ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

На правах рукописи

Цой Юрий Робертович

Нейроэволюционный алгоритм и программные средства для обработки изображений

05.13.01 – Системный анализ, управление и обработка информации (отрасль: информация и информационные системы)

Д и с с е р т а ц и я
на соискание ученой степени
кандидата технических наук

Научный руководитель –
доктор технических наук,
старший научный сотрудник
В.Г. Спицын

Томск – 2007

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

У. Мак-Каллок и В. Питтс в своей основополагающей работе «Логическое исчисление идей, относящихся к нервной активности» [163] показали, что представленный ими формализм описания нейронов позволяет задавать сети, способные (при «оснащении» лентой, считывающим устройством и эффекторами) вычислять те же числа, что и универсальная машина Тьюринга. Во второй половине 50-х годов А.Н. Колмогоров [14] доказал теорему об универсальных аппроксимирующих способностях эквивалентных искусственным нейронным сетям (ИНС) вычислительных структур. В конце 80-х годов Г. Цыбенко была доказана теорема об универсальных аппроксимирующих способностях многослойного перцептрона [79], а в 90-х годах В. Крейнович и А.Н. Горбань независимо доказали, что основным условием существования аппроксимирующих способностей ИНС является нелинейность функции активации нейронов [9, 150]. Тем самым была показана перспективность использования ИНС при решении задач, для которых сложно найти формальный алгоритм решения.

Возможность обучения ИНС является одним из основных преимуществ нейросетевого подхода, позволяющим аппроксимировать функциональные зависимости между входными и выходными данными на основе обучающего множества данных. Существенной для нейросетевой модели является структура ИНС и значения весов ее связей. Однако в настоящее время отсутствует формализованный метод подбора структуры ИНС в зависимости от поставленной задачи и ее характеристик. Поэтому построение качественной нейросетевой модели часто требует высокого уровня квалификации специалиста. Еще одной проблемой «классического» нейросетевого подхода является необходимость формирования обучающего множества, что не всегда осуществимо, например, при решении некорректных задач, либо из-за отсутствия формализованной модели исследуемого объекта.

Комплексное решение этих проблем возможно с использованием эволю-

ционных алгоритмов (ЭА), а соответствующий подход будем называть нейроэволюционным (НЭ). За последние 20-25 лет был выполнен большой объем исследований, посвященных НЭ подходу для решения разнообразных задач, связанных с настройкой структуры и весов связей ИНС. Активно развиваются идеи НЭ подхода применительно к вопросам реализации многоагентных систем [31], адаптивного поведения и искусственной жизни [26], эволюционной робототехники [172]. Указанным проблемам посвящены работы Д. Флореано, Р. Миикkулайнена, С. Нолфи, В.Г. Редько, К. Стенли и других исследователей [2, 21, 26, 96, 101, 110, 157, 172, 206, 230].

Применительно к НЭ подходу Фредериком Груо [117] было показано, что клеточное кодирование[1], использующее набор из 20 специфичных операций (включающих операции безусловного перехода, «расщепления» нейрона, останова и др.) и предназначенное для представления последовательности правил, определяющих параметры ИНС, по выразительности не уступает универсальной машине Тьюринга и клеточным автоматам. Заметим, что идея использования эволюционных принципов для машинного обучения обсуждается и в фундаментальном труде А. Тьюринга «Могут ли машины мыслить?» [217].

Одной из основных проблем, возникающих при разработке НЭ алгоритмов, позволяющих одновременно настраивать структуру и веса связей ИНС, является проблема эффективности, вызванная огромным пространством поиска, включающем не только все возможные (для выбранного способа кодирования) комбинации весов межнейронных связей для различных структур ИНС, но и пространство самих структур. Также необходимо отметить отсутствие единой методологии разработки таких алгоритмов. Как следствие, к настоящему времени создано очень мало эффективных НЭ алгоритмов для одновременной настройки и обучения ИНС. При этом количество парамет-

[1] *Клеточное кодирование* (*cellular encoding*) – метод кодирования в виде дерева с упорядоченными ветвями и помеченными вершинами множества ИНС со схожей структурой. Подробнее о клеточном кодировании см. Gruau F. Neural network synthesis using cellular encoding and the genetic algorithm. Unpublished PhD thesis. – l'Universite Claude Bernard, Lyon, 1994. http://citeseer.ist.psu.edu/

ров в этих алгоритмах достаточно велико, их настройка часто производится экспериментальным путем и требует достаточно высокой квалификации пользователя в области ЭА и ИНС, что затрудняет практическое использование НЭ алгоритмов.

Таким образом, можно сделать вывод об актуальности проблемы разработки НЭ алгоритмов, использующих механизмы адаптации для подстройки значений параметров в процессе работы.

Цель работы и задачи исследования. Целью работы является разработка основанного на методах эволюционных вычислений адаптивного алгоритма для одновременной настройки структуры и весов связей ИНС и программных средств для обработки цифровых изображений.

Для достижения поставленной цели необходимо последовательное решение следующих задач:

1. Общий анализ ЭА и НЭ подхода и формулировка на основе результатов анализа требований к адаптивному НЭ алгоритму для ИНС прямого распространения.

2. Разработка адаптивных генетических операторов на основе анализа ЭА с точки зрения сформулированных требований к НЭ алгоритму. Решение данной задачи предполагает также исследование эффективности предлагаемых генетических операторов.

3. Разработка адаптивного НЭ алгоритма с учетом результатов решения предыдущих задач. Решение данной задачи предполагает также исследование эффективности предлагаемых методов и алгоритмов.

4. Апробация разработанного НЭ алгоритма для решения задачи улучшения визуального качества цифровых изображений на основе приближенной оценки качества работы ИНС.

Методы исследований. В работе использованы методы теории множеств, теории графов, прикладной математики, теории вероятностей, математической статистики, теории оптимизации, цифровой обработки изображе-

ний, теории информации, мягких вычислений.

Апробация работы. Основные результаты работы докладывались и обсуждались на следующих симпозиумах, конференциях и семинарах: Региональная конференция-конкурс «Технологии Microsoft в информатике и программировании» (г. Новосибирск, 2004 г.), II и IV Всероссийская конференция «Молодежь и современные информационные технологии» (г. Томск, 2004 и 2006 гг.), VIII и IX Русско-Корейский международный симпозиум по науке и технологии KORUS-2004 (г. Томск, 2004 г.) и KORUS-2005 (г. Новосибирск, 2005 г.), X и XI Международная конференция «Современные техника и технологии» (г. Томск, 2004 и 2005 гг.), Международная конференция «Интеллектуальные системы AIS'04» (Россия, п. Дивноморское), XIII Всероссийский семинар «Нейроинформатика и ее приложения» (г. Красноярск, 2005 г.), VIII и IX Всероссийские научно-практические конференции «Нейроинформатика – 2006» и «Нейроинформатика – 2007» (г. Москва), USNC/URSI National Radio Science and AMEREM Meetings (Albuquerque, New Mexico, USA, 2006), Всероссийская научная конференция «Нечеткие системы и мягкие вычисления» (г. Тверь, 2006 г.), Десятая национальная конференция по искусственному интеллекту с международным участием (КИИ-06) (г. Обнинск).

По результатам диссертационных исследований опубликовано 26 работ, в том числе 21 статья.

Кратко изложим **основное содержание работы**.

В **первой главе** представлен аналитический обзор эволюционных алгоритмов, искусственных нейронных сетей и нейроэволюционного подхода. Рассмотрены основные способы адаптации параметров в эволюционных алгоритмах, а также отмечена эффективность использования генетических операторов рекомбинации и вариации на уровне фенотипов (декодированных из хромосом решений). Рассмотрены случаи, вызывающие сложности при формировании обучающего множества данных, необходимого для настройки ве-

сов связей ИНС, и показано, что одним из возможных выходов в случае отсутствия обучающего множества может быть использование эволюционного обучения с интегральной оценкой ИНС, отражающей качественные аспекты ее функционирования.

Проанализированы возможности использования ИНС для решения задач обработки и анализа изображений. На основе анализа показаны сложности нейросетевой предобработки изображений и предложено использование ИНС для аппроксимации преобразования яркости пикселей.

В результате анализа основных задач, решаемых с использованием НЭ алгоритмов и включающих как раздельную, так и одновременную настройку структуры и весов связей ИНС, показаны основные преимущества и недостатки применения НЭ подхода. Также рассмотрены примеры решения задач адаптивного управления, адаптивного поведения, эволюционной робототехники и компьютерного творчества с использованием НЭ алгоритмов. На основе анализа ЭА, ИНС и НЭ подхода сформулированы следующие требования к адаптивному НЭ алгоритму для одновременной настройки структуры и весов связей ИНС:

1. Возможность работы с хромосомами переменной длины.
2. Независимость от порядка следования генов в хромосоме.
3. Возможность одновременного существования ИНС с различной структурой в одной популяции.
4. Минимизация вероятности появления НС решений с «плохой» структурой.
5. Возможность поиска структуры ИНС как в сторону усложнения, так и в сторону упрощения.
6. Настройка параметров в процессе работы алгоритма для улучшения получаемых результатов и возможности адаптации к условиям эволюционного поиска при решении различных задач.

Вторая глава посвящена исследованию генетического алгоритма, пред-

лагаемых операторов генного кроссинговера и адаптации размера популяции. Предложен более простой по сравнению с известными способ вычисления времени смешивания для генетических операторов кроссинговера для целочисленного кодирования. Особенностью предлагаемого способа является исключение из анализа динамики популяции, что значительно упрощает процесс получения искомых оценок, которые согласуются как с уже имеющимися аналитическими оценками, так и с результатами экспериментов.

Представлен генный оператор кроссинговера для неупорядоченных целочисленных хромосом переменной длины, в котором при скрещивании хромосом соответствующие гены скрещиваются независимо друг от друга. Результаты исследования эффективности генного ОК показали его лучшую масштабируемость, по сравнению с традиционно используемыми 1-, 2-точечным и однородным ОК, а также меньшее убывание эффективности увеличения размера популяции с ростом размерности оптимизируемой функции.

Предложена стратегия адаптации размера популяции с использованием последовательности Фибоначчи, позволяющая адаптироваться к характеристикам эволюционного поиска. Стратегия разработана на основе результатов проведенного исследования, направленного на изучение влияния изменения размера популяции на процесс эволюционного поиска. Экспериментально показано, что применение разработанной стратегии адаптации размера популяции позволяет в большинстве случаев получить результаты, которые сопоставимы или лучше результатов ГА с постоянным размером популяции. Отмечена эффективность предложенной стратегии адаптации размера популяции для случая малого начального размера популяции.

В **третьей главе** описывается разработанный НЭ алгоритм NEvA для одновременной настройки весов и связей ИНС, разработанный в соответствии с требованиями, сформулированными в Главе 1, и с использованием результатов исследований ГА в Главе 2. Представлены генетические операторы рекомбинации и мутации, работающие на уровне фенотипов и позволяющие

использовать в НЭ алгоритме ИНС различной структуры. Это дает возможность рассмотреть модельную эволюцию структуры ИНС, начиная со структуры без скрытых нейронов, с постепенным недетерминированным увеличением количества скрытых нейронов и межнейронных связей. Использование такого подхода обеспечивает получение достаточно компактных, с точки зрения структуры, ИНС. Представленный адаптивный оператор мутации позволяет уменьшить вероятность появления ИНС с «плохой» структурой, содержащих большое число слабо связанных между собой скрытых нейронов.

Экспериментальная проверка разработанного НЭ алгоритма на задачах классификации и адаптивного нейроуправления и сравнение с рядом известных алгоритмов и методов показали его эффективность как с точки зрения выбранного критерия оценки (количество вычислений целевой функции), так и с точки зрения структуры получаемых ИНС. Отмечено высокое быстродействие алгоритма NEvA, благодаря которому среднее время решения тестовых задач имеет порядок нескольких секунд.

На примере тестовых задач показано, что возможно использование фиксированного набора начальных значений параметров алгоритма NEvA для решения различных по сложности задач за счет реализации в разработанном НЭ алгоритме механизмов адаптации, позволяющих подстраивать значения параметров в зависимости от характеристик эволюционного поиска. Также на примере решения тестовой задачи адаптивного нейроуправления продемонстрировано, что возможность удаления в результате мутации межнейронных связей, позволяет уменьшить размерность исходной задачи за счет удаления связей от входных нейронов, соответствующих малоинформативным признакам.

Описывается разработанная инструментальная библиотека классов EC-Workshop, для проведения исследований в области эволюционных и нейроволюционных вычислений и решения практических задач.

В **четвертой главе** описываются результаты применения разработанно-

го алгоритма NEvA к решению задачи улучшения визуального качества цифровых изображений. Предлагается трехэтапный способ улучшения качества цифровых изображений включающий в себя следующие этапы: (1) предобработка яркости исходного изображения; (2) обработка на локальном уровне с использованием ИНС; (3) обработка на глобальном уровне с применением известного алгоритма автонастройки уровней яркости.

Получены формулы для приближенного вычисления локальных характеристик (среднее и дисперсия) изображения, позволяющие быстро вычислять локальные среднее и дисперсию в прямоугольной окрестности с приемлемой точностью (более 20 дБ для среднего и более 30 дБ для дисперсии). Использование полученных формул позволяет существенно повысить скорость нейросетевой обработки изображений по локально-адаптивному методу по сравнению с использованием точных формул (свыше 30 раз для окрестности 65x65 пикселей).

Экспериментально показано, что использование эволюционирующих нейронных сетей с достаточно грубой оценкой качества их функционирования, учитывающей только некоторые общие требования к характеристикам обработанного изображения, представляет эффективный способ получения нейросетевого решения для быстрой и качественной попиксельной обработки изображений. Время поиска решения составляет около 80 сек., а скорость обработки изображений для окрестности 11x11 пикселей составляет около $0{,}9 \times 10^6$ пикселей/сек. для процессора AMD Sempron 2500, работающего на частоте 1750 МГц. Сравнение результатов разработанного способа трехэтапной нейросетевой обработки изображений с используемой в NASA технологией Multi-Scale Retinex (MSR) показало сопоставимость результатов при небольшом превосходстве MSR, однако вычислительная сложность у предлагаемого способа меньше за счет использования приближенных формул для вычисления локальных характеристик и отказа от применения преобразования Фурье.

Описываются программные средства, реализующие разработанные способы обучения ИНС для обработки изображений и трехэтапный способ улучшения визуального качества изображений.

Научную новизну полученных в работе результатов определяют:

1. Способ вычисления времени смешивания для операторов кроссинговера для целочисленного кодирования, отличающийся от известных меньшей вычислительной сложностью за счет исключения из анализа динамики популяции.

2. Стратегия адаптации размера популяции в процессе работы эволюционного алгоритма, отличающаяся от известных стратегий подстройки размера популяций реализацией подхода, при котором популяция увеличивается при отсутствии улучшения и уменьшается в обратном случае, а также использованием последовательности Фибоначчи.

3. НЭ алгоритм NEvA для одновременной эволюционной настройки структуры и весов связей ИНС, отличающийся от известных НЭ алгоритмов большими возможностями к адаптации в процессе эволюционного поиска.

4. Формулы для приближенного вычисления локальных характеристик изображений, позволяющие значительно ускорить вычисления, необходимые для осуществления обработки изображений и отличающиеся от формул алгоритма box-filtering для ускорения вычисления локальных характеристик меньшими требованиями к объему оперативной памяти.

5. Трехэтапный способ улучшения качества полутоновых и цветных цифровых изображений, отличающийся от известных подходов использованием ИНС для попиксельной локально-адаптивной обработки изображений.

Практическая ценность и реализация результатов работы. Практически значимыми являются методы, алгоритмы, аналитические оценки и формулы, разработанные и полученные в результате диссертационного исследования.

Разработанные программные средства для обработки изображений ис-

пользуются в ОАО «ТомскНИПИнефть ВНК». Результаты внедрения подтверждены соответствующим актом. Результаты диссертационного исследования внедрены в учебный процесс в Томском политехническом университете, в Томском государственном университете систем управления и радиоэлектроники и в Северской государственной технологической академии.

Личный вклад:

1. Постановка задач диссертационного исследования выполнена автором совместно с В.Г. Спицыным.

2. Вывод оценок времени смешивания для операторов кроссинговера и экспериментальная проверка полученных формул выполнены автором.

3. Генный оператор кроссинговера и стратегия адаптации размера популяции разработаны автором. Постановка задачи исследования их эффективности выполнена автором. Им же получены результаты.

4. НЭ алгоритм NEvA для одновременной настройки структуры и весов межнейронных связей ИНС разработан автором. Постановка задач исследования эффективности разработанного НЭ алгоритма и результаты осуществлены и получены автором.

5. Постановка задачи применения НЭ алгоритма для улучшения визуального качества изображений выполнена автором совместно с В.Г. Спицыным.

6. Формулы для приближенного вычисления локальных среднего и дисперсии изображений получены автором. Исследование полученных формул выполнено автором.

Основные положения, выносимые на защиту:

1. Разработанный новый способ вычисления оценок времени смешивания для генетических операторов кроссинговера для целочисленного кодирования позволяет значительно уменьшить вычислительную сложность вывода оценок времени смешивания благодаря исключению из анализа динамики популяции.

2. Разработанный НЭ алгоритм NEvA для одновременной эволюционной настройки структуры и весов связей ИНС позволяет эффективно решать тестовые задачи при сравнении с известными алгоритмами и подходами.

3. Полученные формулы для приближенного вычисления локальных характеристик изображений позволяют значительно ускорить вычисления, необходимые для осуществления обработки изображений, по сравнению с точными формулами при сохранении приемлемой точности результатов.

4. Трехэтапный способ улучшения качества цифровых изображений позволяет быстро и эффективно улучшать визуальное качество изображений.

Автор выражает глубокую благодарность научному руководителю доктору технических наук В.Г. Спицыну за помощь в написании работы, ценные советы, замечания и доброжелательную критику. Автор также благодарит за плодотворные дискуссии и ценные замечания доктора технических наук, профессора, заведующего кафедрой вычислительной техники Н.Г. Маркова, доцентов Томского политехнического университета кандидатов технических наук А.В. Замятина, Ю.Я. Кацмана, А.А. Напрюшкина, кандидата физико-математических наук Ю.Б. Буркатовскую, доцента Московского авиационного института кандидата технических наук Ю.В. Тюменцева, научного сотрудника Института автоматики и процессов управления ДВО РАН кандидата технических наук Б.С. Ноткина. Автор выражает отдельную благодарность доктору физико-математических наук, профессору В.Г. Редько (Институт оптико-нейронных технологий РАН, г. Москва), доктору технических наук, профессору А.Е. Янковской (Томский государственный архитектурно-строительный университет (ТГАСУ)), а также сотрудникам лаборатории интеллектуальных систем ТГАСУ за всестороннюю помощь и поддержку.

ГЛАВА 1. АНАЛИТИЧЕСКИЙ ОБЗОР ЭВОЛЮЦИОННЫХ АЛГОРИТМОВ, ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ И НЕЙРОЭВОЛЮЦИОНОГО ПОДХОДА

В данной главе представлен аналитический обзор используемых в работе концепций, относящихся к области вычислительного интеллекта и мягких вычислений. Рассматриваются примеры использования нейроэволюционного подхода к решению задач адаптивного управления, адаптивного поведения, эволюционной робототехники и компьютерного творчества. С точки зрения задачи одновременной настройки весов и структуры ИНС формулируются общие требования к свойствам и особенностям ЭА.

1.1. Обзор и анализ эволюционных алгоритмов

Эволюционные алгоритмы (*evolutionary algorithms*) (ЭА) [17, 18, 62, 70], используют единую методологию на основе эволюционных принципов и генетических свойств наследственности для решения разнообразных задач моделирования и оптимизации. Сочетание «*наследственность + изменчивость + естественный отбор*» представляет пример «природной» реализации известного метода проб и ошибок, а потому претендует на универсальность применения наряду с последовательным перебором и случайным поиском. Общая схема эволюционного поиска, описывающая большинство существующих моделей ЭА, и соответствующие каждому этапу эволюционные принципы представлены на рис. 1.1. Также на рис. 1.1 для сравнения показана общая схема генетического алгоритма (ГА) [131], одного из самых распространенных видов ЭА. Сама идея использования эволюционных принципов для машинного обучения присутствует также и в известном труде А. Тьюринга, посвященном проблемам создания «мыслящих» машин [217].

Решение оптимизационных задач с использованием ЭА возможно благодаря представлению (*кодированию*) вероятного решения задачи в виде вектора параметров, называемого *хромосомой* или *особью* (пример на рис. 1.2), причем каждый настраиваемый параметр кодируется в отдельном *гене*. Со-

став хромосомы изменяется с течением времени в результате преобразований с использованием *генетических операторов*. Целесообразность преобразований определяется с помощью *функции приспособленности*, которая зависит от поставленной задачи и соответствующей этой задаче целевой функции и вычисляет *приспособленность* каждой хромосомы в зависимости от качества соответствующего ей декодированного решения. Множество особей, сгенерированных на одной этапе, именуют популяцией. Подмножество особей, используемое для генерации новых особей, называют *родительской подпопуляцией*, а множество сгенерированных новых особей именуют, соответственно, *популяцией потомков*. Использование одновременно множества решений, популяции, делает возможным параллельный поиск и часто позволяет компенсировать возможную неопределенность при вычислении приспособленностей [57], а также исправить, «починить» (*repair*) ненужные изменения хромосом [68].

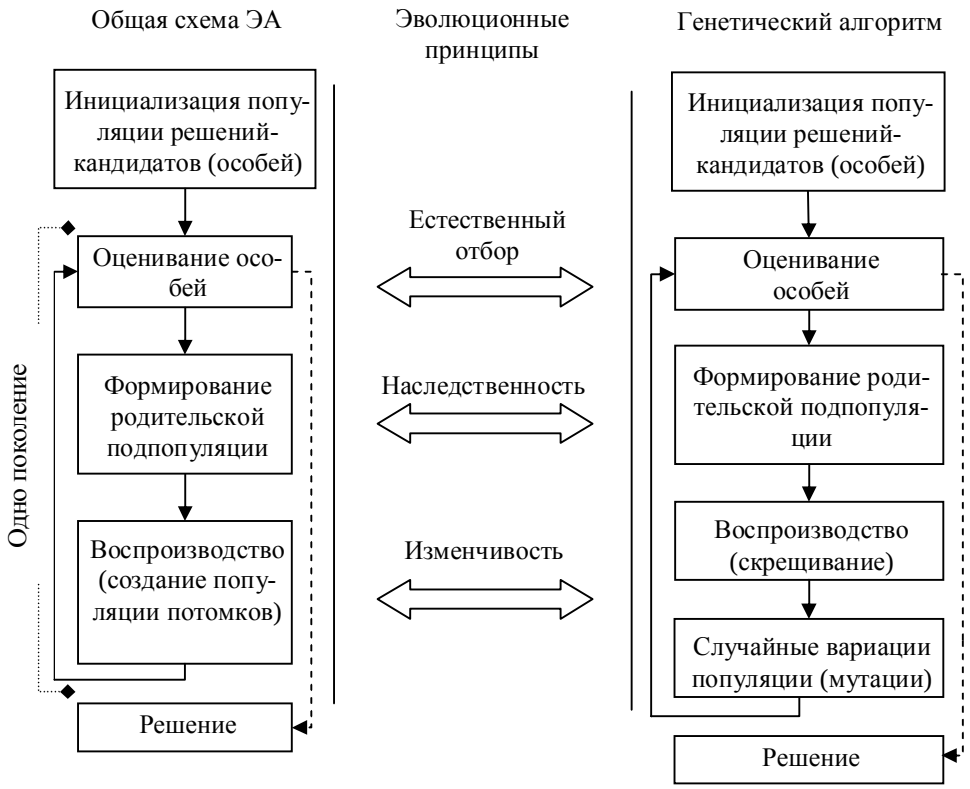


Рис. 1.1. Общая схема эволюционного поиска, пример схемы генетического алгоритма и соответствующие эволюционные принципы

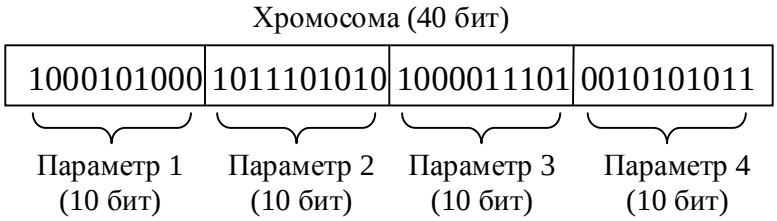


Рис. 1.2. Пример генетического кодирования, целочисленный вариант. Оптимизируемые параметры задачи кодируются в виде бинарной последовательности – хромосомы.

Одним из преимуществ использования ЭА при решении задач оптимизации является их нетребовательность к «знанию» предметной области. Эффективность и качество работы ЭА во многом определяется следующими составляющими и их параметрами:

1. *Генетическое кодирование.* Определяет исследуемое пространство поиска и отношение между *генотипом* (хромосомами) и *фенотипом* (соответствующими решениями) («один-к-одному», «один-ко-многим», «многие-к-одному»).

2. *Генетические операторы*. Используются для рекомбинации и вариации представленных в популяции хромосом и соответствующих им решений. Сюда же относятся операторы селекции и формирования нового поколения.

3. *Функция приспособленности*. Необходима для определения качества получаемых решений и определяет характеристики ландшафта пространства поиска (количество локальных экстремумов, наличие плато, ограничения).

4. *Общая схема эволюционного поиска*. Определяет последовательность выполнения различных операций эволюционного поиска.

Настройка параметров ЭА существенно влияет на его эффективность и качество получаемых решений. Среди общих для различных ЭА численных параметров выделим следующие:

1. Длительность эволюции (число поколений).

2. Размер популяции (число хромосом в одном поколении).

3. Параметры генетических операторов (вероятность применения, количество точек разрыва кроссинговера и др.).

Различные параметры влияют на разные аспекты эволюционного поиска, среди которых выделим два наиболее общих [82]:

1. *Исследование* пространства поиска (*exploration*).
2. *Использование* найденных «хороших» решений (*exploitation*).

Первый аспект отвечает за способности ЭА к эффективному поиску решения и характеризует способности алгоритма избегать локальных экстремумов. Второй аспект важен для постепенного улучшения имеющихся результатов от поколения к поколению на основе уже найденных «промежуточных» решений.

Пренебрежение исследовательскими способностями приводит к существенному увеличению времени работы ЭА и ухудшению результатов из-за «застревания» алгоритма в локальных экстремумах. В итоге становится возможной *преждевременная сходимость* ЭА (также говорят о *вырождении популяции*), когда решение еще не найдено, но в популяции практически все особи становятся одинаковыми и долгое время (порядка нескольких десятков и сотен поколений) не наблюдается улучшения лучшей приспособленности. Игнорирование найденных решений может привести к тому, что работа ЭА будет напоминать случайный поиск, что также отрицательно сказывается на эффективности поиска и качестве получаемых решений.

Основная цель в настройке параметров ЭА и, одновременно, необходимое условие для стабильного получения хороших результатов работы алгоритма – это достижение баланса между исследованием пространства поиска и использованием найденных решений [18, 82]. Взаимосвязь между параметрами генетического алгоритма, а также их влияние на эволюционный процесс носит сложный характер [17, 18, 39, 62]. На рис. 1.3 схематично изображено влияние изменения некоторых параметров ГА на характеристики эволюционного поиска [216].

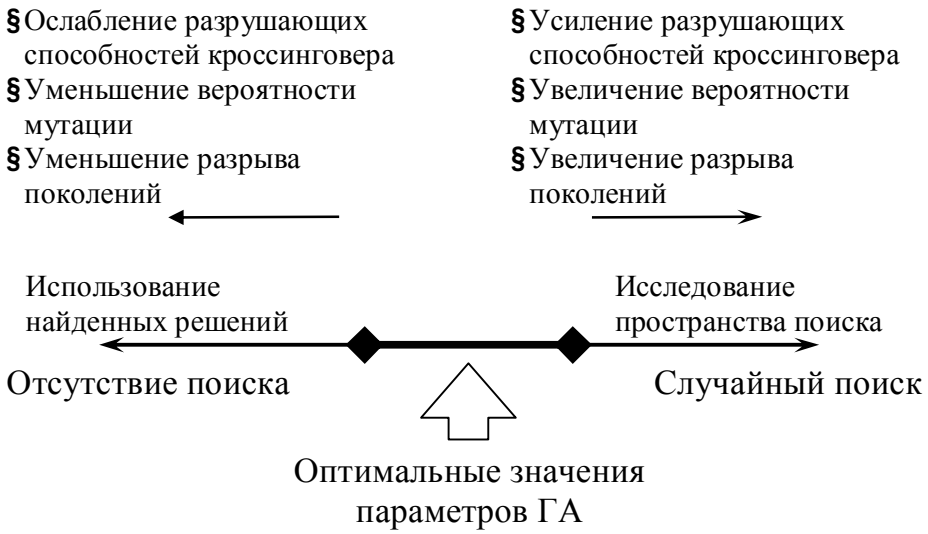


Рис. 1.3. Влияние некоторых параметров ГА[1] на характеристики эволюционного поиска

1.1.1 Классификация методов адаптации в ЭА

Одной из отличительных особенностей ЭА являются их адаптивные способности, что дает возможность реализовать подстройку параметров ЭА в процессе его работы для повышения эффективности ЭА и качества результатов. Целесообразно выделить следующие варианты настройки значений параметров ЭА в процессе его работы [129]:

1. Детерминированная настройка параметров.
2. Адаптивная настройка параметров.
3. Адаптивная самонастройка параметров.

В случае детерминированной настройки параметров ЭА [33, 80, 97, 126, 140, 164] задается процедура, управляющая изменением значений параметров. При этом состояние процесса эволюционного поиска не учитывается, а принимается во внимание только количество поколений, прошедших с момента запуска ЭА. Например, в [126] предложена следующая формула для подстройки вероятности P_m битовой мутации в зависимости от поколения t:

[1] *Разрушающие способности* (*disruption abilities*) кроссинговера определяют вероятность разрушения *шаблона*(*schema*) в зависимости от его длины и порядка [200] (см. также сноску на стр. 48). *Разрыв поколений* (*generation gap*) – параметр, определяющий долю особей в популяции, вытесняемых потомством, и вычисляемый как $G = m/N$, где m – количество потомков, N – размер популяции [80].

$$P_m(t) = \sqrt{\frac{a}{b}} \frac{\exp(-gt/2)}{N\sqrt{L}},$$

где N – размер популяции, L – длина хромосомы в битах, a, b и g – константы.

При использовании варианта с адаптивной настройкой параметров ЭА [18, 41, 43, 61, 63, 90, 118, 136, 137, 156, 190, 195, 197, 199, 203, 218, 225, 231] также определяется процедура изменения значений параметров. Но, в отличие от предыдущего варианта, параметры изменяются в зависимости от характеристик эволюционного поиска. В данном варианте становится доступен уровень адаптации, учитывающий приспособленность, генотипические и фенотипические характеристики каждой отдельной особи. Например, уменьшение вероятности мутации для более приспособленных особей, и увеличение – для менее приспособленных. Из ранних работ необходимо отметить известное эмпирическое правило «1/5» Рехенберга [185], где предполагается, что доля успешных мутаций составляет 1/5 часть от общего числа мутаций, и вероятность мутации адаптивно настраивается, чтобы соответствовать этому правилу. В работе [61] предлагается алгоритм подстройки размера популяции, основанный на идее из [56], где вводится дополнительный параметр, время жизни $\Lambda(i)$ хромосомы, вычисляемый в зависимости от ее приспособленности F_i [61] (для задачи минимизации целевой функции):

$$\Lambda(i) = \begin{cases} \Lambda_{\min} + h \dfrac{F_{worst}(t) - F_i}{F_{worst}(t) - F_{avg}(t)}, & F_i \geq F_{avg}(t) \\ \dfrac{1}{2}(\Lambda_{\max} + \Lambda_{\min}) + h \dfrac{F_{avg}(t) - F_i}{F_{avg}(t) - F_{best}(t)}, & F_i < F_{avg}(t) \end{cases}$$

$$h = \frac{1}{2}(\Lambda_{\max} - \Lambda_{\min}),$$

где $\Lambda_{\min}$ и $\Lambda_{\max}$ – минимальное и максимальное возможное время жизни; $F_{worst}(t)$, $F_{best}(t)$ и $F_{avg}(t)$ – соответственно худшая, лучшая и средняя приспособленность в поколении t. В каждом поколении время жизни всех осо-

бей, кроме лучшей, уменьшается на 1, и при $\Lambda(i) = 0$, хромосома удаляется из популяции.

При адаптивной самонастройке [25, 58-60, 69, 86, 98, 99, 114, 119, 120, 128, 143, 174, 185, 186, 189, 192, 196] значения параметров кодируются в хромосоме особи и, как и оптимизируемые параметры решения, претерпевают эволюционные изменения. Значения закодированных параметров ЭА не влияют на величины приспособленностей особей напрямую, но «хорошие» значения параметров часто способствуют улучшению особей и с течением времени распространяются в популяции. Часто подобным образом кодируют вероятности операторов скрещивания и мутации, а также их параметры. В [189] самоадаптация использовалась для определения позиции и числа точек разрыва кроссинговера. Для этого к L-разрядным хромосомам добавлялись L дополнительных бит, в которых кодировалось наличие («1») или отсутствие («0») точки разрыва в соответствующей позиции хромосомы. При скрещивании точки разрыва кроссинговера определялись по единичным разрядам в дополнительных разрядах рекомбинируемых хромосом.

Необходимо отметить, что возможно комбинирование различных вариантов адаптивной настройки параметров ЭА. К примеру, возможна адаптация размера популяции вместе с самоадаптацией оператора мутации [130, 61].

Адаптация в ЭА возможна не только на уровне параметров, но и на уровне кодирования и использования генетических операторов скрещивания (рекомбинации) и мутации. Неоднократно отмечалось, что использование проблемно-зависимых способов кодирования и операторов позволяет повысить эффективность работы ЭА [18, 65, 149]. Причина заключается в том, что традиционно генетические операторы применяются на уровне хромосом (на уровне генотипов), но не на уровне декодированных решений (фенотипический уровень), что может привести к непредсказуемым изменениям соответствующих решений, однако позволяет реализовать унифицированную схему поиска. Выходом из данной ситуации может быть использование специали-

зированных генетических операторов, работающих на уровне фенотипов, адаптирующихся к характеристикам декодированных решений и учитывающих особенности поставленной задачи, хотя такая организация эволюционного поиска менее универсальна. Отличия между двумя схемами поиска представлены на рис. 1.4.

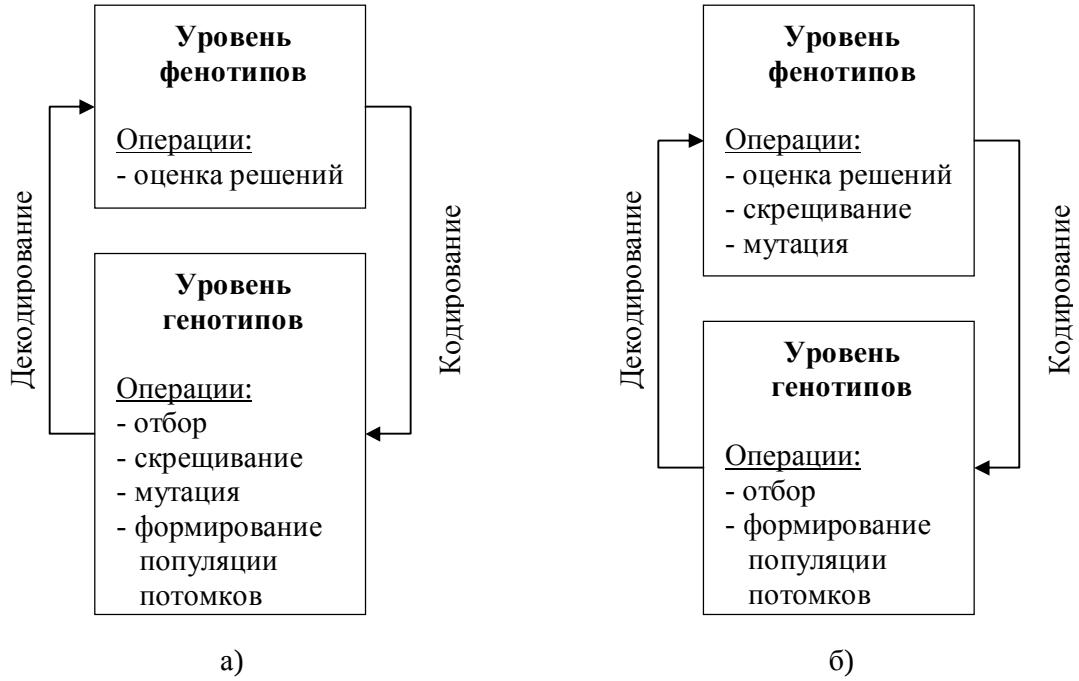


Рис. 1.4. Различие между схемами эволюционного поиска при реализации генетических операторов, работающих на уровне а) генотипов; б) фенотипов

Использование генетических операторов скрещивания и мутации на уровне фенотипов показало свою эффективность в генетическом программировании [149], в задачах настройки и обучения ИНС [117, 204], при решении многих задач на графах [18] и др.

1.2. Искусственные нейронные сети

Искусственные нейронные сети (ИНС) появились в результате применения математического аппарата к исследованию функционирования нервной системы [163]. Полученные при этом результаты успешно применяются при решении проблем распознавания образов, моделирования, выполнения прогнозов, оптимизации и управления [8, 16, 23, 32, 147].

Одним из основных понятий ИНС является *формальный нейрон*, представленный на рис. 1.5 [10], где $x_0, x_1, ..., x_n$ – компоненты вектора входных сигналов, $w_0, w_1, ..., w_n$ – компоненты вектора соответствующих весов входных сигналов, называемые *весами связей*, а y – выходной сигнал нейрона.

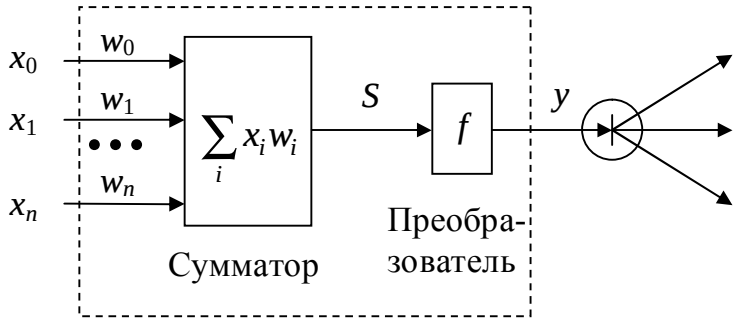


Рис. 1.5. Формальный нейрон (по [10]). Подробности см. в тексте

Функция $f(S)$ называется *функцией активации* или *передаточной функцией нейрона*. Исходя из данного описания, математическая модель формального нейрона может быть представлена следующим образом:

$$y = f(S),$$

$$S = \sum_{i=0}^{n} w_i x_i .$$

Выходной сигнал одного нейрона может служить в качестве входного для другого. Благодаря этому возможны различные схемы организации многих нейронов в одну вычислительную структуру, которая, при соответствующей настройке весов связей всех входящих в эту структуру нейронов, может быть использована для реализации отображения $\mathbf{X} \to \mathbf{Y}$, где $\mathbf{X}$ – пространство входных сигналов ИНС, а $\mathbf{Y}$ – пространство выходных сигналов.

Пример широко распространенной многослойной топологии ИНС приведен на рис. 1.6. Особенностью данной топологии является объединение нейронов в слои, в соответствии с получаемыми входными сигналами. Различают входной, выходной и скрытые слои в зависимости от того, какие сигналы получает/формирует тот или иной слой.

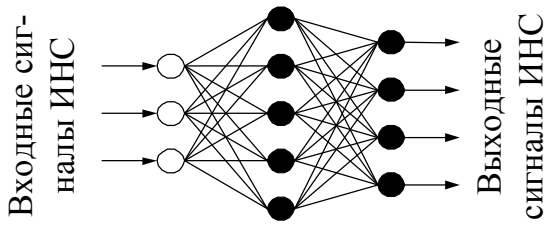


Рис. 1.6. Пример ИНС прямого распространения

Формальным обоснованием использования ИНС является результат цикла работ А.Н. Колмогорова [13, 14] и В.И. Арнольда [1], в результате которых получен положительный ответ на вопрос о возможности представления функции n переменных через суперпозицию функций одного переменного и операций сложения и умножения, путем доказательства следующей теоремы [14] (цитируется по [10]):

Каждая непрерывная функция n *переменных, заданная на единичном кубе* n*-мерного пространства, представима в виде*

$$f(x_1, x_2, ..., x_n) = \sum_{q=1}^{2n+1} h_q \left[\sum_{p=1}^{n} j_q^p(x_p) \right],$$

где функции $h_q(\cdot)$ *непрерывны, а функции* $j_q^p(x_p)$*, кроме того, еще и стандартны, то есть не зависят от выбора функции* f*.*

В частности, данный результат показывает, что для аппроксимации произвольной непрерывной функции n переменных достаточно использовать ИНС с одним скрытым слоем, содержащим $(2n + 1)$ нейронов. Развитием данной теоремы являются результаты, расширяющие теорему Колмогорова [202]; работы, в которых показано, что главным условием универсальных аппроксимирующих способностей ИНС является нелинейность функции активации нейронов [150, 9]; а также исследования аппроксимационных свойств многослойных ИНС [66, 79, 102-107, 124, 134, 145]. В результате были исследованы ИНС с одним скрытым слоем с различными функциями активации

(сигмоида [66, 79], косинусоидальная пороговая функция [104]) и для сигмоидальных сетей вычислены пределы ошибок аппроксимации [66]:

$$R \le \mathrm{O}\left(\frac{C_f^2}{m_1}\right) + \mathrm{O}\left(\frac{m_0 m_1}{N}\log N\right), \qquad (1.1)$$

где m_0 и m_1 – соответственно число нейронов во входном и скрытом слое, N – количество примеров в обучающей выборке, а C_f – первый абсолютный момент в разложении аппроксимируемой функции f в ряд Фурье. Важным следствием неравенства (1.1), является демонстрация известной противоречивости выбора размеров скрытого слоя, а также то, что, игнорируя логарифмический множитель во втором слагаемом в правой части (1.1), можно сделать предположение, что для хорошего качества аппроксимации с ошибкой $e << 1$ размер обучающего множества должен превышать по порядку произведение $m_0 m_1$ [32].

Стоит отметить, что поскольку в теореме Колмогорова выбор функции h_q зависит от аппроксимируемой функции f, то в реальности одного скрытого слоя нейронов может быть недостаточно для аппроксимации с заданной точностью. Более того, в ряде случаев можно считать, что ИНС с двумя скрытыми слоями логически более прозрачны и управляемы, за счет последовательного извлечения в первом скрытом слое локальных признаков входного вектора сигналов, а во втором – глобальных [73, 102]. Также известно, что для решения задачи инверсной кинематики, которая сводится к поиску непрерывной вектор-функции, для которой известна область значений, обратной заданной вектор-функции $\varphi: \Re^M \to \Re^m$, необходимо два скрытых слоя [198]. Также многослойная ИНС, обученная алгоритмом обратного распространения ошибки, способна аппроксимировать недифференцируемые и кусочно-дифференцируемые функции [134].

