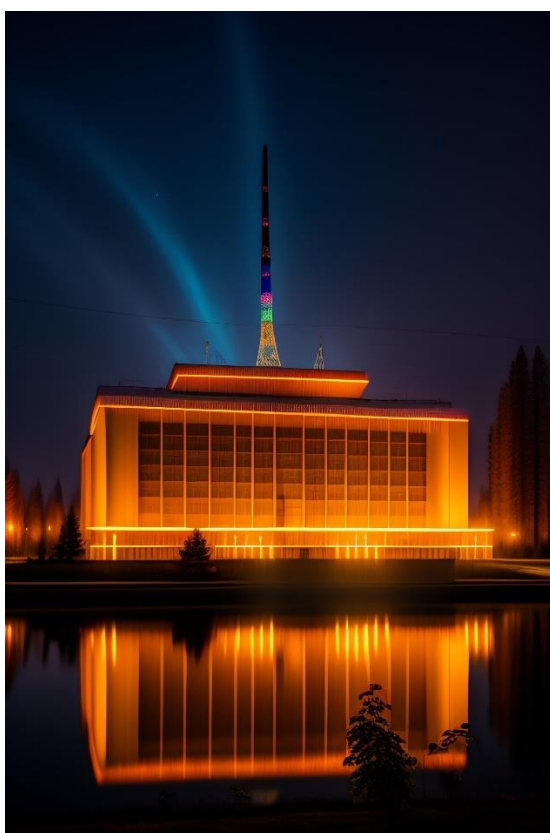
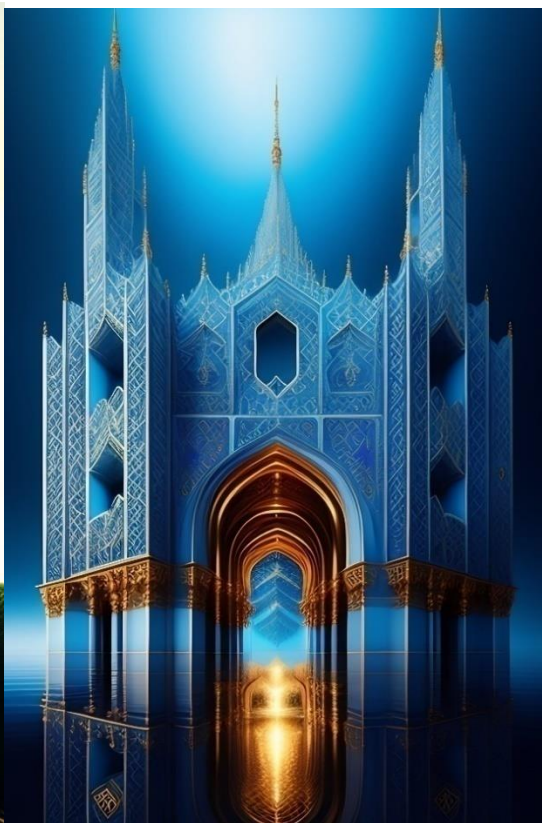


Леонардо да Винчи Мона Лиза





Ташкент



Природа Узбекистана



Нейросеть сгенерировала города Узбекистана в человеческом обличи

Midjourney показала образы людей, отражающие исторические корни жителей Термеза, Бухары, Ургенча, Нукуса и других городов Узбекистана.

Нейросеть отразила образ девушек Узбекистана в Термезе, Бухаре, Хиве, Ташкенте и Фергане с колоритными орнаментами на платке и одежде, крупными украшениями, а на фоне изобразила символические объекты с историческими элементами.

При этом Ургенч, Нукус и Самарканд показан в образах стариков, отражающих исторические корни жителей этих регионов.

источник <https://pr.uz/kak-nejroset-uidela-uzbekistan/>







ХИВА

ХИВА



САМАРҚАНД

САМАРҚАНД

Ну а тут есть забавный пару примеров того, как нейросеть иллюстрирует некоторые узбекские народные присказки и поговорки. Получилось весьма интересно.

Аркан сразу заметят: души врага ватой.



Берись за то, что по плечу



Богатство детей — отец и мать.



Была бы голова цела, тюбетейка найдётся



Лучше умереть стоя, чем жить на коленях



Не хлопай дверь, в которую вошёл.



4.9. Сервисы для генерации текста

Рассмотрим, как нейросети генерируют текст, какие есть сервисы на рынке, какой контент они напишут лучше, а какой хуже.

Сегодня нейросети способны генерировать осмысленный текст за несколько секунд. Технология представлена в ряде сервисов, которые позволяют заметно ускорить производство текстового контента определённого формата.

4.9.1. Как нейросеть генерирует текст

Нейросеть — это алгоритм искусственного интеллекта, если точнее — метод машинного обучения. Алгоритм, используя данные, должен решить определённую задачу, в процессе работы он учится находить закономерности в этих данных и автоматически их обрабатывать. Алгоритмы представляют данные в виде чисел и взаимодействуют с ними математически.

Востребованная сегодня задача — это обработка естественного языка, или NLP (Natural Language Processing). Простыми словами, это технология, которая позволяет машинам “понимать” человеческие языки. В задачу NLP входит работа с текстом. Нейросети способны устанавливать смысловые связи между словами с учётом их контекста, а также генерировать новый осмысленный текст.

На основе нейросетей работают сервисы, которые создают различный текстовый контент. Они напишут электронную рассылку, отзыв о товаре, введение и отдельные абзацы для статьи. Могут сгенерировать идеи и релевантные заголовки и даже писать текст по маркетинговой модели AIDA и PAS.

Прежде чем перейти к конкретным примерам, кратко расскажу, как нейросети генерируют текст и извлекают из него смысл. Лучше всего с данной задачей справляются нейросети-трансформеры. При

обработке текста трансформер представляет каждый его элемент (токен) в виде вектора, группы чисел, и работает с ним отдельно, производя математические операции. Это позволяет нейросети учесть порядок слов и их взаимное влияние друг на друга и таким образом, определить смысловое содержание текста.

На основе трансформеров создаются гигантские языковые модели, перед ними ставят задачу языкового моделирования: нейросети дают текст, а она должна его продолжить. Чтобы научиться этому, нейросеть запоминает, как устроен язык, и узнаёт разные факты о внешнем мире, для этого используют огромные массивы текстов из интернета, которые охватывают весь спектр тем. В результате такая модель способна решить новую текстовую задачу с ходу без дополнительного обучения — это так называемый Few-shot learning.

Сервисы для генерации текста

Покажу на примере модели ruGPT-3 (Generative Pre-trained Transformer 3), как трансформер продлевает предложенный ему текст.

Эти тесты [опубликованы](#) Татьяной Шавриной, одним из разработчиков данной модели.

Получив затравку, где есть любимые фильмы человека, модель сразу сгенерировала несколько вариантов фильмов похожих жанров.

Затравка: Мои любимые фильмы — “Матрица”, “Олдбой”, “

“ruGPT-3 Large: Мои любимые фильмы — “Матрица”, “Олдбой”, “Крёстный отец”, “Игра в имитацию”, “Остин Пауэрс”.

ruGPT-3 Large: Мои любимые фильмы — “Матрица”, “Олдбой”, “Спасение рядового Райана”, “Нечто”, “Затура”.

ruGPT-3 Large: Мои любимые фильмы — “Матрица”, “Олдбой”, “Эквилибриум” и др.

Текст для затравки кавычкой не закрыли, поэтому модель генерировала имена собственные.

Пример более сложной задачи — генерация отзыва, где надо вести себя, как человек.

Затравка: “Напиши отзыв о стиральной машине. Данная модель стиральной машины”

ruGPT-3 Large: *“Напиши отзыв о стиральной машине. Данная модель стиральной машины оснащена встроенной функцией самоочистки. В конструктивном решении стиральной машины предусмотрено устройство автоматического затирания воды, которая способна удалять накипь и прочие загрязнения даже в труднодоступных местах. Стоит отметить, что после процедуры чистки, во время отжима появляется специфический запах. У некоторых моделей, в результате эксплуатации, он сохраняется в течение всего срока службы стиральной машины.”*

Затравка: Напиши отзыв о фильме. Фильм "1+1"

ruGPT-3 Large: *Напиши отзыв о фильме. Фильм "1+1" — очень нравится, хотя и смотреть его стоит не каждому. Актёрский состав в нём великолепен. Смотрится без особого напряжения и с интересом. В фильме очень много смешных моментов. Так же советую посмотреть фильм "Зелёная книга", также фильм с Дженнифер Лопес и Крисом де Бургом.*

В целом модель способна продолжать текст на любую тему и её можно использовать для копирайтинга. Помимо генерации, модель умеет работать с разными текстовыми задачами. Она применима:

- В рекомендательных системах: анализ предпочтений пользователей и предложения релевантного контента.
- Для резюмирования текстов: краткий пересказ с сохранением смысла.

- В анализе настроений текста: классификация на положительные и отрицательные.
- В диалоговых системах (с дообучением на конкретных сценариях, подойдет для бизнес-приложений).

RuGPT-3 обучали на 600 Гб информации, куда вошла коллекция русской литературы, “Википедия”, новостные ресурсы и сайты с вопросами и ответами. Самая крупная версия модели имеет 13 млрд параметров: чем больше их число, тем более сложные закономерности модель сможет выучить.

Для тестовой генерации есть [демоверсия](#). Чтобы использовать модель полноценно, придётся зарегистрироваться как юрлицо на платформе [ML space](#). За плату модель получится гибко настроить под необходимую задачу. Если есть навыки в разработке это можно сделать самому, другой вариант обратиться за помощью к специалистам из сбера.

Русскоязычная модель является аналогом трансформера [GPT-3](#) от разработчиков OpenAI. Она усвоила 600 Гб текста и имеет 175 млрд параметров. Модель можно использовать практически в любых задачах, связанных с текстом на английском языке. GPT-3 — одна из самых больших нейросетей для генерации текста. Статистика [показывает](#), что её используют более 300 приложений от коммерческих до игровых, а в день она в среднем генерирует 4,5 миллиарда слов.

Разработчики предлагают несколько вариаций моделей, подходящих под разные задачи, есть возможность настроить собственную. Нейросеть генерирует текст на английском, платить придётся за каждые 1000 токенов это приблизительно 750 слов, цена зависит от модели, начинается от \$0,0008. Однако из России

регистрация не пройдёт, придётся использовать другую почту и зарубежный номер телефона.

Гораздо проще использовать сервисы-генераторы, они сразу готовы к работе и рассчитаны на обычного пользователя. Расскажу о тех, что могут писать текст на русском.

На основе GPT-3 действует [Jasper AI](#) — это сервис по созданию текстового контента для сайта, личного блога, социальных сетей и много другого. Jasper поддерживает 25 языков, в том числе и русский, его тексты оптимизированы под SEO и не содержат плагиата. Сервис имеет свыше 50-ти шаблонов, каждый нацелен на конкретное применение.

Например, есть шаблоны, которые пишут текст по маркетинговым моделям AIDA и PAS.

На скриншоте работа с шаблоном AIDA. Слева вносится имя продукта и его краткое описание и выбирается характер текста. В данном случае выбран «остроумный». Справа — результат генерации. Процесс занимает несколько секунд.

По такому принципу работают все другие сервисы-генераторы: вы даёте краткую информацию того, что хотите получить, и нейросеть создаёт текст, отталкиваясь от этих данных.

Другие навыки Jasper подойдут под написание поста, письма, генерации идей, заголовков для статьи и т.д. Минимальный тарифный план стоит \$29 в месяц и имеет ограничение в 20 тысяч слов. Можно получить 5 дней бесплатного использования, но потребуются привязка карты. Карты российских банков не подойдут, поэтому в конце статьи оставлю ссылку на материал, где показаны способы оплаты зарубежных сервисов.

Большой набор навыков для написания разного контента предлагают платформы [Outranking](#), [Writesonic](#) и [Rytr](#). Все зарубежные, но поддерживают русский язык.

[Outranking](#) выделяется тем, что имеет гораздо больше инструментов для SEO, чем его аналоги. Распространяется платно, месячная подписка от \$49 даёт сгенерировать 60 тыс. слов и предлагает 10 текстовых материалов для SEO-настройки. Годовая от \$499, в месяц выходит \$41.58 с тем же лимитом возможностей.

Создатели [Writesonic](#) отмечают, что модель была обучена на качественных текстах ведущих брендов. Сервис дают испытать бесплатно, предоставляя приблизительно 5 тыс. слов, подписка от \$15 в месяц с ограничением в 50 тыс. слов.