1.2.1. Проблема формирования обучающего множества данных

Обучение ИНС, с «классической» точки зрения, подразумевает использование обучающей выборки, включающей наборы входных сигналов $\mathbf{X} = \{\mathbf{X}_i : 1 \le i \le N\}$ и соответствующих эталонных значений выходных сигналов $\mathbf{Y} = \{\mathbf{Y}_i : 1 \le i \le N\}$ ИНС. Использование обучающего множества данных дает возможность подстраивать веса связей ИНС с помощью градиентных алгоритмов, на основании информации об отклонении значений действительных выходных сигналов ИНС $\mathbf{F} = \{\mathbf{F}_i : 1 \le i \le N\}$ от требуемых, которое часто представляется в виде ошибки e ИНС, например, для $\mathbf{F}_i = \{f_{ij} : 1 \le j \le K\}$, $\mathbf{Y}_i = \{y_{ij} : 1 \le j \le K\}$:

$$e = \frac{1}{NK} \sum_i \sum_j \left(f_{ij} - y_{ij}\right)^2 .$$

Традиционные подходы к обучению ИНС чаще всего основываются на схеме обратного распространения ошибки [187], обычно использующей алгоритмы градиентного типа, в частности, метод сопряженных градиентов [193], метод Левенберга-Марквардта [161] и ряд других. Однако в ряде случаев формирование обучающей выборки сопряжено со сложностью определения значений компонент вектора $\mathbf{Y}$ из обучающего множества. Данная проблема возникает при решении некорректных задач, а также, если необходимо оценить последовательность выходных сигналов ИНС. К таким задачам относятся задачи, связанные с адаптивным поведением, игровыми стратегиями, обработкой изображений и др.

Одним из возможных решений этой проблемы является обучение ИНС с использованием приблизительной, интегральной оценки, отражающей качественные, внешние особенности ее функционирования, когда оценивается не соответствие выходных сигналов ИНС эталонным, а качество работы ИНС в целом [47, 53]. Примерами таких оценок являются: время поддержания стабильного состояния объекта управления для задачи нейроуправления [3, 40, 110, 135, 204]; процент выигранных игр, либо оценка позиции для задачи по-

иска игровых стратегий [139, 176, 205, 157]; качество изображения для задачи обработки изображений [48, 50, 75, 201, 215] и т.д. Такие оценки более естественны и «интуитивны», однако, как следствие, они не могут быть использованы градиентными алгоритмами обучения для подстройки весов связей. Поэтому «подходящий» обучающий алгоритм должен изменять веса связей ИНС, не имея информации о величине ошибки для каждого выхода этой НС. Одним из подходов, позволяющих использовать подобные «качественные» оценки, является рассматриваемый в следующем разделе нейроэволюционный подход.

1.2.2. Нейросетевая обработка изображений

С самого рождения искусственные нейронные сети тесно связаны с обработкой изображений. Одной из первых задач, стоящей перед ИНС, была задача распознавания образов [28], а одна из нейросетевых моделей, благодаря которой возродился интерес к исследованию ИНС, сеть Хопфилда [133], также часто используется для решения этой задачи. В настоящее время ведутся активные исследования в области нейросетевой обработки изображений [22, 89], включающие следующие основные направления исследований:

1. Распознавание образов. Рассматривается задача определения принадлежности объекта к одному из заранее определенных классов на основе анализа совокупности его признаков. Традиционно для решения этой задачи применяются ИНС прямого распространения, сети Хопфилда [133] и Хемминга [155], неокогнитрон Фукушимы [100]. К рассматриваемому классу задач можно также отнести задачу оптического распознавания символов (OCR, Optical Character Recognition), в которой требуется по изображению текста определить содержание текста, с точки зрения его морфологии и синтаксиса.

2. Предобработка изображений. Включает многочисленные задачи, связанные с повышением контрастности, уменьшением уровня шумов, общее визуальное улучшение (enhancement) и восстановление (restoration) искажен-

ных изображений [89]. Существуют работы, направленные на реализацию нейросетевого детектора краев, например, с использованием биологически правдоподобной структуры ИНС [29].

3. Сжатие изображений. Включает использование ИНС для сжатия (как правило, с потерями) изображений, для уменьшения их размера. При этом используются как традиционные ИНС прямого распространения, так и нейронные сети с радиально-базисными функциями [154], карты Кохонена [147] и ИНС на основе анализа главных компонент [32] (нейросети PCA – Principal Components Analysis).

4. Сегментация изображений. В данном направлении исследуются возможности применения ИНС для выделения на изображении областей с одинаковыми характеристиками в соответствии с выбранным критерием. Данная задача также может рассматриваться как задача классификации, когда для каждого пикселя определяется его класс (например, в зависимости от текстурных характеристик), а выделяемые области содержат пиксели одного класса. Также к этому направлению относится задача кластеризации изображения с целью определения количества возможных классов пикселей [89].

К настоящему времени предобработка изображений является одним из «проблемных» направлений ввиду отсутствия [89] успешных примеров решения задачи автоматического попиксельного улучшения визуального качества изображений с использованием ИНС, когда ИНС обрабатывает яркость пикселей на основе информации об исходных значениях яркости, а также характеристик окрестности пикселей. Одной из таких задач является задача повышения качества изображения посредством изменения яркости пикселей, поскольку замечено, что яркостная компонента играет одну из ключевых ролей в восприятии изображения [11, 19], напрямую влияя на его контрастность, уровень информативности и визуальное качество. Данная задача осложняется отсутствием общепринятого критерия визуального качества изображений, что осложняет обучение ИНС. При этом традиционный подход к

обучению нейронных сетей здесь не применим в силу отсутствия информации о том, какими должны быть значения яркости обработанных пикселей. Данная проблема возникает из-за того, что яркость пикселей (и, следовательно, результат работы ИНС) может быть оценена только с учетом характеристик соседних пикселей. Таким образом, чтобы оценить работу ИНС необходимо предварительно обработать этой ИНС участок изображения. Однако полученная в итоге оценка будет отражать оценку обобщенного результата работы, не связанного с конкретными выходными сигналами ИНС и поэтому не может быть использована для подстройки весов связей.

В общем случае проблема попиксельной обработки изображений может быть представлена в виде проблемы поиска (идентификации) следующего преобразования (или его параметров):

$$I^* = T(I, \mathbf{\Omega}), \tag{1.2}$$

где I и I^* – интенсивность пикселей до и после обработки соответственно; $\mathbf{\Omega}$ – вектор параметров, определяющих локальные и/или глобальные характеристики каждого пикселя обрабатываемого изображения.

В простейшем варианте обработки изображений локальные характеристики пикселей не рассматриваются. Например, известна гамма-коррекция изображения [115], учитывающая нелинейность восприятия яркости человеком, при которой яркость изменяется по степенному закону в зависимости от показателя степени g :

$$I^* = I^g .$$

Более сложный подход к определению вида преобразования (1.2) представлен в работе Мунтеану и Роса [169], где преобразование яркости пикселя осуществляется с использованием функции вида:

$$I^*(x, y) = k \frac{M}{s(x, y) + b}\big(I(x, y) - c \cdot m(x, y)\big) + m(x, y)^a , \tag{1.3}$$

где $I(x, y)$ и $I^*(x, y)$– соответственно значения яркости пикселя (x, y) до преобразования и после; M – средняя яркость всего изображения; $m(x, y)$ и

$s\,(x, y)$ – соответственно среднее и дисперсия яркости в окрестности пикселя (x, y); a, b, c и k – численные параметры, настраиваемые с помощью генетического алгоритма. Недостатком описанного способа является возможная неточность в определении вида преобразования (1.3). Также значения параметров a, b, c и k необходимо подбирать для каждого изображения отдельно, что отрицательно сказывается на скорости обработки.

Ввиду сложности определения вида преобразования (1.2) представляет интерес аппроксимация этого преобразования. Однако решение данной задачи осложняется в силу отмеченной выше сложности оценки функционирования ИНС и последующего применения этой оценки к подстройке весов связей. Отметим, что возможности косвенного оценивания ИНС достаточно для осуществления подстройки параметров ИНС с использованием эволюционного подхода.

1.3. Анализ нейроэволюционного подхода

Комбинация ИНС и ЭА дает возможность совместить гибкость настройки ИНС и адаптивность ЭА, что позволяет реализовать во многом унифицированный подход к решению широкого спектра задач классификации, аппроксимации и моделирования [3, 30, 47, 53, 168, 191, 228]. Также необходимо отметить, что Фредериком Груо [117] была показана возможность использования для эволюционирующих нейронных сетей клеточного кодирования[1], которое предназначено для кодирования последовательности правил, определяющих параметры ИНС, и по выразительности не уступает универсальной машине Тьюринга и клеточным автоматам.

Предлагались различные варианты названий для данного направления исследований. В настоящей работе будет использоваться термин «нейроэволюция» (НЭ) [167], а соответствующий подход будем называть нейроэволюционным. Отметим, что предлагаемый вариант не является устоявшимся и

[1] См. также сноску на стр. 5.

окончательным.

Первые работы, посвященные применению ЭА для обучения и настройки ИНС, появились более 25 лет назад. Исследования в этой области, как правило, рассматривают решение следующих задач:

- поиск значений весов связей ИНС при фиксированной структуре;
- настройка структуры ИНС без поиска весов связей;
- настройка параметров алгоритма обучения;
- настройка параметров активационных функций нейронов;
- фильтрация обучающих данных;
- различные комбинации вышеперечисленных задач.

Рассмотрим некоторые из перечисленных выше задач и отметим их особенности.

1.3.1. Эволюционная настройка весов связей ИНС

Одной из типичных задач, решаемых в рамках НЭ подхода, является задача поиска весов связей ИНС при ее фиксированной структуре [2, 3, 15, 21, 67, 110-113, 127, 135, 148, 159, 166, 176, 188, 209, 210, 219, 223, 226]. При этом, как правило, рассматривается задача минимизации целевой функции (обычно функция ошибки выхода ИНС), а в качестве оптимизируемых параметров используются веса связей, значения которых подбираются с помощью ЭА. Одним из обоснований эволюционного обучения ИНС является «застревание» градиентных алгоритмов в локальных экстремумах в процессе обучения ИНС.

Существенными преимуществами использования эволюционной настройки весов связей ИНС являются:

1. Независимость от структуры ИНС и характеристик функций активации нейронов.
2. Отсутствие необходимости в обучающем множестве данных.

Первое преимущество дает возможность использовать единый подход к обучению ИНС с различной структурой. Второе преимущество позволяет

осуществлять обучение ИНС без информации об эталонных значениях выходных сигналов, а на основе оценки функционирования ИНС «в целом». Для градиентных алгоритмов оптимизации обучающее множество необходимо, чтобы подстраивать веса связей ИНС на основе расхождения значений реального и требуемого выходного сигнала ИНС.

При использовании ЭА для настройки весов связей ИНС достаточно использовать оценку, отражающую качество нейросетевого решения в целом. Здесь важна *адекватность* оценки, то есть если согласно используемой оценке качество первой ИНС лучше качества второй ИНС, то из этого должно следовать, что первая ИНС лучше второй. Кроме адекватности оценки также важно ее *существование* для любого решения, которое может быть закодировано в хромосоме. При этом такая оценка может вычисляться независимо от расхождения выходного сигнала ИНС с эталонным значением. Таким образом, отсутствие обучающей выборки не является серьезным препятствием для эволюционного обучения ИНС, если существует альтернативный способ оценки ее функционирования. При этом появляется возможность использовать приблизительные оценки функционирования ИНС, и оцениваться может не каждый выходной сигнал НС, а последовательность сигналов. Это позволяет расширить круг практических приложений нейроинформатики, а также упростить процесс решения многих задач, связанных с адаптивным нейроуправлением, адаптивным поведением и моделированием.

К недостаткам использования ЭА для обучения ИНС следует отнести следующие:

1. Трудность «тонкой» настройки весов связей на поздних этапах эволюционного поиска.

2. Большие, по сравнению с градиентными алгоритмами, требования к объему оперативной памяти из-за использования популяции ИНС.

3. Проблема конкурирующих решений.

Первая проблема во многом обусловлена использованием целочисленно-

го кодирования весов связей ИНС, что может привести к значительным «скачкам» в пространстве поиска в результате применения операторов скрещивания и мутации. Данное свойство полезно на начальном этапе работы НЭ алгоритма, когда необходимо быстро исследовать пространство поиска, но затрудняет работу алгоритма на более поздних этапах. При использовании вещественного кодирования эта проблема не так актуальна, но не исчезает полностью из-за необходимости адаптивной настройки параметров операторов ЭА, так как характеристики эволюционного поиска изменяются с течением времени.

Проблема конкурирующих решений (*competing conventions problem*) [207, 222] известная также как *проблема перестановки* (*permutation problem*) [182, 188] связана с кодированием весов связей ИНС. Ее суть заключается в том, что генетическое представление допускает существование нескольких вариантов хромосом, кодирующих одну и ту же ИНС (см. пример на рис. 1.7).

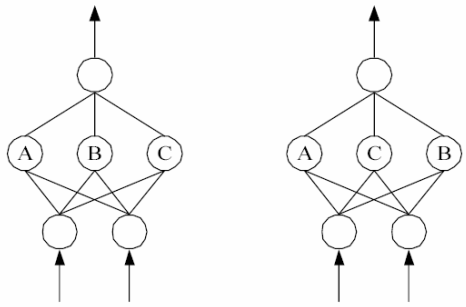


Рис. 1.7. Проблема перестановки. Две одинаковые ИНС могут быть закодированы различным образом «перестановкой» скрытых нейронов. Считается, что «скрещивание» таких сетей практически бесполезно [204]

Предложено несколько вариантов решения проблемы перестановки, среди которых необходимо отметить следующие:

1. Упорядочивание списка весов связей кодируемых ИНС (выравнивание хромосом) по значениям весов [209], либо с использованием специальных меток [204]. Считается, что такие меры позволят уменьшить вероятность появления особей, представляющих одинаковые ИНС. Также возможно раннее удаление особей-дубликатов на этапе формирования популяции следую-

щего поколения.

2. Модификация компонент ЭА, включающая разработку адаптивных операторов скрещивания [182, 188], уменьшение размера популяции, а также повышение вероятности мутации.

3. Уменьшение роли оператора кроссинговера, либо полный отказ от его использования [55, 222, 229]. При этом предполагается, что: (а) вероятность возникновения одинаковых ИНС в результате мутаций различных особей несущественна; (б) в случае, если такие особи все-таки появились, мутации приведут к меньшим «разрушениям», чем скрещивание.

Отметим, что вопрос об актуальности проблемы конкурирующих решений остается открытым. Из-за того, что, в общем случае, вероятность появления в одной популяции двух одинаковых ИНС мала, проблема конкурирующих решений может игнорироваться [160, 167].

1.3.2. Эволюционная настройка структуры ИНС

Еще одной типичной задачей НЭ подхода является задача эволюционной настройки структуры ИНС [6, 7, 34, 95, 144]. В хромосоме кодируется топология ИНС, а настройка весов осуществляется, например, с использованием градиентных алгоритмов. Каждая особь, представляющая сеть той или иной структуры, оценивается в зависимости от результатов обучения с использованием градиентных либо иных алгоритмов: чем лучше результат, тем более приспособлена особь. Поскольку выбор топологии ИНС является, как правило, сложной задачей, решаемой методом проб и ошибок, то эволюционный поиск нейросетевой структуры способен облегчить и в определенной степени автоматизировать процесс решения задачи настройки и обучения ИНС. Из ранних работ в этом направлении отметим исследования А.Б. Глаза и Л.А. Растригина по эволюционной оптимизации структуры многослойного перцептрона [6, 7].

Перечислим преимущества и недостатки эволюционной настройки структуры НС.

Преимущества:

1. Возможность автоматического поиска топологии ИНС и получения более точной нейросетевой модели за счет рассмотрения «нестандартных», нерегулярных топологий.

2. Независимость от характеристик функций активации нейронов.

Для упрощения задачи и повышения качества результатов, в процессе поиска топологии ИНС, возможно использование дополнительных регулирующих ограничений, помогающих избежать чрезмерного «разрастания» сети, которое выражается в быстром увеличении количества скрытых нейронов и связей между ними.

Недостатки:

1. Сложность оценки структуры ИНС без информации о значениях весов связей.

2. Сложность организации поиска топологии ИНС.

Первый недостаток представляет основную проблему эволюционной настройки структуры ИНС. Он, в основном, обусловлен чувствительностью традиционных градиентных алгоритмов обучения к начальным условиям и значениям параметров алгоритма обучения. Таким образом, хромосоме, представляющей ИНС с некоторой структурой, могут соответствовать совершенно разные нейросетевые модели с точки зрения их качества и свойств (в таких случаях говорят, что одному генотипу соответствует множество фенотипов). Для уменьшения влияния случайных факторов на оценку структуры ИНС проводится несколько независимых операций обучения [34, 95], усредненный результат которых и используется в качестве оценки топологии ИНС. Также, несмотря на то, что существуют эвристические правила, согласно которым ИНС с более простой структурой обладают лучшими способностями к обобщению, этот вопрос все еще нельзя считать до конца решенным, так как многое зависит от результатов обучения [35, 72].

Сложность объективной оценки топологии ИНС порождает также слож-

ность оценки отдельных структурных частей сети и возможных структурных модификаций. Данная проблема характерна также и для алгоритмов одновременной настройки и весов связей ИНС (см. п. 1.3.3), поэтому многие дальнейшие замечания относятся в равной степени и к эволюционному поиску структуры ИНС, и к одновременной настройке структуры и весов связей. Для изменения структуры ИНС часто применяются следующие операции:

- добавление/удаление нейронов;
- добавление/удаление связей.

Встречаются также модификации этих операций, включающие, например, «расщепление» существующего нейрона, «перенаправление» связи [168, 229] и др.

Итак, выбор типа структурного изменения ИНС в общем случае неоднозначен из-за того, что оценка необходимости модификации топологии ИНС и варианта этой модификации не всегда возможна. Известны следующие варианты решения проблемы по выбору операции модификации структуры ИНС:

- определение эффективности каждой операции преобразования топологии ИНС на основании оценки качества получаемых НС решений [136, 137];
- использование дополнительной информации об ИНС для ее модификации (например, вычисление «значимости» связей [229], эвристическая оценка структуры ИНС [35, 55]).

Заметим, что существуют неэволюционные подходы к настройке топологии ИНС либо во время ее обучения (алгоритм каскадной корреляции Фальмана [92] и его модификации [170] и др.), либо по завершении процесса обучения (алгоритмы Optimal Brain Damage (OBD) [153], Optimal Brain Surgeon (OBS) [123] и др.). Несмотря на то, что в [183] показана эффективность использования эволюционного подхода для настройки топологии ИНС по сравнению с алгоритмами OBD и OBS, однако детального исследования на эту тему, насколько известно автору, не проводилось.

1.3.3. Одновременная эволюционная настройка весов связей и структуры ИНС

Использование эволюционного подхода позволяет одновременно настраивать веса связей и структуру ИНС [24, 36-38, 40, 41, 44-48, 52, 54, 55, 74, 87, 94, 101, 117, 136, 137, 139, 152, 157, 167, 168, 175, 178, 204-206, 214, 229]. При этом в хромосоме кодируется информация о весах и связях ИНС. Возможно использование как бинарного, так и вещественного кодирования для записи весов связей, а структура сети может быть представлена с использованием различных способов кодирования [36, 228].

В силу того, что в случае одновременной настройки весов связей и структуры ИНС фактически комбинируются две различные задачи, пространство поиска многократно увеличивается и часто включает подпространства различной размерности (для ИНС с разным количеством связей). Для упрощения задачи поиска в таком сложном пространстве часто в явном или неявном виде вводятся ограничения на топологию ИНС и/или ее изменения:

- ограничение количества скрытых нейронов [55, 136, 137, 167, 168];
- ограничение количества связей [94, 167, 168];
- ограничение изменения топологии ИНС (рассматриваются только растущие ИНС [204], либо только многослойные ИНС [24] и др.);

Следствием первых двух ограничений часто является фиксация длины хромосомы, что существенно уменьшает пространство возможных решений и не способствует эффективному поиску структуры ИНС, но существенно упрощает задачу и позволяет использовать «стандартные» операторы скрещивания и мутации. Для повышения гибкости поиска НЭ алгоритма используют генетическое кодирование с переменной длиной хромосомы, которое позволяет добавлять и (в некоторых реализациях) удалять гены в процессе работы алгоритма [36-38, 40, 41, 44-48, 52, 136, 229]. Это ведет к изменению общего порядка следования генов и требует реализации специализированных генетических операторов, учитывающих особенности используемого кодиро-

вания.

Одновременное решение двух отдельных задач: настройки весов связей и структуры ИНС – позволяет в некоторой степени скомпенсировать недостатки, присущие каждой из них в отдельности и объединить их преимущества. С другой стороны, «платой» за это является огромное пространство поиска, а также объединение ряда недостатков, вызванных использованием эволюционного подхода. Суммируя, перечислим преимущества и недостатки.

Преимущества:

1. Независимость от структуры ИНС и характеристик функций активации нейронов.

2. Отсутствие необходимости в обучающей выборке.

3. Возможность автоматического поиска топологии ИНС и получения более точной нейросетевой модели.

Одним из основных преимуществ одновременной эволюционной настройки весов связей и структуры ИНС является возможность автоматизированного поиска ИНС, на основе только критерия оценки ИНС для осуществления эволюционного поиска. При этом, как и для эволюционного обучения ИНС (см. п. 1.3.1), наличие обучающей выборки не является обязательным, а НЭ алгоритм может применяться для поиска ИНС с любыми активационными функциями нейронов.

По сравнению с раздельным эволюционным поиском структуры ИНС и настройкой весов связей, одновременное решение этих задач позволяет избежать некоторых недостатков. Так появление в популяции особей, которым соответствуют ИНС с различными топологиями, уменьшает значимость проблемы конкурирующих решений, а наличие информации о весах связей позволяет обойти проблему субъективной оценки структуры ИНС, в силу того, что оценивается не структура нейросети, а вся ИНС «целиком».

Недостатки:

1. Сложность «тонкой» настройки весов связей на поздних этапах эво-

люционного поиска.

2. Большие, по сравнению с градиентными алгоритмами, требования к объему оперативной памяти из-за использования популяции ИНС.

3. Сложность организации поиска топологии ИНС.

1.3.4. Приложения НЭ подхода

Помимо решения «стандартных» нейросетевых задач классификации и аппроксимации с использованием обучающего множества данных НЭ алгоритмы представляют большой интерес для решения трудноформализуемых задач, где присутствует сложность формирования обучающего множества и/или затруднена оценка качества работы ИНС. Рассмотрим коротко примеры таких задач и некоторые особенности использования НЭ подхода:

Адаптивное управление [3, 36, 112, 137, 167, 204, 212, 214]. При использовании НЭ алгоритмов для решения задач адаптивного управления можно перейти к безмодельному обучению, когда не нужно знать прямую либо обратную математическую модель объекта управления[1] (ОУ), а ИНС оценивается в зависимости от используемого ОУ.

Например, в [3] для обучения ИНС управлению инерционным колебательным звеном второго порядка с передаточной функцией вида $W = \frac{k}{T^2 s^2 + 2TVs + 1}$ использовалась следующая оценка:

$$F = \frac{1}{3}\sum_{i=1}^{3}\left(\frac{1}{t_{\max}} \int_0^{t_{\max}} \left(x(t) - u_r(t)\right)^2 dt \right),$$

где $x(t)$ и $u_r(t)$ – соответственно выход ОУ и сигнал задания в момент времени t; $t_{\max}$ – время интегрирования переходного сигнала в системе.

В [35, 167, 204] при решении задачи балансирования на тележке 1 либо 2 шестов разной длины использовалась следующая оценка ИНС:

$$F = t,$$

[1] Поиск такой модели сам по себе часто является нетривиальной задачей.

где t – количество итераций, в течение которых удавалось удержать маятники от падения. В случае использования 2 маятников без информации о скорости их движения, когда появляется дополнительное требование подавления колебательных движений тележки, в [112, 137, 204] оценка ИНС определяется уже следующим образом (обозначения из [204]):

$$F = 0{,}1 f_1 + 0{,}9 f_2,$$

$$f_1 = t/1000,$$

$$f_2 = \begin{cases} 0, & t < 100, \\ \dfrac{0{,}75}{\sum_{i=t-100}^{t} \left(|x^i| + |\dot{x}^i| + |q_1^i| + |\dot{q}_1^i| \right)}, & t \geq 100, \end{cases}$$

где x^i и $\dot{x}^i$ – положение и скорость тележки, q_1^i и $\dot{q}_1^i$ – положение и угловая скорость первого маятника.

Адаптивное поведение и многоагентные системы [2, 21, 26, 55, 206]. Здесь большой интерес представляет возможность реализации интеллектуальных агентов [31] с использованием ИНС, при этом возможно наличие коллективного поведения [206, 230]. В качестве примера рассмотрим проект NERO (Neuro-Evolution of Robotic Operatives, http://dmc.ic2.org/nero_public) [206]. Ключевой технологией проекта является модифицированный алгоритм NEAT для одновременной настройки структуры и весов связей ИНС, разработанный Кеннетом Стэнли (Kenneth Stanley). Пользователь сначала тренирует на специальном тренировочном полигоне команду НС агентов, а затем его команда «соревнуется» с агентами компьютера или другого пользователя на своеобразном «поле боя». Поведение агентов определяется соответствующими им ИНС, которые настраиваются в процессе обучения на полигоне. Действия агентов зависят от создающейся на тренировочном полигоне ситуации и определяются приоритетами поведения (расстояние до противника, подвижность агентов, активность стрельбы и др.), заданными пользователем. Более приспособленные с точки зрения выбранных приоритетов агенты используются для генерации агентов-потомков, которые замещают худших су-

ществующих агентов.

Для выбора действия (направление движения и стрельба) агентам доступна следующая информация:

- расположение агентов-противников, находящихся в поле зрения агента;
- находится ли агент-противник на линии огня;
- расположение препятствий (стены, различные объекты);
- направление стрельбы ближайшего агента-противника.

Использование для реализации проекта NERO НЭ алгоритма позволяет в реальном времени осуществлять поиск разнообразных стратегий поведения агентов. В процессе обучения агенты учатся решать такие задачи, как поиск маршрута движения в присутствии препятствий, преследование агентов-противников, использование стен для прикрытия от огня и др. Отметим, что во время обучения агентов задачи ставятся в общем виде (дойти до определенной точки, атаковать цель и т.д.) и возникновение сложных стратегий поведения (например, укрытие за стенами) зависит от ситуации, которая создается на тренировочном полигоне.

Эволюционная робототехника (Evolutionary robotics) [76, 172]. Представляет сравнительно молодое направление исследований, направленное на создание и исследование автономных роботов, которые рассматриваются как искусственные организмы, поведение которых развивается без человеческого вмешательства в результате взаимодействия с окружающей средой. Эволюционная робототехника во многом опирается на науки о природе, биологию и этологию, и использует такие концепции и подходы, как нейронные сети, генетические алгоритмы, динамические системы и бионика. Основное отличия исследований в области эволюционной робототехники от исследований адаптивного поведения интеллектуальных агентов заключается в использовании реальных роботов, а не их программных моделей. Это ограничивает условия проведения экспериментов (нельзя выйти за рамки действующих физических законов и технических и материальных ограничений), но добав-

ляет значительно больше реализма за счет необходимости учета инерции, помех, характеристик датчиков, микроконтроллеров и материалов и т.д., что представляет интерес с практической точки зрения. Для управления роботами могут использоваться разнообразные модели ИНС и нейронов. Например, в [96] рассматривается обучение микроробота Alice [71] навигации с избеганием препятствий с использованием эволюционной настройки структуры ИНС с упрощенной моделью спайкового нейрона (далее используются оригинальные обозначения из [96]):

$$o_i^t = \begin{cases} 1 \quad and \quad v_i^t = v_i^{\min}, \quad v_i^t > v_i^{\max} + r^t, \\ 0, \qquad\qquad\qquad\quad v_i^t \le v_i^{\max} + r^t, \end{cases}$$

где o_i^t – выходной сигнал; v_i^t, $v_i^{\min}$ и $v_i^{\max}$ – соответственно текущий, минимальный и максимальный мембранный потенциал нейрона; r^t – случайное целое, необходимое для предотвращения осцилляций и «блокировок» в ИНС с обратными связями. Значение мембранного потенциала v_i^t накапливается с течением времени с учетом входных сигналов нейрона и постоянной утечки. Оценка ИНС определяется в зависимости от скорости движения робота и расстояния до препятствий, которые необходимо максимизировать:

$$F = \sum_t V^t (1 - \Delta V^t)(1 - i),$$

где V^t – суммарная скорость вращения колес робота; ΔV^t – модуль разности скорости вращения левого и правого колеса; i – максимальная текущая активность одного из инфракрасных датчиков препятствий.

Поиск игровых стратегий [139, 157]. Данное направление исследований рассматривает использование ИНС для принятия решений в настольных играх. Часто рассматриваются крестики-нолики [139] и го [157, 176, 205]. Оценка работы ИНС затрудняется большим количеством различных вариантов развития событий и связанной с этим неопределенностью, что делает саму оценкой неточной.

В [205] показано, что при использовании НЭ подхода возможно успеш-

ное обучение ИНС игре в го против программы с детерминированным алгоритмом, реализованным в программе Gnugo [106], на поле 5x5 на основе оценки

$$F = 100 - \left(\frac{2\sum_{i=1}^{n} e_i}{n} + e_f \right),$$

где e_i – счет в партии после i-го хода, n – число ходов в партии, e_f – финальный счет партии. Также отметим вариант с «автообучением», когда настройка ИНС осуществляется путем соревнования одной эволюционирующей популяции ИНС с другой [157] (т.н. конкурентная коэволюция (competitive coevolution)). При этом оценка ИНС формируется в результате серии матчей с ИНС из «противоборствующей» популяции:

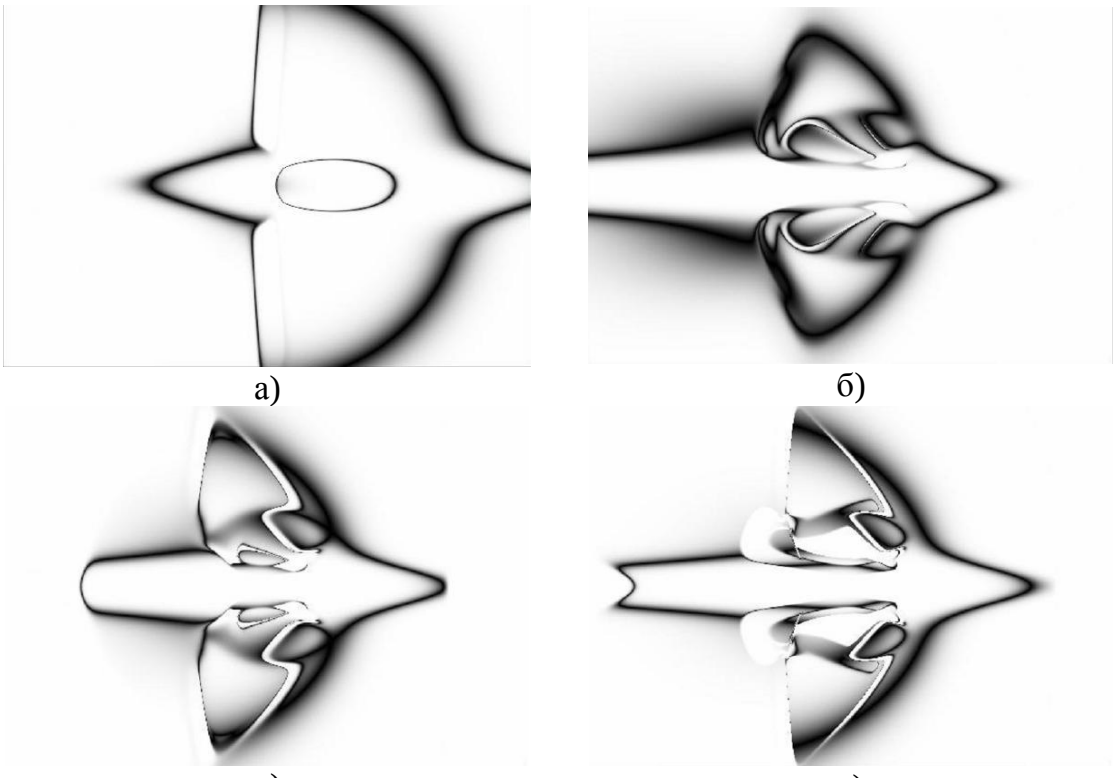
$$F = \sum_{i \in O} \frac{1}{L_i},$$

где O – множество оппонентов, побежденных данной ИНС, а L_i – число проигрышей i-й побежденной ИНС. НЭ обучение направлено на минимизацию оценки F.

Компьютерное творчество [64, 162]. Применение ЭА алгоритмов для компьютерного синтеза изображений [207, 211], звуков и мелодий [74, 88], геометрических форм [171, 207] и т.д. возможно благодаря использованию концепции *эстетической селекции* (*aesthetic selection*, называемой также *aesthetic evolution*, *interactive evolution*) [162]. Особенностью этой концепции является интерактивное взаимодействие ЭА с пользователем, который «вручную» определяет какие решения (изображения, звуки и т.д.) в популяции являются более приемлемыми, и, следовательно, какие особи будут допущены к скрещиванию. Это существенно увеличивает время эволюционного обучения и накладывает значительные ограничения на параметры ЭА и получаемых решений (нельзя использовать популяцию большого размера; решения должны быть такими, чтобы их можно было оценить за сравнительно небольшой промежуток времени и др. [162]), однако позволяет решать постав-

ленную задачу, избежав очевидных и неразрешенных в настоящее время трудностей с формализацией критериев оценки решений.

Например, в [207] описан способ генерации разнообразных геометрических образов и узоров, где ИНС реализует следующее преобразование: $L = g(x_1, x_2, d)$, здесь L – нормированная яркость пикселя с прямоугольными координатами $(x_1; x_2)$, d – расстояние до центра изображения. Таким образом, ИНС используется для «рисования», при этом яркость пикселей зависит от их координат. Отбор ИНС по всей популяции осуществляется пользователем путем выбора понравившихся ему соответствующих образов. Пример эволюции образа, названного «космический корабль» («spaceship»), сгенерированного ИНС в различных поколениях эволюционного поиска, показан на рис. 1.8.



а) б) в) г)

Рис. 1.8. Пример эволюции образа «космический корабль», сгенерированного ИНС в различных поколениях: а) начальный образ; б) появление узоров на «крыльях»; в) изменение формы «хвоста»; г) появление «выхлопов» за «крыльями».

1.4. Цель и задачи исследования

В данной работе рассматривается разработка нейроэволюционного алгоритма и программных средств для обработки цифровых изображений. Актуальность данной задачи объясняется следующими возможностями:

- облегчение решения ряда существующих задач в силу отсутствия необходимости в этапе формирования обучающей выборки, который сам по себе может быть достаточно ресурсоемким;
- нейросетевая аппроксимация преобразования яркости пикселей;
- расширение прикладных областей нейроинформатики и эволюционных вычислений (например, улучшение качества цифровых изображений [37]);
- развитие перспективных направлений, связанных с созданием интеллектуальных агентов [26, 206], робототехникой [172], компьютерным творчеством [162] и др.

В силу специфики рассматриваемой задачи выделим следующие требования, которые необходимо учитывать при разработке НЭ алгоритма для ИНС прямого распространения, реализующего максимально гибкий эволюционный поиск:

1. Возможность работы с хромосомами переменной длины.
2. Независимость от порядка следования генов в хромосоме.
3. Возможность одновременного существования ИНС с различной структурой в одной популяции.
4. Минимизация вероятности появления НС решений с «плохой» структурой (имеются ввиду ИНС с изолированными нейронами и группами нейронов; с «тупиковыми» нейронами, которые не являются входными или выходными и при этом не посылают сигналы другим нейронам и т.д.).
5. Возможность поиска структуры ИНС как в сторону усложнения (конструктивный поиск), так и в сторону упрощения (деструктивный поиск).

6. Настройка параметров в процессе работы алгоритма для улучшения получаемых результатов и возможности адаптации к условиям эволюционного поиска при решении различных задач.

Таким образом, **целью диссертационной работы** является разработка основанного на методах эволюционных вычислений адаптивного алгоритма для одновременной настройки структуры и весов связей ИНС и программных средств для обработки цифровых изображений.

Для достижения поставленной цели необходимо последовательное решение следующих задач:

1. Общий анализ ЭА и НЭ подхода и формулировка на основе результатов анализа требований к адаптивному НЭ алгоритму для ИНС прямого распространения.

2. Анализ ЭА с точки зрения сформулированных требований к НЭ алгоритму и разработка адаптивных генетических операторов. Решение данной задачи предполагает также исследование эффективности предлагаемых генетических операторов.

3. Разработка адаптивного НЭ алгоритма с учетом результатов решения предыдущих задач. Решение данной задачи предполагает также исследование эффективности предлагаемых методов и алгоритмов.

4. Апробация разработанного НЭ алгоритма для решения задачи улучшения визуального качества цифровых изображений на основе неточной оценки качества работы ИНС.

1.5. Основные результаты и выводы по главе

1. Проведен анализ ЭА, в результате которого для повышения эффективности эволюционного поиска показана необходимость использования генетических операторов рекомбинации и вариации на уровне фенотипов.

2. Проанализированы возможности использования ИНС для решения задач обработки и анализа изображений. На основе анализа показаны слож-

ности нейросетевой предобработки изображений и предложено использование ИНС для аппроксимации преобразования яркости пикселей.

3. На основе анализа использования ЭА для настройки весов и связей ИНС выделены преимущества и недостатки НЭ подхода, а также сформулированы общие требования к разработке НЭ алгоритма.

4. В результате анализа НЭ подхода отмечено, что использование НЭ алгоритмов расширяет область применения нейроинформатики, а также представляет интерес для решения задач адаптивного управления, многоагентных систем, эволюционной робототехники и компьютерного творчества.

5. На основе обсуждения проблем разработки эффективных адаптивных ЭА и обучения ИНС сформулированы цель и задачи исследований.

ГЛАВА 2. ИССЛЕДОВАНИЕ ЭВОЛЮЦИОННЫХ АЛГОРИТМОВ

В п. 1.4 был сформулирован ряд требований для разработки НЭ алгоритма, некоторые из которых (возможность работы с хромосомами переменной длины; независимость от порядка следования генов в хромосоме; адаптация параметров ЭА) могут быть выполнены независимо от поставленной задачи путем внесения модификаций в различные составляющие эволюционного алгоритма. В частности, для выполнения первых двух требований рассматривается генный оператор кроссинговера (см. п. 2.2), а адаптация параметров ЭА достигается за счет подстройки размера популяции (п. 2.3).

2.1. Анализ операторов кроссинговера

Оператор скрещивания (оператор кроссинговера, ОК) является основным оператором в *генетическом алгоритме* (ГА) и генетическом программировании (ГП) и используется для получения хромосом потомков путем рекомбинации родительских хромосом (пример для ГА с хромосомами с целочисленным кодированием[1] показан на рис. 2.1). Одним из главных аргументов в пользу использования этого оператора является возможность рекомбинации параметров решений, представленных различными особями. В терминах *гипотезы о строительных блоках*[2] (*building blocks hypothesis*) [107], посредством кроссинговера осуществляется поиск новых строительных блоков и смешивание уже существующих[3]. Однако в эволюционных стратегиях (ЭС) кроссинговер не является основным оператором, и совсем не используется в эволюционном программировании (ЭП).

---

[1] Для краткости далее будем называть такие хромосомы *целочисленными*.

[2] *Гипотеза о строительных блоках* (*building blocks hypothesis*) предполагает существование различимых частей хромосом разной длины – строительных блоков (СБ) и их комбинаций. Согласно этой гипотезе, наличие СБ и комбинаций СБ в некоторой хромосоме явно и практически всегда однозначно (положительно или отрицательно) сказывается на приспособленности соответствующей ей особи. Тогда эволюционный поиск сводится к *выявлению* (*identification*) «хороших» строительных блоков и к поиску их комбинаций, *смешиванию* (*mixing*), способствующих повышению приспособленности особей.

[3] Существует и «биологический» аргумент в пользу кроссинговера [132]. Его суть заключается в том, что скрещивание, отсутствующее у простейших, «изобретено» эволюцией для развития сложных организмов насекомых, рыб, земноводных, птиц и млекопитающих как ответ на усложнение их генетического кода, морфологии и более динамичные условия жизни. Исходя из этих соображений, делается вывод о необходимости использования оператора кроссинговера.

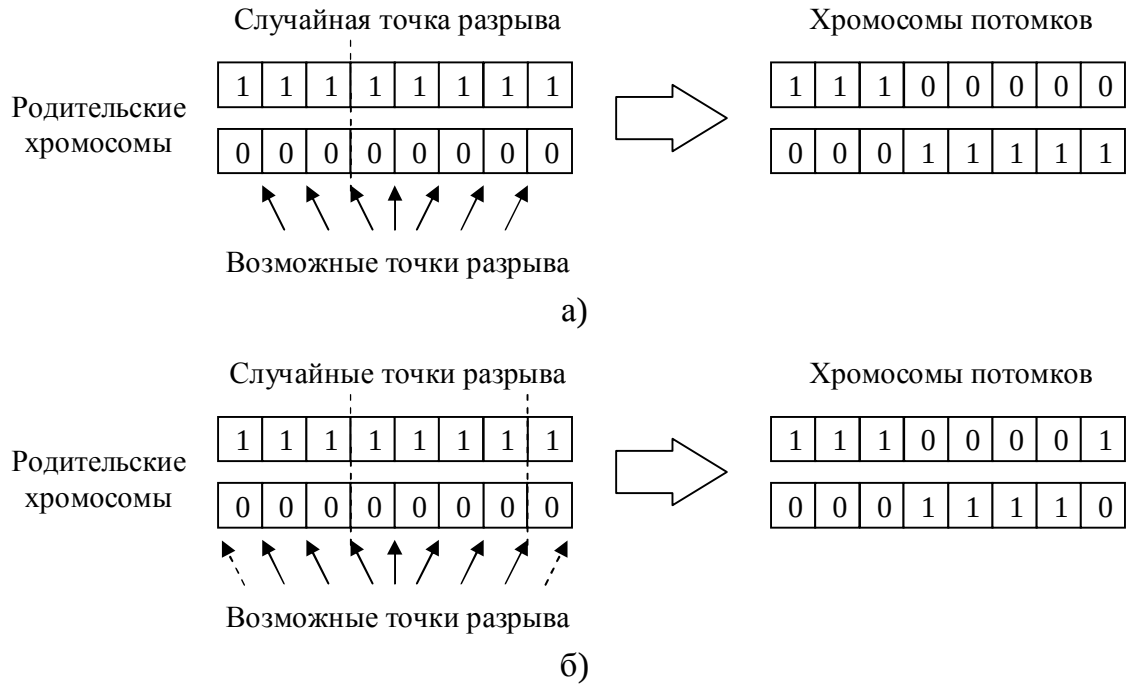


Рис. 2.1. 1-точечный (а) и 2-точечный (б) операторы скрещивания для целочисленных хромосом. Пунктирные стрелки указывают на одну и ту же точку разрыва 2-точечного ОК, которая «склеивает» начало и конец хромосомы (в соответствии с общепринятой условностью, впервые введенной в [81])

Применительно к концепции эволюционирующих ИНС существует серьезный аргумент против использования оператора кроссинговера [55, 228]. Его смысл может быть сформулирован следующим образом. Каждый нейрон в ИНС несет определенную функциональную нагрузку, например, реализуя разделяющую гиперплоскость (для нейронов с сигмоидной, линейной и ступенчатой функциями активации), либо выделяя центр некоторого кластера в пространстве входных параметров (для нейронов с радиально-базисными функциями активации и карт Кохонена). Таким образом, реализуемое ИНС отображение $\mathbf{X} \rightarrow \mathbf{Y}$, где $\mathbf{X}$ – пространство входных сигналов, $\mathbf{Y}$ – пространство выходных сигналов ИНС, существенно зависит от характеристик нейронов, их функциональной нагрузки и векторов весов их входных связей. Использование кроссинговера для рекомбинации параметров двух ИНС может стать причиной существенного искажения преобразований, соответствующих ИНС родительских особей, и, тем самым, привести к потере

существовавших решений. Другими словами, предполагается, что «скрещивание» ИНС без учета их свойств даст скорее неудовлетворительный результат, чем будет способствовать поиску решения.

Если рассматривать проблему перестановки (см. п. 1.3.3), то скрещивание различных хромосом, кодирующих одинаковые ИНС, может привести к тому, что в структуре ИНС особей-потомков одни скрытые нейроны с одинаковой функциональной нагрузкой будут продублированы, а другие скрытые нейроны будут отсутствовать.

Итак, существуют аргументы за и против использования ОК, в том числе и в НЭ алгоритмах, и это обстоятельство требует тщательного анализа свойств ОК для более успешного решения поставленной задачи. Основными свойствами ОК для целочисленных хромосом являются следующие:

1. *Вероятность сохранения* (*survival probability*) шаблона[1].
2. *Время смешивания* (*mixing time*).

2.1.1. Вероятность сохранения шаблона

Согласно [84] вероятность сохранения в результате рекомбинации шаблона H_1 порядка k длины L_1 определяется следующим рекуррентным соотношением:

$$p_{k,s}(h, L, L_1, ..., L_{k-1}) = \sum_{i=0}^{h} \binom{h}{i} \left(\frac{L_1}{L}\right)^i \left(\frac{L - L_1}{L}\right)^{h-i} p_{k-1,s}(i, L_1, ..., L_{k-1}),$$

где h – количество точек разрыва, $L_2, L_3, ..., L_{k-1}$ – соответственно определяющие длины шаблонов порядка $(k-1)$, $(k-2)$,..., 2, входящих в данный (для подробностей см. [200]), $\binom{a}{b} = \frac{a!}{b!(a-b)!}$. Для простейшего случая $k = 2$[2]

[1] *Шаблон* (*schema*) – одно из базовых понятий в теории ГА [131, 100]. Для целочисленных хромосом шаблон описывается как строка, соответствующая сразу множеству хромосом и состоящая из символов {0, 1, #}, где # – масочный символ, обозначающий как 1, так и 0. Приспособленность шаблона определяется как средняя приспособленность соответствующих хромосом в популяции. Существует гипотеза, что успех работы ГА определяется формированием с течением времени все более приспособленных шаблонов, описывающих хромосомы популяции.

[2] При $k < 2$ разрушение шаблона в результате скрещивания не происходит.

и $h = \{1;2\}$:

$$p_{2,s}(1, L, L_1) = \frac{L - L_1}{L} + \frac{L_1}{L} C_{k,s},$$

$$p_{2,s}(2, L, L_1) = \left(\frac{L - L_1}{L} \right)^2 + 2 \frac{L_1}{L} \frac{L - L_1}{L} C_{k,s} + \left(\frac{L_1}{L} \right)^2 = 1 - a(1 - C_{k,s}),$$

где $a = 2 \frac{L_1}{L} (1 - \frac{L_1}{L})$, $C_{k,s}$ – поправочный коэффициент, введенный Спирсом и Де Йонгом для учета вероятности скрещивания родительских хромосом с совпадающими разрядами в определяющих позициях шаблона. Обозначим множество из k определяющих позиций шаблона H_1 через K, а подмножество этих позиций через X, тогда можно вычислить $C_{k,s}$ следующим образом [84]:

$$C_{k,s} = \prod_{d \in X} p_{eq}(d) + \prod_{d \in K - X} p_{eq}(d) - \prod_{d \in K} p_{eq}(d),$$

где $p_{eq}(d)$ – вероятность того, что обе родительские хромосомы имеют одинаковые разряды в определяющей позиции d. При дополнительном условии, что $\forall d \in K : p_{eq}(d) = p_{eq}$, соответствующему началу эволюционного поиска, когда разряды в каждой позиции хромосомы принимают различные значения с одинаковой вероятностью, имеем:

$$C_{k,s} = p_{eq}^{|X|} + p_{eq}^{|K|-|X|} - p_{eq}^{|K|}.$$

В частности, для целочисленных хромосом при $k = 2$:

$$C_{2,s} = 2 p_{eq} - p_{eq}^2 = 0{,}75.$$

2.1.2. Время смешивания

Одним из основных свойств операторов скрещивания является способность к смешиванию, которая характеризует способности ОК к созданию пар хромосом потомков, отличных от хромосом родителей [51]. К примеру, для двух целочисленных родительских хромосом длины L, различающихся в k разрядах (расстояние по Хеммингу между хромосомами равно k), если $k \le (L - k)$, то одноточечный ОК позволяет создать (k + 1) различную пару

хромосом потомков, двухточечный ОК: $\frac{k^2-k+4}{2}$, а однородный ОК: $k+1+\sum_{i=2}^{k-1}\binom{k-1}{i}$ пар различных хромосом. Соответственно считается, что 1-точечный ОК имеет наименьшие способности к смешиванию, а однородный ОК Сисверды [208] является оператором с максимальной смешивающей способностью [83].

Время смешивания (*mixing time*), отражающее способности ОК к смешиванию, является временем, за которое оператор скрещивания, работающий без селекции и мутации, преобразует популяцию строк таких, что каждая пара не содержит одинаковых символов, в популяцию строк одинаковых по составу [181] (рис. 2.2) (порядок следования символов в строках не учитывается). Достаточно очевидно, что ОК с б***о***льшими способностями к смешиванию быстрее перемешивают популяцию.

Несмешанная популяция						Смешанная популяция					
Строка 1:	1	1	1	1	1	Строка 1:	1	3	2	5	4
Строка 2:	2	2	2	2	2	Строка 2:	3	1	5	4	2
Строка 3:	3	3	3	3	3	Строка 3:	5	2	4	1	3
Строка 4:	4	4	4	4	4	Строка 4:	4	5	3	2	1
Строка 5:	5	5	5	5	5	Строка 5:	2	4	1	3	5

Рис. 2.2. Пример несмешанной и смешанной популяции для случая 5-символьного алфавита

В работах Рабани с сотрудниками [181] и Пругеля-Беннета [179] показано, что время смешивания для 1- и 2-точечного ОК имеет порядок $O(L\ln L)$, а для однородного кроссинговера – $O(\ln L)$. Однако использованный в работах [181, 179] математический аппарат довольно сложен. В частности, в [181] эволюция популяции строк, изменяющейся под действием оператора кроссинговера, была представлена в виде динамической системы второго порядка

(quadratic dynamical systems) и получены достаточно точные верхняя и нижняя границы времени смешивания. В [179] оценка времени смешивания кроссинговера производится в результате анализа предложенного Пругелем-Беннетом параметра порядка – меры смешанности строк в популяции. Рассматриваемые операторы кроссинговера единообразно формализуются выражениями:

$$S_i^m(t+1) = c_i S_i^m(t) + (1-c_i) S_i^n(t),$$

$$S_i^n(t+1) = (1-c_i) S_i^m(t) + c_i S_i^n(t),$$

где $S_i^m(t)$ и $S_i^n(t)$ – соответственно значения i-х разрядов в m-й и n-й родительских хромосомах; $S_i^m(t+1)$ и $S_i^n(t+1)$ – значения i-х разрядов в m-й и n-й хромосомах потомков; c_i – параметр, зависящий от выбранного ОК, $c_i \in \{1;0\}$. Например, для однородного ОК:

$$c_i = \begin{cases} 1, x \geq a, \\ 0, x < a, \end{cases}$$

где x – случайная величина, равномерно распределенная на интервале [0; 1]; a – числовой параметр (по умолчанию, a = 0,5). Для 1-точечного ОК:

$$c_i = \begin{cases} 1, i \geq A, \\ 0, i < A, \end{cases}$$

где A – позиция точки разрыва. Аналогично определяется значение c_i и для 2-точечного ОК. Отметим, что специфика работы различных операторов (в частности, различная позиционная неопределенность (positional bias)) стала причиной использования в [179] разной методологии вычисления времени смешивания для разных операторов. Также отметим, что анализ в [181, 179] осложняется рассмотрением популяции строк, так как в этом случае приходится моделировать изменение распределения строк по их составу в пространстве параметров в ходе эволюции.

2.1.2.1 Способ вычисления времени смешивания для ОК для целочисленных хромосом. Далее будет показано, что можно получить оценки,

аналогичные оценкам из ранних исследований Рабани с сотрудниками и Пругеля-Беннета, используя более простой и универсальный подход.

Будем анализировать время t, которое потребуется, чтобы распределить символы некоторой строки $\mathbf{S}_0$ длины L по всей популяции так, чтобы все разряды попали в различные строки. Для вычисления времени смешивания для ОК считаем, что время необходимое для распределения разрядов из $\mathbf{S}_0$ по популяции, совпадает по порядку со временем, необходимым для распределения по популяции символов всех других строк, и, следовательно, совпадает по порядку со временем смешивания t для рассматриваемого ОК.

Рассмотрим нижнюю границу этого времени $t_{\min}$ для ОК, порождающего двух особей-потомков. В самом «благоприятном» варианте в момент времени $t = 0$ исходная строка делится на две равные части (по числу хромосом потомков). Затем эти части также делятся пополам, причем полученные половины переходят к разным хромосомам-потомкам, то есть для $t = 1$ разряды из исходной строки будут присутствовать в 4-х строках популяции. Рассуждая подобным образом, видно, что для полного распределения разрядов строки $\mathbf{S}_0$ по всей популяции необходимо минимум

$$t_{\min} = \log_2 L, \tag{2.1}$$

шагов, что соответствует минимально возможному времени смешивания ОК для случая генерации двух особей-потомков. Отметим, что эта величина совпадает по порядку со временем смешивания $\mathrm{O}(\ln L)$ для однородного кроссинговера t_{uni}, полученным ранее [181, 179].

Далее, для других ОК будем считать, что при скрещивании двух особей одна и только одна родительская хромосома содержит разряды из $\mathbf{S}_0$. То есть предполагаем, что длина L строки мала по сравнению с размером популяции n, то есть $n >> L$. В этом случае вероятность перемешать в результате скрещивания разряды из $\mathbf{S}_0$, которые могли бы присутствовать в каждой из родительских хромосом, невелика и ею можно пренебречь.

Для вычисления времени смешивания для 1-точечного ОК нам, таким образом, достаточно посчитать время, которое необходимо, чтобы точки кроссинговера попали на каждую позицию из $(L-1)$ возможной «внутри» $\mathbf{S}_0$. То есть будем вычислять время, за которое исходная строка будет разделена на отдельные разряды, которые, согласно сформулированному условию, будут распределены по различным хромосомам.

Также полагаем, что использованная единожды точка разрыва в дальнейшем не учитывается. Другими словами, считаем, что как только точка разрыва ОК попадает после i-го разряда $\mathbf{S}_0$, то повторное попадание точки разрыва в эту же позицию не приводит к дальнейшему разбиению $\mathbf{S}_0$ на подстроки и, следовательно, не влияет на оценку времени смешивания.

Для 1-точечного ОК время смешивания t_1 будем вычислять как:

$$t_1 = t_1(1) + t_1(2) + \ldots + t_1(L-1), \tag{2.2}$$

где $t_1(k)$ – время необходимое для попадания точки разрыва 1-точечного ОК в k-ю по счету возможную точку разрыва. Будем вычислять время $t_1(k)$ как величину, обратную частоте, $t_1(k) = p_1^{-1}(k)$, где $p_1(k)$ – вероятность попадания в k-ю позицию разрыва для 1-точечного ОК. Тогда имеем:

$$\begin{aligned} t_1(1) &= \frac{L-1}{L-1}, \\ t_1(2) &= \frac{L-1}{L-2}, \\ &\ldots \\ t_1(L-1) &= \frac{L-1}{1}. \end{aligned} \tag{2.3}$$

Подставляя (2.3) в (2.2), получим:

$$t_1 = (L-1)\sum_{k=1}^{L-1}\frac{1}{k}, \tag{2.4}$$

Известно, что сумма ряда $a_n = \frac{1}{n}$ расходится, при этом справедливо неравенство [4]:

$$\sum_{k=1}^{L-1}\frac{1}{k} > \frac{\log_2 4(L-1)}{2}. \qquad (2.5)$$

Подставляя данное неравенство в (2.4), получим:

$$t_1 > (L-1)\log_2\left(2\sqrt{L-1}\right) \approx 0{,}722(L-1)\ln 4(L-1), \qquad (2.6)$$

что совпадает по порядку со временем смешивания для 1-точечного кроссинговера $O(L\ln L)$, вычисленного в работах [181, 179].

Рассуждая аналогичным образом, вычислим нижнюю границу времени смешивания t_2 для 2-точечного ОК. Согласно известному соглашению, сформулированному в [81], для 2-точечного ОК существует L возможных точек разрыва, причем L-1 точка разрыва совпадает с таковыми для 1-точечного кроссинговера и еще одна «склеивает» начало и конец хромосомы. Будем рассматривать время $t_2(k,k+1) = p_2^{-1}(k,k+1)$, где $p_2(k,k+1)$ – вероятность «выбывания» из рассмотрения k-й и (k +1)-й возможных точек разрыва. Здесь, для более точной оценки t_2, рассматривается только случай, когда обе точки 2-точечного ОК попадают на еще «не использованные» точки разрыва в $\mathbf{S}_0$, поскольку во всех других случаях, в предлагаемом способе анализа будет рассматриваться случайная комбинация 1-точечного и 2-точечного ОК, что, безусловно, повлияет на результат вычисления t_2. Полагая для простоты, что L – четно, по аналогии с выражением (2.2) имеем:

$$t_2 = t_2(1,2) + t_2(3,4) + \ldots + t_2(L-1,L) = L^2\sum_{k=1}^{L/2}\frac{1}{2k(2k-1)} > \frac{L}{2}\sum_{k=1}^{L/2}\frac{1}{k}.$$

Подставив (2.5), получим:

$$t_2 > \frac{L}{2}\log_2\sqrt{2L} \approx 0{,}361L\ln 2L, \qquad (2.7)$$

Данная оценка времени смешивания для 2-точечного ОК совпадает по порядку с оценкой $O(L\ln L)$, полученной в [181, 179]. Заметим, что согласно (2.6) и (2.7): $t_1 \approx 2t_2$, что совпадает с результатом из [179].

Для случая n-точечного ОК имеем:

$$t_n = L^n \sum_{k=1}^{L/n} \frac{1}{nk(nk-1)\ldots(nk-n+1)} > \frac{L}{n} \sum_{k=1}^{L/n} \frac{1}{k}. \qquad (2.8)$$

В результате получаем:

$$t_n > \frac{L}{n} \log_2 2\sqrt{\frac{L}{n}} \approx 1{,}443 \frac{L}{n} \ln 2\sqrt{\frac{L}{n}}\,. \qquad (2.9)$$

Заметим, что с ростом n и L ошибка оценки (2.9) увеличивается, в силу преобразования (2.8), а также ввиду рассматриваемого случая $n >> L$. Для предельного случая $n = L$, соответствующего однородному ОК, получаем очевидное неравенство $t_L > 1$. Это показывает, что полученные оценки задают нижнюю границу времени смешивания для рассмотренных 1- и 2-точечного ОК.

2.1.2.2 Экспериментальная проверка оценок времени смешивания. Проверим экспериментально полученные оценки времени смешивания (2.6) и (2.7). Для этого рассмотрим генетический алгоритм (ГА), использующий только ОК, без селекции и мутации, с хромосомами в виде строк состоящих из A-арных символов. Для запуска будем использовать следующие значения параметров: $\mathrm{A} = n = L = 128$, где A – количество символов в используемом алфавите. Такой подбор параметров делает возможным случай полностью смешанной популяции (пример на рис. 2.2), в которой возможны строки, все символы которых различны. Начальная популяция состоит из несмешанных строк, когда любые две строки не имеют совпадающих символов, причем порядок следования символов не важен.

В качестве меры смешанности строк в популяции будем использовать следующую величину:

$$f = \frac{\sum_{i=1}^{\mathrm{A}} (a - c_i)^2}{\mathrm{A}}, \quad a = \frac{L}{\mathrm{A}},$$

где c_i – количество символов в оцениваемой хромосоме, равных i-му символу используемого алфавита. Таким образом, значение f соответствует отклонению распределения символов в строке от равномерного.

Графики изменения средней меры смешанности популяции в поколении t $\langle f(t)\rangle = \dfrac{\sum f_i(t)}{n}$, усредненные по 100 запускам, показаны на рис. 2.3 и 2.4. Также представлено крупным планом изменение $\langle f \rangle$ в окрестности стационарного значения, и соответствующие значения времени смешивания, вычисленные по формулам (2.6) и (2.7).

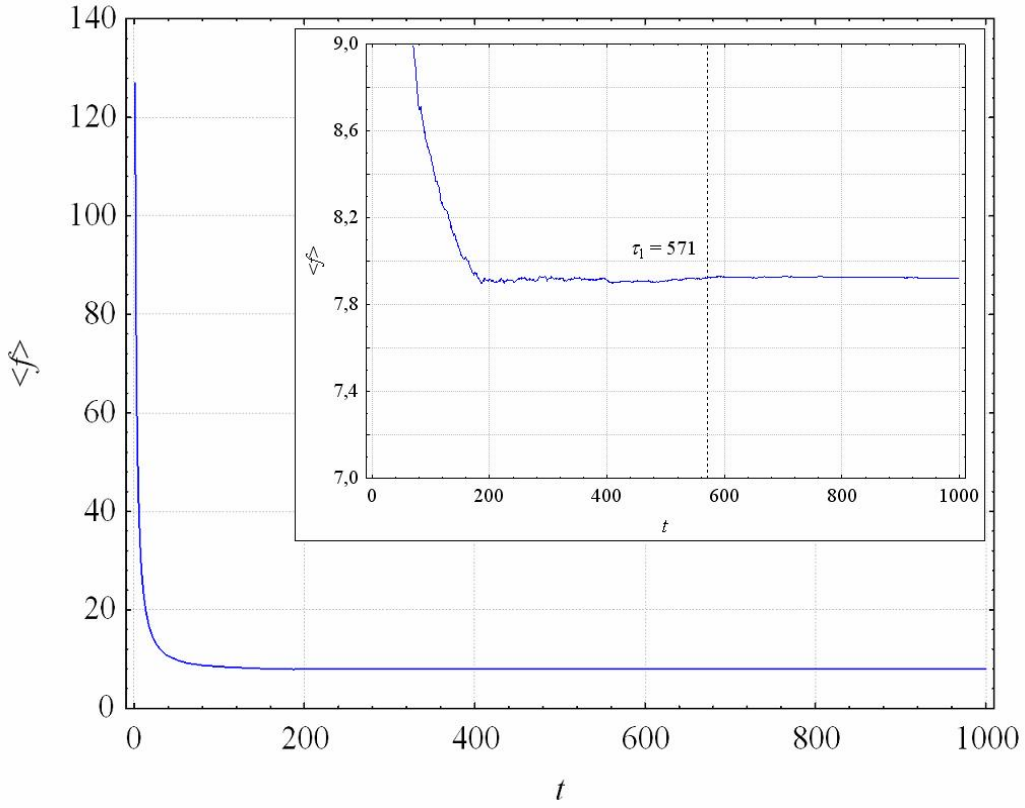


Рис. 2.3. График изменения $\langle f \rangle$ для 1-точечного ОК. Во врезке крупным планом показано изменение $\langle f \rangle$ в окрестности стационарного значения, а также время смешивания $t_2 = 571$, вычисленное по формуле (2.6) (обозначено пунктирной линией)

По графикам на рис. 2.3 и 2.4 видно, что формулы (2.6) и (2.7) дают оценку времени смешивания снизу. Отметим, что в случае двухточечного кроссинговера (рис. П1.1) амплитуда колебаний $\langle f \rangle$ около стационарного значения больше, чем в случае одноточечного ОК (рис. 2.3), что можно объ-

яснить б*о*льшим разнообразием строк в популяции, генерируемых с использованием двухточечного ОК. При этом «стабильное» стационарное значение $\langle f \rangle$, около которого и происходят рассматриваемые колебания, для кривой на рис. 2.4 можно выделить, начиная с момента времени $t = 320$, что превышает вычисленное по формуле (2.7) значение нижней оценки времени смешивания $t_2 = 256$.

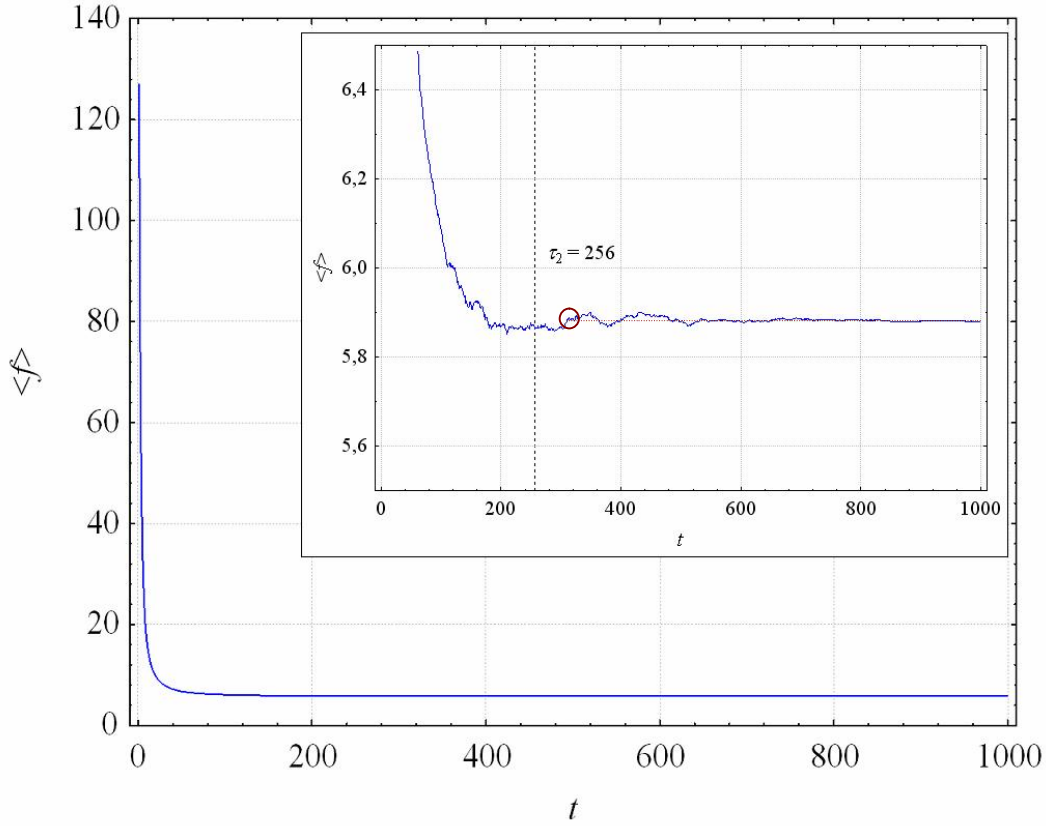


Рис. 2.4. График изменения $\langle f \rangle$ для 2-точечного ОК. Во врезке крупным планом показано изменение $\langle f \rangle$ в окрестности стационарного значения, а также время смешивания $t_2 = 256$, вычисленное по формуле (2.7) (обозначено пунктирной линией)

2.1.2.3 Анализ и обсуждение результатов моделирования. При выводе формул (2.6) и (2.7) рассматривался случай $n >> L$, что позволило, с одной стороны, пренебречь вероятностью скрещивания строк, содержащих разряды из $\mathbf{S}_0$, а с другой, считать, что скорость «распространения» в популяции раз-

рядов из $\mathbf{S}_0$ постоянна. При этом полученные формулы хорошо согласуются с результатами моделирования для случая $n = L$.

Попытаемся объяснить полученный результат. Обозначим общую долю строк в популяции, содержащих разряды из $\mathbf{S}_0$, через x_0 и рассмотрим величину $\frac{dx_0}{dt}$. Очевидно, что $\frac{dx_0}{dt}$ пропорционально $x_0(1-x_0)$ – вероятности скрестить строку, содержащую символы из $\mathbf{S}_0$, со строкой, которая этих символов не содержит, и обратно пропорционально x_0^2 вероятности скрестить строки, обе содержащие символы из $\mathbf{S}_0$. Можем записать:

$$\frac{dx_0}{dt} = a x_0(t)(1-x_0(t)) - b x_0^2(t), \qquad (2.10)$$

где a и b – константы, зависящие от выбора ОК. Будем считать, что случаем, когда скрещивание двух строк, содержащих символы из $\mathbf{S}_0$, привело к уменьшению x_0 можно пренебречь, тогда квадратичный член в правой части уравнения (2.10) можно отбросить, и получим хорошо известную модель логистического роста: $dx_0/dt = a x_0(t)(1-x_0(t))$, решением которой является уравнение:

$$x_0 = \frac{1}{1+\exp(-at)}. \qquad (2.11)$$

Пример графика уравнения (2.11) при $a = 0{,}01$ представлен на рис. 2.5. Характерным свойством логистической модели роста является симметричность графика относительно точки перегиба (в данном случае $t_0 = 50$). Другими словами, увеличение dx_0/dt на начальном этапе и уменьшение на конечном компенсируют друг друга, что, предположительно, и позволяет рассмотреть «выбывание» фиксированного числа точек разрыва в $\mathbf{S}_0$ по очереди. Таким образом, сделано предположение о причинах согласия формул (2.6) и (2.7) с результатами моделирования для случая $n = L$.

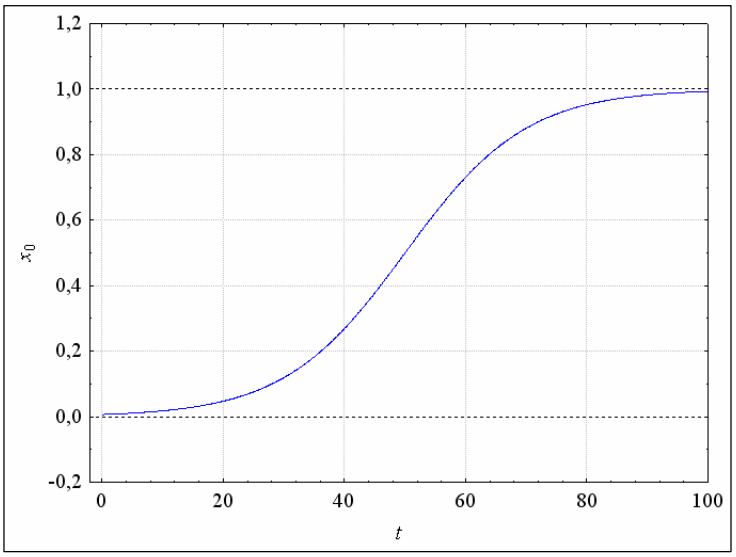


Рис. 2.5. Пример графика логистического роста при $a = 0{,}01$

2.2. Генный оператор кроссинговера

Несмотря на то, что к настоящему времени предложено множество операторов скрещивания для целочисленных хромосом [62], наиболее часто используются «классические» 1-, 2-, *n*-точечный [81] и однородный ОК [208]. Однако, эти операторы не лишены недостатков. В частности, недостатками 1, 2 и *n*-точечного ОК являются:

- зависимость вероятности сохранения шаблона длины L_1 от длины хромосомы L (зависимость от значения отношения L_1 / L) [84]. Зависимость обуславливается наличием *позиционной неопределенности* (*positional bias*) [91] известной также как *неопределенность расстояния* (*length bias*) [84];
- зависимость результата скрещивания от порядка следования параметров в хромосоме, что актуально, когда закодированные в хромосоме параметры имеют различный вес при вычислении целевой функции;
- невозможность скрещивания хромосом с различным количеством генов.

Однородный ОК не имеет позиционной неопределенности и поэтому этот оператор свободен от вышеперечисленных недостатков 1-, 2- и *n*-

точечного ОК. Тем не менее, необходимость при скрещивании 2 особей делать L вызовов датчика случайных чисел увеличивает общую вычислительную сложность ГА, что отрицательно сказывается на продолжительности работы ГА в случае длительной эволюции популяции хромосом большой длины.

Для того чтобы избавиться от недостатков, присущих 1, 2 и n-точечному оператору кроссинговера, предлагается использовать *генный* ОК (per-gene crossover, PGX), при работе которого гены «скрещиваются» независимо друг от друга (рис. 2.3). В таком виде генный ОК обрабатывает каждый ген отдельно, также как и многие операторы кроссинговера для вещественного кодирования [125], например, SBX-кроссинговер [85] или арифметический кроссинговер [164]. Предлагаемый оператор скрещивания назван генным, чтобы подчеркнуть его отличие от известных ОК для целочисленного кодирования, обрабатывающих хромосомы «целиком». Будем рассматривать 1- и 2-точечный генный ОК по количеству точек разрыва на каждый ген (рис. 2.6).

Для возможности скрещивания хромосом различной длины считаем, что каждый ген имеет некоторую метку l, которая отличает его от других генов хромосомы. Во время скрещивания родительские хромосомы предварительно выравниваются с совмещением генов, имеющих одинаковые метки. Затем родительские гены с совпадающими метками (будем называть такие гены совпадающими) скрещиваются с использованием генного ОК, а несовпадающие гены случайно разыгрываются между потомками (рис. 2.7). Реализация такого подхода к скрещиванию хромосом представляющих ИНС с различной структурой описана в п. 3.4.1, и по своей идее схожа с использованным К. Стенли [204] способом скрещивания ИНС на основе исторических меток (historical markings):

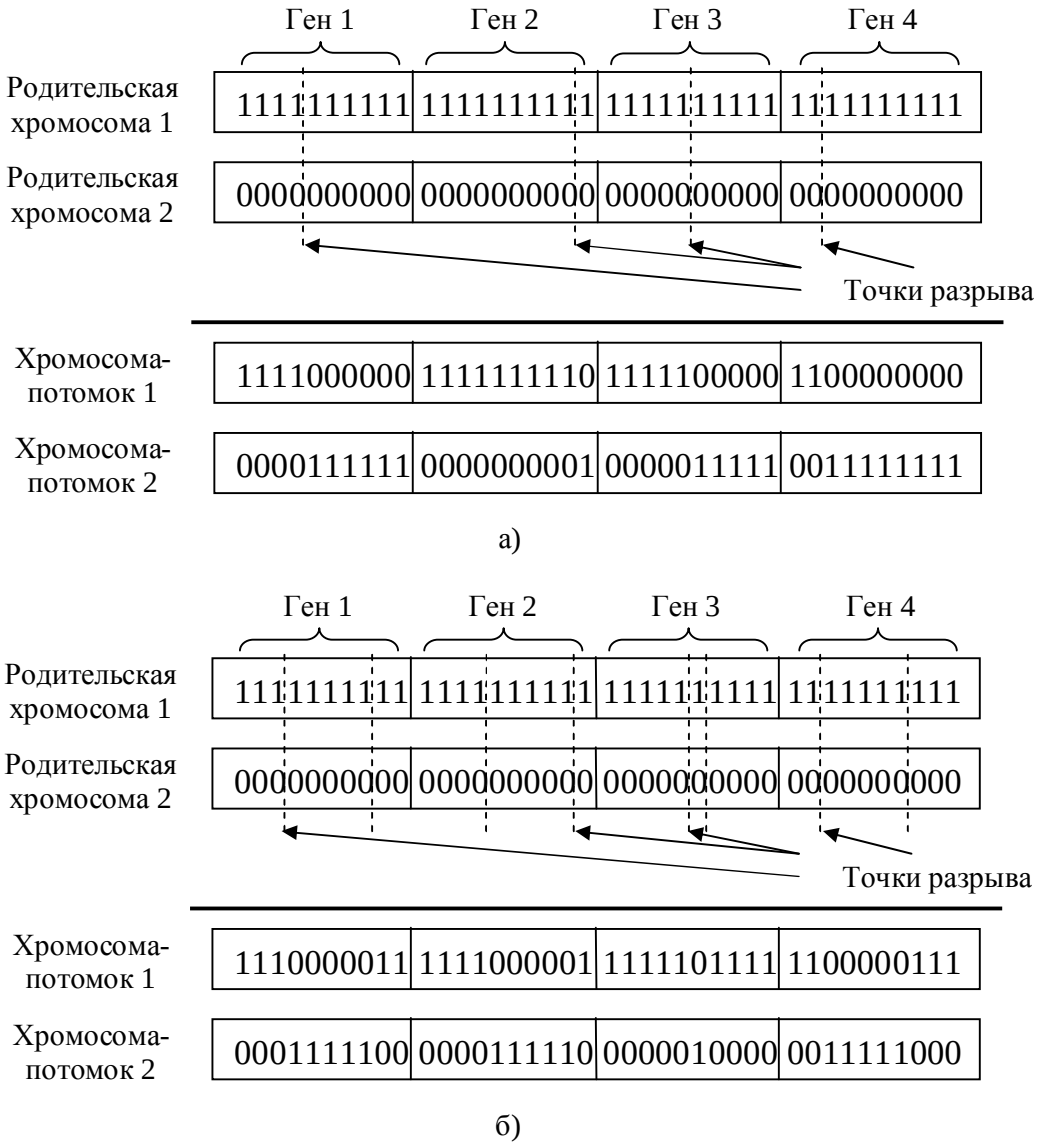


Рис. 2.6. Пример работы 1-точечного (а) и 2-точечного (б) генного ОК.

Использование кроссинговера для каждого гена позволяет избежать 1-го и 3-го недостатков 1-, 2- и *n*-точечного ОК. Применение генного ОК также соответствует случаю простого варианта адаптации ОК к размерности пространства поиска: с ростом числа оптимизируемых параметров разрушающие способности генного ОК также увеличиваются, так как растет количество точек разрыва, что подтверждается результатами исследования, представленными в п. 2.2.1.

Рис. 2.7. Пример скрещивания хромосом различной длины с использованием генного ОК (применяется к генам с метками l_2 и l_5). Гены с метками l_1, l_3 и l_6 разыгрываются между потомками случайным образом. Нумерация генов в родительских хромосомах и хромосомах потомков – независимая.

2.2.1. Исследование эффективности генного ОК

Исследуем экспериментально эффективность предлагаемого генного ОК. Условия проведения экспериментов представлены в табл. 2.1, а используемые тестовые функции и разрядность генов g в табл. 2.2. Выбор значений параметров ГА в первую очередь продиктован необходимостью оценить результаты работы различных ОК при как можно меньшем влиянии других параметров, таких как селекция и оператор мутации.

Для всех используемых тестовых функции необходимо решить задачу минимизации: $F \to \min$.

Будем оценивать усредненное по результатам 100 запусков решение, к которому сойдется популяция. Сходимость ГА, соответствующая случаю, когда в популяции все особи будут одинаковы, возможна так как вероятность мутации равна 0.

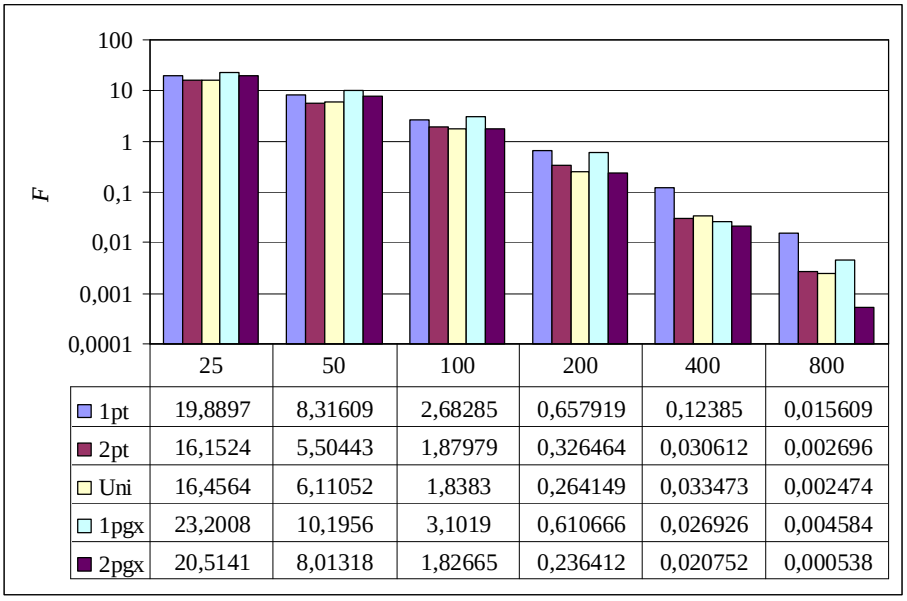
Табл. 2.1. Условия экспериментов

Параметр	**Значение**
Селекция	Турнирная
Размер турнира	2
Вероятность кроссинговера, P_c	1
Вероятность мутации, P_m	0
Размер популяции, N	25, 50, 100, 200, 400, 800
Количество запусков	100

Табл. 2.2. Тестовые функции для оценки эффективности генного ОК. Количество параметров $n \in \{10,30,50\}$

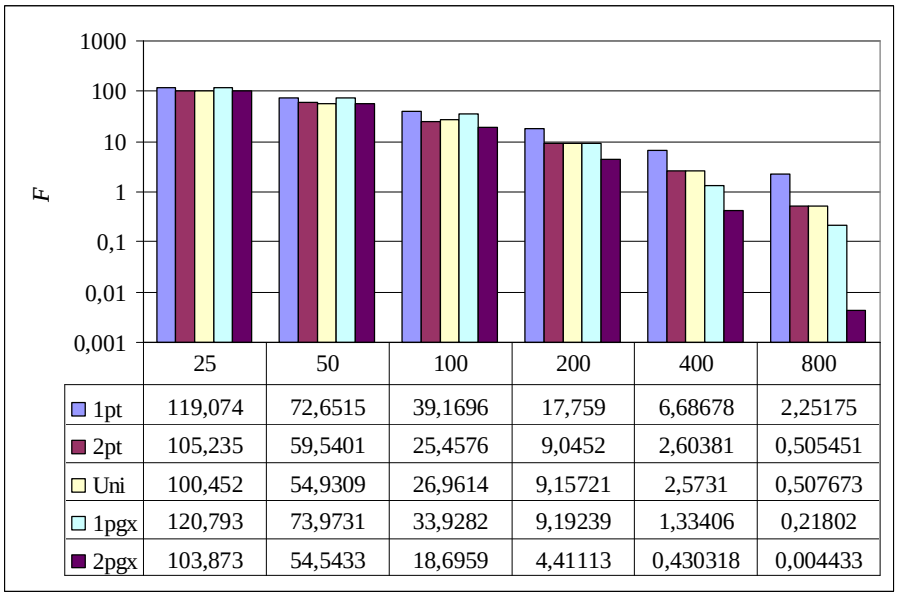
Название	**Формула**
Сферическая функция	$F_1 = \sum_{i=1}^{n} x_i^2, x \in (-5{,}12;\ 5{,}12), g = 10$
Функция Растригина	$F_2 = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)), x \in (-5{,}12;\ 5{,}12), = 10$
Функция Розенброка	$F_3 = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2), x \in (-2{,}048;\ 2{,}048), g = 12$
Функция Швефеля	$F_4 = 418{,}9829n + \sum_{i=1}^{n} - x_i \sin(\sqrt{\lvert x_i \rvert}), x \in (-524{,}288;\ 524{,}288), g = 20$

Результаты экспериментов для тестовых функций (диаграммы и таблицы данных) показаны на рис. 2.8-2.19. Высота столбцов диаграмм показывает значения целевой функции (также содержащиеся в таблице данных), к которым сошлась популяция в результате работы соответствующего ОК. Данные сгруппированы в зависимости от размера популяции. Различные ОК обозначены следующим образом: 1pt – 1-точечный; 2pt – 2-точечный; Uni – однородный; 1pgx – 1-точечный генный; 2pgx – 2-точечный генный.



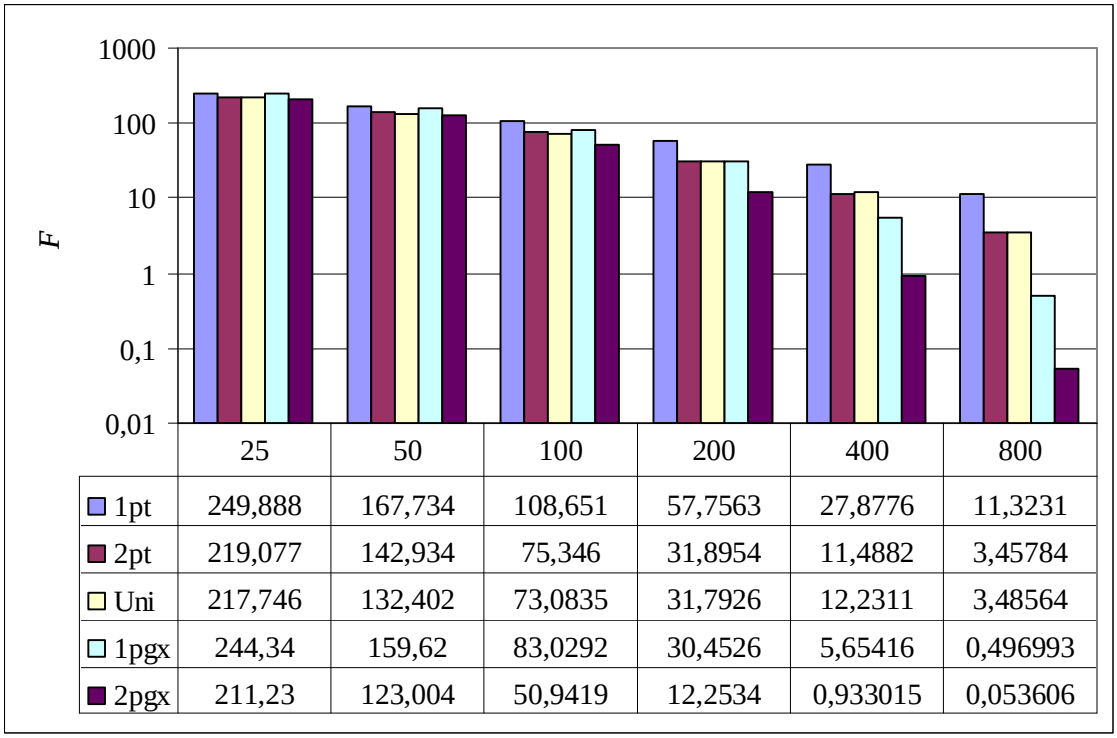
	25	50	100	200	400	800
1pt	19,8897	8,31609	2,68285	0,657919	0,12385	0,015609
2pt	16,1524	5,50443	1,87979	0,326464	0,030612	0,002696
Uni	16,4564	6,11052	1,8383	0,264149	0,033473	0,002474
1pgx	23,2008	10,1956	3,1019	0,610666	0,026926	0,004584
2pgx	20,5141	8,01318	1,82665	0,236412	0,020752	0,000538

Рис. 2.8. Результаты для тестовой функции F_1 от 10 переменных. По вертикальной оси отложены значения целевой функции, к которым сошелся ГА, по горизонтальной оси – использованный размер популяции. Данные сгруппированы в зависимости от значения размера популяции. Обозначения операторов кроссинговера: 1pt – 1-точечный; 2pt – 2-точечный; Uni – однородный; 1pgx – 1-точечный генный; 2pgx – 2-точечный генный



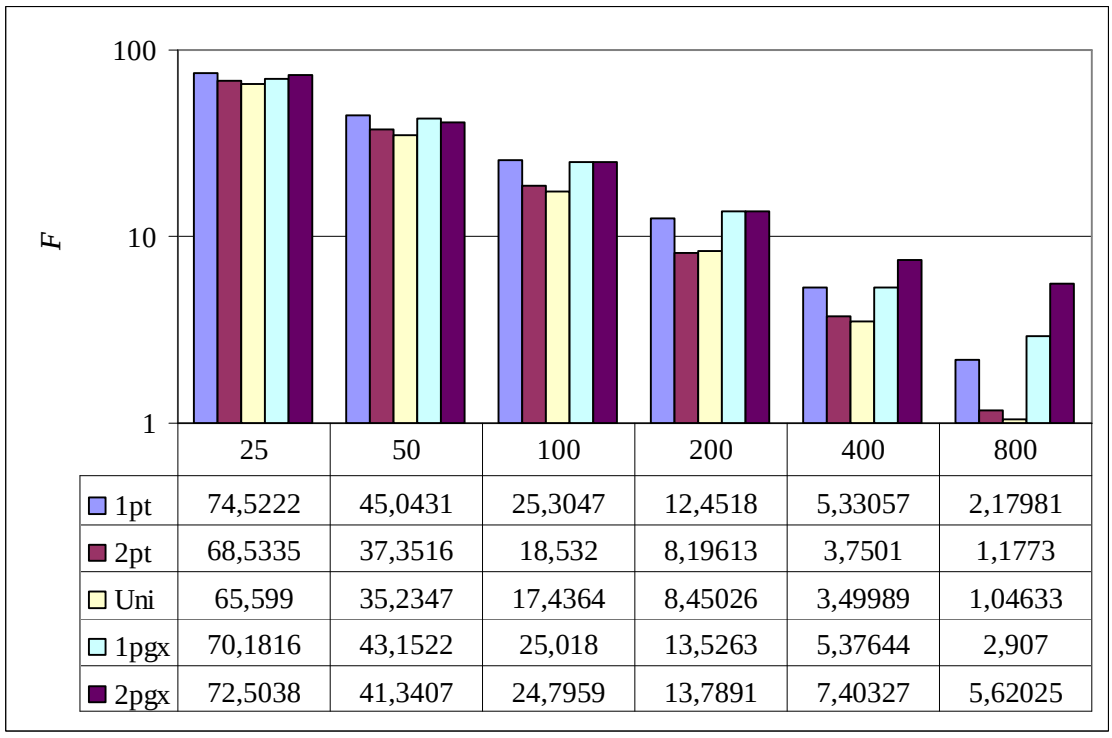
	25	50	100	200	400	800
1pt	119,074	72,6515	39,1696	17,759	6,68678	2,25175
2pt	105,235	59,5401	25,4576	9,0452	2,60381	0,505451
Uni	100,452	54,9309	26,9614	9,15721	2,5731	0,507673
1pgx	120,793	73,9731	33,9282	9,19239	1,33406	0,21802
2pgx	103,873	54,5433	18,6959	4,41113	0,430318	0,004433

Рис. 2.9. Результаты для тестовой функции F_1 от 30 переменных. Обозначения те же, что и на рис. 2.8.