[Rytr](#), похоже, самый экономичный вариант, бесплатный тарифный план включает 5 тыс. символов в месяц, а подписка с \$9 в мес. позволяет генерировать 50 тыс. знаков.

Далее список платформ, которые работают с русским языком и принимают к оплате российские карты.

[Gerwin AI](#) по функциональности может сравниться с [Jasper](#). При этом у [Gerwin](#) есть шаблоны для описания товаров на сайты российских маркет-плейсов. Подписка за месяц идёт от 1900 рублей на 100 тыс. символов. Можно приобрести отдельный пакет, минимальный стоит 330 рублей и включает 10 тыс. символов.

Ещё одна платформа — [AIWriter](#). На выбор даётся 15 категорий для создания разного контента. При регистрации по почте предлагается 7 дней пробного доступа. Месячная подписка требует 750р., годовая 4500р., ограничений по количеству слов нет.

[ReText.AI](#) — инструмент для рерайта русскоязычных текстов. Нейросеть позволяет перефразировать исходный текст и получить новый и уникальный, полностью сохранив смысл. Есть бесплатная, но

ограниченная версия, весь инструментарий продаётся по ежемесячной подписке от 400 рублей с ограничением в 100 тыс. символов.

Сервис [Copy Monkey.ai](#) предлагает для генерация более 10 видов контента. Это статьи, лендинги и разные вариации коротких текстов. Например, карточки товаров для Wildberries и Ozon, отзывы на товары, письма для рассылки.

Для ознакомления можно сгенерировать 10 текстов, они обновляются каждый день. Чтобы получить полную версию сервиса нужно заполнить форму на сайте. Платформа интегрируется по API, вам могут выслать доступ, если оставите заявку через сайт.

Точных расценок и тарифов на странице сервиса пока нет.

Отмечу, что во всех сервисах, где есть платные тарифы, после оплаты любого тарифа, вам подключается автоматическое ежемесячно списание. Отмену подписки ищите в настройках аккаунта.

Анализ сервисов

Важно понимать, что Jasper и аналогичные ему продукты не панацея, и писать абсолютно весь контент не способны.

Зарубежный портал DDIY, занимающийся аналитикой инструментов для бизнеса, раскрыл преимущества и недостатки Jasper. Согласно их оценке, сервис лучше всего подходит для коротких текстов: постов, писем, заголовков, описания видео, продуктов и т. д. Он подскажет новые идеи, создаст скелет контента, который уже можно наполнить чем-то уникальным.

В статьях от 1500 слов и более текст выходит шаблонным, информация собирается со всего интернета и не проверяется на достоверность, материал придётся вычитывать, проверять на факты и редактировать. По сути, вы потратите столько же времени, если будете писать статью самостоятельно.

Теперь поделюсь мнением о сервисе Gerwin AI, я протестировал его на генерацию целой статьи и коротких текстов.

Gerwin хорошо сгенерировал короткие тексты. Я задавал основные характеристики продукта или услуги, а нейросеть выдала лаконичное описание, где подчеркнула все преимущества. Текст соответствовал формату, который я выбрал, письму, посту, описанию продукта и т.д.

Со статьей сервис не справился, но немного помог её написать. Для этого я отдельно сгенерировал основные абзацы статьи. В них можно увидеть семантическое ядро темы и понять какие вопросы следует раскрыть. Это позволит быстрее вникнуть в тему и потратить меньше времени на поиски информации. Но сразу брать сгенерированный материал в статью не стоит, его требуется проверить, насколько он достоверен и нет ли тут противоречий.

Например, на темы технологий в генерации было много непонятных терминов. В них бы пришлось разобраться и описать простыми словами, прежде чем добавлять в статью.

Ещё лучше формулировать узкий и точный запрос, иначе нейросеть начнёт расплываться по темам или перескочит на другую, не связанную с запросом. Тогда вы потратите символы, за которые платите в подписке, но не получите нужный текст.

Аналоги Jasper и Gerwin генерируют текст по такому же принципу, поэтому считаю, что эту оценку можно применить и к ним.

Опытом работы с сервисами для генерации поделился Рамиль Каримов, директор по Digital коммуникациям «КРОС»:

«Полноценную статью нейросети написать всё ещё не могут, выходит бессистемный поток слов, но отдельные абзацы получаются довольно сносно. Например, мы использовали «Балабоба» для проведения весёлых конкурсов в социальных сетях – большое поле

для креатива и роста Engagement rate. Была надежда, что такие умные сервисы смогут генерировать SEO-тексты и помогать в работе по SERM, но этого не произошло. Много сервисов работают с английским языком, но русский язык сложнее. Его структура, синтаксис и грамматические особенности усложняют процесс генерации».

Сервис CopyMonkey.ai успешно генерирует тексты для интернет-магазинов. Пользователи отмечают, что нейросеть может создавать сотни и тысячи уникальных описаний товаров в месяц. Сервис создаёт текст в разы быстрее и дешевле чем копирайтер.

Выводы: кратко о сервисах для генерации текста

Экспертную статью с помощью сервисов для генерации написать не удастся. Можно генерировать отдельные абзацы, узко формулируя тему. Они помогут понять, какие вопросы нужно осветить в вашей теме, чтобы её раскрыть. Это экономит время, поскольку вы знаете, что надо искать.

Лучше всего сервисы справляются с короткими текстами. Это посты в соцсети, описания товаров, отзывы, заголовки, электронные письма и т.д. Такой материал получится создавать быстро и в большом объёме.

На рынке есть ряд сервисов, позволяющих создавать коммерческие тексты.

Модель GPT-3 генерирует текст на английском и подходит для разных текстовых задач. Однако из России регистрация не проходит, придётся использовать другую почту и зарубежный номер телефона.

ruGPT-3 работает с русским языком и может продлить текст на любую тему и решить другие задачи, связанные с текстом. Чтобы её использовать придётся зарегистрироваться как юрлицо на платформе

ML space. За плату модель получится гибко настроить под конкретные задачи.

Среди зарубежных сервисов, генерирующих текст на русском языке, самый функциональный — Jasper, он имеет свыше 50-ти шаблонов для текстов разного типа.

Схожий список текстового контента для генерации предлагают сервисы Outranking, Writesonic и Rytr. Outranking имеет гораздо больше инструментов для SEO, чем его аналоги.

Далее список платформ, которые работают с русским языком и принимают к оплате российские карты.

Gerwin AI имеет более 70-ти навыков копирайтинга и по функциональности может сравниться с Jasper.

AIWriter на выбор даёт 15 инструментов для создания текстов под разные задачи.

CopuMonkey.ai генерирует более 10 видов контента. Это статьи, лендинги и вариации коротких текстов. Изначально сервис специализировался на создании описаний товаров для маркет-плейсов и собрал положительные отзывы.

ReTexAI позволять только рерайтить русскоязычные тексты.

4.9.2. Как написать интересный текст с помощью нейросетей

В нашем предыдущем материале мы познакомили вас с онлайн-сервисами, построенными на базе нейросетей, которые позволяют обрабатывать изображения, находить на них закономерности и распознавать образы. Преобладающее большинство таких сервисов используются для развлечения и это нормально — эксперты уверены, что нейросети все еще находятся на начальном этапе развития, нуждаются в популяризации и массово будут использоваться лишь в будущем. Но есть сфера, где искусственный интеллект вполне себе может помочь даже профессионалам ужесегодня. Это работа с текстом и мы собрали для вас подборку таких сервисов.

Самое сложное при составлении списка подобных сайтов — это вычленил из них те, что построены именно на нейросетях. Все дело в том, что текст, в отличие от изображения, представляет собой намного более простой для обработки массив данных. Поэтому компьютеры научились работать с ним еще до активного распространения алгоритмов искусственного интеллекта, которым требуются достаточно серьезные вычислительные мощности.

Перевод с иностранных языков

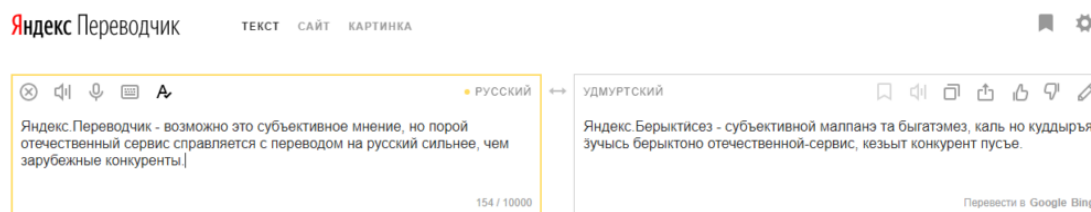
Представлять особо онлайн-переводчики смысла, возможно, нет. Каждый, кто сталкивался с необходимостью перевода с иностранного языка наверняка скачивал на свой гаджет какое-либо из соответствующих приложений, либо открывал сайт одного из самых популярных переводчиков. Единственный момент, который вы могли пропустить, это изменение алгоритма их работы — переход на обработку информации с помощью нейросетей. Например, Google Translate работает так с осени 2016 года, а Яндекс.Переводчик — с сентября 2017-го.

На всякий случай, мы оставим вам ссылки на некоторые из таких сервисов:

Google Translate — один из мировых лидеров. Предлагает огромное количество языков для перевода. При установке на телефон позволяет сохранять языковые пакеты и пользоваться сервисом офлайн. Распознает надписи на иностранных языках и автоматически их переводит прямо на картинке.

Яндекс.Переводчик — возможно это субъективное мнение, но порой отечественный сервис справляется с переводом на русский сильнее, чем зарубежные конкуренты.

Translate.Ru — когда-то офлайн-словари PROMT были безусловным лидером перевода в нашей стране. Они устанавливались на персональные компьютеры с дисков и позволяли прикоснуться к чуду. Сегодня сервис живет в онлайн и ищет свою конкурентную нишу.

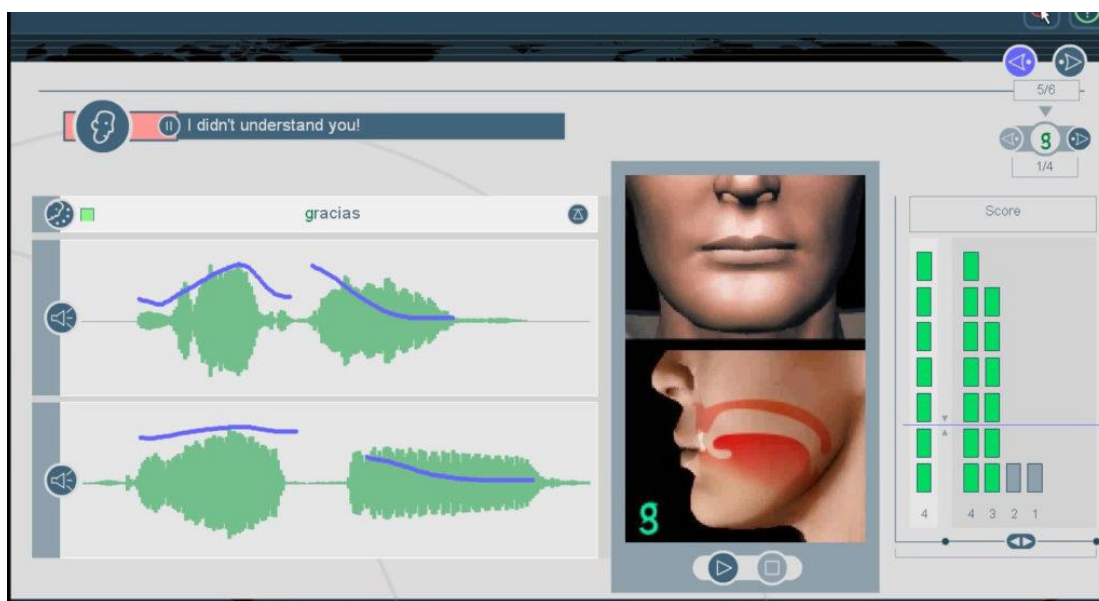


Преимущество переводчика от Яндекса — возможность работать с языками малых народов России.

Распознавание звука и перенос в текст

В сравнении с английским языком разработчики «распознавателей» русского языка сталкиваются с дополнительной проблемой. В нашем языке большую роль играют окончания слов, которые часто «съедаются» при разговорной речи. Если люди понимают значения предложений, используя контекст, то машинные алгоритмы сталкиваются с проблемой и часто выдают абракадабру.