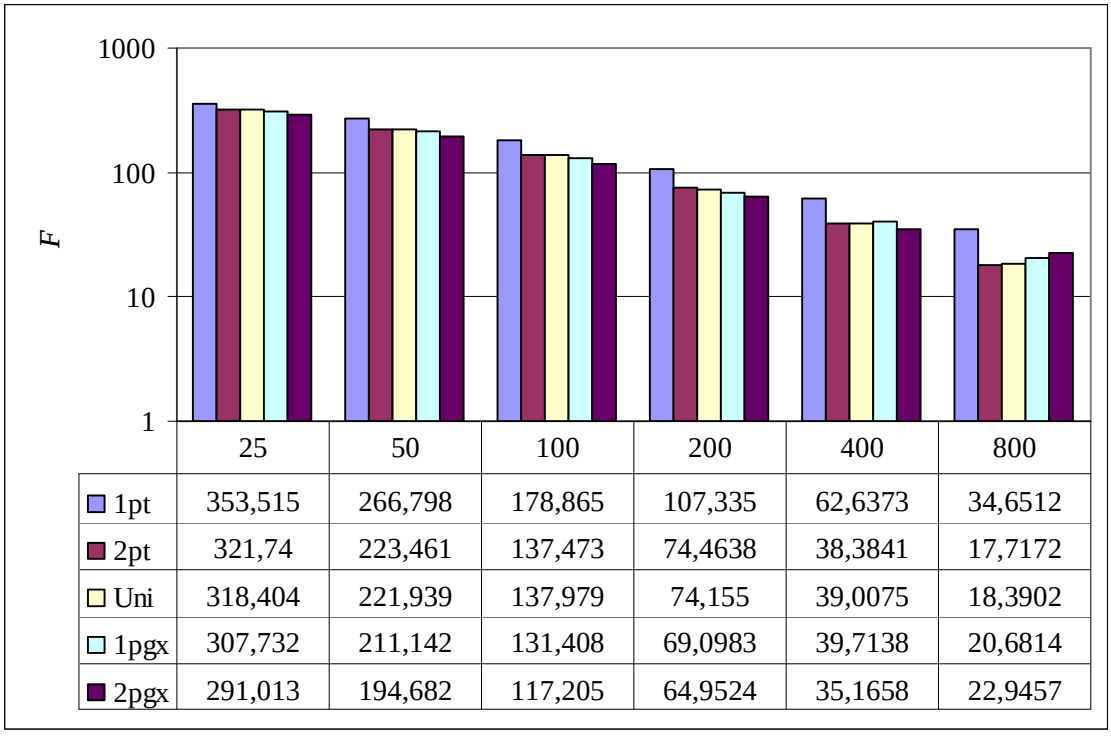
	25	50	100	200	400	800
1pt	249,888	167,734	108,651	57,7563	27,8776	11,3231
2pt	219,077	142,934	75,346	31,8954	11,4882	3,45784
Uni	217,746	132,402	73,0835	31,7926	12,2311	3,48564
1pgx	244,34	159,62	83,0292	30,4526	5,65416	0,496993
2pgx	211,23	123,004	50,9419	12,2534	0,933015	0,053606

Рис. 2.10. Результаты для тестовой функции F_1 от 50 переменных. Обозначения те же, что и на рис. 2.8.



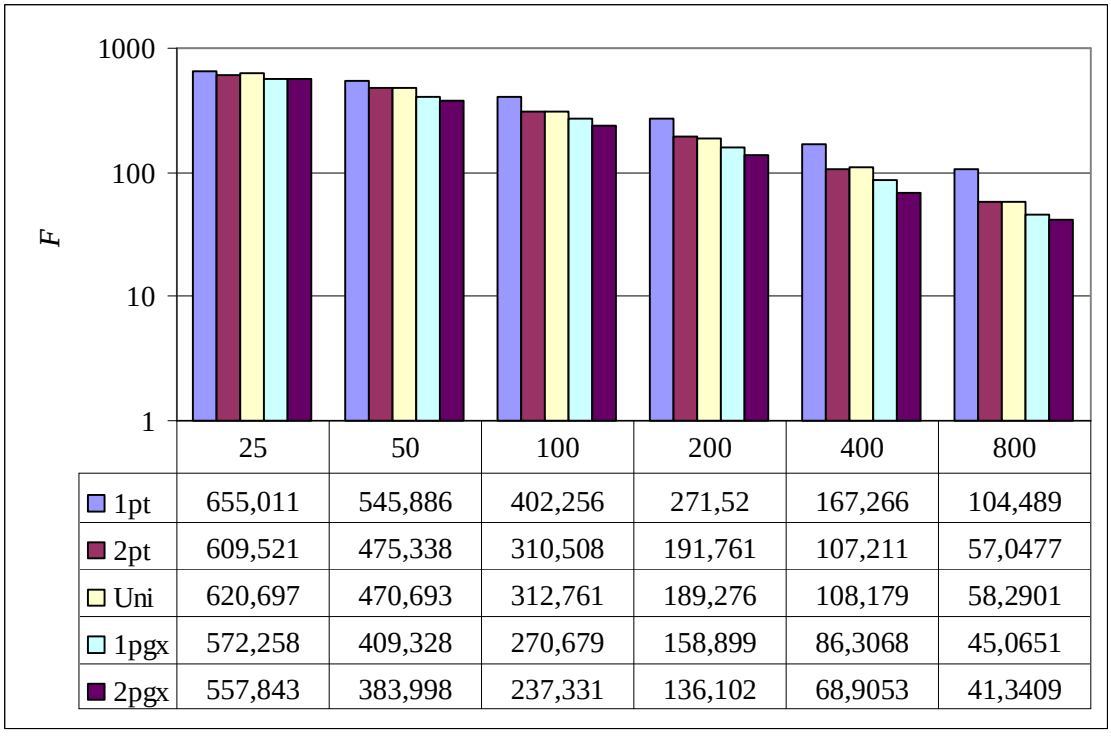
	25	50	100	200	400	800
1pt	74,5222	45,0431	25,3047	12,4518	5,33057	2,17981
2pt	68,5335	37,3516	18,532	8,19613	3,7501	1,1773
Uni	65,599	35,2347	17,4364	8,45026	3,49989	1,04633
1pgx	70,1816	43,1522	25,018	13,5263	5,37644	2,907
2pgx	72,5038	41,3407	24,7959	13,7891	7,40327	5,62025

Рис. 2.11. Результаты для тестовой функции F_2 от 10 переменных. Обозначения те же, что и на рис. 2.8.



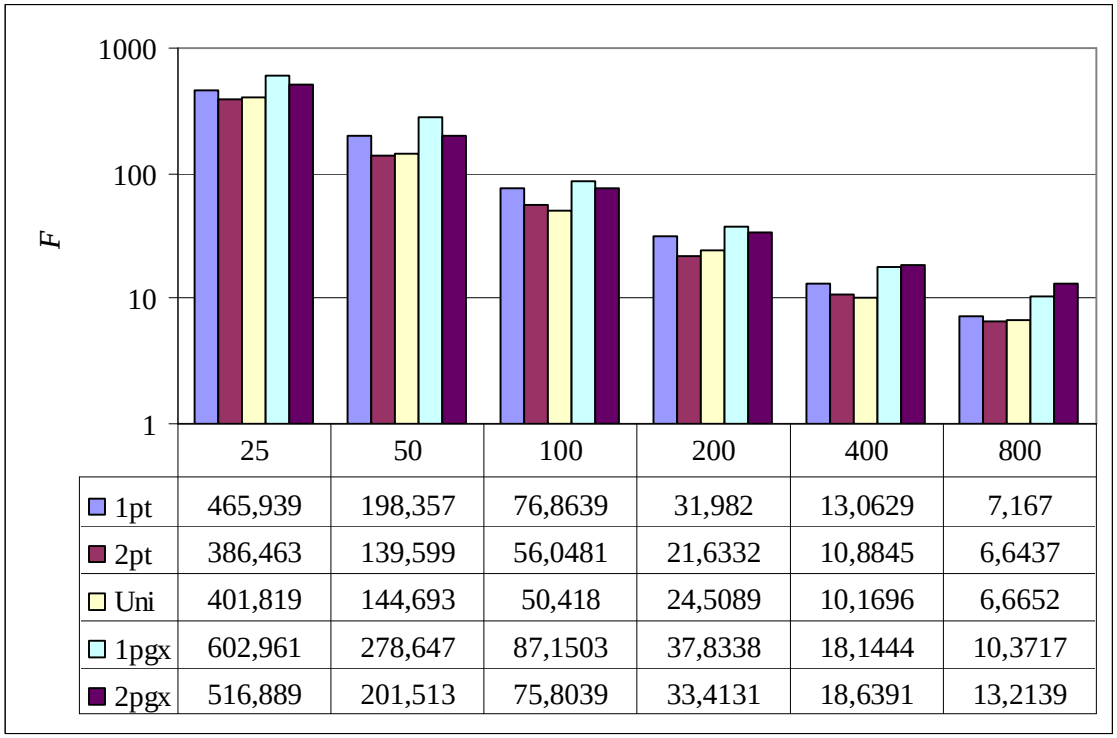
	25	50	100	200	400	800
1pt	353,515	266,798	178,865	107,335	62,6373	34,6512
2pt	321,74	223,461	137,473	74,4638	38,3841	17,7172
Uni	318,404	221,939	137,979	74,155	39,0075	18,3902
1pgx	307,732	211,142	131,408	69,0983	39,7138	20,6814
2pgx	291,013	194,682	117,205	64,9524	35,1658	22,9457

Рис. 2.12. Результаты для тестовой функции F_2 от 30 переменных. Обозначения те же, что и на рис. 2.8.



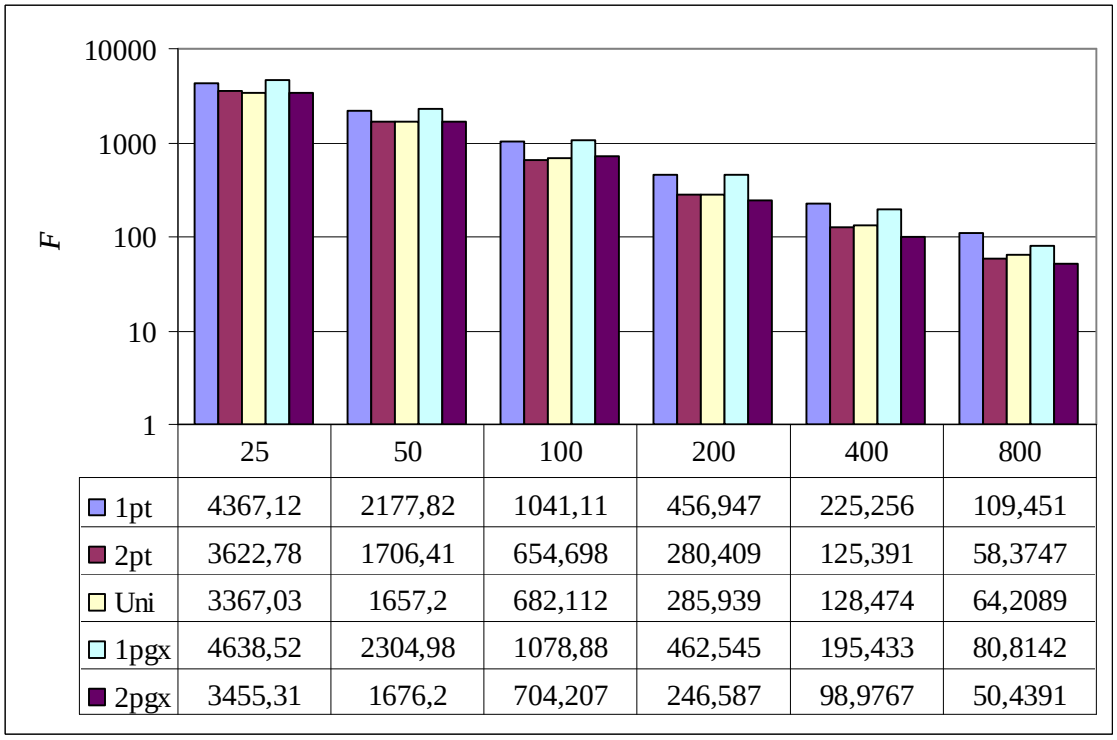
	25	50	100	200	400	800
1pt	655,011	545,886	402,256	271,52	167,266	104,489
2pt	609,521	475,338	310,508	191,761	107,211	57,0477
Uni	620,697	470,693	312,761	189,276	108,179	58,2901
1pgx	572,258	409,328	270,679	158,899	86,3068	45,0651
2pgx	557,843	383,998	237,331	136,102	68,9053	41,3409

Рис. 2.13. Результаты для тестовой функции F_2 от 50 переменных. Обозначения те же, что и на рис. 2.8.



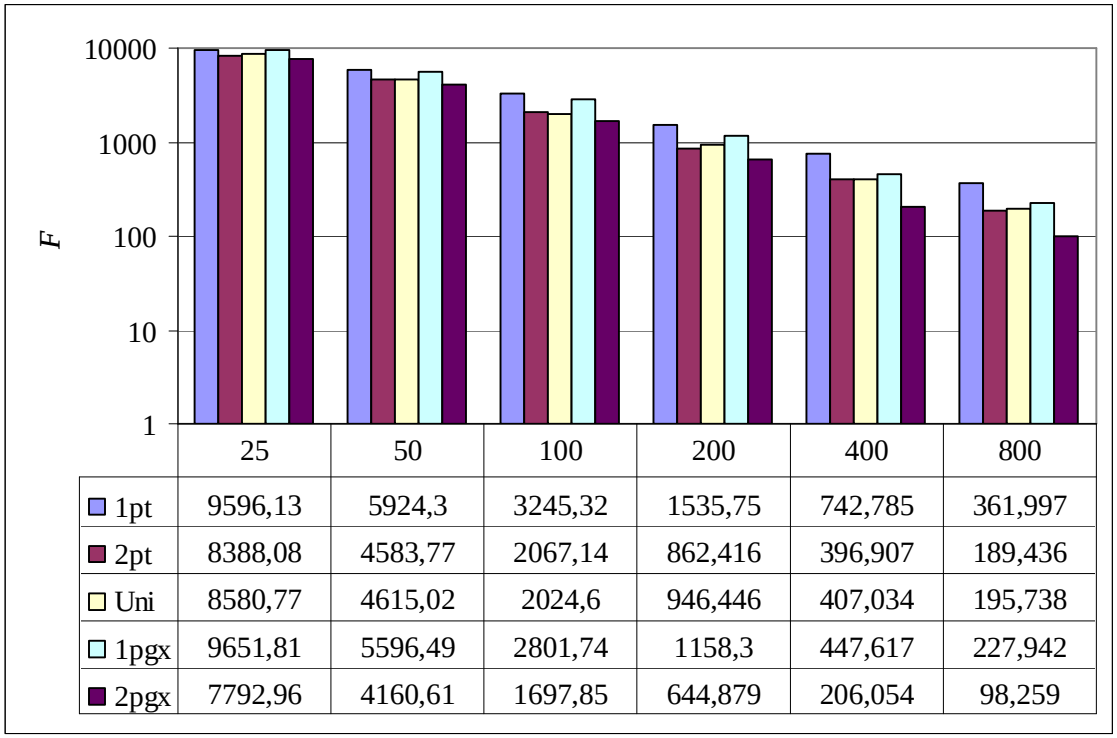
	25	50	100	200	400	800
1pt	465,939	198,357	76,8639	31,982	13,0629	7,167
2pt	386,463	139,599	56,0481	21,6332	10,8845	6,6437
Uni	401,819	144,693	50,418	24,5089	10,1696	6,6652
1pgx	602,961	278,647	87,1503	37,8338	18,1444	10,3717
2pgx	516,889	201,513	75,8039	33,4131	18,6391	13,2139

Рис. 2.14. Результаты для тестовой функции F_3 от 10 переменных. Обозначения те же, что и на рис. 2.8.



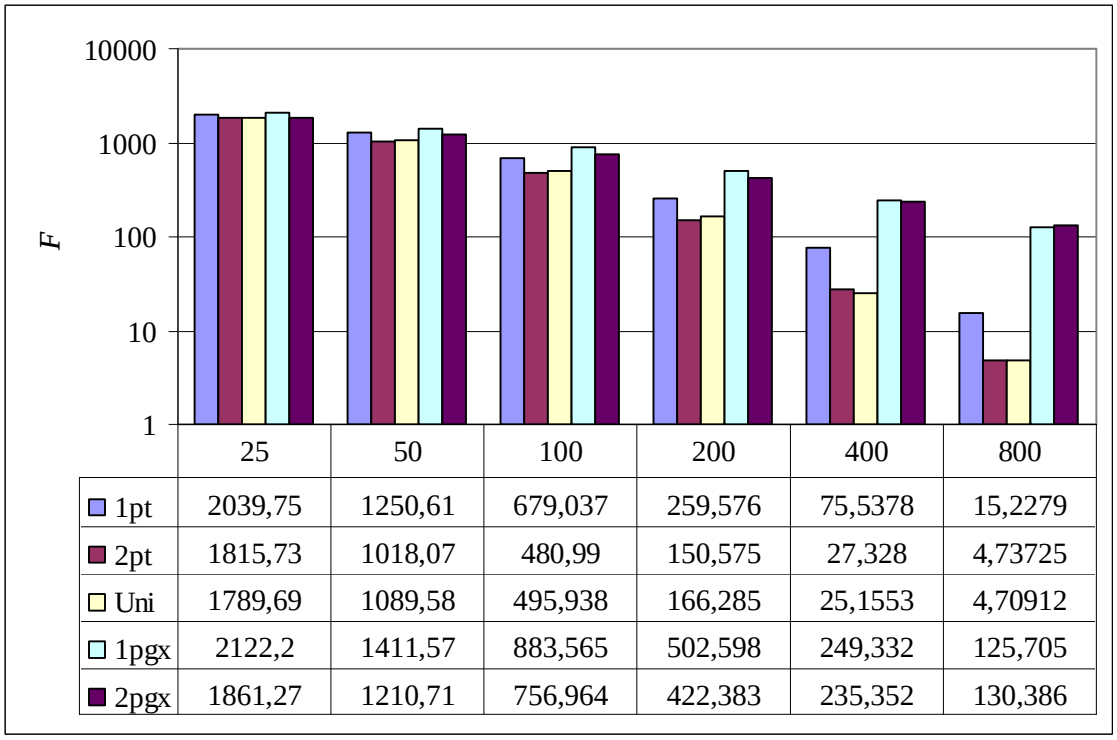
	25	50	100	200	400	800
1pt	4367,12	2177,82	1041,11	456,947	225,256	109,451
2pt	3622,78	1706,41	654,698	280,409	125,391	58,3747
Uni	3367,03	1657,2	682,112	285,939	128,474	64,2089
1pgx	4638,52	2304,98	1078,88	462,545	195,433	80,8142
2pgx	3455,31	1676,2	704,207	246,587	98,9767	50,4391

Рис. 2.15. Результаты для тестовой функции F_3 от 30 переменных. Обозначения те же, что и на рис. 2.8.



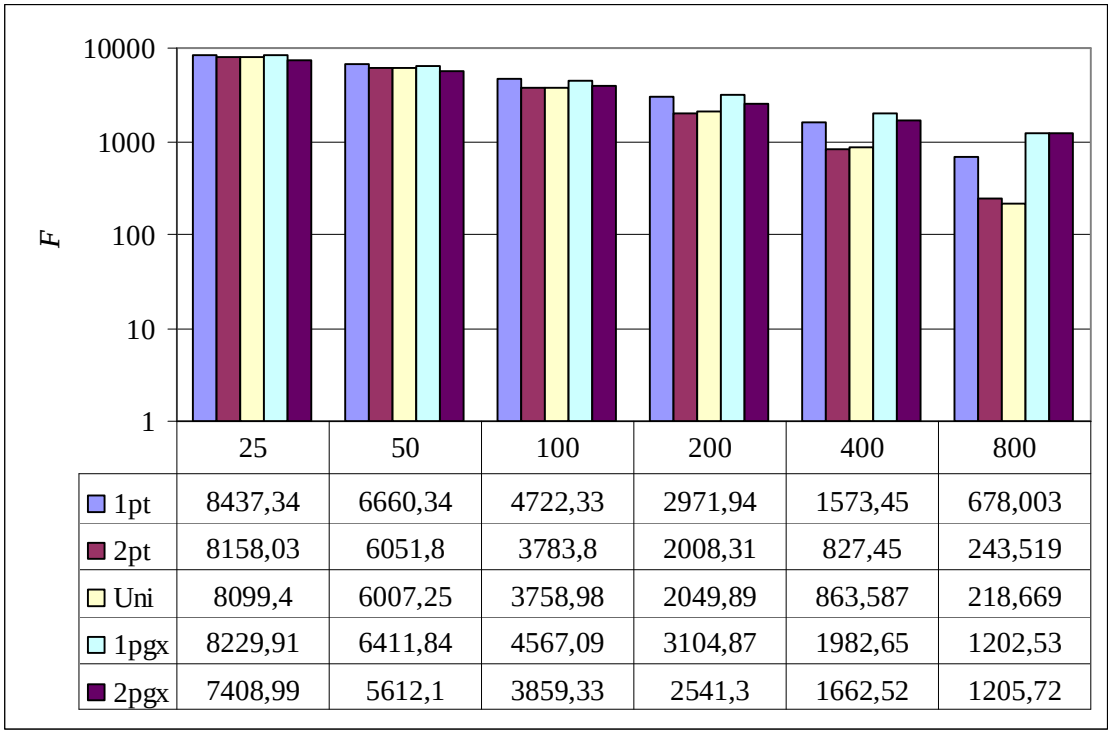
	25	50	100	200	400	800
1pt	9596,13	5924,3	3245,32	1535,75	742,785	361,997
2pt	8388,08	4583,77	2067,14	862,416	396,907	189,436
Uni	8580,77	4615,02	2024,6	946,446	407,034	195,738
1pgx	9651,81	5596,49	2801,74	1158,3	447,617	227,942
2pgx	7792,96	4160,61	1697,85	644,879	206,054	98,259

Рис. 2.16. Результаты для тестовой функции F_3 от 50 переменных. Обозначения те же, что и на рис. 2.8.



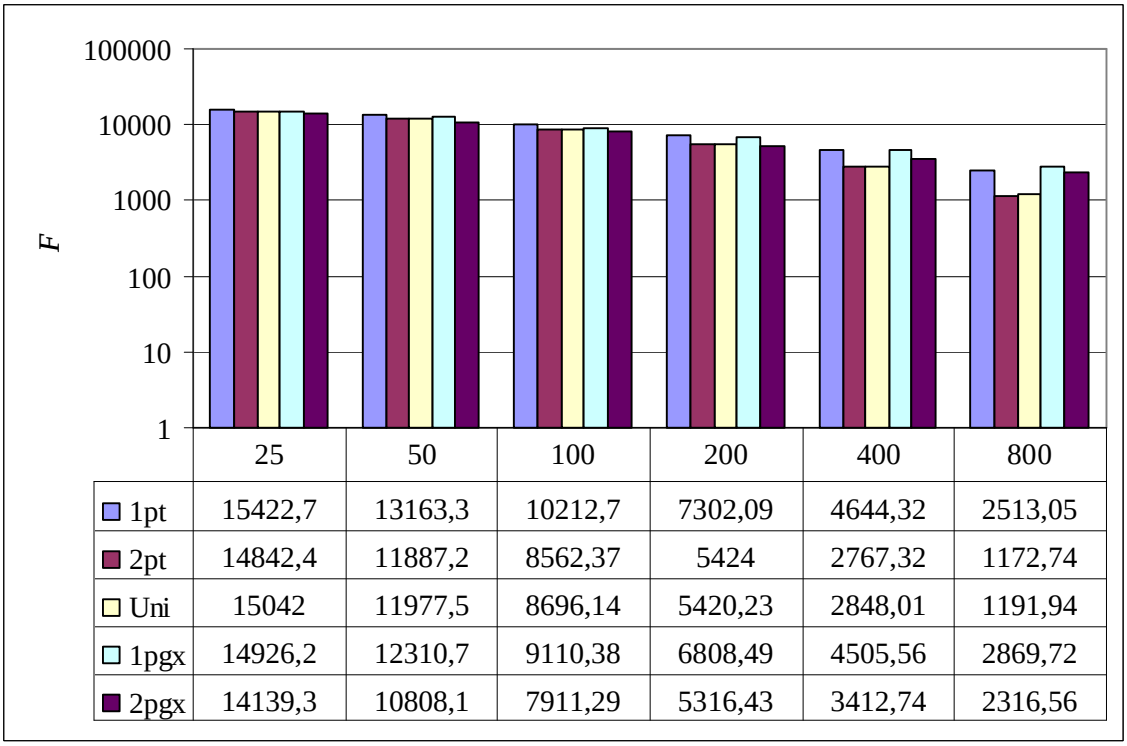
	25	50	100	200	400	800
1pt	2039,75	1250,61	679,037	259,576	75,5378	15,2279
2pt	1815,73	1018,07	480,99	150,575	27,328	4,73725
Uni	1789,69	1089,58	495,938	166,285	25,1553	4,70912
1pgx	2122,2	1411,57	883,565	502,598	249,332	125,705
2pgx	1861,27	1210,71	756,964	422,383	235,352	130,386

Рис. 2.17. Результаты для тестовой функции F_4 от 10 переменных. Обозначения те же, что и на рис. 2.8.



	25	50	100	200	400	800
1pt	8437,34	6660,34	4722,33	2971,94	1573,45	678,003
2pt	8158,03	6051,8	3783,8	2008,31	827,45	243,519
Uni	8099,4	6007,25	3758,98	2049,89	863,587	218,669
1pgx	8229,91	6411,84	4567,09	3104,87	1982,65	1202,53
2pgx	7408,99	5612,1	3859,33	2541,3	1662,52	1205,72

Рис. 2.18. Результаты для тестовой функции F_4 от 30 переменных. Обозначения те же, что и на рис. 2.8.



	25	50	100	200	400	800
1pt	15422,7	13163,3	10212,7	7302,09	4644,32	2513,05
2pt	14842,4	11887,2	8562,37	5424	2767,32	1172,74
Uni	15042	11977,5	8696,14	5420,23	2848,01	1191,94
1pgx	14926,2	12310,7	9110,38	6808,49	4505,56	2869,72
2pgx	14139,3	10808,1	7911,29	5316,43	3412,74	2316,56

Рис. 2.19. Результаты для тестовой функции F_4 от 50 переменных. Обозначения те же, что и на рис. 2.8.

По данным исследований (рис. 2.8-2.19) видно, что результаты работы генного ОК улучшаются с ростом размерности рассматриваемой задачи. Например, для показанных на рис. 2.17-2.19 диаграмм, результаты генного ОК постепенно улучшаются с увеличением числа переменных, при этом для функции 50 переменных и популяций малого и среднего размера (25, 50, 100 и 200 особей) 2-точечный генный ОК превосходит 1-, 2-точечный и однородный ОК, а для популяций большего размера (400 и 800 особей) отставание генных ОК значительно сокращается по сравнению со случаем функции F_4 10 переменных (рис. 2.17). Аналогичные наблюдения можно получить и для остальных функций.

В целом отметим, что результаты работы 2-точечного генного ОК, как правило, превосходят результаты 1-точечного генного кроссинговера.

Также заметим, что для функций F_2, F_3 и F_4 уменьшение размера популяции меньше влияет на результаты генного ОК по сравнению с 1- и 2-точечным и однородным ОК. Так, для функции F_4 от 10 переменных (рис. 2.17) и с размером популяции равным 800 особей видно, что результаты генного ОК хуже результатов «классических» ОК на 1-2 порядка, но уменьшение популяции до 25 особей приводит к тому, что результаты генного ОК по величине становятся сопоставимыми с результатами сравниваемых ОК.

Для более подробной оценки влияния размера популяции на результат работы различных ОК рассмотрим следующую величину, отражающую относительную эффективность увеличения размера популяции:

$$d = \frac{F_{N=N_1}}{F_{N=N_2}} \frac{N_2}{N_1},$$

где $F_{N=N_i}$ – значение целевой функции, к которому сходится ГА при размере N популяции равном N_i. Значения d при $N_1 = 800$, $N_2 = 32$ для различных ОК и используемых целевых функций представлены в табл. 2.3. В скобках в табл. 2.3. показано изменение значений d в процентах относительно случая одноименной функции 10 переменных. Заметим, что сравнивать напрямую

значения d для каждой из функций F_1, F_2, F_3 и F_4 некорректно, так как рассматриваемые подобласти областей определения для каждой из функций различны, также как и значения, принимаемые этими функциями, на них. Поэтому нас будет интересовать изменение d для различных ОК в зависимости от числа переменных у рассматриваемых тестовых функций в отдельности.

Табл. 2.3. Значения d для сравниваемых ОК и используемых тестовых функций при $N_1 = 800$, $N_2 = 32$

Функция	Операторы кроссинговера				
	1-точечный	2-точечный	однородный	1-точ. генный	2-точ. генный
F_1					
10 переменных	39,82 (100 %)	187,226 (100 %)	207,867 (100 %)	158,164 (100 %)	1191,572 (100 %)
30 переменных	1,653 (4,15 %)	6,506 (3,47 %)	6,183 (2,97 %)	17,314 (10,95 %)	732,243 (61,45 %)
50 переменных	0,69 (1,73 %)	1,98 (1,06 %)	1,952 (0,94 %)	15,364 (9,71 %)	123,138 (10,33 %)
F_2					
10 переменных	1,07 (100 %)	1,82 (100 %)	1,96 (100 %)	0,75 (100 %)	0,4 (100 %)
30 переменных	0,318 (29,72 %)	0,567 (31,15 %)	0,541 (27,60 %)	0,465 (62 %)	0,396 (99 %)
50 переменных	0,196 (18,32 %)	0,334 (18,35 %)	0,333 (16,99 %)	0,397 (52,93 %)	0,422 (105,50 %)
F_3					
10 переменных	2,032 (100 %)	1,818 (100 %)	1,884 (100 %)	1,817 (100 %)	1,222 (100 %)
30 переменных	1,247 (61,37 %)	1,939 (106,66 %)	1,639 (87,00 %)	1,794 (98,73 %)	2,141 (175,20 %)
50 переменных	0,83 (40,85 %)	1,38 (75,91 %)	1,41 (74,84 %)	1,32 (72,65 %)	2,48 (202,95 %)
F_4					
10 переменных	4,186 (100%)	11,978 (100%)	11,876 (100%)	0,528 (100%)	0,446 (100%)
30 переменных	0,389 (9,29 %)	1,0469 (8,74 %)	1,157 (9,74 %)	0,214 (40,53 %)	0,192 (43,05 %)
50 переменных	0,192 (4,59 %)	0,396 (3,31 %)	0,394 (3,32 %)	0,163 (30,87 %)	0,191 (42,83 %)

Прежде всего, отметим, что чем больше значение d для одной и той же функции, тем, как правило, лучше полученные результаты, а близкие значения d для различных ОК соответствуют примерно одинаковым по качеству

результатам работы ГА. Таким образом, по данным табл. 2.3. можно примерно судить о результатах работы сравниваемых ОК. В частности, видно, что в случае малого числа переменных результаты генного кроссинговера оказываются хуже результатов 1-, 2-точечного и однородного ОК, но с ростом размерности задачи эффективность увеличения размера популяции для генного ОК убывает[1] значительно медленнее (показано в процентах), чем для «классических» операторов. Заметим, что данные наблюдения согласуются с приведенными выше результатами анализа диаграмм, представленных на рис. 2.8-2.19.

2.2.2. Выводы по исследованию эффективности генного ОК

В результате исследования эффективности генного ОК сделаем следующие выводы:

1. Результаты работы 2-точечного генного ОК для рассматриваемых тестовых функций F_1–F_4, как правило, лучше результатов 1-точечного генного кроссинговера. При этом 1-точечный генный ОК часто оказывается хуже однородного и 2-точечного ОК, в то время как использование 2-точечного генного кроссинговера во многих случаях дает лучшие результаты.

2. С увеличением размерности задачи относительная эффективность d увеличения размера популяции для генного кроссинговера убывает значительно медленнее, чем для 1-, 2-точечного и однородного ОК, что позволяет сделать вывод о лучшей масштабируемости генного ОК. Данная особенность генного кроссинговера является особенно полезной с учетом того, что в предлагаемом НЭ алгоритме (см. Главу 3) рассматриваются ИНС с постепенно увеличивающейся структурной сложностью, когда число генов в хромосоме также растет и, следовательно, потеря эффективности при использовании генного ОК будет меньше, чем, например, при использовании однородного кроссинговера.

---

[1] А для случаев использования 2-точечного генного ОК и функций F_2 и F_3 значение d возрастает.

3. Для генного ОК предпочтительно использовать популяции малого и среднего размера. При повышении размера популяции (как правило, 200 особей и больше) результаты работы 2-точечного и однородного ОК нередко оказываются значительно лучше результатов генного ОК. Отметим, что возможность эффективной работы с меньшим размером популяции приводит к уменьшению требований к объему оперативной памяти, необходимой для работы ГА, и поэтому рассматривается как преимущество.

2.3. Адаптация размера популяции

Адаптация параметров в ЭА, необходимая для повышения качества результатов работы и уменьшения необходимых ресурсов и вычислительных затрат, является одной из поставленных в Главе 1 целей при разработке НЭ алгоритма. При этом адаптивно настраиваться могут различные параметры ЭА, например, параметры кодирования, ОК, мутации, селекции, размер популяции.

В п. 2.2 говорится об адаптации разрушающих способностей генного ОК к росту размерности пространства поиска. В Главе 3 будут описаны генетические операторы скрещивания и мутации, адаптирующиеся к особенностям ИНС, а также будет представлена адаптация вероятности мутации для каждой особи в зависимости от количества межнейронных связей в соответствующей ИНС. В данном разделе будет рассмотрена адаптация размера популяции.

Известно, что малый размер популяции отрицательно сказывается на разнообразии хромосом, что уменьшает эффективность исследования пространства поиска, существенно увеличивает количество поколений эволюционного поиска, часто приводит к ухудшению результатов работы и может стать причиной преждевременной сходимости ГА. Увеличение размера популяции, как правило, частично или полностью решает указанные проблемы, но в ряде случаев приводит к значительному росту количества вычислений целевой функции, что увеличивает сложность ГА и время его работы.

В настоящее время существуют различные подходы к выбору размера популяции, которые можно разбить на три группы:

1. Экспериментальные подходы без адаптации.

2. Подходы, использующие информацию о характеристиках пространства поиска и свойствах целевой функции.

3. Адаптивные подходы.

В подходах первой группы, как правило, рассматриваются результаты работы ГА на наборе разнообразных тестовых функций, и по результатам экспериментов определяется рекомендуемое значение размера популяции, неизменяемое в течение работы ГА [81, 116]. Также отметим подход [78], в котором зависимость величины размера популяции от численных параметров ГА и задачи (размерность пространства поиска, интенсивность селективного отбора и др.) подбирается на основании данных экспериментальных исследований по выявлению зависимости между размером популяции и рассматриваемыми параметрами.

Для подходов второй группы характерен анализ пространства поиска на основе данных о целевой функции F. При этом может предполагаться [109, 121], что функция F декомпозируема, причем можно выделить «главный» параметр x_0, наиболее значимо влияющий на приспособленность особей. Вклад в приспособленность особи остальных параметров рассматривается как аддитивный шум, проявляющийся при вычислении значения приспособленности и влияющий на вероятность селективного отбора особей.

Подходы, рассматриваемые в третьей группе [56, 61, 90, 93, 122, 231, 190], используют ряд правил для адаптивной подстройки размера популяции во время работы ГА в зависимости от характеристик эволюционного поиска. Для этого часто рассматривают изменение во времени средней либо лучшей приспособленности, и размер популяции изменяют таким образом, чтобы обеспечить баланс между ростом приспособленности, с одной стороны, и как можно меньшим общим количеством вычислений целевой функции, с дру-

гой. Также достаточно распространен способ изменения размера популяции на основе использования дополнительной характеристики – времени жизни (lifetime) особи, которая вычисляется в зависимости от приспособленности особи в момент ее первого появления в популяции [56, 61, 93].

Отметим, что результаты применения различных стратегий выбора/адаптации размера популяции зависят от поставленной задачи и выбора параметров ЭА [93] и выявить однозначно лучший подход в настоящее время не представляется возможным.

Подходы, относящиеся к третьей группе, являются наиболее гибкими и представляют наибольший интерес с теоретической и практической точек зрения. Однако до сих пор не известно как влияет изменение размера популяции на характеристики и результат эволюционного поиска. В связи с этим решение задачи адаптации размера популяции целесообразно разбить на два этапа:

1. Исследование влияния изменения размера популяции на характеристики эволюционного поиска.

2. Разработка стратегии адаптации размера популяции на основе результатов, полученных на первом этапе.

2.3.1. Исследование влияния изменения размера популяции на характеристики эволюционного поиска

Для решения задачи исследования влияния изменения размера популяции на эволюционный поиск будем проводить запуски ГА, изменяя количество особей в популяции в соответствии с одной из следующих стратегий:

- «+1» стратегия – популяция увеличивается на 1 особь в каждом поколении. Будем рассматривать запуски с начальным размером популяции $N_0^+ \in \{10,25,50,100\}$ и обозначим соответствующие стратегии изменения размера популяции как «10+1», «25+1», «50+1», «100+1»;
- «-1» стратегия – популяция в каждом поколении уменьшается на 1 особь. Будем рассматривать запуски с начальным размером популя-

ции $N_0^- \in \{25,50,100,200\}$ и обозначим соответствующие стратегии изменения размера популяции как «25-1», «50-1», «100-1», «200-1»;

Размер популяции во всех случаях ограничен снизу 10 и сверху 200 особями. Выбор наибольшего и наименьшего значений размера популяции обусловлен тем, что в популяции меньше 10 особей генетическое разнообразие будет малым для обеспечения эффективного поиска, а использование популяции более 200 особей значительно увеличивает общее количество вычислений целевой функции, снижая тем самым практическую ценность результата.

Будем рассматривать результаты работы простого ГА с турнирным отбором, 1- и 2-точечным генным ОК и побитовой мутацией. Значения параметров ГА приведены в табл. 2.4. Для тестирования выбраны функции, использованные ранее для исследования эффективности различных ОК, и представленные в табл. 2.2.

Табл. 2.4. Условия экспериментов. L – длина хромосомы в битах

Параметр	**Значение**
Селекция	Турнирная
Размер турнира	2
Вероятность кроссинговера, P_c	0,8
Вероятность мутации, P_m	$1/L$
Количество поколений	1000
Количество запусков	100

Для сравнения результатов экспериментов также будем рассматривать результаты работы ГА с фиксированным размером популяции $N \in \{10,25,50,100,200\}$.

Для оценки результатов будем рассматривать усредненное лучшее значение целевой функции найденное ГА. Отметим, что целью данного иссле-

дования является анализ влияния изменения размера популяции на результат работы ГА, но не сравнение 1- и 2-точечного генного ОК, так как такое сравнение требует предварительной настройки параметров ГА, необходимой для раскрытия возможностей этих операторов.

Пример типичного изменения лучшей приспособленности F_{best} в популяции для тестовой функции F_4 показан на рис. 2.20. Диаграммы для значений лучшей приспособленности, полученных в результате 100 запусков, представлены на рис. 2.21-2.24.

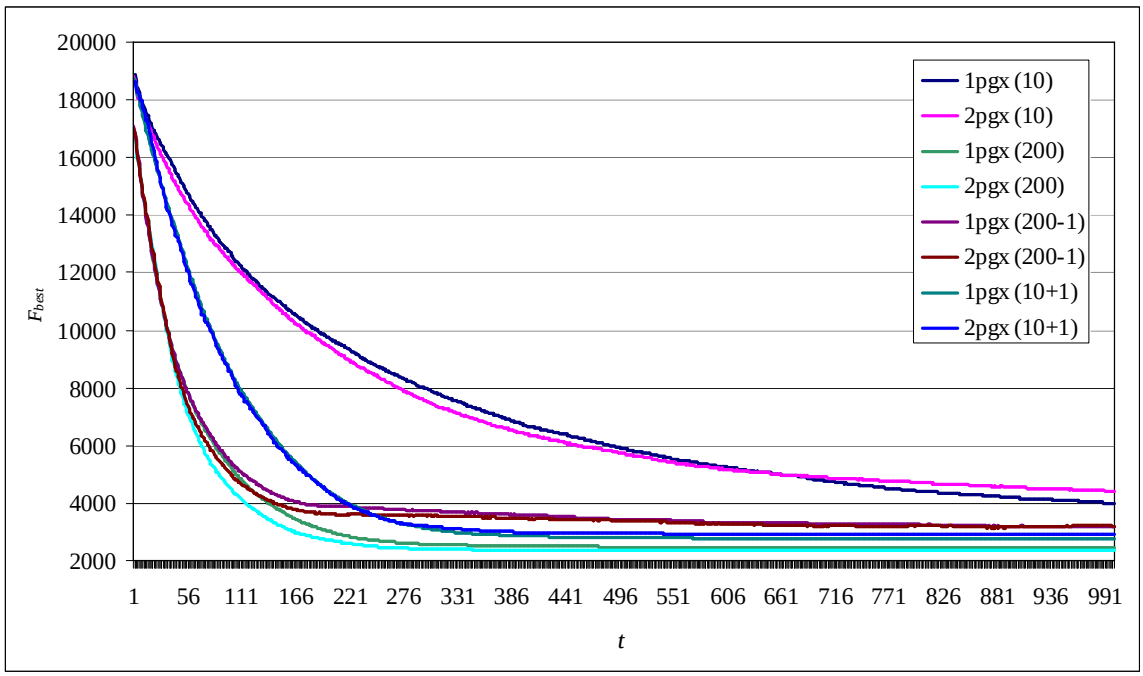
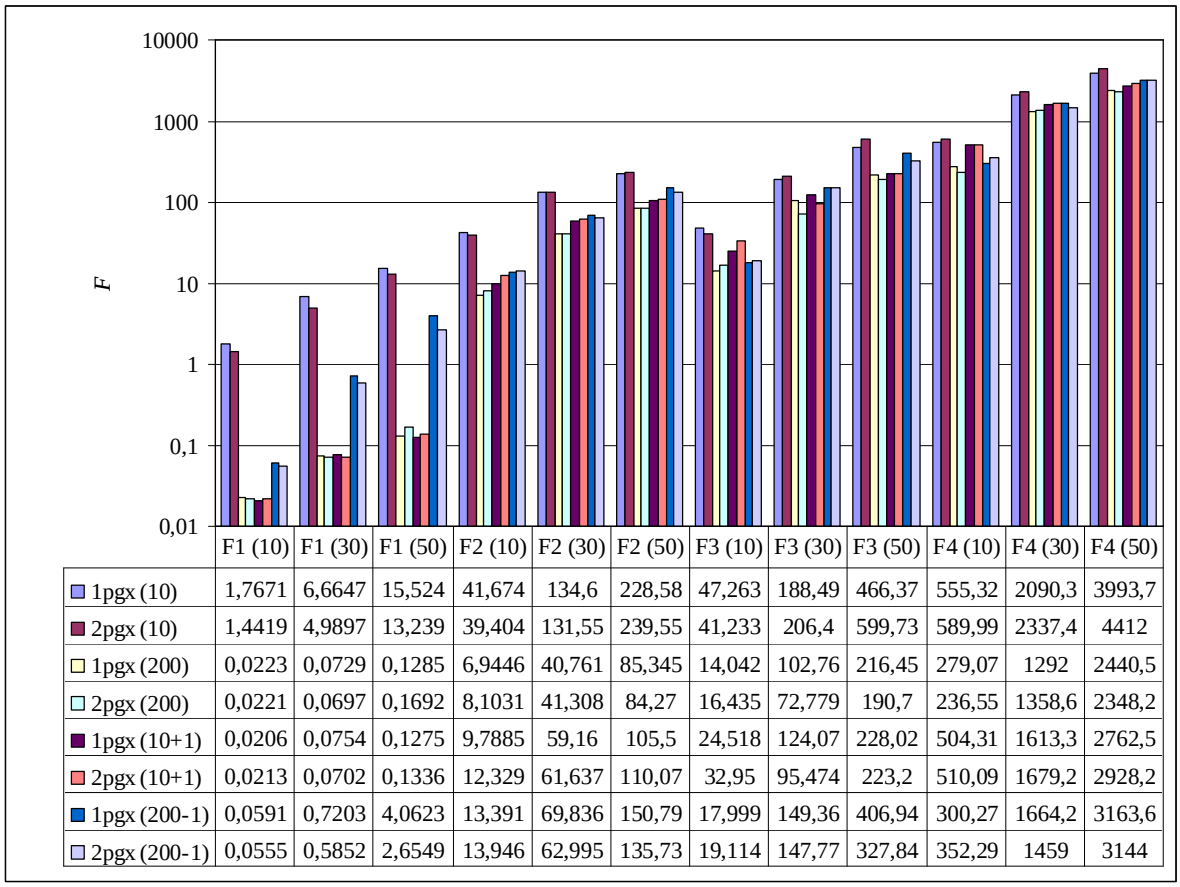


Рис. 2.20. Изменение лучшей приспособленности F_{best} для тестовой функции F_4 в зависимости от номера поколения t. Усреднено по 100 запускам

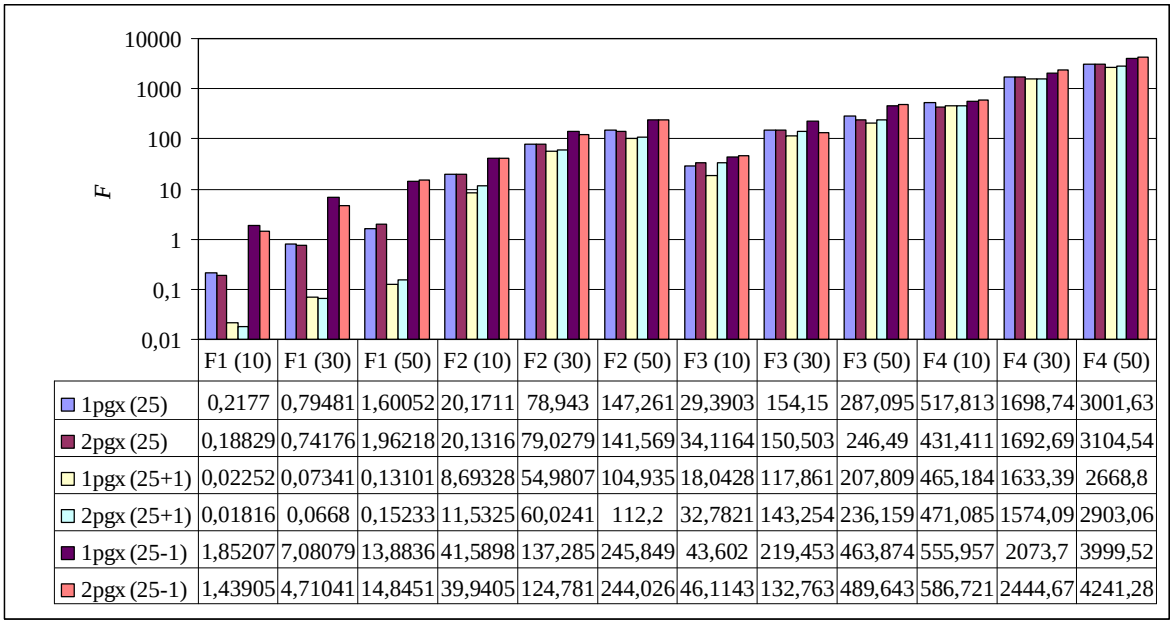
Проанализируем полученные результаты. Большой начальный размер популяции является преимуществом, так как в этом случае изначально рассматривается большая область в пространстве поиска. Это позволяет постепенно уменьшать размер популяции без потери эффективности, однако если уменьшение популяции превосходит некоторый порог, то скорость улучшения приспособленности падает, и дальнейшее уменьшение размера популяции приводит к ранней сходимости ГА со слабым дрейфом, обусловленным

работой операторов кроссинговера и мутации, в сторону областей в пространстве поиска с высокой приспособленностью (см. для примера рис. 2.20, графики для стратегии «200-1»). В случае небольшого начального размера популяции скорость улучшения лучшей приспособленности также невелика. Но постепенное увеличение размера популяции приводит в результате к повышению качества получаемого результата.



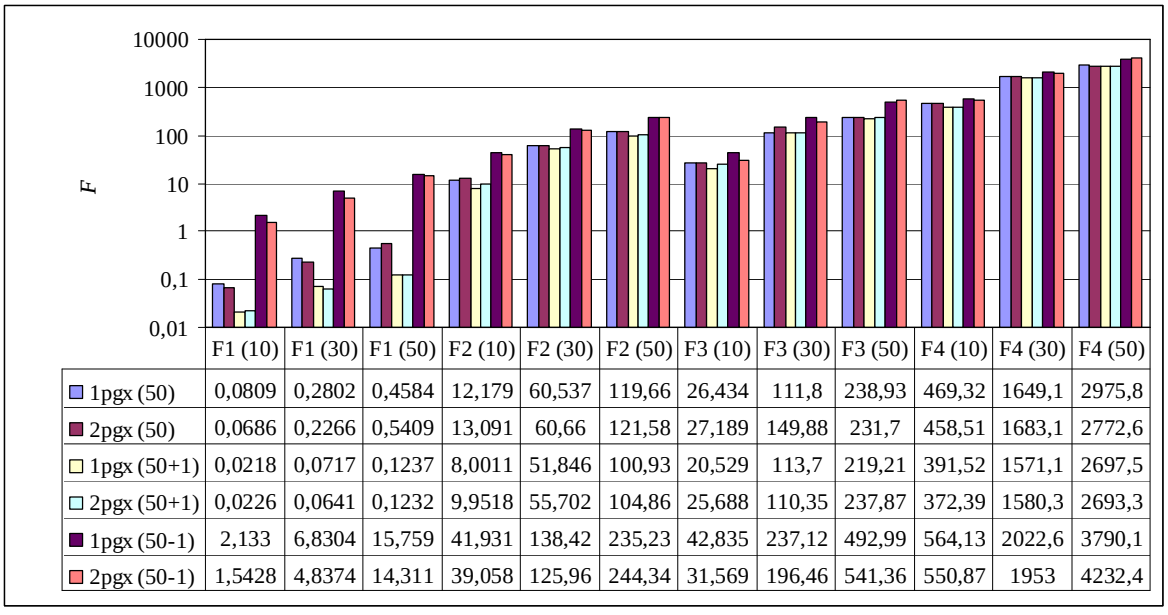
	F1 (10)	F1 (30)	F1 (50)	F2 (10)	F2 (30)	F2 (50)	F3 (10)	F3 (30)	F3 (50)	F4 (10)	F4 (30)	F4 (50)
1pgx (10)	1,7671	6,6647	15,524	41,674	134,6	228,58	47,263	188,49	466,37	555,32	2090,3	3993,7
2pgx (10)	1,4419	4,9897	13,239	39,404	131,55	239,55	41,233	206,4	599,73	589,99	2337,4	4412
1pgx (200)	0,0223	0,0729	0,1285	6,9446	40,761	85,345	14,042	102,76	216,45	279,07	1292	2440,5
2pgx (200)	0,0221	0,0697	0,1692	8,1031	41,308	84,27	16,435	72,779	190,7	236,55	1358,6	2348,2
1pgx (10+1)	0,0206	0,0754	0,1275	9,7885	59,16	105,5	24,518	124,07	228,02	504,31	1613,3	2762,5
2pgx (10+1)	0,0213	0,0702	0,1336	12,329	61,637	110,07	32,95	95,474	223,2	510,09	1679,2	2928,2
1pgx (200-1)	0,0591	0,7203	4,0623	13,391	69,836	150,79	17,999	149,36	406,94	300,27	1664,2	3163,6
2pgx (200-1)	0,0555	0,5852	2,6549	13,946	62,995	135,73	19,114	147,77	327,84	352,29	1459	3144

Рис. 2.21. Значения лучшей приспособленности для стратегий «10», «10+1», «200» и «200-1» на тестовых функциях 10, 30 и 50 переменных (указано в скобках рядом с обозначением функции)



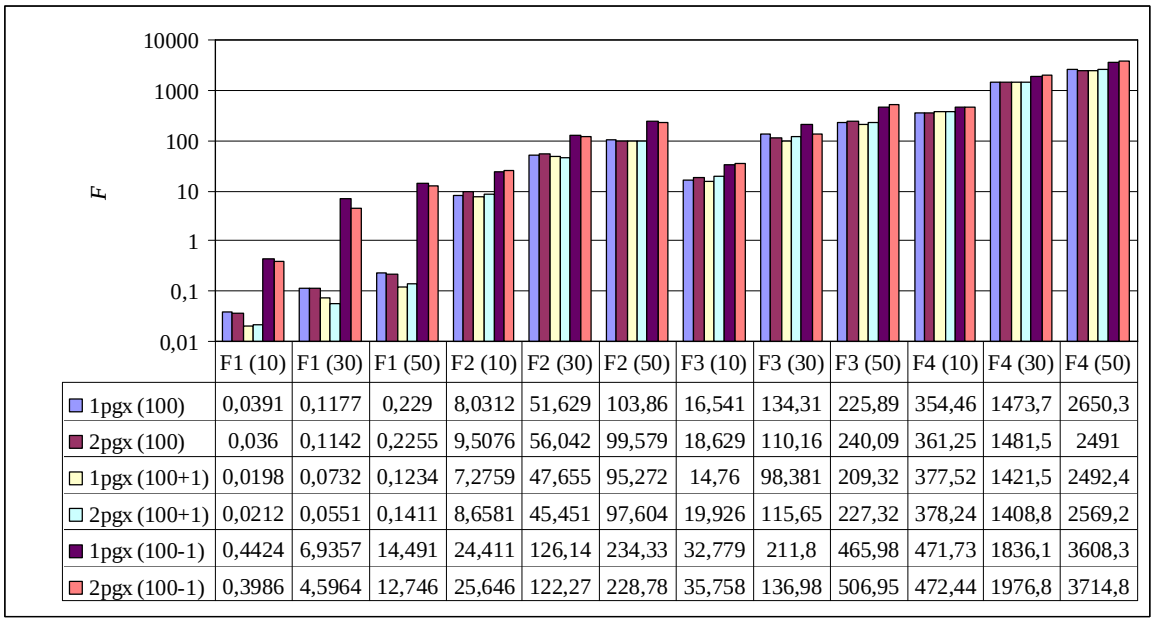
	F1 (10)	F1 (30)	F1 (50)	F2 (10)	F2 (30)	F2 (50)	F3 (10)	F3 (30)	F3 (50)	F4 (10)	F4 (30)	F4 (50)
1pgx (25)	0,2177	0,79481	1,60052	20,1711	78,943	147,261	29,3903	154,15	287,095	517,813	1698,74	3001,63
2pgx (25)	0,18829	0,74176	1,96218	20,1316	79,0279	141,569	34,1164	150,503	246,49	431,411	1692,69	3104,54
1pgx (25+1)	0,02252	0,07341	0,13101	8,69328	54,9807	104,935	18,0428	117,861	207,809	465,184	1633,39	2668,8
2pgx (25+1)	0,01816	0,0668	0,15233	11,5325	60,0241	112,2	32,7821	143,254	236,159	471,085	1574,09	2903,06
1pgx (25-1)	1,85207	7,08079	13,8836	41,5898	137,285	245,849	43,602	219,453	463,874	555,957	2073,7	3999,52
2pgx (25-1)	1,43905	4,71041	14,8451	39,9405	124,781	244,026	46,1143	132,763	489,643	586,721	2444,67	4241,28

Рис. 2.22. Значения лучшей приспособленности для стратегий «25», «25+1» и «25-1» на тестовых функциях 10, 30 и 50 переменных (указано в скобках рядом с обозначением функции)



	F1 (10)	F1 (30)	F1 (50)	F2 (10)	F2 (30)	F2 (50)	F3 (10)	F3 (30)	F3 (50)	F4 (10)	F4 (30)	F4 (50)
1pgx (50)	0,0809	0,2802	0,4584	12,179	60,537	119,66	26,434	111,8	238,93	469,32	1649,1	2975,8
2pgx (50)	0,0686	0,2266	0,5409	13,091	60,66	121,58	27,189	149,88	231,7	458,51	1683,1	2772,6
1pgx (50+1)	0,0218	0,0717	0,1237	8,0011	51,846	100,93	20,529	113,7	219,21	391,52	1571,1	2697,5
2pgx (50+1)	0,0226	0,0641	0,1232	9,9518	55,702	104,86	25,688	110,35	237,87	372,39	1580,3	2693,3
1pgx (50-1)	2,133	6,8304	15,759	41,931	138,42	235,23	42,835	237,12	492,99	564,13	2022,6	3790,1
2pgx (50-1)	1,5428	4,8374	14,311	39,058	125,96	244,34	31,569	196,46	541,36	550,87	1953	4232,4

Рис. 2.23. Значения лучшей приспособленности для стратегий «50», «50+1» и «50-1» на тестовых функциях 10, 30 и 50 переменных (указано в скобках рядом с обозначением функции)



	F1 (10)	F1 (30)	F1 (50)	F2 (10)	F2 (30)	F2 (50)	F3 (10)	F3 (30)	F3 (50)	F4 (10)	F4 (30)	F4 (50)
1pgx (100)	0,0391	0,1177	0,229	8,0312	51,629	103,86	16,541	134,31	225,89	354,46	1473,7	2650,3
2pgx (100)	0,036	0,1142	0,2255	9,5076	56,042	99,579	18,629	110,16	240,09	361,25	1481,5	2491
1pgx (100+1)	0,0198	0,0732	0,1234	7,2759	47,655	95,272	14,76	98,381	209,32	377,52	1421,5	2492,4
2pgx (100+1)	0,0212	0,0551	0,1411	8,6581	45,451	97,604	19,926	115,65	227,32	378,24	1408,8	2569,2
1pgx (100-1)	0,4424	6,9357	14,491	24,411	126,14	234,33	32,779	211,8	465,98	471,73	1836,1	3608,3
2pgx (100-1)	0,3986	4,5964	12,746	25,646	122,27	228,78	35,758	136,98	506,95	472,44	1976,8	3714,8

Рис. 2.24. Значения лучшей приспособленности для стратегий «100», «100+1» и «100-1» на тестовых функциях 10, 30 и 50 переменных (указано в скобках рядом с обозначением функции)

Отметим, что с ростом начального размера популяции N_0^+ использование «+1» стратегии в ряде случаев не приводит к существенному улучшению результатов (см. результаты для стратегии «100+1» на рис. 2.24), при этом количество вычислений целевой функции увеличивается практически вдвое по сравнению с ГА с фиксированным размером популяции $N = 100$. Тем не менее, для этапов эволюционного поиска, отличных от начального, увеличение размера популяции способствует улучшению результата (рис. 2.20).

Итак, из результатов экспериментов следует, что популяцию достаточно большого размера можно уменьшать до некоторого (в общем случае, неопределенного) порогового значения без значительного ущерба для эффективности эволюционного поиска, а популяцию малого размера необходимо увеличивать для улучшения результатов. Однако ввиду использования «+1» и «-1» стратегий неясно, когда следует изменять размер популяции.

Поскольку при решении практических задач наибольшее значение имеет лучшая особь, найденная в результате эволюционного поиска, будем считать, что изменение размера популяции целесообразно производить на основании

динамики лучшей приспособленности в популяции. Для этого предложим следующую «+/-» стратегию изменения размера популяции.

Размер популяции необходимо увеличить, если приспособленность лучшей особи в популяции уменьшается, либо не изменяется, то есть отсутствует прогресс. В случае если приспособленность лучшей особи увеличивается, другими словами, наблюдается эволюционное улучшение, то размер популяции следует уменьшить.

2.3.2. Стратегия адаптации размера популяции

При разработке стратегии адаптации размера популяции необходимо выбрать следующие параметры:

- минимальный $N_{\min}$ и максимальный $N_{\max}$ размер популяции;

- величину Δ_N, на которую изменяется размер популяции;

- условия изменения размера популяции;

- начальный размер популяции.

В качестве верхней границы N возьмем $N_{\max} = 200$, так как дельнейшее увеличение размера популяции нецелесообразно ввиду увеличения количества вычислений целевой функции. Нижнюю границу N будем определять в зависимости от длины L хромосомы в битах: чем больше L, тем больше значение $N_{\min}$. Для определения $N_{\min}$ вычислим размер популяции достаточный для того, чтобы в популяции хромосом длины L после случайной инициализации в каждом разряде встречались в равном количестве как «0», так и «1», что необходимо для обеспечения эффективного эволюционного поиска.

Заметим, что две случайно сгенерированные хромосомы будут отличаться в среднем на $L/2$ разрядов, или, другими словами, расстояние по Хэммингу между этими хромосомами в среднем равно $L/2$. Случайно созданная третья хромосома будет в среднем отличаться от каждой из двух существующих также на $L/2$ разрядов, причем количество совпадающих во всех трех хромосомах разрядов будет равно $L/4$, а количество разрядов, содержащих и «0», и

«1», в трех хромосомах будет в среднем равно $3L/4$. С добавлением случайных четвертой, пятой, шестой и т.д. хромосом, количество разрядов в популяции, содержащих как «0», так и «1», будет постепенно увеличиваться на $L/8$, $L/16$, $L/32$,… до тех пор, пока с добавлением $(2+[\log_2 L])$ хромосомы[1] в каждом разряде популяции не будет присутствовать и «0», и «1» (здесь операция «[·]» означает взятие целой части). В полученной таким образом популяции, пример которой приведен на рис. 2.25а, распределение в каждой позиции «0» и «1» будет неравномерным и для того, чтобы уравнять количество «0» и «1» в популяции добавим к существующим хромосомам еще $(2+[\log_2 L])$ хромосом, полученных инверсией уже сгенерированных (рис. 2.25б). Таким образом, будем вычислять нижнюю границу размера популяции как:

$$N_{\min} = 2\big(2+[\log_2 L]\big).$$

Величину Δ_N, на которую изменяется размер популяции, будем определять таким образом, чтобы в соответствии с результатами исследования в п. 2.3.1, выполнялись следующие условия:

1. Чем в большем количестве последовательных поколений поддерживается улучшение (ухудшение либо стабилизация) лучшей приспособленности F_{best}, тем сильнее следует уменьшать (увеличивать) размер N популяции. Здесь предполагается, что долговременное отсутствие изменения $u(t)$ свидетельствует о том, что существующего размера популяции явно достаточно (недостаточно) для эффективного эволюционного поиска.

2. Размер N популяции не изменяется при колебаниях лучшей приспособленности F_{best}, когда неясно, следует ли увеличивать, либо уменьшать значение N.

---

[1] Здесь учитывается, что, после генерации первой хромосомы, в популяции нет ни одного разряда, в котором содержатся и «0», и «1», а также то, что при $L = 2^k$, $k \in \mathbf{N}$, случайно генерируемые хромосомы с номерами $(1+[\log_2 L])$ и $(2+[\log_2 L])$ увеличивают количество разрядов, содержащих как «0», так и «1», на 1.

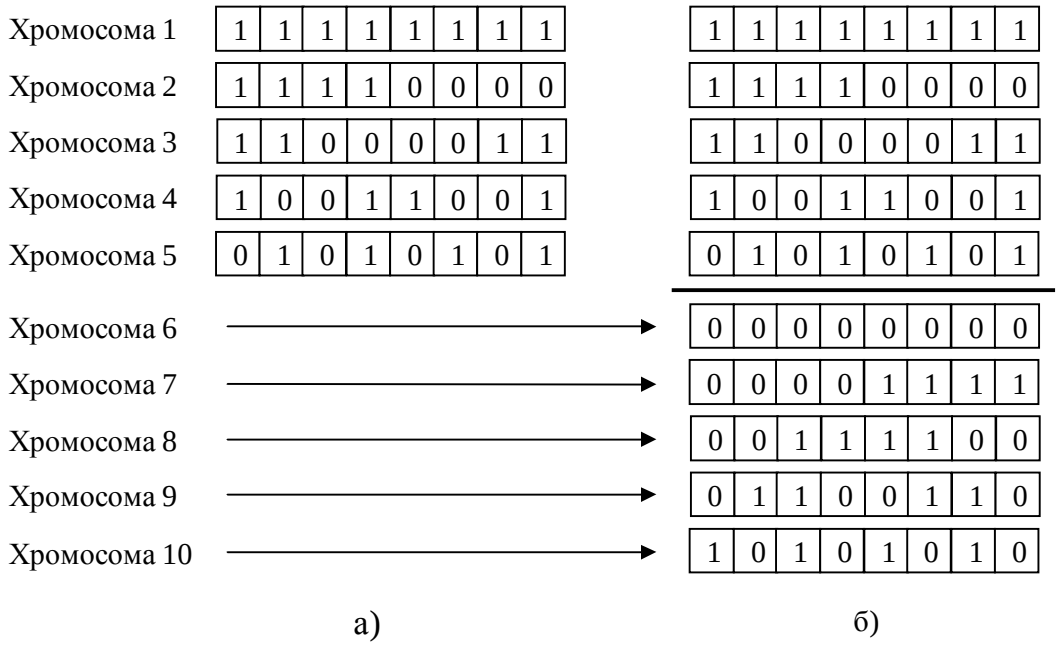


Рис. 2.25. Пример популяции, в которой во всех разрядах присутствуют как «0», так и «1» (а) и полученной из нее популяции с равномерно распределенными нулевыми и единичными разрядами (б). Другие подобные популяции могут быть получены из представленной путем перемены мест *i*-х и *j*-х разрядов во всей популяции

Для выполнения сформулированных выше условий будем определять Δ_N с использованием последовательности Фибоначчи: $a_n = a_{n-1} + a_{n-2}$, $a_0 = a_1 = 1$, – члены которой – целые числа, растущие в геометрической прогрессии, знаменатель которой при $n \to \infty$ стремится к $\frac{1+\sqrt{5}}{2} \approx 1{,}618$ [12]. Будем вычислять Δ_N в поколении t следующим образом:

$$\Delta_N(t) = \begin{cases} a_k, u(t) = u(t-1), \\ a_1, u(t) \neq u(t-1), \end{cases}$$

где $u(t)$ – направление изменения (увеличение или уменьшение) размера популяции в поколении t ; k – количество поколений, в течение которых направление u изменения размера популяции остается постоянным. Таким образом, использование последовательности Фибоначчи обеспечивает увеличение шага изменения размера популяции при неизменном $u(t)$. При этом при

частых изменениях $u(t)$ значение Δ_N будет малым (порядка единиц), что не приведет к значительным изменениям размера популяции. Кроме этого, в силу отсутствия необходимости в округлении Δ_N исчезает опасность последовательного накопления тем самым ошибки вычислений.

Итак, определена стратегия адаптации размера популяции с использованием последовательности Фибоначчи в зависимости от характеристик эволюционного поиска. При использовании этой стратегии будем учитывать тот факт, что эволюционный поиск можно условно разделить на два этапа [27, 77]:

1. Этап активного действия селективного отбора, характеризующийся быстрым улучшением приспособленности в популяции за счет отбрасывания слабо приспособленностью особей и исключения, таким образом, из рассмотрения областей с низкой приспособленностью в пространстве поиска.

2. Этап поиска за счет работы генетических операторов рекомбинации и мутации. На этом этапе скорость улучшения приспособленности существенно падает по сравнению с первым этапом.

Для повышения качества результатов работы ГА необходимо как можно дольше поддерживать первый этап эволюционного поиска, например, за счет генетического разнообразия в популяции для работы оператора селекции. В [108] показано, что порядок времени сходимости (в поколениях) для селекции не меньше $O(\ln N)$ и предполагая, что $L \sim N$ будем считать, что характерное время селекции имеет порядок $O(\ln L)$. Тогда в целях обеспечения условий для селективного отбора будем в течение первых $\ln L$ поколений только увеличивать размер популяции.

Таким образом, на основе полученных в п. 2.3.1 и текущем разделе выводов предлагается следующая стратегия адаптации размера популяции:

$$N(t+1) = N(t) + u(t)\Delta_N,$$

$$u(t)=\begin{cases}1, & F_{best}(t)\ \mathbf{p}\ F_{best}(t-1)\ \text{или}\ t<\ln L\\ -1, & F_{best}(t)\ \mathbf{f}\ F_{best}(t-1)\ \text{и}\ t\geq\ln L\end{cases}$$

где $N(t)$ – размер популяции в поколении t, а значение операций «$\mathbf{p}$» и «$\mathbf{f}$» зависит от типа решаемой задачи:

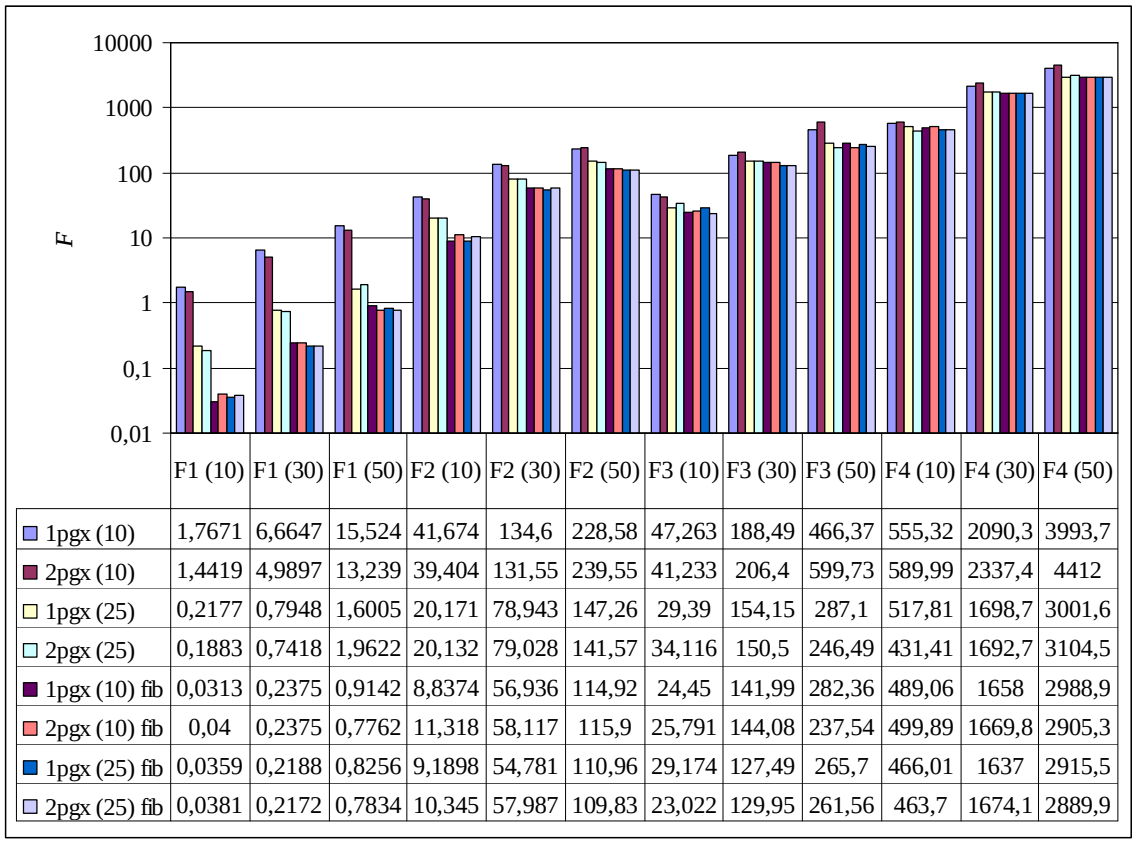
- при $F\to\min$ операции «$\mathbf{p}$» и «$\mathbf{f}$» эквивалентны соответственно «$\geq$» и «$<$»;
- при $F\to\max$ операции «$\mathbf{p}$» и «$\mathbf{f}$» эквивалентны соответственно «$\leq$» и «$>$».

Для определения начального размера N_0 популяции проведем серию экспериментов на тестовых функциях, представленных в табл. 2.2 и сравним результаты работы ГА, использующим адаптацию размера популяции, с результатами из п. 2.3.1. Будем рассматривать значения средней лучшей приспособленности, достигнутые в результате работы ГА.

Значения параметров ГА соответствуют данным табл. 2.3. Максимальный и минимальный размеры популяции равны соответственно $2\left(2+[\log_2 L]\right)$ и 200. Заметим, что намеренно используется стратегия селекции со слабым селективным давлением, для более объективной оценки эффекта от подстройки размера популяции. Результаты работы ГА с различными начальными размерами популяции и использованием предлагаемой стратегии адаптации размера популяции приведены на рис. 2.26-2.28. Также на рис. 2.26-2.28 для сравнения показаны результаты ГА с неизменным размером популяции.

Из приведенных на рис. 2.26-2.28 данных видно, что наибольшее улучшение качества результата (с точки зрения значения целевой функции) достигается при малом начальном размере популяции (10 и 25 особей, рис. 2.26) за счет динамического увеличения размера популяции. При увеличении начального размера популяции до среднего размера (50 и 100 особей, рис. 2.27) разница в результатах между ГА с постоянным размером популяции и ГА с подстройкой числа особей в популяции уменьшается. Однако в большинстве случаев (за исключением случая тестовой функции F_1 от 30 и 50 переменных)

использование адаптивной подстройки размера популяции позволяет получить результаты, которые сопоставимы, либо лучше результатов ГА без адаптации размера популяции.



	F1 (10)	F1 (30)	F1 (50)	F2 (10)	F2 (30)	F2 (50)	F3 (10)	F3 (30)	F3 (50)	F4 (10)	F4 (30)	F4 (50)
1pgx (10)	1,7671	6,6647	15,524	41,674	134,6	228,58	47,263	188,49	466,37	555,32	2090,3	3993,7
2pgx (10)	1,4419	4,9897	13,239	39,404	131,55	239,55	41,233	206,4	599,73	589,99	2337,4	4412
1pgx (25)	0,2177	0,7948	1,6005	20,171	78,943	147,26	29,39	154,15	287,1	517,81	1698,7	3001,6
2pgx (25)	0,1883	0,7418	1,9622	20,132	79,028	141,57	34,116	150,5	246,49	431,41	1692,7	3104,5
1pgx (10) fib	0,0313	0,2375	0,9142	8,8374	56,936	114,92	24,45	141,99	282,36	489,06	1658	2988,9
2pgx (10) fib	0,04	0,2375	0,7762	11,318	58,117	115,9	25,791	144,08	237,54	499,89	1669,8	2905,3
1pgx (25) fib	0,0359	0,2188	0,8256	9,1898	54,781	110,96	29,174	127,49	265,7	466,01	1637	2915,5
2pgx (25) fib	0,0381	0,2172	0,7834	10,345	57,987	109,83	23,022	129,95	261,56	463,7	1674,1	2889,9

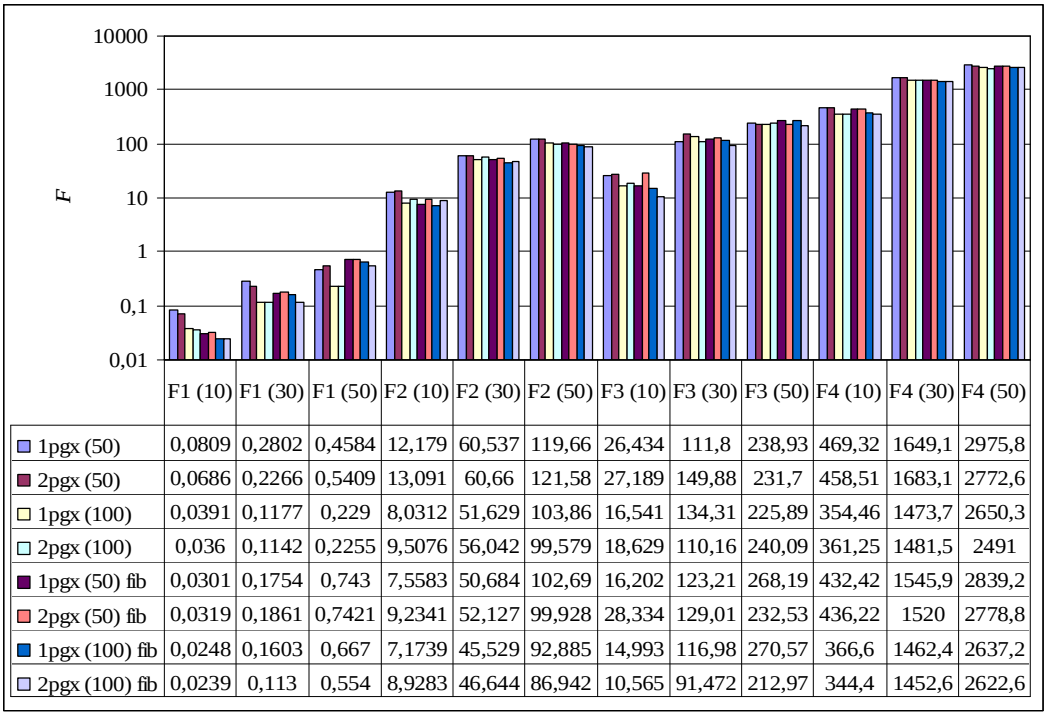
Рис. 2.26. Значения средней лучшей приспособленности для ГА с начальным размером популяции 10 и 25 особей с адаптацией размера популяции (обозначено как «fib») и без адаптации. Количество переменных указано в скобках рядом с обозначением тестовых функций

Отдельно отметим результаты для ГА с начальным размером 200 особей (рис. 2.28). Так как при выбранных значениях параметров ГА $N_{\max} = 200$, то в этом случае размер популяции не мог увеличиться больше начального размера. Однако приведенные на диаграмме на рис. 2.28 данные позволяют сделать вывод об эффективности использованной стратегии адаптации размера популяции, поскольку полученные результаты во многих случаях не уступают результатам работы ГА с большим размером популяции, при работе кото-

рого используется заведомо большее количество вычислений целевой функции. Это позволяет сделать следующее

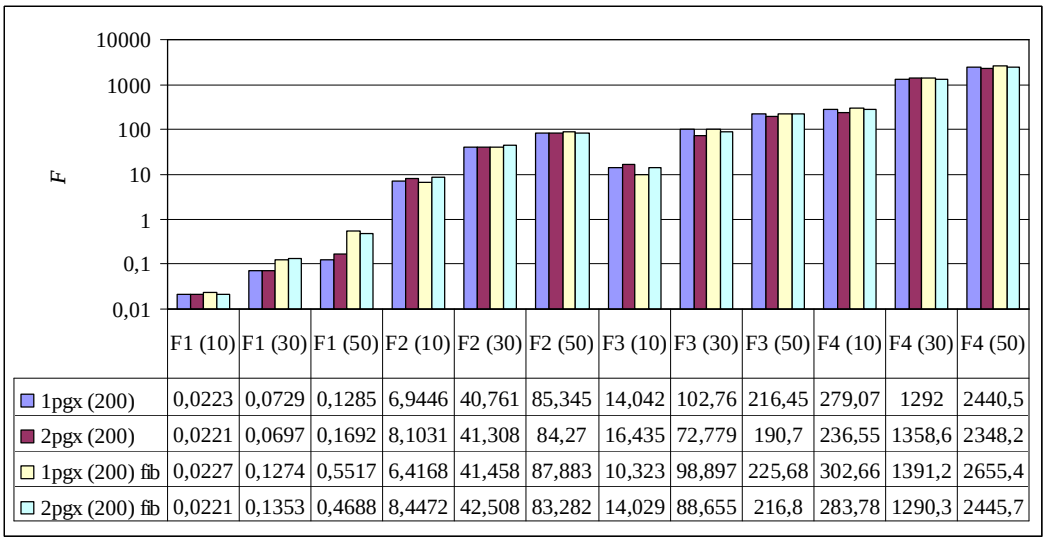
Предположение: Во многих случаях не требуется популяция, размер которой превышает 200 особей. При этом улучшение результатов работы ГА возможно за счет модификации схемы эволюционного поиска и используемых операторов.

В случае использования параллельных ГА [62], использующих несколько популяций, данное предположение может быть сделано для отдельной популяции. Отметим, что использованные настройки параметров ГА во многом соответствуют базовой модели простого ГА, эффективность которой может быть значительно увеличена за счет использования стратегий селекции с большим давлением и путем введения в популяцию элитных особей, что положительно отразится на результатах предложенной стратегии адаптации размера популяции.



	F1 (10)	F1 (30)	F1 (50)	F2 (10)	F2 (30)	F2 (50)	F3 (10)	F3 (30)	F3 (50)	F4 (10)	F4 (30)	F4 (50)
1pgx (50)	0,0809	0,2802	0,4584	12,179	60,537	119,66	26,434	111,8	238,93	469,32	1649,1	2975,8
2pgx (50)	0,0686	0,2266	0,5409	13,091	60,66	121,58	27,189	149,88	231,7	458,51	1683,1	2772,6
1pgx (100)	0,0391	0,1177	0,229	8,0312	51,629	103,86	16,541	134,31	225,89	354,46	1473,7	2650,3
2pgx (100)	0,036	0,1142	0,2255	9,5076	56,042	99,579	18,629	110,16	240,09	361,25	1481,5	2491
1pgx (50) fib	0,0301	0,1754	0,743	7,5583	50,684	102,69	16,202	123,21	268,19	432,42	1545,9	2839,2
2pgx (50) fib	0,0319	0,1861	0,7421	9,2341	52,127	99,928	28,334	129,01	232,53	436,22	1520	2778,8
1pgx (100) fib	0,0248	0,1603	0,667	7,1739	45,529	92,885	14,993	116,98	270,57	366,6	1462,4	2637,2
2pgx (100) fib	0,0239	0,113	0,554	8,9283	46,644	86,942	10,565	91,472	212,97	344,4	1452,6	2622,6

Рис. 2.27. Значения средней лучшей приспособленности для ГА с начальным размером популяции 50 и 100 особей с адаптацией размера популяции (обозначено как «fib») и без адаптации. Количество переменных указано в скобках рядом с обозначением тестовых функций



	F1 (10)	F1 (30)	F1 (50)	F2 (10)	F2 (30)	F2 (50)	F3 (10)	F3 (30)	F3 (50)	F4 (10)	F4 (30)	F4 (50)
1pgx (200)	0,0223	0,0729	0,1285	6,9446	40,761	85,345	14,042	102,76	216,45	279,07	1292	2440,5
2pgx (200)	0,0221	0,0697	0,1692	8,1031	41,308	84,27	16,435	72,779	190,7	236,55	1358,6	2348,2
1pgx (200) fib	0,0227	0,1274	0,5517	6,4168	41,458	87,883	10,323	98,897	225,68	302,66	1391,2	2655,4
2pgx (200) fib	0,0221	0,1353	0,4688	8,4472	42,508	83,282	14,029	88,655	216,8	283,78	1290,3	2445,7

Рис. 2.28. Значения средней лучшей приспособленности для ГА с начальным размером популяции 100 и 200 особей с адаптацией размера популяции (обозначено как «fib») и без адаптации. Количество переменных указано в скобках рядом с обозначением тестовых функций

Пример изменения размера популяции с $N_0 = 100$ для тестовой функции F_2 от 50 переменных, а также динамика средней лучшей приспособленности в популяции показаны на рис. 2.29. Видно, что пока скорость изменения средней лучшей приспособленности F_{best} достаточно велика, размер популяции постепенно уменьшается, но при значительном замедлении улучшения популяция начинает постепенно увеличиваться.

2.4. Основные результаты и выводы по главе

1. Представлен новый вычислительно менее сложный способ вычисления оценок времени смешивания для генетических операторов скрещивания, работающих с хромосомами с целочисленным кодированием. Результаты вычисления времени смешивания совпадают по порядку с известными результатами и подтверждены экспериментально.

2. Представлен генный оператор кроссинговера (PGX) для работы с неупорядоченными хромосомами переменной длины. Результаты исследования эффективности генного ОК показали его лучшую масштабируемость, по

сравнению с традиционно используемыми 1-, 2-точечным и однородным ОК, а также меньшее убывание эффективности d увеличения размера популяции с ростом размерности оптимизируемой функции.