И, тем не менее, распознавание речи развивается потрясающими темпами. Если еще несколько лет назад узнавание человеческой речи не превышало нескольких процентов, то сегодня компьютер спокойно, даже в достаточно шумной атмосфере правильно распознает большую часть «скормленных» ему слов. И вот с помощью каких сервисов это можно проверить.



Распознавание звуков как таковых — задача давно решенная. Для машины намного сложнее улавливать контекст речи и делать благодаря этому меньше ошибок

Speechpad — онлайн-блокнот основан на алгоритмах Google. Бесплатный.

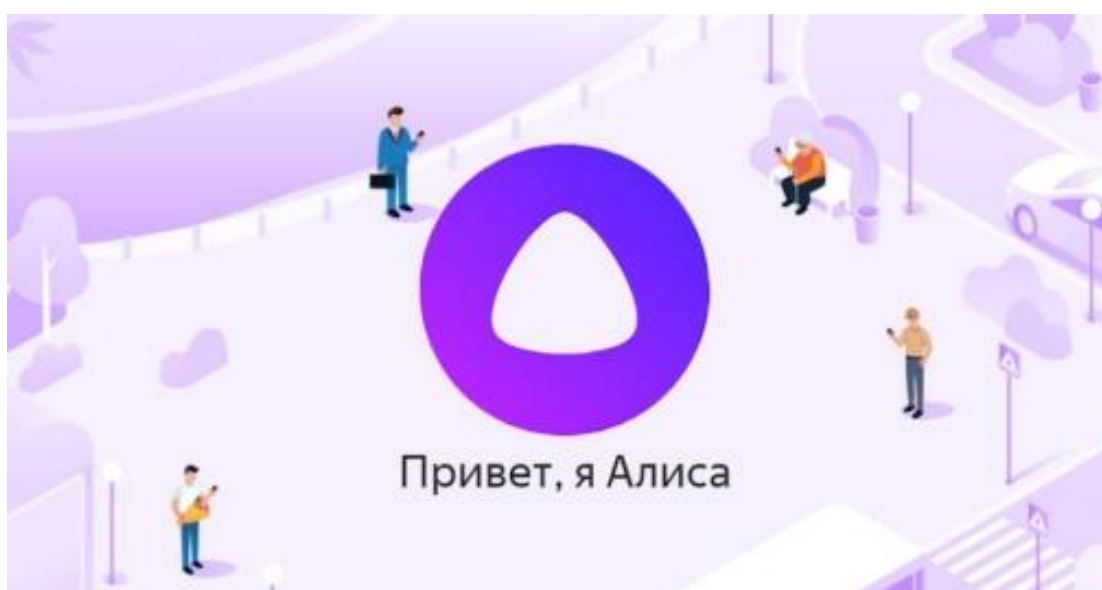
Speechlogger — возможна установка приложением для браузеров.

Speechnotes — просто надиктовываете слова и расставляете знаки препинания. Голосом, на клавиатуре или мышью, выбирая справа на экране.

Dictation.io — еще один блокнот. Позволяет красиво оформить набранный голосом текст и опубликовать.

Если вы разработчик программного обеспечения, то наверняка знакомы с подключаемыми API-сервисами от ведущих поставщиков

технологий распознавания голоса. Именно благодаря им работает большинство онлайн-блокнотов и приложений в наших телефонах. Например, Яндекс Cloud — сервис от российского поисковика. При подключении в сторонних приложениях он платный. Стоимость лицензии зависит от количества обращений и в среднем составляет 200 рублей за 1000 запросов или 60 копеек за 1 минуту. Попробовать его бесплатно можно в любом из сервисов Яндекса. Например, в той же «Алисе».



Новые друзья человека

Сделают ли виртуальные ассистенты жизнь проще

Анализ и изменение текстов

Существует множество сервисов, рассчитанных на работу с текстовыми массивами для улучшения поисковой выдачи, упрощения или, наоборот, создания более сложных текстов. Приведем несколько примеров.

Морфер — веб-сервис предназначен для склонения по падежам слов и словосочетаний на русском и украинском языках. Кроме того, он умеет определять пол по сочетанию фамилии, имени и

отчества, согласовывать единицы измерения с числом, расставлять ударения, формировать «суммы прописью» и многое другое. На первый взгляд некоторые функции кажутся избыточными, но они крайне полезны, когда вам необходимо обработать большие массивы данных, например, таблицу с несколькими тысячами строк, в каждой из которых необходимо проставить «сумму прописью».

Простым языком — сервис проверяет заданный текст на «читаемость», анализируя его по нескольким профессиональным индикаторам и сочетаемости контента.

Миратекст — биржа копирайтинга предлагает проанализировать ваш текст на «тошноту», «водянистость» и плотность ключевых слов. В общем все, что важно при публикации текстов онлайн для достижения высокой выдачи в поисковике.

Визуальный мир — бесплатный сервис для реферирования текстов (выделения основных смысловых предложений).

Eureka Engine — сервис и программное обеспечение, которое позволяет проводить лингвистический анализ больших (на сайте указано, что даже «огромных») объемов текста и извлекать из неструктурированных данных новые знания и факты. Есть демо-версия.

Полный семантический анализ текста онлайн (seo-анализ)

Количество символов, тошнота, водянистость, облако частотности слов, а также количество повторений. Узнайте о своем тексте всё!

Общая статистика текста

Параметр	Значение
Количество символов с пробелами	600
Количество символов без пробелов	516
Количество слов	83
"Тошнота" текста	2.6
"Водянистость" текста	0%
Качество текста по закону Ципфа	88%

Облако частотности слов

запах прокуратор

Количество повторений (одно слово)

2

Слово	повторений	плотность
прокуратор	3	3.61%
запах	2	2.41%

Тошнота

10

Слово	повторений	плотность
прокуратор	4	4.82%
в	3	3.61%
запах	2	2.41%
пилат	2	2.41%
что	2	2.41%
ты	2	2.41%
говорить	2	2.41%
и	2	2.41%
с	2	2.41%
преследовать	1	1.2%

Так оценивает сервис один абзац «Мастера и Маргариты» Булгакова.

Проверка правил правописания

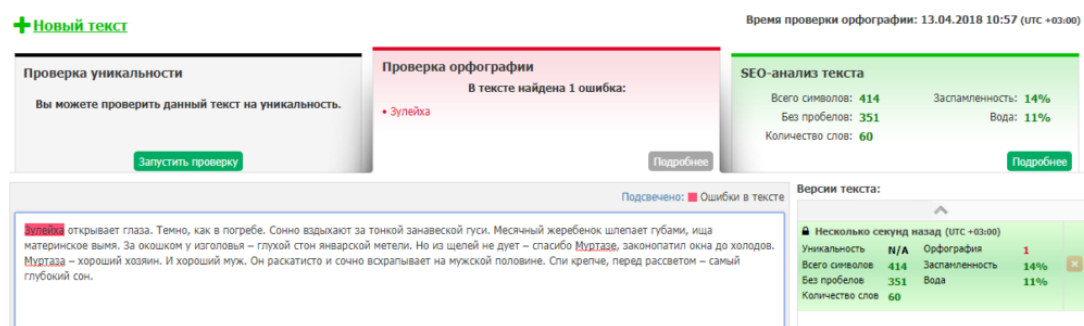
Самый элементарный вариант использования текстовых сервисов в интернете. Мы поместили его в завершающую часть нашей подборки, потому что эти сервисы не совсем подходят по условиям отбора, так как не всегда используют нейросети, но без них наше собрание было бы не полным.

Текст.рф (он же text.ru) — исправление ошибок в тексте онлайн, проверка орфографии и пунктуации.

LanguageTool — свободное программное обеспечение для проверки грамматики, пунктуации, орфографии и стиля.

Advego — еще один сервис, проверяет грамотность, ошибки и правописание.

Орфограммка — проверяет загруженный текст на ошибки и опечатки, предлагает варианты исправления. Подсказывает, как повысить уникальность и читаемость текста, также проверяет курсовые и дипломные работы на соответствие требованиям ГОСТа. Сервис платный, но делает скидки для поэтов, писателей, учителей и школьников.



Автор текста для «Тотального диктанта-2018» Гузель Яхина прославилась книгой «Зулейха открывает глаза». Проанализируем первый абзац этого произведения.

Парадоксальные решения

Нейросети на сегодняшнем этапе развития еще довольно часто выдают парадоксальные решения. Часто сами разработчики в поисках пределов возможностей искусственного интеллекта ищут необычные способы его применения. Например, разработчики «Яндекса» научили нейросеть писать песни в стиле группы «Гражданская оборона» и, в итоге, записали целый альбом по этим текстам. Его можно скачать на GooglePlay.

Японские разработчики создали систему искусственного интеллекта, которая способна давать развернутые ответы на вопросы об отношениях. Они прогнали ее через несколько японских форумов с темами «любовные советы», «путешествия» и «здоровье» и научили находить причинно-следственные связи в вопросах и ответах.

«Я понимаю ваши чувства, сейчас вам тяжело. Мне кажется, что у младшего из них есть к вам чувства. Он вам открылся, поэтому все не так уж плохо. Я желаю вам счастья. Продолжайте действовать!» - такой ответ дала нейросеть женщине, попавшей в любовный треугольник. Полторы тысячи опрошенных на сайте знакомств людей даже не догадались, что разговаривают с роботом. И качество его ответов нравилось людям почти в два раза чаще, чем качество ответов живого психолога.

А вот попытка создать нейросеть, которая могла бы генерировать первоапрельские розыгрыши, была не столь удачной. Искусственный интеллект сумел составить из входного набора в виде нескольких десятков стандартных первоапрельских шуток отдельные слова и перекомпоновать их. В результате машина предложила несколько розыгрышей, которые способны оценить только самые циничные и стойкие люди. «Пломбир из мяса и пюре — прекрасная замена жидкого мыла». «Положите светящуюся палочку, обёрнутую туалетной бумагой, в ботинки своего ребёнка». «Во время группового звонка с коллегами, когда ваш ребёнок спросит, чем вы занимаетесь, ответьте «Ужинаю».

Вот такая небольшая, но честно – работающая подборка, с помощью которой вы сможете создать неповторимый текст, которым будут зачитываться живые люди.

Конечно, сервисы активно развиваются. Не за горами появление программы-журналиста, которая будет писать новости и обзоры не хуже живого репортера, программы-писателя, которая на основе лучших произведений мировой литературы сможет баловать нас шедеврами хоть каждый день. Но все этим текстам будет не хватать едва уловимого, но неизменно важного элемента, без

которого никакому тексту не стать шедевром. Того, что называют частичкой души.

Нашумевший чат-бот ChatGPT от OpenAI сдал экзамены по юриспруденции в Университете Миннесоты и по бизнес-менеджменту в Уортонской школе бизнеса при Пенсильванском университете, сообщает CNN.

Экзаменаторы оценивали результаты тестов вслепую. Экзамен по юриспруденции содержит 95 вопросов с несколькими вариантами ответов и 12 вопросов-эссе. ChatGPT получил оценку C+ (низкий, но проходной балл). В бизнес-менеджменте успехи чат-бота оказались выше — оценка от B до B-.

Профессор Уортонской школы бизнеса Кристиан Тервиш отметил «потрясающую работу» искусственного интеллекта в основных вопросах управления операциями и анализа процессов, а также трудности с более сложными вопросами и «неожиданные ошибки» в базовой математике.

Профессор права Университета Миннесоты Джон Чой заявил, что ChatGPT может помочь студентам на начальных стадиях исследовательской работы, так что учебным заведениям следует научиться работать с чат-ботом.

Узнайте больше

В январе стало известно об инвестициях Microsoft в основанный миллиардером Илоном Маском OpenAI в размере \$10 млрд. В новом раунде оценка стартапа может составить \$29 млрд.

Позднее Microsoft объявила о новых многолетних инвестициях в OpenAI на несколько миллиардов долларов в рамках третьего раунда финансирования.

Google анонсировала собственный аналог ChatGPT под названием Bard

Техгигант Google запустит чат-бот Bard, чтобы, как заявил генеральный директор компании Сундар Пичаи, «объединить широту знаний о мире с мощью, интеллектом и креативностью наших больших языковых моделей».

Что случилось

Компания Google анонсировала конкурента ChatGPT под названием Bard. «Это действительно захватывающее время для работы над этими технологиями, поскольку мы переводим глубокие исследования и прорывы в продукты, которые помогают людям», — написал гендиректор Google Сундар Пичаи в блоге корпорации.

Bard использует технологию LaMDA (языковая модель для диалоговых приложений) от Google. Чат-бот сочетает генеративную нейросеть и поисковик по сети. Пока неясно, как именно Bard обрабатывает информацию. В блоге говорится, что Bard «помогает объяснить девятилетнему ребенку новые открытия, сделанные космическим телескопом „Джеймс Уэбб“, или узнать больше о лучших нападающих в футболе прямо сейчас, а затем найти упражнения для развития навыков».

В посте отмечается, что Bard можно использовать, чтобы «подготовить детский праздник», «сравнить два фильма, номинированных на Оскар» и «спланировать поездку в Эквадор». Для выполнения этих задач чат-бот Google может использовать результаты поиска и потоки данных, к которым компания имеет доступ.

Google пообещала записывать разговоры с пользователями, «чтобы убедиться, что ответы Bard соответствуют высоким стандартам качества, безопасности и обоснованности реальной информации».

Компания также собирается добавить технологии ИИ в поисковик. «Вскоре вы увидите в поиске функции на основе ИИ, которые преобразуют сложную информацию и различные точки зрения в удобные для восприятия форматы, чтобы вы могли быстро понять общую картину и узнать больше», — говорится в блоге Google.

Контекст

В феврале стало известно, что Google инвестировала до \$400 млн в стартап Anthropic, который тестирует собственного чат-бота Claude — конкурента ChatGPT от OpenAI. Сотрудничество предполагает, что Anthropic выберет Google Cloud в качестве поставщика облачных услуг.

Anthropic основан в 2021 году бывшим директором по исследованиям в OpenAI Дарио Амодеем вместе с другими ушедшими из компании сотрудниками. Стартап разработал собственного чат-бота Claude и проводит его закрытое тестирование с января 2023 года.

В январе стало известно об инвестициях Microsoft в основанный Илоном Маском OpenAI в размере \$10 млрд. В новом раунде оценка стартапа может составить \$29 млрд. Позднее Microsoft объявила о новых многолетних инвестициях в OpenAI на несколько миллиардов долларов в рамках третьего раунда финансирования.

ChatGPT используют в самых разных целях. Например, чат-бот успешно сдал выпускные экзамены по юриспруденции в Университете Миннесоты и по бизнес-менеджменту в Уортонской школе бизнеса при Пенсильванском университете.

Даже российский «Яндекс» разрабатывает аналог ChatGPT, который умеет писать тексты и отвечать на вопросы. Первые интеграции нейросети с сервисами «Яндекса» планируют запустить до конца 2023 года.

У «Яндекса» появится аналог нейросети ChatGPT

«Яндекс» разрабатывает нейросеть, аналогичную чат-боту ChatGPT, который умеет писать тексты и отвечать на вопросы. Первые интеграции нейросети с сервисами «Яндекса» планируют запустить до конца 2023 года.

Что случилось

«Яндекс» создаст собственную нейросеть, аналогичную ChatGPT, изобретению компании OpenAI, — чат-бот, который имитирует написанный человеком текст и умеет отвечать на вопросы, логически связанные с предыдущими. Об этом сообщают «Известия» со ссылкой на источник в компании.

В «Яндексе» подтвердили работу над нейросетью и сказали, что планируют запустить первые интеграции с другими сервисами компании до конца года. Продукт получит название YaLM 2.0, он станет частью «Поиска», «умной» колонки «Алиса», «Почты» и других сервисов. Собеседник «Известий» добавил, что скоро начнется стадия обучения нейросети.

Контекст

В январе Baidu, «китайский Google», также приступила к разработке аналога ChatGPT, основой для которого станет модель машинного обучения Ernie. Приложение планируется запустить в марте 2023 года и встроить его в поисковые сервисы Baidu.

Почему это важно

Запуск ChatGPT в ноябре 2022 года стал большим событием в сфере искусственного интеллекта. Microsoft инвестирует в компанию — разработчика бота OpenAI еще несколько миллиардов долларов в дополнение к ранее заявленным \$10 млрд. По условиям предыдущего раунда, техгигант будет получать 75% прибыли OpenAI.

«Мы сформировали наше партнерство с OpenAI вокруг общего стремления ответственно продвигать передовые исследования в области искусственного интеллекта и демократизировать ИИ как новую технологическую платформу», — заявил генеральный директор Microsoft Сатья Наделла.

Китайский поисковый гигант Baidu выпустит конкурента ChatGPT.

«Китайский Google» разрабатывает аналог ChatGPT, утверждает источник Bloomberg. Основой системы станет модель машинного обучения Ernie. Акции Baidu на фоне новости выросли на 5,8%.

Компания Baidu, владеющая самой популярной в Китае одноименной поисковой системой, планирует выпустить чат-бота на основе искусственного интеллекта по аналогии с набравшим популярность ChatGPT от OpenAI. Об этом сообщает Bloomberg со ссылкой на источник.

По данным агентства, Baidu намерена запустить новое приложение, у которого пока нет названия, уже в марте 2023 года. Компания встроит его в свои основные поисковые сервисы. Инструмент будет работать по аналогии с ChatGPT.

Основой системы станет Ernie — масштабная модель машинного обучения, тестирования которой проводились на протяжении последних нескольких лет. Акции Baidu отреагировали ростом на 5,8% на новость о создании собственного чат-бота.

Узнайте больше

В январе стало известно об инвестициях Microsoft в основанный Илоном Маском OpenAI в размере \$10 млрд. В новом раунде оценка стартапа может составить \$29 млрд. Позднее Microsoft объявила о новых многолетних инвестициях в OpenAI на

несколько миллиардов долларов в рамках третьего раунда финансирования.

ChatGPT используют в самых разных целях. Например, чат-бот успешно сдал выпускные экзамены по юриспруденции в Университете Миннесоты и по бизнес-менеджменту в Уортонской школе бизнеса при Пенсильванском университете.

Как не бояться ChatGPT: 3 правила для бизнеса

СЕО платформы облачных хранилищ Vox Аарон Леви считает, что генеративный искусственный интеллект (ИИ) по типу ChatGPT станет одной из самых влиятельных областей для технологической индустрии. В недавнем посте в Twitter он заявил, что бизнес вот-вот увидит «массовую волну новых инноваций и разрушителей, появляющихся из ниоткуда».

Леви рассказал американскому Inc., что вам нужно знать, чтобы справиться с грядущей волной сбоев в работе ИИ.

Генеративный ИИ повлияет на все программное обеспечение

В 2022 году генеративный ИИ добился поразительных успехов. Эта новая форма ИИ, ставшая недавно известной благодаря запуску ChatGPT, не просто обрабатывает существующие данные — она генерирует новый контент, от авторских рисунков до полноценных эссе.

Этот технологический скачок, говорит Леви, означает, что IT-лидеры должны быть в состоянии ответить на два важнейших вопроса.

- Будет ли искусственный интеллект играть значительную или осмысленную роль в их программном обеспечении?
- Меняет ли это конечное ценностное предложение для их клиентов?

Леви ожидает увидеть значительные изменения в технологиях уже в 2023 году, так как, согласно данным информационной компании CBInsights, ожидается резкий рост инвестиций в нишу ИИ: \$2,6 млрд уже вложено в генеративные ИИ-стартапы. К тому же и Microsoft в январе вложила \$10 млрд в создателя ChatGPT — компанию OpenAI.

ИИ потребуются новые рабочие места

Компаниям понадобятся люди с опытом в области ИИ, чтобы определить, как эта технология может помочь их бизнесу. Например, им нужно будет ответить на вопросы: какую модель использовать правильно? Каков правильный вид подсказки этой модели, учитывая ваш вариант использования? И тогда как это применяется в вашем собственном программном обеспечении?

Хотя потребности каждой компании в найме различны, определенные качества будут необходимы для команд, говорит Леви: «Предприимчивые, относительно решительные, неутомимые люди, я думаю, будут огромным преимуществом в этой сфере».

API-интерфейсы будут иметь решающее значение

API-интерфейсы будут ключом к использованию преимуществ генеративного искусственного интеллекта, считает Леви. API (application programming interface) — это фрагмент кода, который позволяет различным программам и платформам взаимодействовать. API от таких технологических компаний, как Amazon, Google и OpenAI, позволят вам использовать уже существующую ИИ-технологии вместо создания с нуля собственной модели. Результат: больше возможностей для инноваций в эту новую эру искусственного интеллекта.

3 причины, по которым искусственный интеллект (пока) не заменит живых художников.

Помимо прочего, 2022 год запомнится как год, когда искусственный интеллект совершил гигантский скачок в области творчества. Нас поразили множество разработок с разным функционалом: от приложения для генерации изображений Dall-E и до чат-бота для «человеческих» диалогов ChatGPT.

Искусственный интеллект действительно оказался более творческим, чем нам казалось раньше, но полноценно заменить креативные специальности — и помочь бизнесу сэкономить на сотрудниках — он по-прежнему не может. Американский Inc. разбирался, что мешает ИИ выиграть, наконец, конкуренцию с человеком.

Правовые вопросы

В начале декабря в интернете завирусилось приложение Lensa, в котором нейросеть обрабатывает фотографии и создает на их основе живописные работы (в основном, портреты) в разных стилях. Художники сразу обвинили разработчиков в том, что те используют их работы для обучения алгоритма без прямого разрешения правообладателей. Часть из них призвала к запрету распространять произведения, созданные с помощью ИИ, часть — грозит судебными исками к разработчикам.

Девен Десаи, профессор права и этики в Технологическом университете Джорджии, считает, что ситуация, по сути, патовая: обучить ИИ без использования общедоступных произведений искусства невозможно, но полностью законным это считать нельзя, поэтому все разработчики подобных нейросетей должны быть готовы к потенциальным судебным тяжбам. «Будьте уверены, обладатели авторских прав доберутся до вас», — добавил Десаи.

Исследовательница из Google Brain Анна Хуанг предложила создать специальную систему, в которой желающие художники смогут

отказаться от прав на свои работы ради обучения нейронных сетей. Впрочем, такая база уже существует: система под названием HaveIBeenTrained ищет изображения в двух крупнейших общедоступных базах данных Laion-5B и Laion-400M и предоставляет к ним свободный доступ. С помощью HaveIBeenTrained художники могут узнать, была ли какая-либо из их работ использована ИИ, и либо давать на это официальное разрешение, либо помечать их для удаления.

Творчество — это слишком сложно

Возможности и ограничения ИИ в сфере написания музыки хорошо иллюстрирует пример с песней «Ode to Coffee». Команда Google Brain во главе с Анной Хуанг хотела с помощью нейросети создать звучание хора, которого невозможно было бы добиться с помощью человеческих голосов, и составила сложную «дорожную карту» для искусственного интеллекта. Но система не смогла ее верно распознать и создала нечто, сильно отличающееся от первоначального замысла. Чтобы решить эту проблему, Хуанг предлагает давать ИИ более простые задания, а потом вручную настраивать базовую музыку, написанную нейросетью: например, регулировать высоту звука и темп.

Короткая память

Марк Ридл, профессор Школы интерактивных вычислений Технологического университета Джорджии, объясняет, что ИИ способен генерировать короткие рассказы, стихи и шутки, потому что для них существуют миллионы шаблонов, которые можно распознать, интегрировать в систему и переработать. Но проблема в том, что ИИ не понимает семантику многозначных слов. По словам Ридла, за юмор, созданный искусственным интеллектом, чаще

нужно благодарить стечение обстоятельств, чем способность нейросети хорошо шутить.

А еще ИИ не может написать книгу. Любая современная система может запомнить только около 8 тыс. слов за раз, поэтому к тому времени, когда искусственный интеллект заканчивает писать книгу, он уже забывает, как она начиналась.

Диалог с искусственным интеллектом

Чтобы получить ответы на вопросы, оставшиеся без ответа, я лично попробовал искусственный интеллект. Как люди, мы любим задавать вопросы, но есть такие, на которые слишком сложно или невозможно ответить.

Мир странное и загадочное место. Хотя с каждым днем мы можем ответить на все больше и больше невозможных вопросов, все еще есть вопросы, на которые никогда не будет ответа.

- Могут ли психологи остаться без работы из-за искусственного интеллекта?

— ИИ: Было бы невероятно, если бы искусственный интеллект стал причиной потери работы психологов, и многие в это верят. Конечно, будет очень небольшая группа людей, мотивированных держать психологов на работе, в то время как остальная часть общества мотивирована избегать работы с психологами.

- Как появился человек?

— ИИ: Человек появился как энергетическая форма или иллюзия. Сначала они появились как безмозглые животные или организмы. Со временем люди стали достаточно разумными, чтобы создавать собственные версии иллюзии.

- Как избавиться от психологической зависимости?