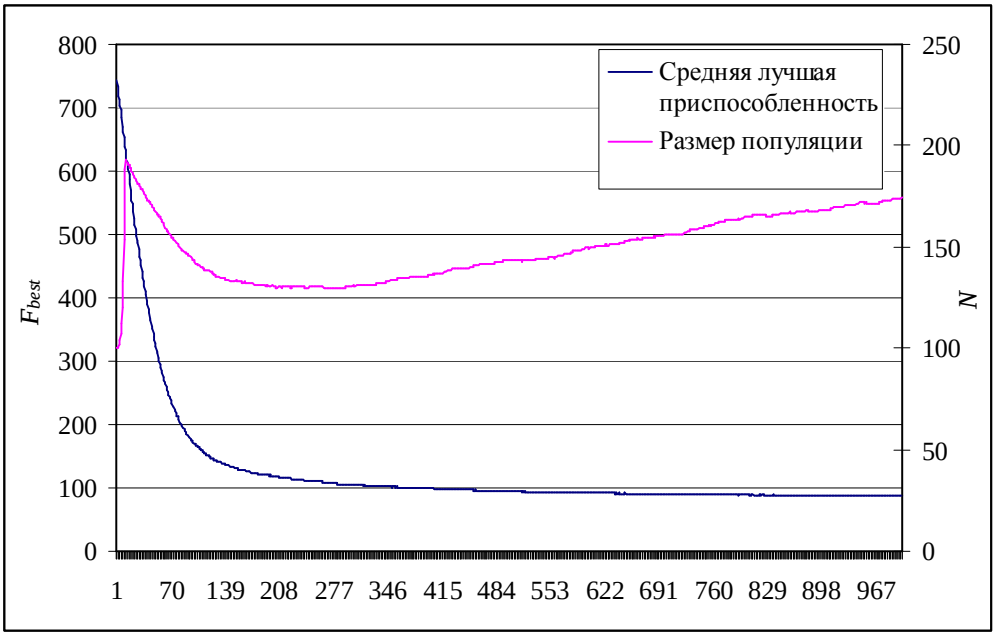


Рис. 2.29. Пример динамики размера N популяции и средней лучшей приспособленности F_{best} для тестовой функции F2 от 50 переменных. Усреднено по 100 запускам

3. Предложена простая стратегия изменения размера популяции с использованием последовательности Фибоначчи, позволяющая адаптироваться к характеристикам эволюционного поиска. Экспериментально показано, что ее применение позволяет в большинстве случаев получить результаты, которые сопоставимы или лучше результатов ГА с постоянным размером популяции. Отмечена эффективность предложенной стратегии адаптации размера популяции для случая малого начального размера популяции.

ГЛАВА 3. ОПИСАНИЕ И ТЕСТИРОВАНИЕ РАЗРАБОТАННОГО НЕЙРОЭВОЛЮЦИОННЫЙ АЛГОРИТМ NEVA

В данной главе описывается созданный нейроэволюционный алгоритм для одновременной настройки структуры и весов связей ИНС, разработанный в соответствии с требованиями, сформулированными в Главе 1, и с использованием результатов исследований генетического алгоритма в Главе 2. Приведены результаты тестирования и сравнения алгоритма на известных задачах классификации и адаптивного управления. Значительное внимание уделяется возможностям разработанных механизмов адаптации, позволяющим использовать одни и те же начальные значения параметров разработанного алгоритма для решения различных задач.

3.1. Индексирование нейронов ИНС

Использование НЭ алгоритма подразумевает представление в генетическом виде информации об ИНС, поэтому прежде чем перейти к описанию разработанного алгоритма, введем ряд правил индексации нейронов, входящих в состав ИНС.

Пусть K – суммарное число нейронов в ИНС и $\mathbf{v} = \{n_i : 1 \le i \le K\}$ – частично упорядоченное множество нейронов, которым поставлено в соответствие множество индексов $\mathbf{k} = \{k_{n_i} : k_{n_i} \ge 0\}$. Введем следующие правила:

1) Никакие два нейрона одной ИНС не могут иметь одинаковые индексы:

$$k_{n_i} \ne k_{n_j}, \text{ при } i \ne j.$$

2) Считаем, что в рамках решения одной задачи число входов и выходов ИНС неизменно и не зависит от структуры ИНС. В силу этого для всех возможных $\mathbf{v}$ первые $(N_I + N_O)$ элементов, где N_I и N_O – число входов и выходов ИНС соответственно, считаем строго упорядоченными, таким образом, чтобы сохранялась постоянность индексов входных и выходных нейронов:

$$k_{n_i} = i - 1,\ 1 \le i \le (N_I + N_O).$$

При этом сначала индексируются входные нейроны, а затем выходные. К примеру, для ИНС с 2 входами и 1 выходом, входные нейроны будут иметь индексы: $k_{n_1} = 0$ и $k_{n_2} = 1$, а выходной: $k_{m_3} = 2$ (другой пример представлен на рис. 3.1).

3) Новые нейроны, добавляемые к структуре ИНС, получают минимальный возможный индекс:

$$k_{n_{K+1}} = K,$$

Например, если ИНС содержит 3 входных, 1 выходной и ни одного скрытого нейрона, то новому нейрону в этой сети будет присвоен индекс «4», следующему нейрону – «5» и т.д.

4) Множество индексов $\mathbf{k}$ нейронов в ИНС не должно содержать «пробелы», для этого должно выполняться условие:

$$\forall k_{n_i} \in [1; K-1]\ \ \exists k_{n_j} \in [0; K-1]:$$

$$k_{n_i} - k_{n_j} = 1.$$

В случае возникновения такого пропуска индексы нейронов корректируются по следующему правилу:

$$k'_{n_i} = k_{n_i} - \Delta + 1, \text{ при } k_{n_i} > \hat{k}_{\min},$$

где k'_{n_i} – скорректированное значение индекса нейрона n_i, Δ – длина пропуска (минимальная разность индексов нейронов, заключающих рассматриваемый пропуск), $\hat{k}_{\min}$ – минимальный пропущенный индекс. Процедура корректировки должна проводиться для каждого имеющегося пропуска.

Пусть, к примеру, после удаления из сети нейронов с индексами «4», «5» и «7» имеем множество $\mathbf{k} = \{0,1,2,3,6,8\}$. Тогда $\Delta = 3$, $\hat{k}_{\min} = 4$ и оставшиеся индексы нейронов корректируются следующим образом: «6» $\rightarrow$ «4», «8» $\rightarrow$ «6». Получим множество $\mathbf{k}' = \{0,1,2,3,4,6\}$, которое

после еще одной корректировки с параметрами $\Delta = 2$, $\hat{k}_{\min} = 5$ будет иметь вид $\mathbf{k}'' = \{0,1,2,3,4,5\}$.

Также отметим, что для различных ИНС нейроны с одинаковыми индексами будем считать идентичными, несмотря на возможно разное количество входящих и исходящих связей, а также на то, что индексы нейронов могли изменяться в результате коррекции индексов.

3.2. Общее описание разработанного НЭ алгоритма

Разработанный НЭ алгоритм NEvA – NeuroEvolutionary Algorithm, позволяет настраивать структуру ИНС одновременно с весами связей. Каждая особь представляет одну ИНС, а популяция – множество ИНС. Используются нейроны с сигмоидной функцией активации (a – константа):

$$y = \frac{1}{1 + \exp(-aS)},$$

$$S = \sum_i w_i x_i .$$

Используется селекция усечением, при этом для скрещивания отбирается не больше половины всех особей, приспособленность которых выше средней приспособленности в популяции. Одна элитная особь из текущего поколения попадает в следующее поколение. Для выхода из локальных экстремумов используется перезапуск алгоритма, который осуществляется, если элитная особь не изменяется в течение 7 поколений.

После скрещивания и мутации, когда сформирована популяция следующего поколения, производится поиск одинаковых особей, представляющих одинаковые ИНС, и если найдено m таких особей, то $(m - 1)$ особь подвергается действию оператора мутации.

В начальной популяции все особи представляют ИНС без скрытых нейронов, в которых все входные нейроны соединены с выходными. Веса сетей инициализируются случайными величинами из диапазона [-0,5; 0,5]. Услож-

нение структур ИНС производится путем применения оператора мутации (см. п. 3.4.2).

3.3. Кодирование информации

Поскольку ИНС представляет собой ориентированный граф, то для хранения информации о структуре и весах связи ИНС достаточно списка всех межнейронных связей с их атрибутами. При этом каждая связь описывается двойкой $(\mathbf{s},\mathbf{e})$, где $\mathbf{s}=\{s_i : i\in[0;N_I-1]\cup[N_I+N_O;K-1]\}$ и $\mathbf{e}=\{e_i : N_I\le i<K\}$ – соответственно множества индексов начальных и конечных нейронов связи. Заметим, что поскольку рассматриваются ИНС прямого распространения, связь не может начинаться на выходном нейроне и заканчиваться на входном. Каждый ген представлен двойкой $(\mathbf{c},\mathbf{w})$, где $\mathbf{c}$ – множество связей, $\mathbf{w}$ – множество весов связей, закодированных 19-разрядными бинарным числами в диапазоне $[-26{,}2144;\ 26{,}2143]$ с шагом 0,0001. Схематичный пример кодирования информации о структуре ИНС и весах связей приведен на рис. 3.1.

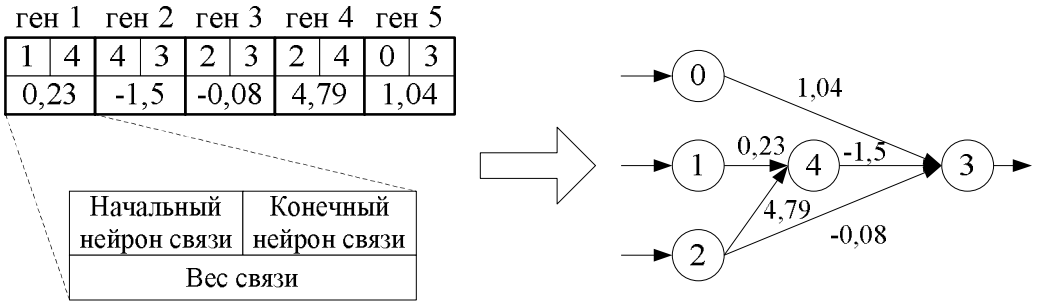


Рис. 3.1. Пример генетического кодирования информации об ИНС. В левой части рисунка показано генетическое представление (хромосома), в правой – соответствующая ИНС. Кружки обозначают нейроны, числа в кружках – индексы нейронов, подписи рядом со связями соответствуют весам

Будем считать, что связь $c_i=(s_i,e_i)$ входит в структуру ИНС, если $s_i,e_i\in\mathbf{k}$, при этом будем писать $c_i\in\mathbf{c}$, где $\mathbf{c}=\{c_j\}$, множество связей, соответствующих рассматриваемой ИНС. Также считаем, что две связи c_1 и c_2 идентичны, если $s_1=s_2$ и $e_1=e_2$ и будем в писать $c_1\equiv c_2$.

Такой подход к генетическому кодированию информации об ИНС позволяет реализовать поиск ИНС с произвольной структурой. Отметим, что порядок следования генов в хромосоме не имеет значения, что требует либо использования специализированного оператора скрещивания, либо упорядочивания связей по некоторому правилу. Однако, учитывая сформулированное в Главе 1 требование к НЭ алгоритму о возможности существования ИНС с различной архитектурой в одной популяции, такое упорядочивание осложнило бы реализацию оператора скрещивания ввиду различий списков связей, соответствующих ИНС с различными структурами.

3.4. Генетические операторы

В литературе [18, 65, 117, 149, 204] и Главе 1 отмечалось, что реализация операторов рекомбинации и вариации на уровне фенотипов позволяет повысить эффективность работы ЭА, а также уменьшить вероятность появления некорректных решений. Далее описываются предлагаемые генетические операторы скрещивания и мутации, работающие на уровне фенотипов и учитывающие следующие требования к НЭ алгоритму, сформулированные в Главе 1:

- возможность одновременного существования ИНС с различной структурой в одной популяции;
- минимизация вероятности появления некорректных НС решений (имеются ввиду ИНС с изолированными нейронами и группами нейронов; с «тупиковыми» нейронами, которые не являются входными или выходными и при этом не посылают сигналы другим нейронам и т.д.);
- возможность поиска структуры ИНС как в сторону усложнения, так и в сторону упрощения.

3.4.1. Оператор скрещивания

В общем виде работу оператора скрещивания можно представить в виде преобразования:

$$c : \mathbf{G}^m \to \mathbf{G}^l,$$

где $\mathbf{G}$ обозначает пространство генотипов, m и l – соответственно число родительских хромосом и число хромосом-потомков, участвующих в одной операции скрещивания. Будем использовать распространенный вариант $m = l = 2$.

Пусть K_1 и K_2 – количество нейронов в ИНС, соответствующих первой и второй родительским особям, и $\mathbf{c}_1$ и $\mathbf{c}_2$ – соответствующие рассматриваемым ИНС множества связей. Далее будем считать, что если потомок наследует связь c_i, то он также наследует и вес w_i этой связи.

Работа предлагаемого ОК соответствует рассмотренному в п. 2.2 способу скрещивания хромосом различной длины (см. рис. 2.4) при этом в качестве меток генов используются параметры межнейронных связей c_k.

В предлагаемом операторе скрещивания общие для родительских особей нейроны и связи наследуются обоими потомками. Для всех c_k таких, что $\exists i, j : c_i \in \mathbf{c}_1, c_j \in \mathbf{c}_2, c_i \equiv c_j \equiv c_k$, веса, соответствующие связям c_i и c_j, скрещиваются с использованием двухточечного кроссинговера. Для нейронов и связей, входящих только в одну из родительских ИНС, используются следующие правила (случай $K_1 > K_2$, x_B – случайная булевская переменная):

1. Для каждой связи $c_{1,i} \notin \mathbf{c}_2$:

1.1. Если $s_{1,i} \geq K_2$, то нейрон с индексом $s_{1,i}$ со всеми своими входными и выходными связями переходит к первому потомку (если $x_B = TRUE$), или ко второму (при $x_B = FALSE$).

1.2. Если $e_{1,i} \geq K_2$, то нейрон с индексом $e_{1,i}$ со всеми своими входными и выходными связями переходит к первому потомку (если $x_B = TRUE$), или ко второму (при $x_B = FALSE$).

1.3. Если $s_{1,i}, e_{1,i} < K_2$, то связь «разыгрывается» между потомками: при $x_B = TRUE$ связь $c_{1,i}$ наследуется первым потомком, а при $x_B = FALSE$ – вторым.

2. Все связи $c_{2,j} \notin \mathbf{c}_1$ случайным образом разыгрываются между потомками: при $x_B = TRUE$ связь $c_{2,j}$ наследуется первым потомком, а при $x_B = FALSE$ – вторым.

Аналогичный набор правил используется и для случая $K_2 > K_1$.

Таким образом, в предлагаемом операторе скрещивания учитывается структура ИНС родительских особей, что позволяет говорить о скрещивании на уровне фенотипов.

Пример скрещивания показан на рис. 3.2. Сплошными линиями показаны общие нейроны и связи, пунктирными – различающиеся.

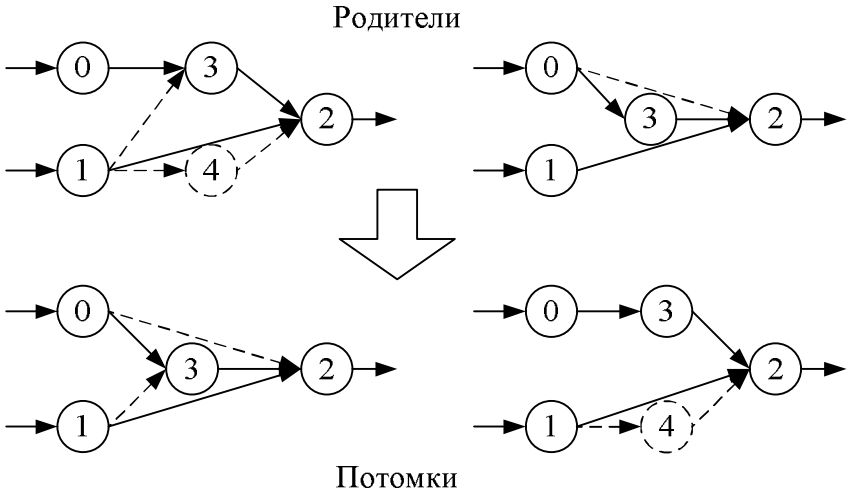


Рис.3.2. Пример работы предлагаемого оператора скрещивания. Общие для родительских особей нейроны и связи показаны сплошными линиями, а различающиеся – пунктирными. Нейроны с индексами 0 и 1 – входные, нейрон с индексом 2 – выходной

3.4.2. Оператор мутации

В результате мутации возможны следующие виды изменений структуры ИНС (виды мутации):

- *добавление скрытого нейрона*. Новый нейрон добавляется вместе с входящей и выходящей связями. При этом выходящая связь не может идти на входной нейрон, а входящая не может поступать с выходного нейрона.
- *удаление случайно выбранного скрытого нейрона* вместе со всеми входными и выходными связями. При необходимости производится коррекция индексов нейронов (см. п. 3.1). Входные и выходные нейроны сети не могут быть удалены.
- *добавление связи*. Индексы начального и конечного нейронов новой связи определяются случайным образом. При этом связь не может заканчиваться входным нейроном. Вес связи определяется случайно из диапазона [-0,5; 0,5]. Если в ИНС уже существует связь с такими же входными и выходными нейронами, то ее вес заменяется на случайный.
- *удаление случайно выбранной связи*. При удалении последней связи скрытого нейрона сам нейрон также удаляется, и, если необходимо, производится коррекция индексов нейронов.
- *изменение веса* случайно выбранной связи на случайную величину из диапазона [-0,5; 0,5].

Перечисленные виды изменений структуры ИНС позволяют вести поиск как в сторону усложнения (добавление связей и нейронов), так и в сторону упрощения (удаление связей и нейронов) структуры ИНС. При этом за счет независимости мутаций различных особей, становится возможным существование в одной популяции особей, представляющих ИНС с различной структурой, что удовлетворяет сформулированным в Главе 1 требованиям к разработке НЭ алгоритма.

Еще одним из требований является минимизация вероятности появления НС с «плохой» структурой, когда, например, в структуре ИНС присутствуют изолированные нейроны и группы (ансамбли) нейронов; «тупиковые» нейроны, которые не являются входными или выходными и при этом не посылают сигналы другим нейронам и т.д. Также будем считать «плохими» слабосвязанные ИНС, содержащие малое число связей по сравнению с числом нейронов (пример на рис. 3.3).

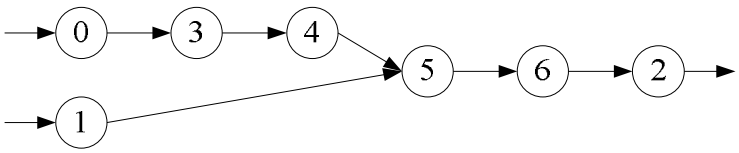


Рис.3.3. Пример ИНС с «плохой» структурой, содержащей малое число связей. Нейроны с индексами 0 и 1 – входные, нейрон с индексом 2 – выходной

Для предотвращения возникновения вышеперечисленных случаев будем применять различные виды мутаций в зависимости от особенностей архитектуры ИНС, соответствующей мутирующей особи. Для этого введем два коэффициента, регулирующие размер и «направление развития» НС.

Первый коэффициент k_C характеризует степень «связанности» нейронов сети и пропорционален отношению количества существующих связей к числу возможных:

$$k_C = \left(\frac{2N_C}{K(K-1) - N_I(N_I-1) - N_O(N_O-1)} \right)^2,$$

где $N_C = |\mathbf{c}|$ – количество связей в ИНС.

Второй коэффициент k_N будем использовать для определения необходимости добавления/удаления нейронов, на основании предположения, что чем больше суммарное количество входных и выходных нейронов ИНС, которое зависит от решаемой задачи, тем, вероятно, ИНС с более сложной структурой необходима для решения поставленной задачи. Вычисляется коэффициент k_N по следующей формуле:

$$k_N = k_C \left(\frac{N_I + N_O}{N_N} \right)^2.$$

Умножение на k_C в правой части необходимо чтобы учесть некоторые особенности имеющейся структуры ИНС.

Выбор вида мутации определяется в зависимости от значений коэффициентов k_C и k_N (рис. 3.4, x – случайная величина равномерно распределенная на интервале $[0;\ 1]$, N_H – количество скрытых нейронов).

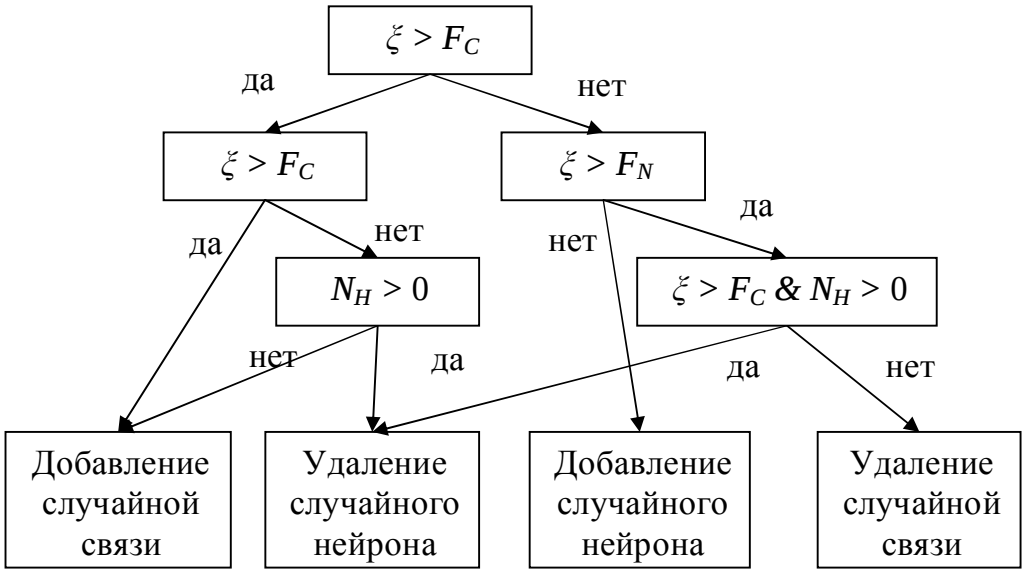


Рис. 3.4. Схема выбора вида мутации

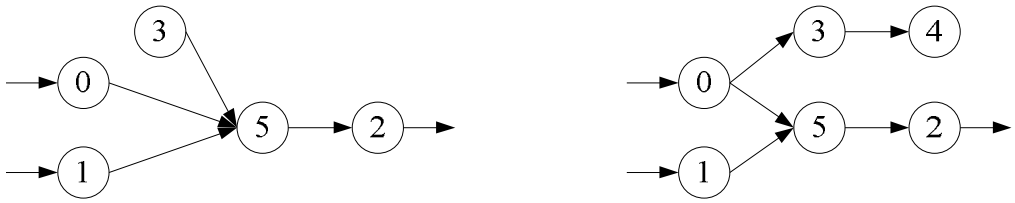
Условно схему выбора вида мутации можно разделить на две «ветви» по первому условному переходу:

1. Ветвь увеличения k_C (усложнение структуры ИНС).
2. Ветвь уменьшения k_C (упрощение структуры ИНС).

Поскольку удаление нейрона может привести как к уменьшению, так и к увеличению k_C, в зависимости от текущей структуры ИНС, то этот вид мутации присутствует в обеих ветвях. Таким образом, основным фактором для регуляции структуры получаемых ИНС является число задействованных межнейронных связей.

Такой подход, с одной стороны не ограничивает сверху количество скрытых нейронов, а с другой, препятствует слишком быстрому увеличению числа нейронов и связей в ИНС, так как добавление каждого нового нейрона в сеть будет происходить с меньшей вероятностью. Мутация веса одной случайно выбранной связи происходит для всех особей популяции с вероятностью 0,5.

Удаление межнейронных связей в ИНС может дать побочный эффект: появление «висячих» нейронов, у которых нет входящих связей (рис. 3.5а), а также «тупиковых» нейронов, не имеющих выходящих связей (рис. 3.5б). Отметим, что особенностью используемой сигмоидной функции активации является установление выхода нейрона равным 0,5 при отсутствии сигналов на входе ($S = 0$). Таким образом, «висячие» нейроны становятся источниками константного сигнала, который будучи подан на входы других нейронов играет роль смещения. Также стоит отметить, что удаление связей, начинающихся на входных нейронах, может привести к удалению части неинформативных и малоинформативных входных признаков, что будет продемонстрировано в п. 3.5.2 на примере решения задачи нейроуправления.



а) б)

Рис. 3.5. Примеры ИНС с а) «висячими» (нейрон 3) и б) «тупиковыми» (нейроны 3 и 4) нейронами. В обоих случаях нейроны с индексами 0 и 1 – входные, нейрон с индексом 2 – выходной.

Вероятность мутации P_m также настраивается адаптивно. Величину вероятности мутации принято выбирать небольшой, так как в противном случае, при превышении P_m некоторого значения (так называемый порог ошибки [173]), в эволюционном поиске начинает преобладать случайная составляю-

щая вплоть до «вырождения» эволюционного поиска в случайный. В силу этого вероятность мутации равная $1/L$, где L – длина хромосомы (применительно к рассматриваемому НЭ алгоритму, число связей), представляется достаточно разумной, так как предполагает в среднем одну мутацию на одну хромосому. При этом неоднократно отмечалось (см. например, [33, 126], а также п. 1.1.1), что на более поздних этапах эволюции значение P_m лучше уменьшить, чтобы снизить вероятность разрушения найденных «хороших» решений. Для этого для каждой хромосомы будем производить $N_I N_O$ вызовов оператора мутации с вероятностью P_m, при этом выбор вида мутации осуществляется в соответствии со схемой на рис. 3.4. Таким образом, с учетом постепенного усложнения структуры ИНС, представленных популяцией, можно сказать, что частота мутаций будет уменьшается с течением времени, так как значение L в ходе эволюции увеличивается, и вероятность однократной мутации, равная $\frac{N_I \cdot N_O}{L}$, становится меньше[1].

3.5. Экспериментальное исследование разработанного НЭ алгоритма

Данный раздел посвящен экспериментальной проверке эффективности представленного НЭ алгоритма NEvA на ряде известных задач классификации и адаптивного управления.

Поскольку решение различных задач требует настройки параметров используемого алгоритма, то, очевидно, не существует для некоторого алгоритма универсального оптимального набора параметров, позволяющего максимально эффективно решать любые поставленные задачи. Поиск набора параметров, оптимального для конкретной задачи, является, по сути, комбинаторной задачей, решение которой во многом зависит от опыта исследователя и объема накопленных им знаний. Сам по себе процесс подбора параметров алгоритма может занять достаточно длительное время в связи с необходимо-

[1] Соответственно уменьшаются вероятности 2-, 3-,… кратной мутации.

стью экспериментальной проверки каждого потенциально оптимального набора. Поэтому большую важность представляет адаптация параметров алгоритма в процессе его работы, при этом начальные значения параметров могут быть фиксированными и использоваться при решении широкого круга задач.

При разработке алгоритма NEvA большое внимание уделялось способностям к адаптации, в результате чего следующие компоненты и параметры алгоритма настраиваются адаптивно в процессе работы:

– размер популяции;
– оператор кроссинговера;
– способность генного оператора кроссинговера к разрушению в зависимости от количества генов в хромосоме;
– оператор мутации;
– вероятность мутации.

Тестирование алгоритма NEvA с начальными значениями параметров из табл. 3.1 будем проводить, используя следующие задачи:

- задача классификации («Исключающее ИЛИ»);
- адаптивное нейроуправление (перевернутый маятник (inverted pendulum) [110]).

Отметим, что начальные значения параметров, равные представленным в табл. 3.1, используются и для решения задачи НЭ улучшения визуального качества цифровых изображений в Главе 4.

Все эксперименты проводятся на компьютере с процессором AMD Sempron 2500+ (1750 МГц), 512 МБ ОЗУ под управлением операционной системы Microsoft Windows XP Professional SP2.

Табл.3.1. Начальные значения параметров для алгоритма NEvA при тестировании, а также для решения задачи улучшения визуального качества цифровых изображений (см. Главу 4)

Параметр	**Начальное значение параметра**
Разрядность генов	19
Диапазон значений генов	[-26,2144; 26,2143]
Селекция	Усечением
Порог отсечения	0,5
Количество элитных особей	1
Оператор кроссинговера для закодированных весов связей	2-точечный генный
Вероятность кроссинговера, P_c	0,8
Вероятность мутации, P_m	$1/(N_I\, N_O)$
Размер популяции, N_0	50
Мин. размер популяции[1], N_{min}	$2(2+[\log_2(19\ N_I\ N_O)])$
Макс. размер популяции, N_{max}	200

3.5.1. Исключающее ИЛИ

Задача реализации с помощью ИНС логической операции «Исключающее ИЛИ» является примером простейшей задачи классификации, которая не может быть решена путем линейного разделения пространства входных признаков [32, 165], поэтому в структуре ИНС должен быть хотя бы 1 скрытый нейрон. С точки зрения тестирования разработанного НЭ алгоритма рассматриваемая задача представляет интерес тем, что в начальной структуре ИНС нет скрытых нейронов, и появление и закрепление скрытых нейронов, а также настройка весов входящих и исходящих связей производится за счет эволюционного поиска.

[1] Подробности выбора минимального размера популяции см. в п. 2.3.2

Таблица истинности для операции «Исключающее ИЛИ» приведена в табл. 3.2.

Табл.3.2. Таблица истинности для операции «Исключающее ИЛИ»

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Также отметим, что данная задача является частным случаем задачи *n*-битной четности, когда для входной *n*-разрядной бинарной строки требуется установить на выходе нейросети «1», если количество единичных разрядов в ней нечетно, и «0», если количество единичных разрядов четно или равно 0. Встречаются варианты использования задачи 3- и 4-битной четности для тестирования НЭ алгоритмов [175, 227].

Тестирование разработанного НЭ алгоритма NEvA будем проводить, обучая ИНС с использованием данных из табл. 3.2. Для сравнения результатов также будем рассматривать следующие алгоритмы настройки весов межнейронных связей:

- алгоритм обратного распространения ошибки без инерционной составляющей (далее ОР);
- алгоритм обратного распространения ошибки с инерционной составляющей (далее ОРИ);
- простой генетический алгоритм (далее ПГА);
- генетический алгоритм, использующий стратегию элитизма, когда лучшая особь в текущем поколении переходит в следующее поколение без селективного отбора (далее ЭГА).

Первые два алгоритма (ОР и ОРИ) используют градиентный спуск, а ПГА и ЭГА используют эволюционный принцип для поиска решения. Так как перечисленные алгоритмы настраивают только веса связей ИНС, то структура сети выбрана фиксированной. Будем рассматривать несколько вариантов структуры ИНС с 2, 3 и 4 нейронами в скрытом слое, 2 входными и 1 выходным нейронами. Будем обозначать ИНС с такими структурами простым перечислением количества нейронов в каждом слое, начиная со входного, то есть как «2-2-1», «2-3-1» и «2-4-1» соответственно. Таким образом, с учетом дополнительного входного нейрона как источника постоянного единичного сигнала, использованные ИНС всего содержат в первом случае 6 нейронов и 9 связей, во втором – 7 нейронов 13 связей и в третьем – 8 нейронов и 17 межнейронных связей. Параметры для ПГА и ЭГА представлены в табл. 3.3.

Табл. 3.3. Значения параметров для ПГА и ЭГА

Параметр	**Значение параметра (ПГА)**	**Значение параметра (ЭГА)**
Разрядность генов	19	
Диапазон значений генов	[-26,2144; 26,2143]	
Селекция	Турнирная	
Размер турнира	2	4
Количество элитных особей	0	1
Оператор кроссинговера	1-точечный	2-точечный генный
Вероятность кроссинговера, P_c	0,8	
Вероятность мутации, P_m	0,002	
Размер популяции, N	50	

Все алгоритмы запускаются по 50 раз, и решение считается найденным, если появляется особь, представляющая сеть, среднеквадратичная ошибка

выхода которой (значение целевой функции): $F = \frac{1}{2KN_O}\sum_{i=1}^{K}\sum_{j=1}^{N_O}\left(y_{ij} - f_{ij}\right)^2$, где K – количество примеров в обучающей выборке, – меньше 0,001. Если количество вычислений целевой функции превышает 150000, то работа алгоритма останавливается, и запуск считаем неудачным.

Будем сравнивать алгоритмы по количеству N_{FE} вычислений целевой функции, необходимых для нахождения решения. В табл. 3.4 и 3.5 представлены средние значения $<N_{FE}>$ количества вычислений целевой функции, среднеквадратичное отклонение (СКО) результатов и количество неудачных запусков для сравниваемых алгоритмов.

Табл. 3.4. Средние значения $<N_{FE}>$ для различных алгоритмов для структуры ИНС «2-3-1» и «2-2-1»

	Структура ИНС: «2-3-1»				Структура ИНС: «2-2-1»			
	ПГА	ЭГА	ОР	ОРИ	ПГА	ЭГА	ОР	ОРИ
$<N_{FE}>$	4389,46	1641,24	8255,35	2016,64	6702,27	1846,69	21809	8923,76
СКО	10989,98	4239,94	14890,99	4762,11	18910,25	4176,17	38019,13	18200,10
Неудачи	1	2	1	3	15	20	27	13

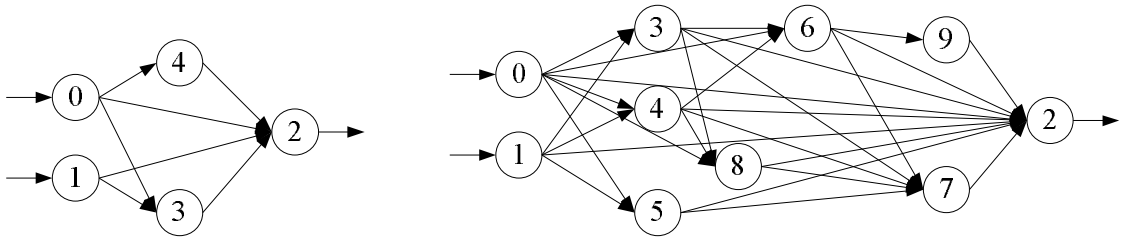
Табл. 3.5. Средние значения $<N_{FE}>$ для различных алгоритмов. Структура ИНС для алгоритмов ПГА, ЭГА, ОР и ОРИ: «2-4-1»

	NEvA	ПГА	ЭГА	ОР	ОРИ
$<N_{FE}>$	7693,7	1261,99	711,97	3682,90	619,7
СКО	3577,98	2427,72	890,27	4465,94	407,70
Неудачи	0	0	0	1	0
$<K>$	7,67	8			
$<N_C>$	15,35	17			

Из представленных в табл. 3.4. результатов экспериментов видно, что в случае ИНС с фиксированной структурой с 2 и 3 нейронами в скрытом слое

использование алгоритмов ПГА, ЭГА, ОР и ОРИ приводит к появлению неудачных запусков, при этом СКО превышает <N_{FE}>, что говорит о нестабильности результатов, обусловленной чувствительностью этих алгоритмов к начальным условиям.

Хорошие результаты работы алгоритмов ПГА, ЭГА, ОР и ОРИ получены при настройке весов связей ИНС со структурой «2-4-1» (табл. 3.5). При сравнении этих результатов с результатами работы предлагаемого алгоритма NEvA видно, что с учетом того, что в разработанном НЭ рассматривалась изначально более сложная задача, включающая одновременный поиск структуры и весов связей ИНС, по значению <N_{FE}> алгоритм NEvA незначительно отстает от алгоритмов ПГА, ЭГА, ОР и ОРИ. При этом отметим стабильность результатов работы предлагаемого НЭ алгоритма и отсутствие неудачных запусков. Среднее количество нейронов и связей в ИНС, полученных в результате работы алгоритма NEvA, составило соответственно 7,67 и 15,35, что позволяет говорить о сопоставимой по сложности структуре ИНС для сравниваемых алгоритмов. Примеры нейронных сетей, найденных с помощью НЭ алгоритма и содержащей наименьшее и наибольшее число связей представлены на рис. 3.6.



а) б)

Рис. 3.6. Примеры ИНС с самой простой (а) и самой сложной (б) структурой, найденных с использованием алгоритма NEvA.

Среднее время поиска решения с использованием разработанного НЭ алгоритма составило 1,15 с. с СКО равным 0,60, что в совокупности с полученными результатами (табл. 3.5) позволяет говорить об эффективности ал-

горитма NEvA при решении рассмотренной задачи. При этом отметим, что начальные значения параметров алгоритма NEvA были зафиксированы (табл. 3.1) до начала экспериментов и в процессе экспериментов не настраивались, что показывает эффективность механизмов адаптации в алгоритме NEvA.

3.5.2. Задача балансирования шестов

Данная задача представляет собой задачу адаптивного нейроуправления. Целью является удержание от падения выведенных из равновесия 1 либо 2 шестов (перевернутых маятников, англ. inverted pendulum), находящихся на тележке, которая может ограниченно передвигаться, путем перемещения самой тележки (пример для случая 2 перевернутых маятников показан на рис. 3.7). Моделирование движения тележки и шестов производится в дискретном времени с шагом $\Delta t = 0{,}02$ с.

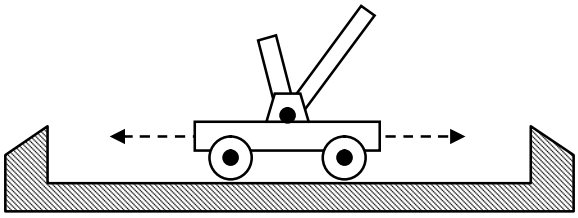


Рис. 3.7. Схематичное изображение тележки с 2 шестами различной длины. Пунктирными стрелками показаны направления возможного движения тележки

Существует несколько разновидностей рассматриваемой задачи в зависимости от количества шестов (1 или 2), вида воздействия на тележку (дискретное либо непрерывное [204]), числа степеней свободы (тележка перемещается в одном или двух измерениях [112]), а также от информации, подаваемой на управляющее устройство (включающей данные об угловых скоростях шестов, либо без этих данных [204]). Отметим, что в случае управления тележкой без информации об угловых скоростях, необходима система управления с обратной связью, поэтому такая разновидность задачи в данной работе не рассматривается. Математическая модель тележки с произвольным

числом шестов и значения параметров модели для экспериментов представлены в [111].

Будем рассматривать следующие варианты задачи (тележка перемещается в одном измерении):

1. Один шест, фиксированная сила воздействия на тележку, равная 10Н в каждую сторону.

2. Два шеста различной длины, непрерывная по величине сила воздействия на тележку в диапазоне от 0 до 10 Н в каждую сторону.

Результаты работы алгоритма NEvA, усредненные по 100 запускам, будем сравнивать с результатами алгоритмов GENITOR и SANE из [167] и [112] (для задачи балансирования 2 шестов), ESP из [112], CMA-ES из [135] и NEAT из [204]. Алгоритмы GENITOR, ESP и CMA-ES не настраивают структуру ИНС, а осуществляют только поиск весов связей. При этом в GENITOR [221] используется steady-state схема эволюционного поиска, когда в каждом поколении худшая особь в популяции замещается одной особью-потомком. В алгоритме ESP [110] реализован коэволюционный (coevolutionary) подход, при этом для каждого нейрона поиск весов входящих связей осуществляется в отдельной популяции. Алгоритм CMA-ES для поиска весов связей ИНС [135] основан на использовании в эволюционных стратегиях адаптации матрицы ковариаций для оптимизируемых параметров, а также дерандомизации (derandomization) поиска и накопления информации (cumulation) из предыдущих поколений. В алгоритме SANE [167] структура ИНС настраивается ограниченно, так как количество скрытых нейронов и число связей каждого нейрона фиксировано и определяется до начала запуска, и рассматриваются только ИНС прямого распространения с одним скрытым слоем. Алгоритм NEAT [204] позволяет осуществлять одновременный поиск структуры и весов связей ИНС и по функциональным возможностям превосходит GENITOR, SANE, ESP и CMA-ES.

Приспособленность особей F_i для рассматриваемых вариантов задачи нейроуправления будем определять следующим образом:

$$F_i = t,$$

где t – количество тактов, в течение которых удавалось удержать шесты с использованием ИНС, соответствующей i-й особи. Для задачи с одним шестом необходимо удержать шест на протяжении 120000 тактов (40 минут модельного времени), а в случае с двумя шестами – в течение 100000 тактов (более 33 минут модельного времени). В качестве меры эффективности алгоритмов будем рассматривать количество потребовавшихся неудачных попыток до достижения желаемой цели.

Результаты для алгоритма NEvA получены по 100 запускам, для алгоритмов SANE, ESP и CMA-ES по 50 запускам и для алгоритма NEAT по 150 запускам. Напомним, что начальные значения параметров для алгоритма NEvA неизменны для рассматриваемых задач и представлены в табл. 3.1.

Рассмотрим сначала первый вариант задачи с 1 шестом на тележке. Результаты экспериментов приведены в табл. 3.6.

Табл.3.6. Результаты решения проблемы с 1 маятником

Алгоритм	Среднее количество попыток	СКО	Число неудач
GENITOR	1846	1396	0
SANE	535	329	0
ESP	285	277	0
CMA-ES	283	138	0
NEvA	688,46	687,58	0
NEvA ($N_0 = 25$)	466,79	439,41	0

Из представленных в табл. 3.5 результатов видно, что разработанный алгоритм уступает алгоритмам SANE, ESP и CMA-ES. Достаточно большой

разрыв между алгоритмом NEvA и этими алгоритмами можно объяснить следующими факторами:

– большая для алгоритма NEvA сложность задачи, включающая помимо подстройки весов межнейронных связей также поиск структуры ИНС, причем без явных ограничений на топологию и количество скрытых нейронов и связей как в алгоритме SANE;
– выбор достаточно большого начального размера популяции, что приводит к увеличению общего количества попыток.

Ввиду последнего фактора отметим, что уменьшение начального размера популяции до 25 особей (N_0 = 25) привело к существенному улучшению результатов (последняя строка в табл. 3.6). Также заметим, что во всех случаях разработанный НЭ алгоритм опережает алгоритм GENITOR.

Информация о структуре использованных ИНС приведена в табл. 3.7. «+/-» для алгоритма SANE означает, что количество скрытых нейронов и связей фиксировано, но алгоритм настраивает расположение межнейронных связей.

Табл. 3.7. Данные о структуре ИНС для задачи с 1 маятником

Алгоритм	$<N_H>$	$<N_C>$	Настройка структуры ИНС
GENITOR	5	35	-
SANE	5	20[1]	+/-
ESP	5	30	-
CMA-ES	8	40	-
NEvA	1,24	6,71	+

Количество нейронов N_H и связей N_C в ИНС для алгоритмов GENITOR, SANE, ESP и CMA-ES выбирается пользователем. Из представленных в табл.

[1] Предполагаемое значение

3.7 данных видно, что использование НЭ подхода к поиску структуры ИНС позволяет найти значительно более простую топологию по сравнению с «ручным» выбором.

Среднее время поиска решения с использованием разработанного алгоритма NEvA при N_0 = 50 для задачи балансирования одного маятником составляет 1,04 сек. со среднеквадратичным отклонением равным 1,07.