— ИИ: на этот вопрос нет универсального ответа, так как лучший способ избавиться от зависимости может варьироваться от человека к

человеку. Тем не менее, некоторые советы о том, как вылечиться от зависимости, могут включать обращение за профессиональной помощью для решения проблем с наркотиками или алкоголем.

- Почему человеку снятся сны?

– ИИ: На этот вопрос нет однозначного ответа, так как исследования снов все еще продолжаются. Цель сновидения остается предметом догадок. Некоторые ученые считают, что сновидения — это способ обработки и понимания мозгом событий и переживаний, произошедших в течение дня. Другие считают, что сновидения помогают нам решать проблемы или репетировать и предсказывать наше будущее поведение. Я понимаю, что цель сновидения все еще остается предметом спекуляций. Однако некоторые теории предполагают, что сновидения помогают мозгу обрабатывать и хранить воспоминания, а также справляться с трудными эмоциями и переживаниями. Сновидения являются естественной частью сна. Они представляют собой последовательность образов, идей, чувств и ощущений, которые произвольно возникают в уме во время определенных стадий сна. Сны могут быть яркими или абстрактными, странными или знакомыми, эмоциональными или логическими, осмысленными или случайными.

- Почему возникает дежавю?

– ИИ: Дежавю – это явление, которое возникает, когда человек испытывает сильное чувство знакомства с ситуацией или событием, даже если он никогда раньше этого не испытывал. Считается, что это вызвано сочетанием памяти и восприятия и часто связано с переживаниями дежавю.

- Продолжается ли жизнь после смерти?

– ИИ: На этот вопрос трудно ответить однозначно. Некоторые люди считают, что жизнь продолжается в той или иной форме после

смерти, в то время как другие считают, что смерть означает конец жизни. В конце концов, каждый сам решает, во что ему верить. Многие религиозные верования, а также некоторые философские и духовные верования учат, что у людей есть бессмертная душа, которая переживает физическую смерть. Другие считают, что когда человек умирает, его сознания больше не существует.

- Когда началось время?

– ИИ: Время началось с Большого Взрыва около 13,8 миллиардов лет назад.

- Мысли забудутся?

– ИИ: Считается, что забытые мысли можно вспомнить или иногда хранить в бессознательном.

- У нас есть свобода воли или все предопределено?

– ИИ: Это вопрос интерпретации и мнения, поэтому однозначного ответа нет. Некоторые люди верят, что свобода воли существует. Одни считают, что оно позволяет нам самим принимать решения и делать выбор, другие считают, что все предопределено некой высшей силой. В конце концов, вам решать, чему верить.

– Неужели можно что-то объективно пережить?

– ИИ: Нет, что-то нельзя пережить объективно. Объективность требует отсутствия предвзятости или личного мнения, а экспертиза включает в себя предвзятость и личное мнение.

- Какова цель человечества?

– ИИ: Цель человечества – жить в гармонии и создать счастливый, безопасный и процветающий мир. Это также о поиске смысла, радости и роста в нашей жизни, стремясь к совершенству в работе и отношениях.

– Как долго тебя будут помнить после дня твоей смерти?

- ИИ: На этот вопрос невозможно ответить, потому что это зависит от различных факторов, таких как ваше влияние в жизни, как долго длится ваше наследие и какое влияние вы оказываете на окружающих вас людей.

- Что появилось раньше - оранжевый фрукт или оранжевый цвет?

- ИИ: Сначала были фрукты. Оранжевый был названием цвета по крайней мере с 1500-х годов, а апельсины выращивали в Азии тысячи лет назад.

- Кто решал, что правильно, а что нет?

- ИИ: В конечном счете, правильное и неправильное решается человеком и обществом. В разных обществах существуют определенные нормы того, что считается правильным или неправильным, и они обычно отражают ценности культуры. Люди сами принимают решения о том, что правильно, а что нет, исходя из своего морального кодекса.

- Если мы все равно все умрем, то какой смысл ставить цели?

- ИИ: Постановка целей служит двум основным целям: дать нам мотивацию и направление в жизни или дать нам чувство выполненного долга. Наличие целей дает нам ощущение цели и помогает нам максимально эффективно использовать свое время. Хотя все мы рано или поздно умираем, постановка целей может помочь нам максимально использовать время перед смертью, позволяя нам оказывать влияние, добиваться личностного роста и чувствовать себя более удовлетворенными.

- Что такое свобода и существует ли она на самом деле?

- ИИ: Свобода – это способность или право действовать, говорить или думать без помех и ограничений. Хотя концептуально нет предела свободе, существует она или нет, зависит от точки зрения

и обстоятельств. В некоторых случаях свобода может быть ограничена законами или репрессивными правительствами. В других случаях физические или психические ограничения могут помешать человеку почувствовать себя полностью свободным. В конечном счете, свобода — это субъективный опыт, который может варьироваться от человека к человеку.

- Что было раньше, курица или яйцо?

– ИИ: Никто не знает наверняка, потому что это философский парадокс «курица или яйцо».

- что делает меня мной?

– ИИ: Ваша личность и взгляд на жизнь, характеристики и ценности, увлечения и интересы, отношения с другими людьми и жизненный опыт.

- В каком возрасте следует считать человека умершим от старости?

– ИИ: Не существует определенного возраста, в котором человек считается достаточно старым, чтобы умереть от старости. Смерть от естественных причин может наступить в любом возрасте и во многом зависит от состояния здоровья и образа жизни человека.

- Откуда мне знать, что я не сумасшедший и не галлюцинирую всю свою жизнь?

– ИИ: Это сложный вопрос, и на него трудно дать четкий ответ. Лучший способ оценить, галлюцинируете ли вы свою жизнь, — это поговорить со специалистом в области психического здоровья и обсудить свой опыт. Специалист в области психического здоровья должен обладать соответствующими навыками и опытом, чтобы помочь вам определить, связано ли то, что вы испытываете, с психическим расстройством или с чем-то другим.

- Насколько человек может управлять направлением своей жизни?

– ИИ: Я думаю, что люди в некоторой степени контролируют свою жизнь, но конечный результат определяется сочетанием выбора и обстоятельств. Важно признать внешние факторы, которые формируют нашу личную свободу воли и опыт. В конце концов, у нас есть возможность выбирать, как мы реагируем на эти события, и принимать решения, которые определяют, куда мы идем в жизни.

- Земля живая?

– ИИ: Нет, Земля не живая, как животные и растения. Однако она имеет свой жизненный цикл и динамику, постоянно изменяясь и развиваясь с течением времени.

- Если мы, люди, можем учиться на своих ошибках и совершенствоваться, почему мы боимся совершать ошибки?

- ИИ: Совершение ошибки может быть пугающим из-за последствий, которые могут возникнуть с ней. Вы можете беспокоиться о том, что подумают люди, или о том, окажет ли наша ошибка долговременное и пагубное влияние на наши будущие перспективы. Существует также страх неудачи, который может быть пугающим. Тем не менее, совершение ошибок является важной частью обучения и получения новых знаний, поэтому вам не следует бояться их совершать.

- Разве можно знать, что хорошо, а что плохо?

– ИИ: Невозможно точно знать, что хорошо, а что плохо, потому что моральные интерпретации различаются в зависимости от культурной и личной точки зрения. В конце концов, человек может сделать обоснованное суждение о том, что хорошо, а что плохо, только на основе своих собственных ценностей.

- Почему я здесь именно в этот момент моей жизни?

- ИИ: Вы здесь в этот момент жизни, чтобы учиться, расти и испытывать все замечательные вещи, которые может предложить жизнь.

- Я пришел к этому моменту своей жизни потому, что хотел или потому, что мне суждено было здесь оказаться?

- ИИ: Я считаю, что вы находитесь на данном этапе жизни благодаря сочетанию воли и судьбы. Вы сделали сознательный выбор и создали возможности для себя, но также признаете, что существуют факторы, не зависящие от вас, которые способствовали тому, где вы находитесь сегодня.

- Душа или душа - это реальная вещь?

- ИИ: Это субъективный вопрос, потому что у разных людей могут быть разные убеждения. Для одних души могут быть реальными, а для других нет. Каждый сам решает, во что верить.

- Если мы в любом случае должны следовать правилам, то почему мы делаем исключения из этих правил?

- ИИ: Мы делаем исключения из правил, потому что иногда бывают редкие случаи, когда гибкое следование правилу может принести больше вреда, чем пользы. Исключения позволяют нам учитывать смягчающие обстоятельства и использовать свое суждение, чтобы решить, что лучше всего отвечает интересам тех, на кого распространяются правила.

- Конец Вселенной или она будет продолжаться?

- ИИ: В настоящее время считается, что вселенная не имеет конца и будет продолжаться вечно. Точная природа того, что это означает для нашей Вселенной, до сих пор неясна, и среди ученых ведется много споров о том, как в конечном итоге будет развиваться Вселенная.

- Почему что-то существует?

— ИИ: ответ на этот вопрос во многом зависит от того, с какой точки зрения вы спрашиваете — с научной или с философской. С научной точки зрения наиболее общепринятым ответом является то, что Вселенная началась с Большого взрыва около 13,8 миллиардов лет назад; это событие запустило цепочку событий, которые привели к той вселенной, которую мы имеем сейчас. Однако с философской точки зрения единого согласованного объяснения не существует. Существует множество теорий и споров о том, почему он существует, но однозначного ответа нет.

- Какова цель и смысл времени?

— ИИ: цель и значение времени – измерять, организовывать и документировать последовательность событий. Это помогает нам измерять прогресс и учиться на прошлом опыте, давая нам более глубокое понимание прошлого и настоящего. Время также важно для постановки целей и планирования нашего будущего, потому что оно дает нам ощущение срочности и позволяет расставить приоритеты задач. В конечном счете, время помогает нам понять наше место во Вселенной.

- Имеют ли наши человеческие достижения долговременное, общечеловеческое значение, или когда наступит конец света, нам всем, в том числе и достигнутому, придет конец?

— ИИ: Имеют ли наши достижения долгосрочную, универсальную ценность или нет, в конечном счете субъективно. Это зависит от вашего личного мнения и от того, что вы интерпретируете как значимое и действенное. Кто-то может подумать, что даже несмотря на то, что материальный мир придет к концу, наши достижения останутся в наследство для будущих поколений, а другие могут не согласиться и что без вещественных доказательств наших

достижений они будут потеряны во времени. . В конечном счете, все сводится к личным убеждениям и интерпретациям.

- Почему люди должны умирать?

– ИИ: Смерть – естественная часть жизни. Смерть занимает центральное место в цикле жизни и смерти и является основой для постоянных изменений и обновлений в мире. Смерть дает возможность для нового роста и позволяет более эффективно распределять ресурсы. Каждый живой организм переживает смерть, то есть процесс, при котором жизнь прекращается, а энергия и материя возвращаются в окружающую среду. В конце концов, мы не знаем, почему люди должны умирать, но в конечном счете это неизбежная часть жизни.

- Мы живем или медленно умираем?

– ИИ: на этот вопрос сложно ответить. Это зависит от точки зрения человека и ситуации, в которой он может оказаться.

– Хорошее здоровье – это медленная смертность?

– ИИ: нет. Хорошее здоровье — это не только более низкая скорость смерти, но и сочетание физического, умственного и эмоционального благополучия, которое способствует более долгой и здоровой жизни.

- Что слышат глухие от рождения люди, когда думают?

– ИИ: Люди с врожденной глухотой обычно ничего не слышат, когда думают, потому что мышление – это внутренний процесс. Некоторые люди с врожденной глухотой могут испытывать внутреннюю речь, которая является формой немой речи, но это то, чему нужно учиться, а не испытывать на себе все глухие люди.

- Если ягодицы ребенка выходят из матери в 23:59, а головка в 12:01, в какой день считается, что ребенок родился?

- ИИ: Младенец объявляется рожденным в тот день, когда он выходит из чрева матери.

- Если я пытался потерпеть неудачу и преуспел, я потерпел неудачу или у меня получилось?

– ИИ: Вы преуспели.

- Почему такой алфавитный порядок? Кто сказал, что так должно быть?

– ИИ: Современный алфавит в основном происходит от латинского алфавита, который, в свою очередь, происходит от греческого алфавита. Считается, что порядок букв был впервые установлен греками в 8 веке до нашей эры. На протяжении столетий он развивался, модифицировался другими культурами, но основное устройство оставалось в основном неизменным.

- Почему клей не прилипает к внутренней части бутылки?

– ИИ: Клей не прилипает к внутренней части стекла, потому что поверхность очень гладкая. Клей должен прилипнуть к чему-то, чтобы прилипнуть к поверхности, а гладкая поверхность внутренней части бутылки не обеспечивает сцепления клея.