Результаты решения задачи с двумя шестами и информация о структуре использованных ИНС представлены соответственно в табл. 3.8 и 3.9. Также как и в случае задачи с одним шестом будем оценивать количество попыток необходимых для поиска ИНС, позволяющей удержать шесты в течение требуемого количества тактов (в данном случае 100000 тактов или более 30 минут модельного времени).

Табл. 3.8. Результаты решения задачи с 2 маятниками.

Алгоритм	Среднее количество попыток	Размер популяции	Число неудач
SANE	12600	200	0
ESP	3800	200	0
CMA-ES	895	3	0
NEAT	3578	150	0
NEvA	1448,55	18 – 200	0

Результаты разработанного алгоритма NEvA значительно превосходят результаты алгоритмов SANE, ESP и NEAT, но уступают результатам CMA-ES. Здесь необходимо отметить, что в алгоритме CMA-ES не производится поиск структуры ИНС, и, следовательно, рассматривается более простая задача.

Анализ использованных различными алгоритмами структур ИНС (табл. 3.9) позволяет сделать вывод об эффективности алгоритма NEvA, не усту-

пающему лучшему на данный момент результату, полученному с использованием алгоритма NEAT, и значительно превосходящему результаты настройки топологии ИНС для алгоритмов SANE, ESP и CMA-ES, когда параметры (количество скрытых нейронов и межнейронных связей) задаются пользователем.

Табл. 3.9. Данные о структуре ИНС для задачи с 2 маятниками. В скобках для алгоритма NEvA представлены средние значения

Алгоритм	N_H	N_C
SANE	7	35[1]
ESP	5	35
CMA-ES	6	42
NEAT	0 – 4	6 – 15
NEvA	0 – 3 (0,71)	5 – 14 (7,6)

Необходимо отметить, что реализованная в разработанном алгоритме NEvA возможность удаления в результате мутации межнейронных связей при решении задачи балансирования двух шестов привело к удалению малоинформативных входных сигналов и уменьшению, тем самым, размерности задачи. В частности, несколько раз были удалены сигналы о скорости передвижения тележки или ее координатах. Соответствующие решения были найдены:

– в 4 случаях при решении задачи балансирования одного шеста;
– в 11 случаях при решении задачи балансирования двух шестов.

Среднее время поиска решения с использованием разработанного алгоритма NEvA для задачи с двумя маятниками составляет 2,89 сек. с СКО равным 3,07.

[1] Предполагаемое значение

3.6. Разработка библиотеки классов для эволюционных вычислений

3.6.1. Обзор существующих инструментальных библиотек для эволюционных вычислений

В настоящее время наиболее известными и функциональными являются следующие инструментальные библиотеки для эволюционных вычислений:

- Open BEAGLE (Beagle Engine is an Advanced Genetic Learning Environment) [103].

- ECJ (Evolutionary Computation in Java) [158].

- EO (Evolving Objects) [142].

Библиотеки Open BEAGLE и EO реализована на языке C++, а библиотека ECJ – на языке программирования Java. Все рассматриваемые библиотеки разработаны с использованием объектно-ориентированных принципов и представляют мощные и функциональные средства для решения различных исследовательских и практических задач с использованием эволюционного подхода. В частности, для повышения гибкости в этих библиотеках используются следующие возможности:

- работа с различными способами представления данных;

- работа с различными способами записи приспособленности, не зависящими от способа представления данных и операторов;

- работа с различными операторами;

- работа с различными моделями эволюции;

- управление параметрами запуска ЭА;

- конфигурируемый вывод информации о результатах работы ЭА.

Библиотека ECJ является популярной библиотекой для эволюционных вычислений, написанной с использованием языка программирования Java, но ее недостатком является требовательность к вычислительным ресурсам [177] и низкое быстродействие.

Библиотека EO изначально разрабатывалась для работы с любыми

структурами данными и содержит богатый набор инструментальных и вспомогательных программных структур и функций [142]. Имеются возможности гибкой настройки параметров запуска и сбора статистики. Среди недостатков ЕО отметим сложность работы с этой библиотекой [103] и отсутствие возможности сбора статистики по многократным запускам эволюционного алгоритма.

В библиотеке Open BEAGLE представлены богатые возможности для программирования эволюционных алгоритмов. Сама библиотека разработана с учетом современных подходов и требований к разработке программного обеспечения. Используется модель управления памятью на основе подсчета ссылок и умных указателей, а для управления нестандартными/аварийными ситуациями применяется обработка исключений. Для определения параметров работы ЭА, свойств внутренних структур данных и вывода данных в файл отчета используется формат XML. Достоинством библиотеки Open BEAGLE является обширная и подробная документация по использованию и установке, а также легкость работы с имеющимися инструментальными средствами. Недостатком является сложность внутренней организации библиотеки Open BEAGLE, которая влечет сложность расширения библиотеки. При этом при добавлении новых объектов необходимости учитывать ряд требований, отрицательно сказывающихся на быстродействии (обязательное определение методов `read` и `write`, работающих с форматом XML). Быстродействие также снижается за счет использования в Open BEAGLE механизма обработки исключений [20].

3.6.2. Общие требования к библиотеке классов

Рассмотрим общие требования к разрабатываемой инструментальной библиотеке классов.

Современные взгляды на разработку программного обеспечения (ПО), и инструментальных библиотек в частности, включают следующие требования:

1. *Модульность*. Необходимо для обеспечения возможности независимо-

го проектирования, разработки и сопровождения (обновления/замены) отдельных элементов (модулей) программы без влияния на работоспособность и стабильность всей программы в целом.

2. Возможность *повторного использования* (*reusability*) отдельных модулей при различных сценариях работы программы. Необходимо для уменьшения времени разработки и размера исполняемого кода, а также для удобства сопровождения библиотеки.

3. *Расширяемость*. Необходимо для дальнейшего роста функциональности библиотеки. Для выполнения этого требования важно выполнение требования *модульности*, так как расширяемость подразумевает также добавление элементов, не предусмотренных на этапе проектирования библиотеки, которое не должно привести к модификации исходных текстов библиотеки классов и/или нарушению стабильности работы.

4. *Мультиплатформенность*. Необходимо для возможности создания программ, использующих библиотеку, независимо от используемой операционной системы или программно-аппаратной платформы.

5. Библиотека классов не должна определять особенности пользовательского интерфейса конечной программы. Необходимо для обеспечения гибкости использования библиотеки.

Для удовлетворения всех вышеперечисленных требований будем использовать *паттерны проектирования* (*design patterns*) [5], которые разрабатывались для создания гибкого и надежного ПО, предусматривающего повторное использование, и появились в результате обобщения опыта работы многих разработчиков ПО.

Отметим, что помимо «архитектурных» требований, перечисленных выше, существенным является требование к *эффективности* разрабатываемого ПО, что необходимо для обеспечения высокого быстродействия при разумных затратах вычислительных ресурсов.

3.6.3. Выбор средств разработки

Исходя из сформулированных выше требований к разрабатываемой библиотеке классов, рассматривались следующие языки программирования:

1. Java.

2. C#.

3. C++.

Все рассматриваемые языки программирования поддерживают средства объектно-ориентированного программирования.

Языки Java и C# используются в большом классе приложений, однако недостатком первого является меньшая по сравнению с другими языками скорость работы, а второго – сложность разработки мультиплатформенных приложений. Общим недостатком при использовании языков Java и C# является необходимость установки дополнительных программных средств поддержки: Java Virtual Machine и .NET Framework соответственно.

Язык C++ позволяет разрабатывать эффективные приложения и для обеспечения мультиплатформенности при использовании стандартных языковых средств (C Run Time библиотек, средств Standard Template Library – STL) во многих случаях достаточно перекомпилировать исходный код под конкретную платформу. Таким образом для разработки инструментальной библиотеки классов был выбран язык программирования C++.

Для разработки библиотеки использовалась среда программирования Microsoft Visual Studio .NET 2003, так как эта среда программирования имеет развитые средства разработки и отладки программных проектов и удобный пользовательский интерфейс, облегчающий такие рутинные процедуры как поиск количества и типов параметров вызываемой функции, поиск методов и переменных класса и др.

3.7. Структура разработанной библиотеки классов ECWorkshop

3.7.1. Общее описание

Укрупненная структурная схема разработанной инструментальной библиотеки показана на рис. 3.8. Основными модулями библиотеки являются:

1) Производящий модуль.

2) Эволюционный алгоритм.

3) Пространство задачи.

4) Модуль обработки результатов.

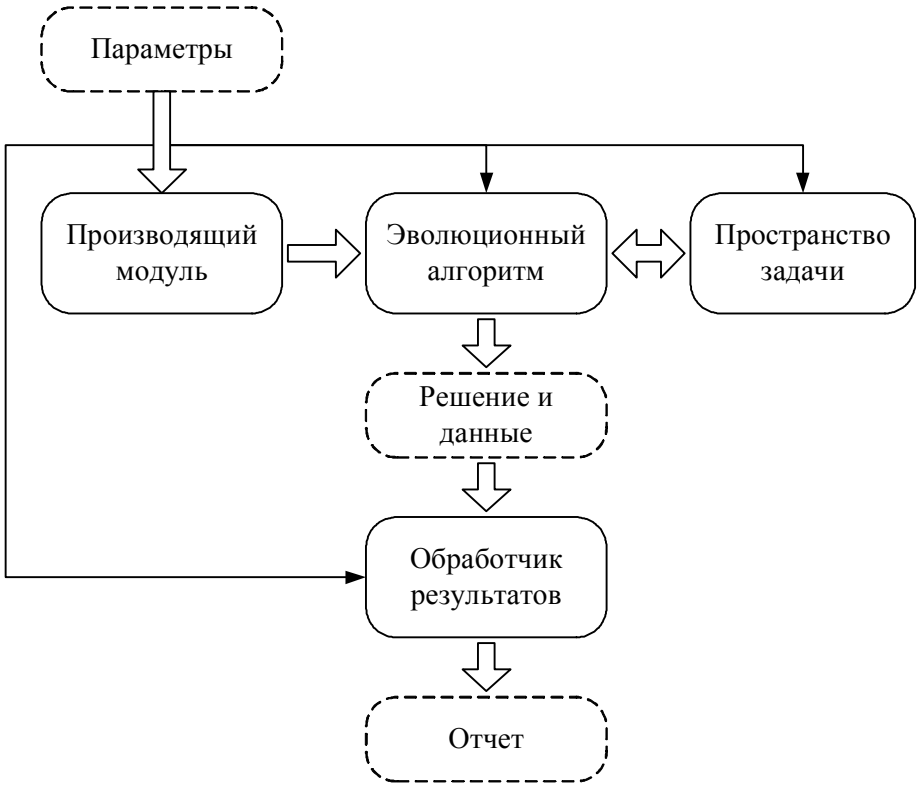


Рис. 3.8. Укрупненная схема организации модулей библиотеки

Все входные параметры записываются в структуру `QEAParameters`, содержащую поля, представленные в табл. 3.10, и в дальнейшем доступны для чтения всем модулям и операторам библиотеки.

Производящий модуль реализован с использованием паттерна Фабрика (Factory) [5], и используется для создания всех операторов и структур данных ЭА. Необходимость использования фабричного паттерна вызвана тем, что используемые эволюционные операторы зависят от используемого способа кодирования решений. Такой подход позволяет упростить программный код ЭА за счет абстрагирования от способа кодирования и избежать случая, когда эволюционные операторы не подходят для работы с выбранным способом кодирования.

Табл. 3.10. Описание полей структуры `QEAParameters`

Имя поля	**Назначение**
`encodingType`	Тип кодирования.
`chromosomeLength`	Длина хромосомы (количество генов).
`arity`	Мощность используемого алфавита (для хромосом, представленных символьными строками).
`geneSize`	Длина гена в битах (символах).
`geneOffset`, `genePrecision`	Соответственно шаг и смещение, используемые для перекодирования генов из целочисленного вида в вещественный по формуле: g_{real} = `genePrecision`*g_{int} + `geneOffset` где g_{real} и g_{int} – соответственно вещественное и целочисленное значения гена.
`populationSize`	Начальный размер популяции.
`minPopSize`, `maxPopSize`	Минимальный и максимальный допустимый размер популяции соответственно.
`sizingStep`, `sizingDirection`	Соответственно величина изменения размера популяции и направление изменения («1» = увеличение, «-1» = уменьшение) для «+» и «-» стратегий (см. п. 2.3.1).
`generationsNumber`	Количество поколений эволюционного поиска.
`errorLevel`	Требуемое значение целевой функции.
`selectionType`	Тип оператора селекции.
`tournamentSize`	Начальный размер турнира для турнирной селекции.

Имя поля	Назначение
`eliteCount`	Количество элитных особей.
`parentSelectorType`	Тип оператора для выбора скрещиваемых родительских особей.
`parentNumber, childrenNumber`	Соответственно количество родительских особей, участвующих в скрещивании, и количество генерируемых особей-потомков.
`xType, xRate`	Соответственно тип и начальная вероятность оператора кроссинговера.
`mutationType, mRate`	Тип и начальная вероятность оператора мутации соответственно.
`populationSizerType`	Тип оператора изменения размера популяции.
`problem`	Указатель на объект, представляющий пространство задачи (используется ЭА для оценивания приспособленности особей).

Сгенерированные производящим модулем структуры данных и операторы используются в модуле, реализующем *эволюционный алгоритм*, для осуществления эволюционного поиска. При этом оценивание приспособленности особей производится путем обращения из модуля эволюционного алгоритма в модуль *пространства задачи* с передачей указателя на оцениваемую особь. В процессе работы модуля «Эволюционный алгоритм» могут быть выполнены множество независимых запусков ЭА. При этом промежуточные и конечные результаты работы ЭА в каждом запуске сохраняются в структуре `QRunSummary`, представленной в табл. 3.11. Отметим, что содержимое структуры `QRunSummary`, содержащей данные о текущем запуске ЭА, доступно для чтения всем эволюционным операторам.

Табл. 3.11. Описание полей структуры `QRunSummary`

Имя поля	Назначение
`generation`	Номер текущего поколения.

Имя поля	Назначение
`feCount`	Количество вычислений целевой функции с начала запуска ЭА.
`means, devs`	Соответственно массивы для хранения значений средней приспособленности и дисперсии приспособленности в каждом поколении.
`bests, worsts`	Соответственно массивы для хранения значений лучшей и худшей приспособленности в каждом поколении.
`popSizes`	Массив для хранения информации о размере популяции.
`startClock, finishClock, firstHitClock`	Соответственно время начала запуска, время конца запуска и время, когда было найдено первое решение.
`time, firstHitTime`	Соответственно время работы ЭА и время, необходимое для нахождения первого решения в секундах.
`firstHitFECount, firstHitGeneration`	Соответственно количество вычислений целевой функции и количество поколений, необходимых для нахождения первого решения.
`convergenceGeneration`	Номер поколения, на котором произошло вырождение популяции.
`currentBest`	Лучшая особь в текущем поколении

Полученные в результате работы модуля «Эволюционный алгоритм» структуры типа `QRunSummary` обрабатываются в *модуле обработки* для проведения статистического анализа результатов и сравнения с результатами работы ЭА с различными настройками параметров.

3.7.2. Описание структур данных

Данные об абстрактных (базовых) классах для используемых структур данных для генетического представления решений приведены в табл. 3.12.

Табл. 3.12. Описание базовых классов для генетического кодирования

Имя класса	Краткое описание
QGene	Базовый класс для кодирования информации об отдельном гене.
QIndividual	Базовый класс особи. Представляет собой контейнер для объектов класса QGene и содержит методы для записи/чтения приспособленности особи, вспомогательные методы для работы с массивом генов, а также метод для вывода информации об особи в указанный поток вывода.
QPopulation	Базовый класс популяции. Представляет собой контейнер для объектов класса QIndividual и содержит вспомогательные методы для работы с массивом особей, а также метод вывода информации о популяции в указанный поток вывода.

Определение новых типов кодирования осуществляется созданием производных классов от классов QGene, QIndividual и QPopulation. Для облегчения добавления нового типа кодирования разработаны классы-образцы. Ниже приведено описание класса-образца для нового класса гена QSomeGene, в квадратных скобках “[]” приведены участки кода, подлежащие изменению:

```
class QSomeGene : public QGene {
protected:
     [type] _value;
public:
     QSomeGene (void);
     QSomeGene (const QSomeGene& argGene);
     virtual ~QSomeGene (void);

     virtual QSomeGene& operator= (const QSomeGene& argGene);
     virtual QGene* clone () const;
     virtual QEncodingType getEncodingType () const;
     virtual bool equalsTo (const QGene* argGene) const;
     virtual int assign (const QGene* argGene);

     virtual [type] getValue () const;
     virtual int setValue (const [type] argValue);
};
```

Схожим образом выглядят образцы классов, для классов особи и популяции. Все классы для описания генетического кодирования должны перегружать метод `getEncodingType`, возвращающий используемый тип кодирования, зарегистрированный в перечислении `QEncodingType`, и метод `clone` для создания нового объекта, использующего такой же тип кодирования, что и вызываемый объект.

Также используются структуры `QEAParameters` и `QRunSummary` для хранения информации о параметрах запуска и результатах работы соответственно, описанные в табл. 3.10. и 3.11. Заполнение большинства полей структуры `QRunSummary` производится в процессе работы ЭА после оценивания особей (см. п. 3.7.5).

3.7.3. Описание классов эволюционных операторов

К эволюционным операторам относятся все классы, используемые в процессе эволюционного поиска для модификации популяции. К таким классам относятся следующие базовые классы:

- `QInitializer` – базовый класс оператора для инициализации популяции.
- `QSelection` – базовый класс оператора для селекции особей.
- `QParentSelector` – базовый класс оператора для выбора особей для скрещивания.
- `QCrossoverOperator` – базовый класс оператора скрещивания.
- `QMutationOperator` – базовый класс оператора мутации.
- `QPopulationSizer` – базовый класс оператора изменения размера популяции.

Отметим, что возможно расширение приведенного выше списка путем добавления новых операторов.

Базовым классом для всех операторов является абстрактный класс QOperator, описание которого приведено ниже:

```
class QOperator {
protected:
     const QEAParameters* _params;

public:
     QOperator (void);
     QOperator (const QOperator& argOperator);
     virtual ~QOperator (void);

     QOperator& operator= (const QOperator& argOperator);

     virtual int operate () = 0;
     virtual int setParameters (const QEAParameters& argParams);
};
```

Основным методом класса является метод `operate`, в котором должны быть реализованы особенности функционирования оператора.

Использование большинства эволюционных операторов перечисленных выше стандартно, однако работа оператора `QPopulationSizer` требует пояснений. Рассматриваемый оператор используется для изменения размера популяции путем изменения размера популяции потомков (популяция следующего поколения). Поскольку популяция потомков формируется оператором кроссинговера (`QCrossoverOperator`), то для определения точки останова в процессе формирования популяции потомком необходимо использовать оператор `QPopulationSizer`. Простейший вариант оператора `QPopulationSizer` не изменяет размер популяции, оставляя его равным начальному. Такая функциональность реализована в классе `QConstPopulationSizer`.

Пример схемы использования / взаимодействия различных операторов для генетического алгоритма и их связь со структурами данных представлен на рис. 3.9 (см. также п. 3.5.7). Блок «Параметры ЭА; Результаты» содержит данные о параметрах запуска ЭА и текущих результатах работы эволюционного алгоритма.

Для обеспечения требования модульности внутренняя логика работы операторов не зависит от других операторов. Другими словами функциони-

рование одного оператора не влияет на внутреннюю логику функционирования другого оператора (хотя влияние на результат работы возможно).

3.7.4. Описание класса задачи

Для вычисления приспособленности особей используется класс задачи `QProblem`. Использование отдельного класса для вычисления приспособленности позволяет «разделить» эволюционный поиск и особенности решаемой задачи [213]. Тем самым достигается возможность реализации единого набора классов для решения различных оптимизационных задач.

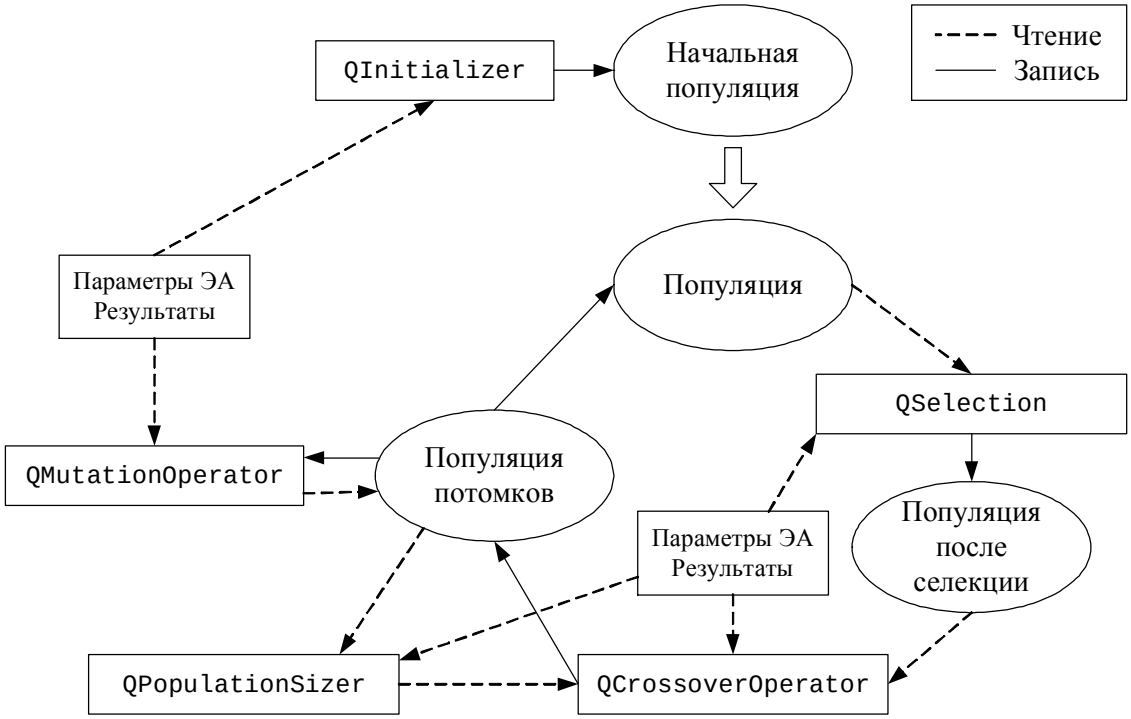


Рис. 3.9. Пример использования эволюционных операторов для генетического алгоритма. Пунктирными стрелками показаны операции чтения, сплошными – операции записи. Блок «Параметры ЭА; Результаты» указан два раза для упрощения схемы

При определении новой функции приспособленности необходимо создать класс, производный от `QProblem` (табл. 3.13) и перегрузить виртуальные методы `evaluateIndividual` и `evaluatePopulation`. Вычислен-

ное значение приспособленности должно записываться в переменную `_fitness` класса `QIndividual`. Поскольку возможны различные способы генетического кодирования решений, при реализации методов `evaluateIndividual` и `evaluatePopulation` необходимо учитывать особенности реализации конкретных способов кодирования.

Табл. 3.13. Краткое описание класса `QProblem`

Имя переменной	**Краткое описание**
`_params`	Указатель на структуру `QEAParameters`, содержащую информацию о параметрах текущего запуска ЭА.
Имя метода	**Краткое описание**
`setParameters`	Передача объекту класса `QProblem` указателя на структуру `QEAParameters`, содержащую информацию о параметрах текущего запуска ЭА.
`evaluateIndividual`	Метод для оценки приспособленности особи.
`evaluatePopulation`	Метод для оценки приспособленности популяции.

Отметим, что в реализации библиотеки рассматривается только задача *минимизации* приспособленности, поэтому при реализации значение приспособленности особи должно *уменьшаться* с ростом качества соответствующего решения.

Ниже приведен пример для класса `QOneMaxProblem` для целочисленного кодирования для известной проблемы Onemax, где требуется найти битовую строку с максимальным количеством единиц.

```
class QOneMaxProblem : public QProblem {
public:
    QOneMaxProblem (void) {}
    virtual ~QOneMaxProblem (void) {}

    virtual int evaluatePopulation (QPopulation* argPopul) {
        int i, popSize = argPopul->getSize ();
        for (i=0; i<popSize; i++) {
            evaluateIndividual (argPopul->getIndividual( i ));
        }
        return Q_OK;
```

```
    }

    virtual int evaluateIndividual (QIndividual* argInd) {
        // проверка способа кодирования для оцениваемой особи
        if (argInd->getEncodingType() == QET_INTEGER) {
            QIntegerIndividual* tempInd =
                    (QIntegerIndividual*)argInd;
            int nParams = _params->chromosomeLength;
            int er = nParams;
            double x;

            for (int i=0; i<nParams; i++) {
                // преобразование из целочисленного вида
                // к вещественному
                x = tempInd->getGeneValue(i) * _params->genePrecision
                     + _params->geneOffset;
                er -= x;
            }
            argInd->setFitness (er);
            return Q_OK;
        } else {
            cerr<<"\n\n[QOneMaxProblem::evaluatePopulation] error:\n"
                <<"\tIncorrect encoding type! (encoding ID = "
                <<argInd->getEncodingType()<<")\n\n";
            return Q_FAILURE;
        }
        return Q_OK;
    }
};
```

3.7.5. Описание класса эволюционного алгоритма

Для реализации конкретной схемы эволюционного поиска используется отдельный класс, производный от класса `QEvolutionaryAlgorithm`, в котором реализованы базовые методы для обеспечения эволюционного поиска, включающие инициализацию операторов и структур данных, а также связывание базовых операторов с данными. Последовательность использования операторов зависит от выбранного вида ЭА и схемы эволюционного поиска.

Краткое описание основных методов класса `QEvolutionaryAlgorithm` представлено в табл. 3.14.

Табл. 3.14. Краткое описание основных методов класса `QEvolutionaryAlgorithm`

Имя метода	Назначение
`setParameters`	Запись параметров запуска ЭА в структуру `QEAParameters` и создание операторов и структур данных.
`initializeComponents`	Инициализация операторов и структур данных в соответствии с содержимым структуры `QEAParameters`.
`evaluatePopulation`	Оценивание приспособленности особей в популяции и обновление полей в структуре `QRunSummary`.
`selectParentIndividuals`	Селекция особей по результатам оценивания.
`selectElite`	Выбор и запись элитных особей в популяцию потомков.
`crossPopulation`	Скрещивание особей для формирования популяции потомков. При вероятности скрещивания равной 0, популяция потомков формируется из случайных пар особей из популяции, полученной в результате селекции.
`mutatePopulation`	Мутация популяции потомков.
`nextGeneration`	Переход к следующему поколению.
`finishWork`	Удаление структур данных и операторов.
`run`	«Чистый» виртуальный метод, в котором должна быть реализована последовательность вызова эволюционных операторов для осуществления эволюционного поиска.

Ниже приведен пример реализации (перегрузки) метода `run` для класса генетического алгоритма `QGeneticAlgorithm`, работающего по схеме, изображенной на рис. 3.9:

```
#define   Q_FAILURE 0
#define   Q_OK      1

int QGeneticAlgorithm::run () {
    if (_params.generationsNumber > 0) {
        int i;
        QRunSummary summary;

        _summary.reset ();    // инициализация структуры
                              // QRunSummary
        _summary.startClock = clock ();
```

```
        initializeComponents ();   // инициализация ЭА

        for (i=0; i<_params.generationsNumber; i++) {
            evaluatePopulation ();      // оценка особей
            selectParentIndividuals (); // селекция
            selectElite ();             // выбор элитных особей
            crossPopulation ();         // скрещивание
            mutatePopulation ();        // мутация
            nextGeneration ();          // новое поколение
            _summary.generation++;
        }
        _summary.finishClock = clock ();

        return Q_OK;
    } else {
        cerr<<"\n\n[QGeneticAlgorithm::run] error:\n"
            <<"\tGenerations number is undefined!\n\n";
        return Q_FAILURE;
    }
}
```

Пример использования класса `QGeneticAlgorithm` и перегруженного метода `run` показан ниже:

```
int main () {
    QEvolutionaryAlgorithm* ea = new QGeneticAlgorithm;
    QEAParameters params;
    QProblem* problem;
    QSummaryProcessor sumProcessor;

    /**
    Чтение параметров запуска,
    заполнение структуры QEAParameters
    и создание объекта класса QProblem
    */

    // Передача параметров запуска ЭА
    problem->setParameters (&params);
    ea->setParameters (params);
    sumProcessor.setParameters (params);

    // Организация 100 запусков ЭА
    for (i=0; i<100; i++) {
        ea->run ();      // запуск ГА
        // сохранение результатов запуска
        sumProcessor.addSummary (ea->getSummary());
    }
    sumProcessor.operate ();  // обработка результатов

    ea->finishWork ();   // удаление структур данных и операторов
```

```
    // удаление объектов ЭА и задачи
    delete ea;
    delete problem;

    return 0;
}
```

3.8. Реализация нейронной сети

Для программной реализации ИНС, нейросеть представлена в виде ориентированного графа и описывается в виде списка связей. Это позволяет описать любую структуру ИНС, в том числе и с перекрестными и обратными связями. Отметим, что такой «обобщенный» подход к описанию структуры ИНС усложняет программную реализацию, поскольку при этом приходится отказаться от удобной и простой концепции многослойных ИНС, позволяющей рассматривать ИНС в виде иерархии «нейрон-слой-сеть».

Для программной реализации ИНС разработаны классы `CConnection`, `CFlexNeuron` и `CFlexNetwork`. Опишем эти классы.

Класс `CConnection` предназначен для описания межнейронной связи. Для этого в классе содержатся переменные, представленные в табл. 3.15.

Табл. 3.15. Переменные класса `CConnection`

Переменная	Описание
`type`	Тип связи: прямая или обратная. Определяется в зависимости от характеристик структуры ИНС.
`weight`	Вес связи.
`start`	Указатель на нейрон, с которого начинается данная связь.

Класс `CFlexNeuron` описывает отдельный нейрон со всеми входящими в него связями. Переменные класса описаны в табл. 3.16.

Табл. 3.16. Переменные класса `CFlexNeuron`

Переменная	Описание
`inputs`	Список объектов класса `CConnection` для определения нейронов, выходы которых связаны с данным.
`a`	Численный параметр, используемый для определения наклона (крутизны) активационной функции.
`offset`	Численный параметр, определяющий дополнительное слагаемое (смещение) к взвешенной сумме связей.
`sum`	Взвешенная сумма связей с учетом переменной смещения `offset`
`output`	Значение выхода нейрона
`index`	Уникальный идентификатор (индекс) нейрона в сети
`tag`	Вспомогательный параметр

Для хранения информации об ИНС разработан класс `CFlexNetwork`, который содержит переменную-список всех нейронов, входящих в ИНС. Также в классе `CFlexNetwork` реализованы необходимые функции для работы с ИНС, включающие вычисление выхода ИНС с произвольной структурой с различными функциями активации (поддерживаются следующие виды функции активации: линейная, сигмоидальная и гиперболический тангенс), чтение/запись информации об ИНС из потока/файла.

Классы, реализующие ИНС, интегрированы в состав библиотеки EC-Workshop и используются в программной реализации разработанного нейроэволюционного алгоритма NEvA.

3.9. Основные результаты и выводы по главе

1. Представлен разработанный НЭ алгоритм NEvA для одновременной эволюционной настройки структуры и весов связей ИНС. Экспериментальная проверка разработанного алгоритма на задачах классификации и адаптивного нейроуправления и сравнение с рядом известных алгоритмов и методов показали его эффективность как с точки зрения выбранного критерия оценки (количество вычислений целевой функции), так и с точки зрения

структуры получаемых ИНС. Также необходимо отметить высокое быстродействие алгоритма NEvA, благодаря которому среднее время решения тестовых задач имеет порядок нескольких секунд.

2. Представлены генетические операторы рекомбинации и мутации, работающие на уровне фенотипов и позволяющие использовать в НЭ алгоритме ИНС различной структуры. Это дает возможность рассмотреть модельную эволюцию структуры ИНС, начиная со структуры без скрытых нейронов, с постепенным недетерминированным увеличением количества скрытых нейронов и межнейронных связей. Использование такого подхода обеспечивает получение достаточно компактных, с точки зрения структуры, ИНС. Отметим, что использование адаптивного оператора мутации позволяет уменьшить вероятность появления ИНС с «плохой» структурой, содержащих большое число слабо связанных между собой скрытых нейронов.

3. На примере тестовых задач показано, что возможно использование фиксированного набора начальных значений параметров разработанного алгоритма NEvA для решения различных по сложности задач за счет реализации в алгоритме NEvA механизмов адаптации, позволяющих подстраивать значения параметров в зависимости от характеристик эволюционного поиска.

4. Возможность удаления в результате мутации межнейронных связей, позволяет уменьшить размерность исходной задачи за счет удаления связей от входных нейронов, соответствующих малоинформативным признакам, что показано на примере решения тестовой задачи адаптивного нейроуправления.

5. Разработана инструментальная библиотека классов для работы с эволюционными алгоритмами. Использование паттернов проектирования позволило создать расширяемую библиотеку, которая может быть легко модифицирована для решения практических задач.

ГЛАВА 4. УЛУЧШЕНИЕ КАЧЕСТВА ЦИФРОВЫХ ИЗОБРАЖЕНИЙ

В Главе 1 подчеркивалось, что использование эволюционной настройки весов связей ИНС позволяет решать трудноформализуемые задачи, для которых сложно сформировать обучающую выборку. В данной главе будет рассмотрена одна из таких задач – задача улучшения визуального качества цифровых изображений. Задача осложняется тем, что помимо отсутствия обучающей выборки значительную роль при оценке изображения играет субъективность восприятия, для исключения которой необходимо использовать критерии оценки качества общего характера. Несмотря на то, что это обстоятельство приводит к «загрублению» оценки изображения, будет показано, что ее использование для обучения ИНС с использованием разработанного НЭ алгоритма позволяет достаточно быстро получить нейросетевое решение для локально-адаптивной обработки изображения.

4.1. Вводные замечания

Обработка изображений является сложной задачей ввиду проблемы объективной оценки качества изображений, вычислительных затрат и др. Из-за огромного количества практических областей и, следовательно, разнообразия требований, предъявляемых к качеству изображений, основная цель: создание универсального алгоритма, либо подхода к улучшению качества изображений, – представляется недостижимой. Данный факт является одним из основных мотивов для исследователей к созданию новых способов обработки изображений. К настоящему времени предложено множество подходов к улучшению качества изображений [115, 138]. Некоторые из них весьма просты (например, гамма-коррекция [115]), в то время как другие достаточно сложны (к примеру, алгоритм Multi-Scale Retinex [224], использующий модель цветовосприятия человека [151]).

В общем случае проблема попиксельной обработки изображений может быть представлена в виде проблемы поиска следующей функции преобразо-

вания (или ее параметров):

$$I^* = T(I, \mathbf{\Omega}), \tag{4.1}$$

где I и I^* – интенсивность пикселей до и после обработки соответственно; $\mathbf{\Omega}$ – вектор параметров, определяющих локальные и/или глобальные характеристики каждого пикселя обрабатываемого изображения.

В данном разделе предлагается новый способ улучшения качества изображений посредством аппроксимации функции T на основании ее некоторых общих свойств. Функция T аппроксимируется с помощью искусственных нейронных сетей (ИНС), которые обучаются с использованием эволюционного алгоритма (нейроэволюционный (НЭ) подход) для попиксельной обработки изображений.

4.2. Общие положения предлагаемого способа нейросетевой обработки изображений

Будем рассматривать локально-адаптивный подход к обработке изображений. Особенность подхода заключается в независимой обработке каждого пикселя изображения, исходя из имеющегося набора его локальных и глобальных характеристик (например, локальная и глобальная средняя интенсивность) [11]. Будем использовать ИНС для попиксельной обработки яркостной составляющей изображений.

В отличие от известных подходов, где изображение поступает для обработки на вход ИНС «целиком» (неокогнитрон [100], сети Хопфилда [133] и карты Кохонена [146]), в предлагаемом способе обработки изображений ИНС обучается обработке одного пикселя. Таким образом, уменьшаются требования к объему оперативной памяти, необходимой для хранения информации об ИНС. При этом появляется возможность обрабатывать изображения произвольных размеров. Необходимо отметить, что такой подход допускает объединение множества ИНС, обрабатывающих по одному пикселю, в двухмерный массив.

Будем рассматривать преобразование T (4.1) в следующем виде:

$$L^*(x,y) = T\left(L(x,y),\ D_{(x,y)},\ m_{(x,y)}\right), \tag{4.2}$$

где $L^*(x,y)$ 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 (2) ИНС, аппроксимирующая функцию T, должна иметь 3 входа и 1 выход.

Поскольку субъективная природа человеческого восприятия затрудняет оценку визуального качества изображения, при обработке изображений всегда имеется неопределенность, вызванная непредсказуемостью реакции наблюдателя. Кроме этого, часто различные изображения должны быть обработаны по-разному с учетом контекстной информации. Поэтому выбранных локальных характеристик (яркость и ее статистические характеристики) может быть недостаточно для обработки изображения. Данные соображения ведут к заключению, что обученная ИНС «должна знать», как обрабатывать изображения в некотором абстрактном общем случае. В силу приведенных рассуждений будем обучать ИНС аппроксимации функции T*, несколько отличной от функции T, представленной выражением (4.2):

$$L^*(x,y) = T^*\left(m_{(x,y)},\ D_{(x,y)},\ \langle L\rangle\right), \tag{4.3}$$

где $\langle L\rangle$ – средняя яркость на всем изображении (глобальная средняя яркость). Таким образом, ИНС обучается работать с усредненными параметрами вместо точных.

При обработке цветных изображений, сначала изображения преобразуются в полутоновые, затем осуществляется обработка с использованием ИНС, а после этого информация о цвете восстанавливается:

$$Z_i^*(x,y) = \frac{L*(x,y)}{L(x,y)} Z_i(x,y), \tag{4.4}$$

где $Z_i^*(x,y)$ и $Z_i\,(x,y)$ – соответственно восстановленная после обработки и исходная i-е цветовые компоненты пикселя $(x\,;y)$.

4.3. Оценка функционирования ИНС

Оценка качества изображения играет ключевую роль в обучении ИНС, т.к. от нее непосредственно зависит результат НС обработки изображений. В настоящее время отсутствует общепринятый подход к оценке качества изображений, поэтому разные авторы используют различные оценки в зависимости от поставленных задач. Так при анализе алгоритмов сжатия в качестве оценок качества традиционно используются разнообразные меры попиксельного различения двух изображений: исходного и сжатого, – например, соотношение сигнал/шум (signal-to-noise ratio, *SNR*):

$$f_{SNR}(z,\hat{z}) = 10\lg\left(\frac{\sum_{x=1}^{N}\sum_{y=1}^{M}[z(x,y)]^2}{\sum_{x=1}^{N}\sum_{y=1}^{M}[z(x,y)-\hat{z}(x,y)]^2}\right),$$

где $f(x,y)$ и $\hat{f}(x,y)$ – значения интенсивности пикселя с координатами (x,y) на исходном и сжатом изображениях соответственно, N и M – ширина и высота изображения.

Оценки изображений, использующие два изображения, исходное (эталон) и измененное, называют *объективными* (*objective*), а оценки, в которых оценивается только одно изображение, – *субъективными* (*subjective*), когда отсутствует эталон, с которым можно сравнить данное изображение.

Отметим, что большинство объективных оценок качества не отражают визуальное качество изображения, а только показывают насколько одно изображение попиксельно отличается от другого без учета сюжетных сцен изображений. При этом, если $f(A,B)$ – объективная оценка изображений A и B, то часто оказывается, что если поменять местами эталон и измененное изображение, то значение оценки не позволяет сделать никаких выводов о том, как соотносятся по качеству изображения A и B. Например, значения соотношения сигнал/шум для случаев $f_{SNR}(A,B)$ и $f_{SNR}(B,A)$ без информации о самих изображениях не позволяют сказать, какое из двух изображений луч-

ше. Также, если известно, что изображение B визуально лучше, чем изображение A, а изображение C, в свою очередь визуально лучше, чем B, то, к примеру, зная $f_{SNR}(A,B)$ и $f_{SNR}(B,C)$, ничего нельзя сказать о значении $f_{SNR}(A,C)$, например, в общем случае нельзя ответить на вопрос «Верно ли, что $f_{SNR}(A,C) > f_{SNR}(A,B)$?».

Таким образом, использование для сравнения визуального качества изображений мер попиксельного различия приводит к ряду сложностей логического характера, т.к. мера различия двух изображений и их визуальное качество не обязательно взаимосвязаны.

Использование субъективных оценок визуального качества значительно осложняется отсутствием эталонного, референсного изображения, и в качестве «ориентира» в существующих субъективных оценках визуального качества изображения традиционно используются особенности световосприятия человека. В силу того, что в общем случае при анализе изображения изначально неизвестны характеристики сигнала и шума, задача субъективной оценки качества изображения является родственной с задачей слепого разделения сигналов и потому некорректна.

4.3.1. Анализ и модификация оценки Мунтеану-Роса

При выборе способа оценки качества изображения для обучения ИНС важна скорость вычисления этой оценки, т.к. в процессе обучения оценка вычисляется от нескольких сотен до нескольких тысяч раз. В связи с этим во время обучения ИНС будем использовать субъективную оценку качества.

Для выбора используемой оценки качества изображений проанализируем оценку Мунтеану-Роса (далее MP-оценка) [169], которая была успешно использована для улучшения визуального качества изображений с помощью генетического алгоритма и вычисляется по следующей формуле:

$$f_{MR} = \ln(\ln(E + e))\frac{h}{MN}\exp(H), \qquad (4.5)$$

$$H = -\sum_{i=0}^{255} l_i \ln l_i ,$$

где E – суммарная интенсивность пикселей на контурах на изображении, h – количество пикселей на контурах; N и M – соответственно ширина и высота изображения; l_i – доля пикселей с i-м уровнем яркости. Оценка Мунтеану-Роса основана на том свойстве, что для человеческого восприятия большее значение имеет перепад яркости в соседних пикселях, чем значение яркости в каждом пикселе, что позволяет сделать вывод о необходимости максимизации E и h. В то же время, присутствие большого числа уровней градации яркости на изображении и равномерность гистограммы изображения, определяющая значение H положительно сказывается на восприятии изображения [11]. Согласно MP-оценке, чем больше значение f_{MR}, тем лучше визуальное качество.

Будем сравнивать MP-оценку со следующими известными объективными оценками визуального качества изображений:

1. Оценка VIF (Visual Information Fidelity) [194]: f_{VIF}.

2. Оценка rVIF (reversed VIF): f_{rVIF}.

Оценка f_{VIF} используется для сравнения двух изображений, исходного и измененного, и вычисляется с учетом количества информации на рассматриваемых изображениях и целостности (fidelity) сигнала на измененном изображении по сравнению с исходным. Оценка f_{VIF} обладает следующими важными свойствами:

1. Интерпретируемость. Если сравниваемые изображения одинаковы, то $f_{VIF} = 1$. Если исходное изображение A лучше измененного B, то $f_{VIF}(A,B) > 1$, в противном случае $f_{VIF}(A,B) < 1$.

2. Ассиметричность. Для двух различных изображений A и B справедливо неравенство: $f_{VIF}(A,B) \neq f_{VIF}(B,A)$.

Тестирование оценки f_{VIF} на наборе из более 750 изображений показало ее высокую корреляцию с субъективным мнением, а также корректную реакцию на различные виды искажений исходного изображения: белый шум, размытие, артефакты сжатия [194].