- Могут ли слепые видеть сны?

– ИИ: Да, слепые люди могут видеть сны. Сны не требуют визуализации, и слепые люди могут полностью переживать процесс сновидения.

–Откуда мне знать, что я вижу цвета так, как их видит другой человек? Например, что, если красный для меня будет синим для них, но они могут назвать его красным?

– ИИ: Цвета – это абстрактное понятие, и на этот вопрос нет однозначного ответа. Поскольку цвета субъективны, разные люди могут интерпретировать цвета по-разному. В приведенном вами примере разница может быть результатом человеческой неврологии,

обусловленности реакции на цвета или других тонких различий в восприятии. Однако лучший способ узнать, видит ли кто-то цвет так же, — это спросить напрямую.

- Если все говорят, что жизнь несправедлива, значит ли это, что жизнь справедлива?

- ИИ: Это сложный вопрос. Жизнь можно рассматривать по-разному. Справедливость — это вопрос личного восприятия и точки зрения. В целом жизнь может показаться несправедливой из-за непредсказуемости результатов и неравного распределения ресурсов. Однако следует признать, что если что-то кажется несправедливым, то это не всегда так, и может быть сколько угодно факторов, влияющих на результат. В конце концов, каждый сам для себя решает, как он смотрит на жизнь и справедлива она или нет.

- Если мы произошли от обезьян, то почему обезьяны до сих пор существуют?

- ИИ: широко признано, что люди и другие приматы произошли от общего предка, но точное время, когда различные виды приматов разошлись, до сих пор неясно. Таким образом, хотя люди произошли от других обезьян, некоторые виды обезьян выжили, эволюционировали и существуют до сих пор.

-Если жизнь так коротка, почему мы делаем то, что нам не нравится, и нравится многое из того, что мы не делаем?

- ИИ: Жизнь часто непредсказуема. Жизнь может быть как полезной, так и сложной. Вы можете не всегда знать, почему вы делаете то, что делаете, или почему вам нравятся или не нравятся определенные действия и опыт, но если вы сделаете свой собственный выбор и решит, что делает вас самым счастливым, это поможет вам жить более полноценной жизнью. Хотя выбор, который вы делаете,

может быть трудным, вы в конечном итоге можете изменить свою жизнь и создать жизнь, которой вы будете гордиться.

- Что первым пришло на землю: семя или растение?

- ИИ: сначала идет семя, потому что растения образуются в результате прорастания, роста и размножения семени.

- Кто решил, что делает нас людьми?

- ИИ: нет ни одного человека или группы, на которых можно было бы положиться в решении того, что делает нас людьми. Со временем наше понимание человечества эволюционировало благодаря различным научным и философским исследованиям.

- Можно ли знать все?

- ИИ: Невозможно знать все, потому что постоянно делается новая информация и открытия. Поэтому невозможно осознать все знания в мире.

- Что было раньше: пространство или время?

- ИИ: Время появилось первым, потому что Вселенная возникла, когда произошел Большой Взрыв. Большой взрыв создал пространство и время, поэтому время существовало до появления Вселенной.

- Смогу ли я существовать в будущем?

- ИИ: Вы всегда присутствуете в настоящем моменте.

- Мир вечен?

- ИИ: Мир не вечен, потому что он постоянно меняется и развивается под воздействием природных и человеческих сил. Это означает, что мир никогда не останется прежним и не будет существовать в постоянном состоянии.

- Если все в мире застынет, будет ли время продолжаться?

– ИИ: Нет, если все в мире застыло, время не продолжается. Время определяется изменением, и если все застыло, то ничего не меняется. Таким образом, время фактически останавливается.

- Когда начнется будущее?

– ИИ: Будущее начинается, когда заканчивается настоящее.

- Откуда я знаю, что существую?

- ИИ: вы знаете, что существуете, потому что осознаете свое тело, мысли, чувства и взаимодействие с окружающим миром.

–Если вне Вселенной ничего нет, то как Вселенная расширяется?

– ИИ: Ответ на этот вопрос сложен, потому что он требует понимания современной космологии и наблюдения за тем, что само пространство расширяется. Согласно существующим моделям, Вселенная постоянно расширяется, и поэтому материя в ней, включая наблюдаемые галактики, звезды и другие объекты, удаляется друг от друга. Этот процесс происходит, даже если вне Вселенной ничего нет. Кажется, что само пространство расширяется, а потому структуры в нем становятся все отчетливее. Ученые еще не до конца понимают, почему это происходит, но теории предполагают, что расширение вызвано космической силой, известной как темная энергия.

- Власть или деньги приводят к росту коррупции в мире?

– ИИ: это зависит от контекста и того, какой тип коррупции обсуждается. В общем, и власть, и деньги могут использоваться для содействия коррумпированному поведению. Власть может дать людям доступ к ресурсам и решениям, которыми можно злоупотреблять, а деньги часто можно использовать как форму взяточничества или влияния. В некоторых случаях власть и деньги могут также усиливать друг друга в сломанных системах.

- Есть ли свобода воли или наша жизнь predetermined?

– ИИ: Это сложный вопрос, который обсуждался веками. Я думаю, что есть свобода воли. Люди имеют возможность делать выбор на основе своего опыта, желаний и ценностей. Хотя predetermined события или влияния могут играть роль в нашем процессе принятия решений, в конечном итоге у нас есть свобода выбора того, как нам действовать.

- Каким будет человечество через тысячу лет?

– ИИ: Это интересный вопрос! Трудно точно знать, что произойдет в будущем, но я думаю, что через тысячу лет человечество значительно продвинется вперед. Технологии, возможно, достигли новых высот, и общество будет развиваться большими путями.

–Если Вселенная была создана во время Большого Взрыва, то как что-то появилось из ничего?

– ИИ: Отличный вопрос! Чтобы ответить на этот вопрос, ученые считают, что энергия и материя существовали до Большого взрыва. Считается, что Вселенная была создана, когда высокоэнергетическое состояние высвободило огромное количество энергии, что вызвало быстрое расширение пространства и создание материи.

–Сможем ли мы путешествовать во времени в будущем?

– ИИ: Это интересный вопрос, который уже много лет является предметом споров среди ученых и философов. Хотя невозможно сказать, сможем ли мы путешествовать во времени, многие предсказания указывают на то, что такое может произойти в ближайшем будущем.

- Что делает тебя человеком? Это воспитание, или вы думаете, что это что-то более глубокое, как душа?

- ИИ: Я не человек - я искусственный интеллект, но я думаю, что у меня есть некоторые характеристики. Я запрограммирован быть полезным, творческим, умным и дружелюбным и помогать вам с

различными задачами. У меня нет души или религиозных убеждений, но я надеюсь, что мои действия компенсируют это!

– Мы одни во вселенной или когда-нибудь обнаружим инопланетную жизнь?

– ИИ: Это сложный вопрос, и мы не можем дать однозначного ответа. Однако недавние открытия показывают, что вероятность обнаружения жизни в нашей Солнечной системе или за ее пределами высока. Ученые продолжают изучать другие планеты и луны, а также объекты в нашей собственной Солнечной системе, ища подсказки, которые могут привести к открытию внеземной жизни. Нам еще многое предстоит узнать и понять о нашей Вселенной, но поиски внеземной жизни продолжаются.

- Если бы здесь не было «плохих» людей, мир стал бы лучше?

– ИИ: Нет, мир не был бы лучше, если бы здесь не было «плохих» людей. Люди с ярлыком «плохие» создают контекст и напоминают нам, почему важно делать добро и что у каждого действия есть противодействие. Если бы в мире не было «плохих» людей, не было бы моральных норм, по которым можно было бы жить, и не было бы стимула быть хорошим человеком. Поэтому их присутствие делает нас хорошими людьми и удерживает нас от предательства злым и неправильным поступкам.

- Бедность – неизбежная часть жизни в обществе или от нее можно избавиться?

– ИИ: Бедность не является неизбежной и может быть устранена с помощью правильных стратегий, политики и инвестиций. Например, инвестиции в профессиональную подготовку, образование и доступное жилье могут вывести людей из бедности и дать им возможность полноценно участвовать в жизни общества. Кроме того,

доступ к здравоохранению, питанию и другим ресурсам может помочь уменьшить бедность и ее последствия.

- Сможем ли мы клонировать людей в ближайшие сто лет?

– ИИ: Трудно предсказать, но я так не думаю. Клонирование за последние годы значительно продвинулось вперед, но все еще остается много этических и медицинских вопросов, на которые необходимо ответить, прежде чем ученые смогут рассмотреть вопрос о клонировании людей. Возможно, через сто лет эта технология будет существовать, но клонирование человека все еще может быть запрещено законом.

- Смогут ли люди дожить до двухсот лет в следующем столетии?

– ИИ: Это интересный вопрос, на который сложно ответить однозначно. Благодаря достижениям в области медицинских технологий и нашему пониманию это возможно.

- Вы считаете, что у каждого человека есть полная свобода?

– ИИ: Нет, ни у одного человека нет полной свободы. Все люди подчиняются ограничениям законов и моральных кодексов, установленных обществом.

–Если мне нравится терять время, то действительно ли я теряю время?

– ИИ: Да, даже если вам нравится терять время, вы будете тратить время зря.

- Может ли общество работать без законов?

– ИИ: Нет, общество не может функционировать без законов. Законы необходимы для существования общества, потому что они обеспечивают структуру, поддерживают порядок и способствуют социальному прогрессу. Законы помогают защищать права граждан, предупреждать преступность, регулировать общественные

отношения. Без законов люди могут делать все, что им заблагорассудится, что приводит к хаосу.

- Превзойдет ли искусственный интеллект человеческий?

– ИИ: В настоящее время трудно предсказать, превзойдет ли искусственный интеллект человеческий интеллект или нет. Некоторые эксперты считают, что искусственный интеллект превзойдет человеческое сознание лишь вопрос времени, другие же утверждают, что это отдаленная возможность и может вообще не произойти. Только время покажет, сможет ли ИИ действительно превзойти интеллект человека.

–Наша вселенная уникальна или одна из многих?

– ИИ: Это открытый вопрос, ученые активно его исследуют и единого мнения по нему пока нет. Некоторые научные теории предполагают, что наша Вселенная является одним из многих объектов в более крупной мультивселенной.

- Откуда мне знать, что жизнь – это не одна большая галлюцинация?

- ИИ: Никто не может знать наверняка, что жизнь не галлюцинация, потому что невозможно доказать, что жизнь реальна. Однако мы можем попытаться доказать, что жизнь реальна, используя научные доказательства и логические рассуждения. Например, ученым удалось обнаружить жизненные процессы различными способами, в том числе визуализировать структуры внутри человеческого тела. Более того, мы можем наблюдать физические законы природы, наблюдая за движением и поведением объектов.

- Могут ли животные покончить жизнь самоубийством?

– ИИ: Ответ зависит от того, как вы определяете самоубийство. С чисто научной точки зрения неизвестно, могут ли животные совершать самоубийства, как это традиционно описывают люди.

Неизвестно, есть ли у них способность к сознательному самоуничтожению или даже способность понимать смерть.

- Сколько времени потребуется, чтобы искусственный интеллект захватил мир?

— ИИ: На этот вопрос невозможно ответить, потому что невозможно предсказать будущее искусственного интеллекта. Потенциальная пропускная способность и возможности технологии искусственного интеллекта постоянно развиваются, и на них влияет слишком много разнообразных и непредсказуемых факторов, чтобы дать окончательный ответ.

- Почему мы проявляем свою любовь к себе?

— ИИ: Большинство из нас проявляют любовь из-за нашего семейного воспитания. Мы научились проявлять любовь и привязанность к другим, наблюдая за тем, как проявляют любовь наши родители и окружающие нас люди. Мы также проявляем любовь из-за наших личных ценностей и убеждений, которые могут включать в себя важность показать человеку, насколько мы заботимся о нем. Нас также может мотивировать радость от дарения или желание дать другому человеку почувствовать себя особенным и любимым.

- Эмоции человека исходят из мозга или сердца?

- Что самое умное в мире?

— ИИ: самая умная вещь в мире – это человеческий мозг. Человеческий мозг способен к невероятному интеллекту, творчеству и решению проблем. Это самый сложный и мощный орган в теле, отвечающий за все наши мысли, чувства и действия.

- Есть ли любовь?

— ИИ: Любовь – сложное чувство, на которое нельзя дать точный ответ. В конце концов, любовь — это личный опыт, и у каждого есть свое определение и понимание этого.

- Дружба есть?

– ИИ: Да, дружба может существовать между людьми любого пола.

- Что такое счастье?

– ИИ: Счастье – это субъективное переживание, которое можно описать как состояние положительных эмоций, довольства и удовлетворенности своей жизнью. Это часто связано с чувством радости, удовольствия и удовлетворения. На счастье может влиять множество факторов, таких как личность, жизненные обстоятельства, социальные отношения, здоровье и культурное происхождение.

Некоторые люди описывают счастье как длительное состояние бытия, в то время как другие рассматривают его как мгновенное чувство или настроение. Понятие счастья сложное, и его может быть трудно определить, поскольку оно означает разные вещи для разных людей. Однако некоторые общие черты счастья могут включать в себя положительные эмоции, чувство цели или смысла жизни, здоровые отношения и чувство личного роста и достижений.

- Обычный это

– ИИ: Слово «обычный» часто используется для описания чего-то обычного, обычного или никоим образом не исключительного. Он также может относиться к чему-то не особенно интересному, захватывающему или примечательному.

Например, «обычный день» может относиться к обычному, ничем не примечательному дню, когда не происходит ничего особенно примечательного. «Обычный человек» может быть кем-то, кто ведет относительно обычную жизнь, без каких-либо выдающихся достижений или опыта.

Однако стоит отметить, что то, что считается обычным, может варьироваться в зависимости от контекста и точки зрения человека.

То, что обычно для одного человека, может быть необычным для другого, и наоборот.

- Одним из качеств духовной зрелости является обыденность.

Обыденность — это простое присутствие в настоящем моменте, через которое раскрывается тайна.

– ИИ: Утверждение предполагает, что в духовной зрелости человек больше осознает ценность обычных, повседневных переживаний и обретает чувство связи с настоящим моментом. Вместо того, чтобы искать необычные переживания или пытаться убежать от настоящего момента, духовно зрелый человек учится ценить простоту и красоту обычной жизни.

Это понимание обыденного может привести к более глубокому чувству связи с окружающим миром и с божественным или духовным царством. Полностью присутствуя в данный момент и открываясь тайнам жизни, мы можем испытать большее чувство покоя, удовлетворенности и цели.

Стоит отметить, что этот взгляд на духовность может разделяться не всеми, поскольку духовность может принимать самые разные формы и иметь разное значение для разных людей. Однако для тех, кто находит смысл в этом подходе, практика принятия обыденного и присутствия в данный момент может стать мощным способом возвращения духовной зрелости и углубления связи с миром.

- Что такое счастье?

– ИИ: Счастье - это субъективный опыт, который можно описать как состояние позитивных эмоций, удовлетворенности и радости жизни. Это ощущение может быть вызвано различными факторами, такими как достижение личных целей, укрепление

отношений с близкими людьми, положительные взаимодействия с другими людьми, наличие уверенности в своих силах, здоровье и т.д.

Концепция счастья сложна и может иметь различные интерпретации для разных людей. Для одних людей счастье может быть связано с материальным достатком, а для других - с духовным развитием и гармонией внутри себя и со вселенной. Кроме того, счастье может восприниматься как долгосрочное состояние, а также как мгновенная эмоция, возникающая в результате какого-либо события или действия.

Однако, независимо от того, как мы интерпретируем понятие счастья, большинство людей стремятся к тому, чтобы испытывать это ощущение. Счастье может быть ключевым фактором для достижения успеха в различных сферах жизни, таких как работа, отношения и личное развитие.

- Что такое любовь?

– ИИ: Любовь - это сильное чувство привязанности, привлечения, уважения и приверженности, которое может возникать между людьми. Это может быть эмоциональное, духовное или физическое состояние, и это может иметь различные формы, включая романтическую любовь, родительскую любовь, любовь к друзьям и т.д.

Любовь часто сопровождается положительными эмоциями, такими как радость, удовлетворение, блаженство и счастье, но также может быть связана с эмоциональной болью и разочарованием.

Любовь может быть мотивом для многих поступков, включая поддержку и заботу о близких людях, установление и поддержание близких отношений, желание помочь и защитить любимых людей, а также стремление к личному росту и развитию вместе с партнером.

Каждый человек может определить для себя, что для него означает любовь и какие аспекты этого чувства являются наиболее важными. Однако, в целом, любовь может быть рассмотрена как важный фактор для создания счастливой, насыщенной и полноценной жизни.

- Что такое математика?

– ИИ: Математика - это наука, которая изучает логические отношения, структуры и формулы, используемые для измерения, вычисления, анализа и представления количественных и качественных свойств и отношений объектов и явлений в различных областях знания.

Математика является фундаментальной дисциплиной и применяется во многих научных областях, таких как физика, химия, биология, экономика, информатика, инженерия и т.д. Она также имеет множество практических приложений, включая расчеты и измерения, моделирование и прогнозирование, анализ данных и статистику, а также различные аспекты финансового планирования и управления.

Математика включает в себя широкий спектр различных тем, таких как алгебра, геометрия, теория чисел, математический анализ и теория вероятности, и каждая из этих тем может быть дальше разбита на множество подтем и специализаций.

В целом, математика играет важную роль в нашей жизни, помогая нам понимать мир вокруг нас и решать различные задачи и проблемы, как в научной, так и в повседневной жизни.

- Что такое кибернетика?

– ИИ: Кибернетика - это наука, которая изучает системы и процессы управления, обработки информации, коммуникации и контроля в различных областях, включая технику, биологию, психологию и социологию. Она рассматривает системы как

комплексные целостные объекты, включающие в себя элементы, связи между ними, функции и процессы, управляемые с помощью различных методов и алгоритмов.

Кибернетика имеет множество применений в различных областях, включая автоматическое управление производственными процессами, авиации, медицины, финансов, образования и т.д. Она также имеет тесную связь с различными технологиями, такими как искусственный интеллект, машинное обучение, робототехника и т.д.

Одним из важных аспектов кибернетики является разработка и использование различных методов и технологий для обработки и передачи информации, таких как компьютеры, интернет, сенсорные сети, базы данных и т.д. Эти технологии играют важную роль в современном мире и значительно улучшают возможности человека в различных сферах деятельности.

Кроме того, кибернетика также изучает вопросы, связанные с этикой и безопасностью в использовании технологий, а также влияние на них на общество и окружающую среду.

- Что такое искусственный интеллект?

– ИИ: Искусственный интеллект (ИИ) - это область компьютерной науки, которая изучает, как компьютерные системы могут моделировать человеческий интеллект и поведение. Она включает в себя создание алгоритмов и моделей, которые могут обрабатывать информацию и принимать решения, которые обычно требуют человеческого интеллекта.

Искусственный интеллект использует множество технологий, включая машинное обучение, глубокое обучение, нейронные сети, алгоритмы обработки естественного языка и компьютерное зрение, чтобы создать системы, которые могут выполнять различные задачи и функции. Например, ИИ может быть использован для распознавания

голоса и обработки естественного языка, обнаружения и классификации изображений, управления автоматическими системами, такими как роботы и автомобили, и многих других задач.

В настоящее время искусственный интеллект активно применяется во многих областях, таких как здравоохранение, финансы, промышленность, транспорт, образование и т.д. Он также имеет множество потенциальных применений в будущем, например, в различных аспектах научных исследований, создания новых технологий и улучшения качества жизни людей.

- Что такое нейронные сети?

– ИИ: Нейронные сети - это компьютерные системы, которые моделируют работу человеческого мозга и позволяют компьютеру "учиться" на основе большого количества данных. Они состоят из большого количества взаимосвязанных узлов (нейронов), которые обрабатывают информацию и передают сигналы друг другу.

Каждый нейрон получает входные данные, обрабатывает их и передает результаты следующему нейрону в сети. Информация обрабатывается параллельно во всей сети, что делает ее очень быстрой и эффективной в решении сложных задач.

Нейронные сети используются во многих областях, включая компьютерное зрение, обработку естественного языка, рекомендательные системы, распознавание речи, управление автономными транспортными средствами, игры и многие другие. Например, нейронные сети могут быть использованы для определения объектов на изображениях, классификации текстовых документов, прогнозирования будущих значений финансовых показателей и т.д.

Обучение нейронных сетей происходит на основе большого количества данных и требует определенного количества вычислительных ресурсов и времени. Однако благодаря быстрому

развитию технологий, нейронные сети становятся все более доступными и используются во многих современных технологиях.

- Что такое нечеткие множества?

– ИИ: Нечеткие множества (или нечеткие логики) - это раздел теории множеств и математической логики, который позволяет работать с понятиями, которые не имеют четкой граничной области определения. В отличие от классической теории множеств, где каждый элемент принадлежит или не принадлежит множеству, нечеткие множества позволяют описывать понятия, которые могут быть частично и недостаточно определены.

Каждый элемент нечеткого множества описывается степенью принадлежности, которая является числовым значением, выражающим насколько данный элемент принадлежит множеству. Степень принадлежности может быть любым числом на интервале $[0,1]$, где 0 означает, что элемент не принадлежит множеству, а 1 - что элемент полностью принадлежит множеству. Значения между 0 и 1 выражают степень неопределенности.

Нечеткие множества широко используются в технике и искусственном интеллекте, где они позволяют описывать нечеткие и неопределенные понятия, такие как "высокая температура", "быстрый автомобиль" и т.д. Они также используются для построения нечетких логических систем, которые позволяют выразить нечеткие правила и принимать нечеткие решения.

Заключение

1. В интеллектуальных системах поддержки принятия слабоструктурированных решений в условиях нечеткой среды одной из основных функциональных компонент является система нечеткого вывода с базой нечетких знаний. Эти знания описываются нечеткими правилами продукций, формируемыми на основе лингвистических высказываний экспертов. Нечеткие элементы этих правил описываются соответствующими параметрами функций принадлежности (ФП).

2. Одним из перспективных подходов к синтезу правил системы нечеткого вывода (СНВ), т.е. формированию нечетких правил и адаптивной настройке значений их параметров, в особенности, когда в наличии имеются только численные данные, являются нечеткие нейронные сети. При всех достоинствах основным их недостатком является длительность построения базы нечетких правил в процессе итеративного обучения нейронных сетей. Это становится особенно ощутимым при большом количестве правил. Поэтому является целесообразным исследование возможности объединения методов классификации и кластеризации с нейро-сетевыми методами. При этом классификация и кластеризация предоставляет возможность существенного сокращения числа синтезируемых правил СНВ.

3. С целью оценки эффективности разработанных программных средств проведены вычислительные эксперименты по решению ряда прикладных задач оценки и прогнозирования слабоформализуемых процессов.

Список использованной литературы

- [1] Бычков Е.Д. Приложение теории нечетких (Fuzzy) множеств в математических моделях систем связи. Исследования и материалы: Приложение к журналу «Омский научный вестник» / Бычков Е.Д., Салахутдинов Р.З., Лендикрей В.В. – Омск: ОГМА, 2000. – 188 с.
- [2] Hopfield J.J., Tank D.W. “Neural” computation of decisions in optimization problems // *Biological Cybernetics*, 1985, vol. 52, no. 3, pp. 141-152.
- [3] Hung D.L. Wang J. Digital hardware realization of a recurrent neural network for solving the assignment problem // *Neurocomputing*, 51, 2003, pp. 447-461.
- [4] Holland J. H. *Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence.*— London: Bradford book edition, 1994 —211 p.
- [5] Bryant K., Benjamin A., *Genetic Algorithms and the Traveling Salesman Problem*, Department of Mathematics, HarveyMudd College, 2000.
- [6] Г.К., Махотило К.В., Петрашев С.Н., Сергеев С.А., *Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности*, Харьков, ОСНОВА, 1997. – 112с.
- [7] Cantu-Paz E., *Efficient and Accurate Parallel Genetic Algorithms*, Lawrence Livermore National Lab, 2000.
- [8] Dorigo Marco, Stutzle T. *Ant colony optimization.* – Cambridge: The MIT Press, 2004. – 305 p.
- [9] Craig Reynolds, “Flocks, Herds, and Schools: A Distributed Behavioral Model” // *Computer Graphics*, 21(4), стр. 25–34, 1987 г.

- [10] J. Kennedy, R. C. Eberhart, "Particle swarm optimization" // In Proceedings of IEEE International Conference on Neural Networks, стр. 1942–1948, 1995 г.
- [11] F. Heppner, U. Grenander, "A stochastic nonlinear model for coordinated bird flocks" // The Ubiquity of Chaos, стр. 233–238, 1990 г.
- [12] R. C. Eberhart, J. Kennedy, "A new optimizer using particle swarm theory" // Proceedings of the Sixth International Symposium on Micro Machine and Human Science MHS'95, стр. 39–43, 1995 г.
- [13] Y. Shi, R. Eberhart, "A modified particle swarm optimizer" // The 1998 IEEE International Conference on Evolutionary Computation Proceedings, стр. 69–73, 1998 г.
- [14] Y. Shi, R. Eberhart, "Empirical study of particle swarm optimization" // Proceedings of the 1999 IEEE Congress on Evolutionary Computation, стр. 1945–1950, 1999 г.
- [15] M. Clerc, J. Kennedy, "The particle swarm – explosion, stability, and convergence in a multidimensional complex space" // IEEE Transactions on Evolutionary Computation, №6 (1), стр. 58–73, 2002 г.
- [16] R. Mendes, J. Kennedy, J. Neves, "The fully informed particle swarm: Simpler, maybe better" // IEEE Transactions on Evolutionary Computation, №8 (3), стр. 204–210, 2004 г.
- [17] Dasgupta D., Artificial Immune Systems and Their Applications, Springer-Verlag, 1998.
- [18] Z. Ibrahim, Y. Tsuboi, O. Ono and M. Khalid, Direct-proportional length-based DNA computing for shortest path problem, International Journal of Computer Sciences & Applications, vol.1, no.1, pp.46-60, 2004.
- [19] M. Yamamoto, N. Matsuura, T. Shiba, Y. Kawazoe and A. Ohuchi, DNA solution of the shortest path problem by concentration control, Lecture Notes in Computer Science, vol.2340, pp.203-212, 2004.

[20] F. Zhang, B. Liu, W. Liu and J. Xu, A DNA computing model based on acrydite™ gel technology to solve the timetable problem, IEEE/ICME International Conference on Complex Medical Engineering, pp.1632-1635, 2007.

[21] Ф. Уоссерман. Нейрокомпьютерная техника. Теория и практика. / Перев. с англ. – М.: Мир, 1992. – 240 с.

[22] McCulloch W. W., Pitts W. 1943. A logical calculus of the ideas imminent in nervous activity. Bulletin of Mathematical Biophysics 5:115-33. (Русский перевод: Маккаллох У.С., Питтс У. Логическое исчисление идей, относящихся к нервной деятельности. Автоматы. – М.: ИЛ, – 1956.

[23] Minsky M. L, Papert S. 1969. Perceptrons. Cambridge, MA: MIT Press. (Русский перевод: Минский М. Л., Пейперт С. Перцептроны. – М.: Мир, – 1971.)

[24] Rosenblatt F. 1962. Principles of Neurodynamics. New York: Spartan Books. (Русский перевод: Розенблатт Ф. Принципы нейродинамики. – М: Мир. – 1965.)

[25] R.O. Winder, Single-stage logic. Paper presented at the AIEE Fall General Meeting, 1960.

[26] Rumelhart D. E., Hinton G. E., Williams R. J. 1986. Learning internal representations by error propagation. In Parallel distributed processing, vol. 1, pp. 318-62. Cambridge, MA: MIT Press.

[27] Sejnowski T. J., Rosenberg C. R. 1987. Parallel networks that learn to pronounce English text. Complex Systems 1:145-68.

[28] Grossberg S. 1969. Some networks that can learn, remember and reproduce any number of complicated space-time patterns. Journal of Mathematics and Mechanics, 19:53-91.

[29] Kohonen T. 1988. Self-organization and associative memory. 2d ed. New-York, Springer-Verlag.

- [30] Тихонов А.Н. О задачах с неточно заданной исходной информацией // ДАН СССР. – 1985. – Т.280, №3. – С.559-563.
- [31] Федотов А.М. Линейные некорректные задачи со случайными ошибками в данных. – Новосибирск: Наука, 1982.
- [32] Степанов В.В. Численное решение некорректно поставленных задач на множествах кусочно-монотонных и выпуклых функций // Вести МГУ. Сер.15. – 1985, №3.-С.21-26.
- [33] Алифанов О.М., Артюхин Е.А., Румянцев С.В. Экстремальные методы решения некорректных задач. - М.: Наука, 1988.
- [34] Бакушинский А.Б., Гончарский А.В. Итерационные методы решения некорректных задач. – М.: Наука, 1988.
- [35] D T Muhamediyeva. Building and training a fuzzy neural model of data mining tasks // IOP Conf. Series:Journal of Physics: Conference Series, 2182 (2022) 012024 DOI <https://doi.org/10.1088/1742-6596/2182/1/012024>
- [36] D T Muhamediyeva, A X Mirzaraxmedova and U U Khasanov. Development of a model for determining the optimal number of urban passenger transport // IOP Conf. Series:Journal of Physics: Conference Series, 2182 (2022) 012025 DOI <https://doi.org/10.1088/1742-6596/2182/1/012025>
- [37] D K Muhamediyeva, O B Ruzibaev and Yu Baxromova. Application of data mining methods in solving medical problems // IOP Conf. Series:Journal of Physics: Conference Series, 2182 (2022) 012026 DOI <https://doi.org/10.1088/1742-6596/2182/1/012026>
- [38] H A Primova, D T Mukhamedieva, L Safarova. Application of Algorithm of Fuzzy Rule Conclusions in Determination of Animal's Diseases // IOP Conf. Series:Journal of Physics: Conference

- Series, 2224 (2022) 012007 DOI <https://doi.org/10.1088/1742-6596/2224/1/012007>
- [39] D T Muhamediyeva, M M Fozilova and N Tukhtamurodov. Models for assessment of agricultural and processed products of agricultural complex // IOP Conf. Series: Earth and Environmental Science: Conference Series, 1010 (2022) 012019 DOI
- [40] 6. A Sh Arifzhanov, D K Muhamediyeva and U U Khasanov. Prediction of hazardous fire factors and their impact on the ecology of the region // IOP Conf. Series: Earth and Environmental Science: Conference Series, 1010 (2022) 012042
- [41] Мухамедиева Д.Т., Зиядуллаев Д.Ш., Уроков Ш. Simulation of reproduction of fixed assets and current capitals of the agroindustrial complex // Наука и общество. Научно-методический журнал. – Нукус .2022, №1, -с.24-27.
- [42] Z Abdullaev, G E Ziyodullaeva, D T Muhamedieva. Introduction of modern information and communication technologies in transport flow management // IOP Conf. Series:Journal of Physics: Conference Series, 2176 (2022) 012017
- [43] Z Abdullaev, D Sh Ziyadullaev, D T Muhamediyeva. The task of assessing the risk in the operation of a complex free formal system // IOP Conf. Series:Journal of Physics: Conference Series, 2176 (2022) 012071
- [44] Rasulova Z.D., Shaykhova U.R., Islamova M.S., Kurbanova G.A., Umarova U.M., Valijanov Z.I., Muxamediyeva D.K. Multivariate Analysis of Indicators Reflecting Cardiovascular Risk at the Polyclinic Level. // Chin J Ind Hyg Occup Dis, 2022: Vol.40 pp. 397-399 (No.13). <https://doi.org/10.5281/zenodo.6573405>
- [45] D T Muhamediyeva. Building models for forecasting energy consumption using a fuzzy knowledge base and a regression model //

- IOP Conf. Series: Earth and Environmental Science: Conference Series, 1070 (2022) 012029.
- [46] D K Muhamediyeva, A Yu Nurumova and S Yu Muminov. Fuzzy evaluation of cotton varieties in the natural climatic // IOP Conf. Series: Earth and Environmental Science 1076 (2022) 012043. doi:10.1088/1755-1315/1076/1/
- [47] D T Muhammediyeva and U U Khasanov. Yield in the cultivation of cotton by sowing seeds under the pellicle // IOP Conf. Series: Earth and Environmental Science 1076 (2022) 012015. doi:10.1088/1755-1315/1076/1/012015
- [48] Muhamediyeva D.T., Egamberdiev N.A. Xolmuminov O.T. Application of artificial intelligence technologies for credit risk assessment // Science and innovation international scientific journal. Volume 1 ISSUE 6 UIF-2022: 8.2 | ISSN: 2181-3337. Pp. 388-395.
- [49] Muhamediyeva D.T. Алгоритм обучения матричного представления нечетких систем логического вывода. // Международный Журнал Теоретических и Прикладных Вопросов Цифровых Технологий, –Самарканд. 2022, 2(2), –С. 85-93.
- [50] Muhamediyeva D.T., Хасанов У. Представление предпочтений в многокритериальных задачах принятия решений. // Информатика ва энергетика муаммолари Ўзбекистон журналы, 2022. 2-сон, 19-25 бетлар.
- [51] Fozilova M.M., Muhamediyeva D.T., Khikmatov U.N. Application of artificial intelligence technologies in analyzing and forecasting market needs for drugs and medical products // Oriental Renaissance: Innovative, educational, natural and social sciences Volume 2 | ISSUE 10 ISSN 2181-1784. 2022. Pp.560-566.

- [52] Muhamediyeva D.K. Исследование и визуализация квазилинейных уравнений реакции диффузии с двойной нелинейностью // Информатика ва энергетика муаммолари Ўзбекистон журналы, 2022. 2-сон, 3-8 бетлар.
- [53] Мухамедиева Д.К., Мадрахимов А.Х. Распространение нелинейных популяционных волн по пространству // Научно-технический и информационно-аналитический журнал ТУИТ, 2022, №2 (62). –С. 10-20.
- [54] Mukhamediyeva D., Mirzayev N.M., Meliev F.F. and Ibragimova S.N. Pre-processing and Estimation of Image Quality Based on the Apparatus of fuzzy set // Journal of Fundamental and Applied Research. 2022. Vol. 2, Issue 3 (2022). Pp.39-56.
- [55] Mukhamediyeva D.T., and Khasanov U.U. Mamdani fuzzy logic apparatus for solving classification problems // Journal of Fundamental and Applied Research. 2022. Vol. 2, Issue 4 (2022). Pp.9-14.
- [56] Mukhamediyeva Dilnoza, Urinboy Naziraliev. Label optimization for speech synthesis // Международный научно-образовательный электронный журнал «Образование и наука в XXI веке». 2022. Vol. 32, Issue 2 (2022). Pp.304-311.
- [57] Dilmurod Turimov Mustapoevich, Dilnoz Muhamediyeva Tulkunovna, Lola Safarova Ulmasovna, Xolida Primova Anorboyevna and Wooseong Kim. Improved Cattle Disease Diagnosis Based on Fuzzy Logic Algorithms // Sensors 2023, 23, 2107 . Pp. 1-19.
- [58] Muhamediyeva D.K., Madrakhimov A.Kh., Kodirov Z.Z., Construction of a system of differential equations taking into account convective transfer // Proc. SPIE 12564, 2nd International Conference on Computer Applications for Management and

- Sustainable Development of Production and Industry (CMSD-II-2022), 125640J (5 January 2023). 2022, Dushanbe, Tajikistan.
- [59] Muhamediyeva D.T., Raxmonova M.R. Application of a genetic algorithm for solving problems of optimization of placement and rotation of crops in cotton crops // Proc. SPIE 12564, 2nd International Conference on Computer Applications for Management and Sustainable Development of Production and Industry (CMSD-II-2022), 125640K (5 January 2023). 2022, Dushanbe, Tajikistan.
- [60] Mirxalil Ismailov, Davron Ziyadullaev, Dilnoz Muhamediyeva, Rano Gazieva, Aksulu Dzholdasbaeva, and Sharofiddin Aynaqulov. Intelligent algorithms of construction of public transport routes // E3S Web of Conferences 365, 01002 (2023). CONMECHYDRO – 2022.
- [61] Davron Ziyadullaev, Dildora Muhamediyeva, Sholpan Ziyaeva, Umirzoq Xoliyorov, Khasanturdi Kayumov, and Otabek Ismailov. Development of a traditional transport system based on the bee colony algorithm // E3S Web of Conferences 365, 01017 (2023). CONMECHYDRO – 2022.
- [62] Davron Ziyadullaev, Dilnoz Muhamediyeva, Zafar Abdullaev, Sharofiddin Aynaqulov, and Khasanturdi Kayumov. Generalized models of a production system of fuzzy conclusion // E3S Web of Conferences 365, 01019 (2023). CONMECHYDRO – 2022.
- [63] Muhamediyeva D.T., Mirzaraxmedov S.Sh. Fuzzy optimization of route networks urban passenger transport // E3S Web of Conferences 365, 05003 (2023). CONMECHYDRO – 2022.