Несмотря на достоинства оценки f_{VIF}, отмечены результаты ее некорректной работы, когда исходное изображение подвергается комплексной обработке. Для решения этой проблемы предлагается использовать перевернутый (reverse) вариант оценки f_{VIF}, обозначаемый далее как f_{rVIF}. Отличие этой оценки от f_{VIF} заключается в том, что исходное и измененное изображения «меняются местами». Например, если A и B соответственно исходное и измененное изображения, то $f_{rVIF} = f_{VIF}(B, A)$, при этом считаем, что чем меньше значение f_{rVIF}, тем лучше изображение B. Таким образом, если оценка f_{VIF} показывается насколько изображение *B лучше*, чем *A*, то оценка f_{rVIF} используется для того, чтобы показать насколько изображение *A хуже*, чем *B*.

В данной работе значение оценок f_{VIF} и f_{rVIF} вычислялось с использованием доступной на сайте разработчиков оценки f_{VIF} [180] Matlab-реализации.

Будем сравнивать рассматриваемые оценки, используя набор из более 200 изображений, которые можно условно разбить на две группы:

1. Группа **A** исходных изображений.

2. Группа **B** измененных изображений.

Изображения группы **B** сформированы путем искажения изображений из группы **A** с использованием различных преобразований как одиночных: точечного и Гауссова шума, размытия, растяжения контраста, JPEG-сжатия с потерями, преобразование гистограммы, – так и их комбинаций.

Вычисление оценок f_{MR}, f_{VIF} и f_{rVIF} будем проводить следующим образом. Пусть A_i – изображение из группы **A** и B_j – изображение из группы **B**,

полученное из A_i применением некоторого преобразования, например, добавлением шума. Тогда значения оценок f_{VIF} и f_{rVIF}:

$$f_{VIF} = f_{VIF}(A_i, B_j),$$

$$f_{rVIF} = f_{VIF}(B_j, A_i).$$

Поскольку при вычислении оценок f_{VIF} и f_{rVIF} используется два изображения, одно из которых считается исходным, то для обеспечения сопоставимости результатов объективных и субъективных оценок вместо субъективной оценки f_{MR} введем новую объективную оценку f^*_{MR}:

$$f^*_{MR} = \frac{f_{MR}(B_j)}{f_{MR}(A_i)}. \qquad (4.6)$$

Из формулы (4.6) следует, что если согласно оценке f_{MR} изображение B_j визуально лучше, чем A_i, то $f^*_{MR} > 1$, в противном случае: $f^*_{MR} < 1$.

Полученные в результате вычисления данные для каждой из оценок были нормализованы согласно формуле:

$$j_i = \frac{f_i - m_f}{s_f}, \qquad (4.7)$$

где f_i и j_i – соответственно исходное и нормализованное значения оценки, m_f и s_f – соответственно среднее и дисперсия ненормализованной оценки.

Корректность работы MP-оценки проверялась путем вычисления коэффициента корреляции r Пирсона для пар оценок (f^*_{MR}, f_{VIF}) и (f^*_{MR}, f_{rVIF}). Получены следующие значения:

$$r(f^*_{MR}, f_{VIF}) = -0{,}0323,$$

$$r(f^*_{MR}, f_{rVIF}) = -0{,}0737.$$

Абсолютные значения r очень малы, но, как известно, малое абсолютное значение коэффициента корреляции свидетельствует об отсутствии только линейной зависимости между величинами, и при этом возможно сущест-

вование нелинейной зависимости. Проверим это, изменив соотношение между величинами E, h и H.

Заметим, что в формуле (4.5) для MP-оценки значения E, h и H вносят различный вклад в результат. Так количество пикселей на контурах ограничено сверху единицей и имеет наименьший вес, энтропия распределения уровней яркости H ограничена сверху 256. Суммарная интенсивность краевых пикселей E может неограниченно расти, но практически имеет порядок $O(1)$ ввиду медленного роста логарифмической функции.

Вместо мультипликативной оценки (4.5) введем аддитивную оценку визуального качества изображения:

$$f_{\text{mod}} = y_1(E) + y_2(h) + y_3(H),$$

где $y_1(E)$, $y_2(h)$ и $y_3(H)$ – некоторые функции зависящие от своих аргументов. Ввиду того, что визуальное качество изображения не может увеличиваться до бесконечности, потребуем, чтобы значение функции f_{mod} уменьшалось с ростом визуального качества изображения. При этом вид функций y_i должен быть таким, чтобы обеспечивать такое соотношение между величинами E, h и H, которое было бы характерно и человеческому восприятию. Экспериментальным путем, с помощью сопоставления значения функции f_{mod} с субъективным восприятием и объективными оценками f_{VIF} и f_{rVIF} был определен следующий вид функции f_{mod}:

$$f_{\text{mod}} = \frac{MN - h}{MN} + \frac{256 - \exp(H)}{192} + \frac{255 - E/h}{255}. \qquad (4.8)$$

Исследование оценки проводилось с использованием той же серии изображений, что и для оценки f_{MR} и для сопоставимости результатов с результатами для оценок f_{VIF} и f_{rVIF} вычислялись значения коэффициента:

$$f^*_{\text{mod}} = \frac{f_{\text{mod}}(A_i)}{f_{\text{mod}}(B_j)}.$$

Также как и в случае коэффициента f^*_{MR}, если $f^*_{\mathrm{mod}} > 1$, то согласно оценке f_{mod} изображение B_j визуально лучше, чем изображение A_i, а если $f^*_{\mathrm{mod}} < 1$, то изображение B_j визуально хуже, чем изображение A_i. Полученные значения для коэффициента f^*_{mod} были нормализованы по формуле (4.7) и для нормализованных данных вычислены коэффициенты корреляции с оценками f_{VIF} и f_{rVIF}, приведенные в табл. 4.1. Для сравнения в табл. 4.1 также представлены значения коэффициентов корреляции для MP-оценки.

Табл. 4.1. Значения коэффициентов корреляции для пар оценок (f^*_{MR}, f_{VIF}), (f^*_{MR}, f_{rVIF}), (f^*_{mod}, f_{VIF}) и (f^*_{mod}, f_{rVIF})

	f_{VIF}	f_{rVIF}
f^*_{MR}	-0,0323	-0,0737
f^*_{mod}	0,1504	-0,6718

Из приведенных в табл. 4.1 данных видно, что использование аддитивной оценки качества изображений позволяет значительно усилить корреляцию между субъективной оценкой качества f_{mod} и объективными оценками качества f_{VIF} и f_{rVIF}, которые, в свою очередь, хорошо согласуются с субъективными оценками визуального качества изображения [194]. Значения коэффициента корреляции для пар оценок (f^*_{mod}, f_{VIF}) и (f^*_{mod}, f_{rVIF}) значимы с уровнем значимости 0,05, в то время как значения коэффициента корреляции для пар оценок (f^*_{MR}, f_{VIF}) и (f^*_{MR}, f_{rVIF}) не значимы с этим же уровнем значимости, на основании критерия значимости:

$$T = r\sqrt{\frac{n-2}{1-r^2}},$$

где n – количество наблюдений, равное в данном исследовании 246.

На основании полученных результатов будем использовать оценку (4.8) для оценивания качества изображения для обучения ИНС. Таким образом, качество обработанного изображения оценивается в зависимости от двух достаточно общих факторов:

1. Общее количество и характеристики пикселей на контурах на оцениваемом изображении.
2. Распределение различных уровней яркости на оцениваемом изображении.

Оценивание функционирования ИНС осуществляется по последовательности $N * M$ ее выходных сигналов. Формула (4.8) представляет довольно грубую оценку качества изображения, так как в ней рассматривается только контрастность и распределение яркости обработанного изображения. Однако, как будет показано далее, такой подход достаточен для эволюционной настройки структуры и весов ИНС.

Отметим, что возможно обучение ИНС на наборе изображений. В этом случае оценка каждой ИНС вычисляется как средняя оценка обработанных изображений. После обучения полученная ИНС может быть использована для обработки изображений, отсутствовавших в обучающей выборке. Таким образом, сохраняется «классическая» методология обучения ИНС. Данное свойство позволяет сократить время обработки изображений из-за отсутствия необходимости в обучении ИНС обработке каждого нового изображения.

4.3.2. Ранняя проверка ИНС

При вычислении оценки ИНС (4.8) используется только общая информация о распределении яркости на обработанном изображении. Такой подход приводит к тому, что ИНС, преобразующая данное изображение в негатив, также может иметь высокую оценку качества, так как при вычислении выражения (4.8) не учитывается соответствие темных и светлых областей на исходном и обработанном изображении.

Чтобы избежать использования ИНС, инвертирующих яркость на изображении, предлагается перед вычислением (4.8) использовать раннюю проверку ИНС. Для этого рассмотрим преобразование однотонных темного и светлого изображений с использованием оцениваемой ИНС и потребуем, чтобы в результате нейросетевого преобразования первое изображение не стало светлым, а второе – темным. При этом, очевидно, достаточно рассмотреть преобразование яркости только одного пикселя от темного и светлого изображения. Таким образом, считаем, что ИНС не прошла раннюю проверку, если выполняется хотя бы одно из следующих условий:

$$net(0{,}9;\ 0;\ 0{,}9) < 0{,}25,$$

$$net(0{,}1;\ 0;\ 0{,}1) > 0{,}75,$$

где $net\left(m_{(x,y)};\ D_{(x,y)};\ \langle L\rangle\right)$ – значение выходного сигнала ИНС соответствующее нормированному значению преобразованной яркости пикселя в зависимости от нормированных значений локальных средней $m_{(x,y)}$ и дисперсии $D_{(x,y)}$ яркости, а также от средней яркости $\langle L\rangle$ всего изображения. Границы 0,25 и 0,75 выбраны по результатам экспериментов.

ИНС, не прошедшая раннюю проверку, не оценивается с использованием изображений для обучения, а сразу получает наихудшую оценку и, тем самым, соответствующая ей особь не принимает участие в скрещивании. Отметим, что использование ранней проверки позволяет ускорить процесс обучения популяции ИНС за счет исключения из процесса оценивания «инвертирующих» ИНС.

4.4. Трехэтапная обработка

Анализ получаемых НС решений [37] и практические исследования показали необходимость использования пред- и постобработки изображений. В результате проведенного исследования, для предварительной обработки была выбрана эвристическая мультипликативная подстройка яркости исходного изображения:

$$\mathbf{L} = a\mathbf{L}_0,$$

$$a = \begin{cases} \sqrt{\dfrac{97}{\langle L_0 \rangle}}, \langle L_0 \rangle < 97, \\ 1,\ 97 \le \langle L_0 \rangle \le 157, \\ \sqrt{\dfrac{157}{\langle L_0 \rangle}}, \langle L_0 \rangle > 157, \end{cases}$$

где матрицы $\mathbf{L}_0$ и $\mathbf{L}$ описывают распределение яркости на изображении соответственно до и после предобработки, $\langle L_0 \rangle$ – средняя яркость исходного изображения. Границы 97 и 157 выбраны по результатам экспериментов.

Согласно преобразованию (4.1) ИНС обрабатывает пиксели с использованием локальных характеристик, поэтому для более эффективной обработки представляется разумным применение, в качестве алгоритма для следующего этапа обработки, «глобального» алгоритма улучшения качества изображений. Для этой цели предлагается использовать алгоритм автоматической настройки уровней яркости, реализованный во многих графических пакетах, и заслуживший признание, благодаря своему качеству и скорости работы. Согласно этому алгоритму, каждый цветовой канал изображения обрабатывается отдельно и интенсивность z_{ij} обработанного пикселя определяется по следующей формуле:

$$z_{ij} = \begin{cases} 0,\ z_{ij} < b_0, \\ 255\dfrac{z_{ij} - b_0}{b_1 - b_0},\ b_0 \le z_{ij} \le b_1, \\ 255,\ z_{ij} > b_1, \end{cases}$$

где b_0 и b_1 – соответственно нижняя и верхняя границы «отсечения» по гистограмме яркости, такие что:

$$b_0 = \max k : \sum_{i=0}^{k} r_i < 0{,}005,$$

$$b_1 = \min c : \sum_{i=c}^{255} r_i < 0{,}005,$$

где r_i – доля пикселей на исходном изображении с i-м уровнем интенсивности, и, очевидно, $\sum r_i = 1$. При этом границы b_0 и b_1 вычисляются отдельно для каждого канала.

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.

Преобразование яркости, соответствующее предлагаемому трехэтапному способу обработки, можно схематично представить следующим образом:

$$\mathbf{L}_0 \xrightarrow{1} \mathbf{L} \xrightarrow{2} \mathbf{L}^* \xrightarrow{3} \mathbf{L}_1,$$

где $\mathbf{L}_1$ описывает распределение яркости на обработанном изображении, полученное после третьего этапа, а символ $\xrightarrow{k}$ обозначает применение k-го этапа обработки.

4.5. Приближенное вычисление локальных характеристик

В соответствии с предлагаемым подходом, применение обученных ИНС подразумевает использование локальных средней и дисперсии яркости (4.2), поэтому время обработки изображений существенно зависит от скорости вычисления этих характеристик. Ясно, что с увеличением размера окрестности вычислительная сложность также увеличивается, так как увеличивается количество обрабатываемых пикселей.

Далее описывается вывод приближенных формул для локальных средней и дисперсии яркости в прямоугольной окрестности.

Считаем, что распределение яркости $L = \{l_{ij} \mid i = 1,..., M,\ j = 1,..., N\}$, где l_{ij} – яркость пикселя, находящегося на пересечении i-й строки и j-го столбца

изображения, задает совместное распределение двумерной случайной величины (X, Y), где $X \in [1;N]$, $Y \in [1;M]$:

$$p_{xy}(i,j) = l_{ij} \Big/ \sum_i \sum_j l_{ij}, \tag{4.9}$$

где $p_{xy}(i,j)$ – значение совместной плотности распределения в точке с прямоугольными координатами (j; i). Тогда плотности распределения случайных величин X и Y:

$$p_x(j) = \sum_i p_{xy}(i,j),$$

$$p_y(i) = \sum_j p_{xy}(i,j),$$

Полагаем, что окрестности каждого пикселя имеют прямоугольную форму и ограничены точками (j_1; i_1) и (j_2; i_2), причем $i_2 \geq i_1$ и $j_2 \geq j_1$. Используя понятие условной вероятности, а также предполагая скореллированность распределения яркости в соседних строках (то есть рассматриваем случай «гладкого» распределение яркости на изображении), запишем:

$$\sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} p_{xy}(i,j) = \sum_{i=i_1}^{i_2} p_y(i) \sum_{j=j_1}^{j_2} p_x(j \mid i) \approx l_y \sum_{j=j_1}^{j_2} p_x(j \mid y_l) \sum_{i=i_1}^{i_2} p_y(i), \tag{4.10}$$

$$l_y = \frac{\sum_{i=i_1}^{i_2} p_y(i)}{(i_2 - i_1 + 1) p_y(y_l)},$$

где y_l номер некоторой строки в промежутке [i_1 ; i_2], а l_y – коэффициент пропорциональности. В результате несложных выкладок получим:

$$\sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} p_{xy}(i,j) \approx l_y \frac{\sum_{i=i_1}^{i_2} p_y(i)}{p_y(y_l)} \sum_{j=j_1}^{j_2} p_x(j \mid y_l) p_y(y_l) = l_y^2 (i_2 - i_1 + 1) \sum_{j=j_1}^{j_2} p_{xy}(y_l, j).$$

Используя аналогичные соображения легко получить приближенные значения первого и второго начальных моментов, а затем и приближенные формулы для среднего $\tilde{m}_{(x,y)}$ и дисперсии $\tilde{D}_{(x,y)}$:

$$\tilde{m}_{(x,y)} = \frac{l_y^2}{(j_2 - j_1 + 1)} \sum_{j=j_1}^{j_2} p_{xy}(y_l, j), \tag{4.11}$$

$$\tilde{D}_{(x,y)} = \frac{l_y^2}{j_2 - j_1 + 1} \left(\sum_{j=j_1}^{j_2} p_{xy}(y_l, j) \left(b_y p_{xy}(y_l, j) - \tilde{m}_{(x,y)} \right) \right), \tag{4.12}$$

$$b_y = \frac{\sum_{i=i_1}^{i_2} p_y^2(i)}{(i_2 - i_1 + 1) p_y^2(y_l)},$$

Заметим, что коэффициенты l_y и b_y являются отношением усредненной величины к частному значению этой величины. Тогда, предполагая достаточно малый размер окрестности, делаем приближения $l_y = 1$ и $b_y = 1$, что позволяет упростить формулы (4.11) и (4.12):

$$\tilde{m}_{(x,y)} = m_{(x)}(y_l) = \frac{\sum_{j=j_1}^{j_2} p_{xy}(y_l, j)}{(j_2 - j_1 + 1)}, \tag{4.13}$$

$$\tilde{D}_{(x,y)} = D_{(x)}(y_l) = \frac{\sum_{j=j_1}^{j_2} p_{xy}^2(y_l, j)}{j_2 - j_1 + 1} - m_{(x)}^2(y_l). \tag{4.14}$$

Таким образом, возможно приближенное вычисление локальных среднего и дисперсии для двумерной прямоугольной области по произвольной строке из этой области, используя предположение о коррелированности распределений яркости в соседних строках и рассматривая окрестность достаточно малого размера. Отметим, что формулы аналогичные (4.13) и (4.14) могут быть выведены и для произвольного столбца рассматриваемой окрестности.

Чтобы скомпенсировать потерю информации, вызванную вычислением локальных характеристик двумерной области по одномерному массиву, соответствующему произвольной строке или столбцу пикселей, будем использовать следующие формулы:

$$\tilde{m}_{(x,y)} = \frac{m_{(x)}(y_l) + m_{(y)}(x_k)}{2}, \tag{4.15}$$

$$\tilde{D}_{(x,y)} = \frac{D_{(x)}(y_l) + D_{(y)}(x_k)}{2}. \tag{4.16}$$

где $y_l \in [i_1;\ i_2],\ x_k \in [j_1;\ j_2]$.

Чтобы избежать резких перепадов при вычислении средней яркости соседних пикселей, будем использовать среднюю яркость на всем изображении в качестве «стабилизирующего слагаемого»:

$$\tilde{m}_{(x,y)} = \frac{m_{(x)}(y_l) + m_{(y)}(x_k) + \langle L \rangle}{3}, \tag{4.17}$$

Чтобы обобщить результат, полученный в данном разделе, отметим, что таким образом удалось уменьшить вычислительную сложность вычислений локальных характеристик с $O(n^2)$ до $O(n)$. Дополнительное исследование (см. п. 4.5.1) показало, что для набора искусственных и реальных изображений соотношение сигнал-шум при использовании формул (4.15) и (4.16) составляет от 20 дБ и выше для приближенного среднего, и от 30 дБ и выше для приближенной дисперсии, что вполне удовлетворительно, принимая во внимание рассуждения из п. 4.2 настоящей главы об отсутствии необходимости в обработке точных значений локальных характеристик ввиду субъективной природы человеческого восприятия и большого разнообразия изображений. Использование приближенных формул позволяет ускорить вычисления более чем в 30 раз для окрестности размером 65x65 пикселей.

4.5.1 Экспериментальное исследование точности приближенных формул для вычисления локальных характеристик

Для оценки точности формул (4.15) и (4.16) будем сравнивать результаты, полученные с использованием этих формул, с результатами «стандартных» формул для среднего и дисперсии в означенной окрестности:

$$m_{(x,y)} = \frac{\sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} p_{ij}}{S}, \tag{4.18}$$

$$D_{(x,y)} = \frac{1}{S} \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} \left(p_{ij}\right)^2 - m_{(x,y)}^2, \tag{4.19}$$

$$S = (i_2 - i_1 + 1)(j_2 - j_1 + 1).$$

Для экспериментов будем использовать два набора полутоновых сгенерированных изображений и один набор реальных. Для наборов сгенерированных изображений $\mathbf{P}_1 = \{P_{1,i}\}$ и $\mathbf{P}_2 = \{P_{2,i}\}$, $i = 1,...,11$, (рис. 4.1 и 4.2) изображения $P_{1,1}$ и $P_{1,2}$ совпадают (другими словами, яркости соответствующих пикселей изображений $P_{1,1}$ и $P_{1,2}$ равны) и генерируются случайно, а все последующие изображения определяются следующим образом:

$$P_{1,i} = \aleph(P_{1,1}, i-1), \ i = 2,...,11,$$

$$P_{2,i} = \aleph(P_{2,i-1}, 1), \ i = 2,...,11,$$

где $\aleph(P, r)$ обозначает применение к изображению P медианной фильтрации с радиусом r.

Таким образом, изображения из набора $\mathbf{P}_1$, начиная со второго, получены применением к изображению $P_{1,1}$ медианного фильтра с постепенно увеличивающимся радиусом, а изображения из второго набора – применением к предыдущим изображениям медианного фильтра с единичным радиусом. Полученные таким образом наборы сгенерированных изображений представлены на рис. П4.1 и П4.2. Набор $\mathbf{P}_3 = \{P_{3,i}\}$, $i = 1,...,16$ из 16 реальных изображений представлен на рис. П4.3.

По рис. 4.1 и 4.2 видно, что перепады яркости на последующих изображениях менее резкие, чем на предшествующих. Таким образом, согласно использованному при выводе формул (4.15) и (4.16) предположению, о плавном изменении яркости на изображении, можно ожидать, что с ростом индексов изображений $P_{1,i}$ и $P_{2,i}$ формулы (4.18) и (4.19) будут давать все более точные

результаты. В то же время, при увеличении радиуса окрестности, погрешность приближенных формул также должна возрастать, т.к. увеличивается количество пикселей, яркости которых не учитываются при расчетах.

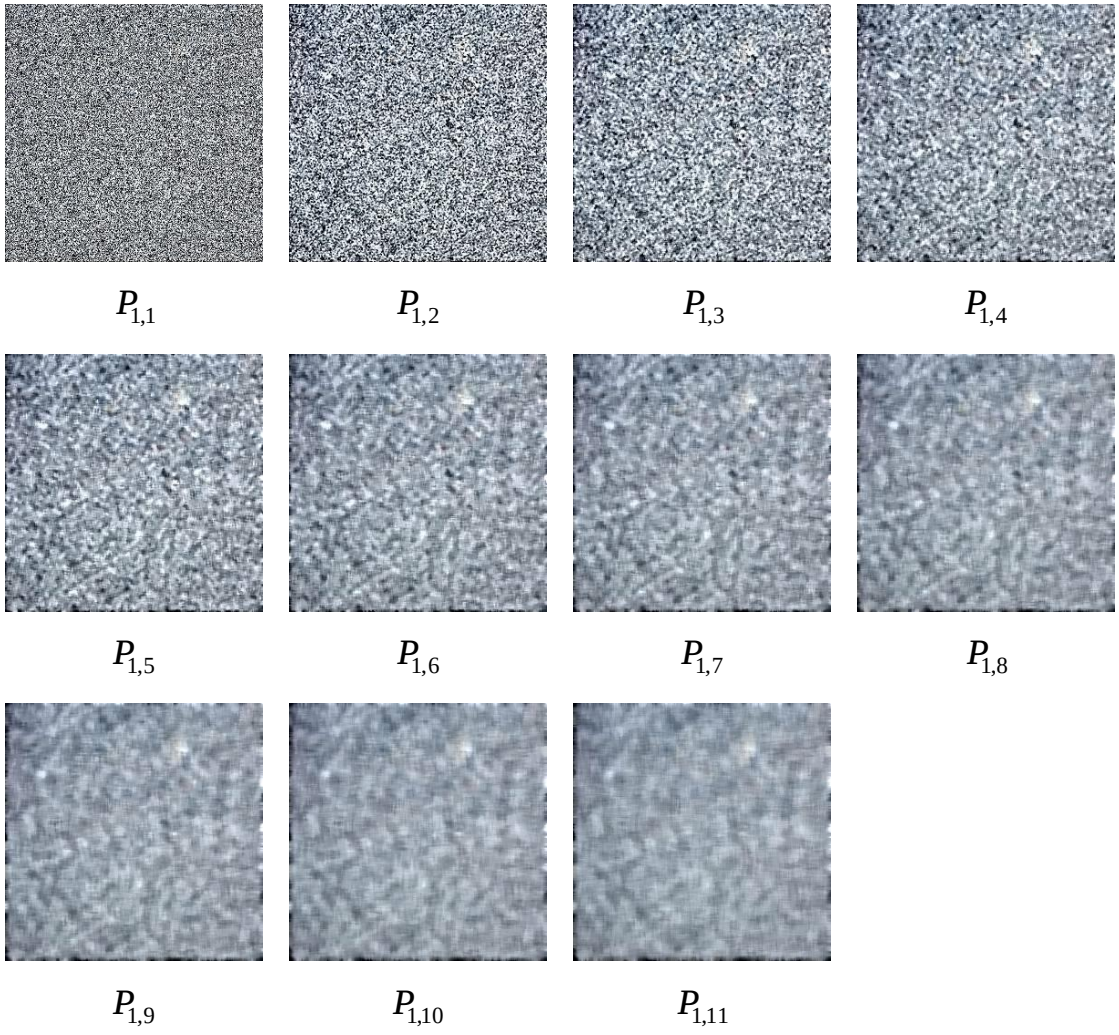


Рис. 4.1. Сгенерированные изображения из набора $\mathbf{P}_1$. Представлены последовательно (с ростом индекса) слева-направо, сверху-вниз. Подробности см. в тексте.

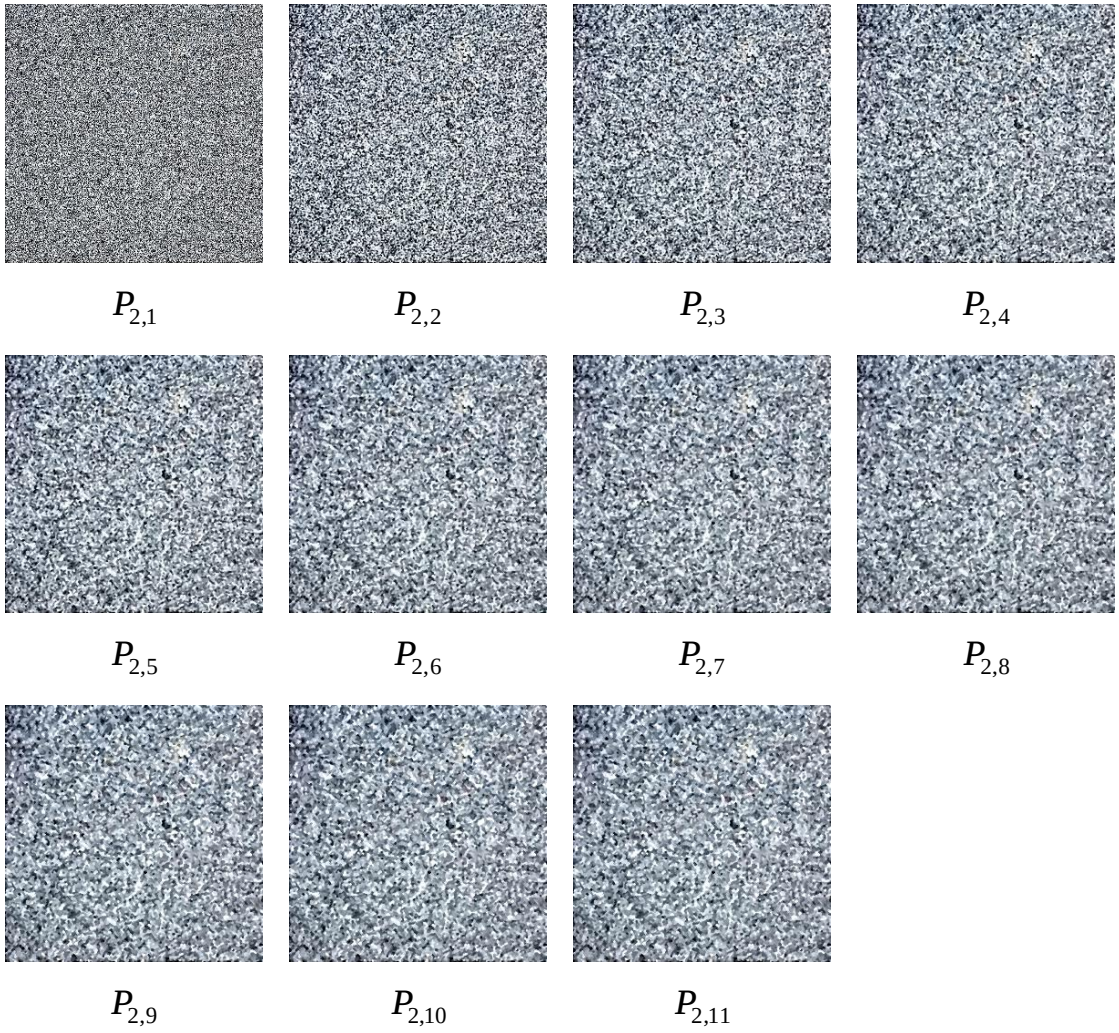


Рис. 4.2. Сгенерированные изображения из набора $\mathbf{P}_2$. Представлены последовательно (с ростом индекса) слева-направо, сверху-вниз. Подробности см. в тексте.

Для исследования формул (4.15) и (4.16) будем рассматривать окрестности радиусом $r \in \{1,3,7,12,16,22,32\}$, при этом будем выбирать следующие значения y_l и x_k: $y_l = y$ и $x_k = x$, где x и y координаты точки в центре окрестности (т.е. координаты точки, для которой построена рассматриваемая окрестность). Будем оценивать время вычислений и соотношение сигнал/шум:

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{MN}\sum_{i,j}\left(z_{1,ij} - \tilde{z}_{1,ij}\right)^2},$$

где M и N – соответственно ширина и высота изображения, а $z_{1,ij}$ и $\tilde{z}_{1,ij}$ обозначают соответственно значения характеристик яркости, вычисленных точно (по формулам (4.18) и (4.19)) и приближенно (по формулам (4.15) и (4.16)).

$P_{3,1}$ $P_{3,2}$ $P_{3,3}$ $P_{3,4}$

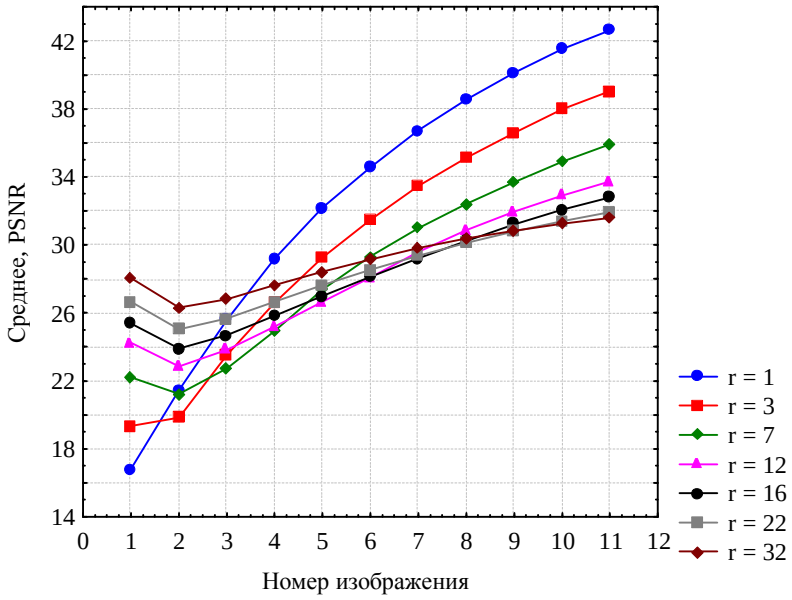
$P_{3,5}$ $P_{3,6}$ $P_{3,7}$ $P_{3,8}$

$P_{3,9}$ $P_{3,10}$ $P_{3,11}$ $P_{3,12}$

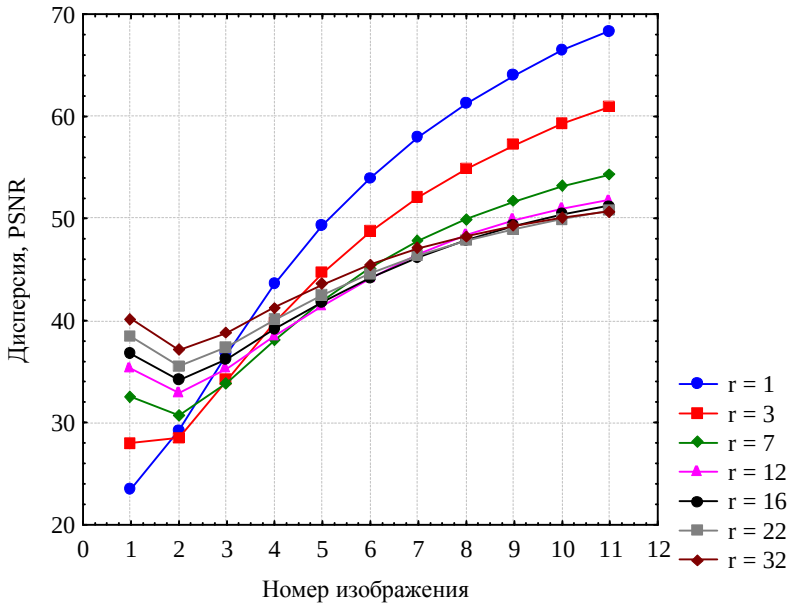
$P_{3,13}$ $P_{3,14}$ $P_{3,15}$ $P_{3,16}$

Рис. 4.3. Реальные изображения. Изображения пронумерованы последовательно слева-направо, сверху-вниз.

Графики соотношений сигнал/шум для различных значений радиуса окрестности r (в пикселях) представлены на рис. 4.4-4.6.

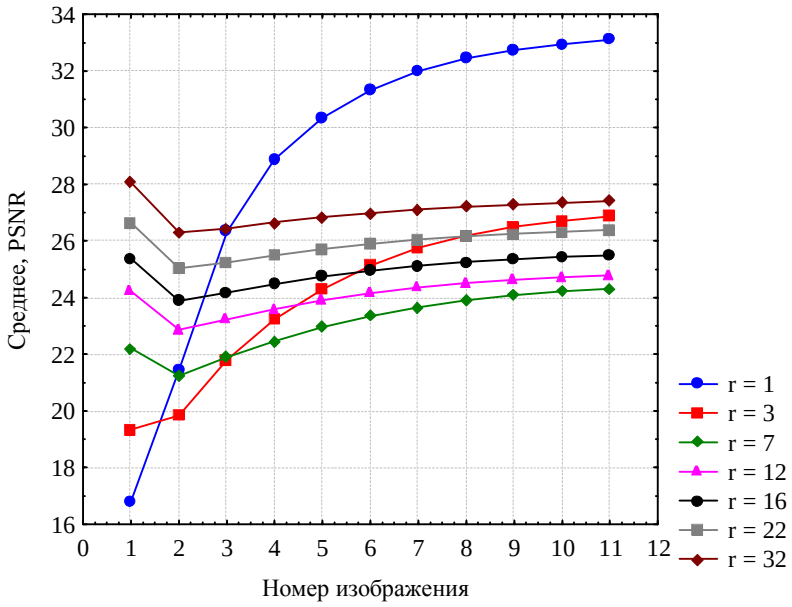


а)

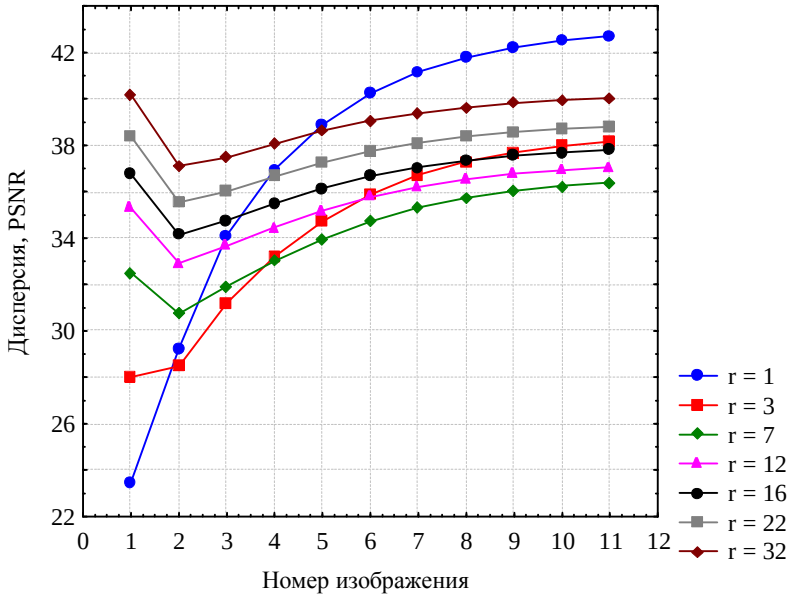


б)

Рис. 4.4. Соотношение сигнал-шум для приближенных локальных среднего (а) и дисперсии (б) яркости, вычисленных по формулам (4.15) и (4.16) на первом наборе изображений (рис. 4.1)

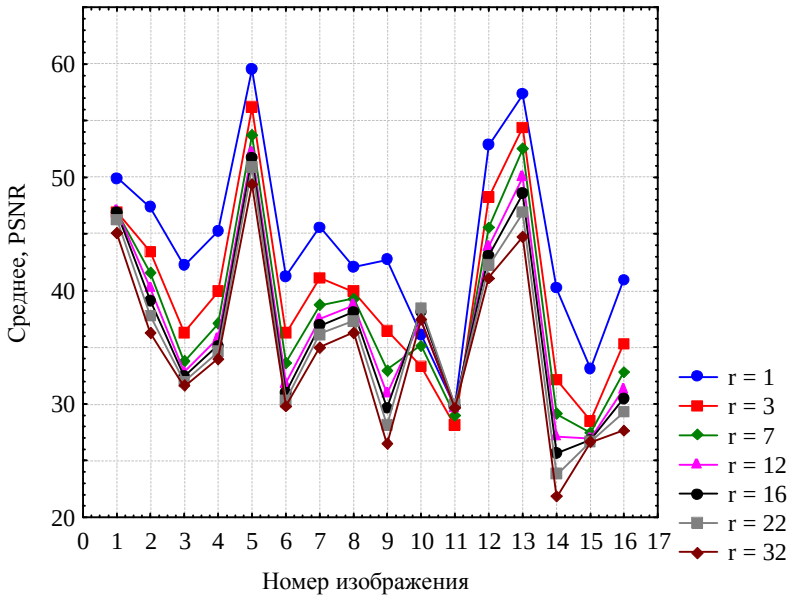


а)

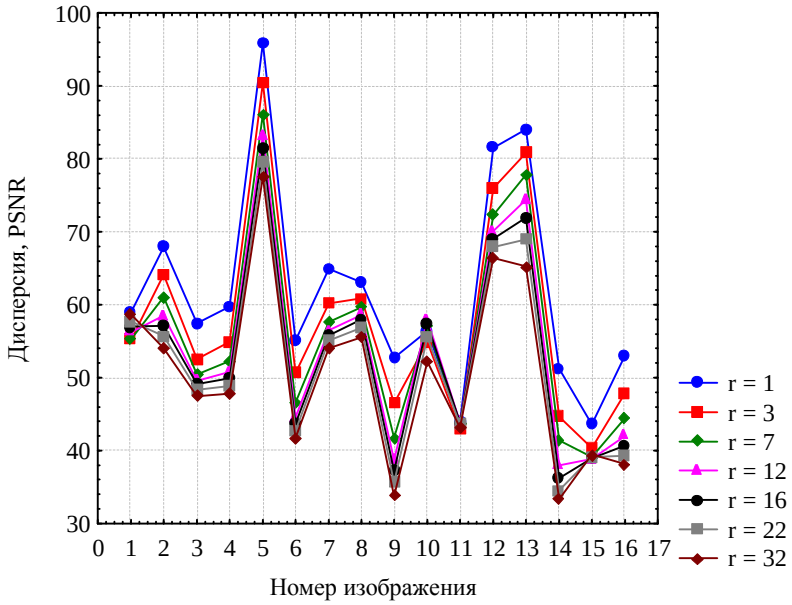


б)

Рис. 4.5. Соотношение сигнал-шум для приближенных локальных среднего (а) и дисперсии (б) яркости, вычисленных по формулам (4.15) и (4.16) на втором наборе изображений (рис. 4.2)



а)



б)

Рис. 4.6. Соотношение сигнал-шум для приближенных локальных среднего (а) и дисперсии (б) яркости, вычисленных по формулам (4.15) и (4.16) на третьем наборе изображений (рис. 4.3)

Анализ результатов, показанных на рис. 4.4-4.6 целесообразно разбить на две части: для сгенерированных и реальных изображений.

По графикам на рис. 4.4, 4.5 видно, что при уменьшении перепадов яркости на изображении, что соответствует увеличению номеров изображений, точность приближенного вычисления средней и дисперсии яркости увеличивается. При этом на графиках присутствует «провал», соответствующий изображениям $P_{1,2}$ и $P_{2,2}$. Данный «провал» можно объяснить следующим предположением.

В случае, когда яркость всех пикселей случайна (изображения $P_{1,1}$ и $P_{2,1}$), распределение яркостей пикселей в соседних строках (столбцах) примерно одинаково и равномерно, хоть и декоррелированно. В этом случае при увеличении радиуса окрестности, «похожесть» распределений будет, очевидно, только увеличиваться (т.к. увеличивается число пикселей в каждой строке и каждом столбце), и, следовательно, будет расти и точность приближенных формул (см. графики на рис. 4.4, 4.5 для изображений с номером 1). При увеличении «упорядоченности» распределения пикселей, например, с помощью использованного медианного фильтра, распределения яркости в соседних строках (столбцах) начинают коррелировать. При этом нарушается равномерность распределения яркостей пикселей в пределах одной строки или столбца (и тем более на их участках, лежащих внутри окрестности), что приводит к снижению точности вычислений по сравнению с вычислениями для случайных изображений. По всей видимости, этот факт обуславливает характерные «провалы» на графиках на рис. 4.4 и 4.5, которые особенно сильно проявляются в случае окрестностей со сравнительно большим радиусом и отсутствуют в случае использования окрестностей малого радиуса.

В дальнейшем, при еще большем «упорядочивании» яркости пикселей (в данном случае посредством медианного фильтра) размер окрестности, на которую влияет яркость каждого пикселя, увеличивается, что приводит к перераспределению яркости на изображении. Это перераспределение выражается, например, в последовательном увеличении «серости» изображений из первого набора (рис. 4.1), а также в образовании достаточно «устойчивых» темных и

белых областей для изображений из второго набора (рис. 4.2). Таким образом, в наборах сгенерированных изображений, распределение яркости с увеличением номера изображения становится все более «плавным», что способствует увеличению точности приближенных формул. С другой стороны, уменьшение случайности распределения яркости отрицательно влияет на точность формул (4.1) и (4.2) при увеличении радиуса окрестности, т.к. одна строка или столбце, принадлежащие этой окрестности, уже недостаточно точно характеризуют распределение яркости в этой окрестности. В случае набора изображений $\mathbf{P}_2$ данный фактор привел к тому, что результаты вычислений по приближенным формулам для изображений с номерами 5 и выше в случае окрестностей достаточно большого радиуса не превышают либо незначительно превосходят точность результатов для случайного изображения $P_{2,1}$.

Результаты экспериментов на третьем наборе изображений и приведенные рассуждения дают некоторые обоснования к выбору радиуса окрестности при использовании приближенных формул (4.15) и (4.16). Малый радиус окрестности оказывается предпочтительнее, т.к. в большинстве случаев это обеспечивает более высокую точность вычислений по сравнению с использованием большего радиуса окрестности. Однако на зашумленных изображениях и на изображениях с большим количеством перепадов яркости (изображения $P_{3,10}$ и $P_{3,11}$ на рис. 4.3, см. также соответствующие участки графиков на рис. 4.6) использование большего радиуса окрестности дает более точный результат. Поэтому предположим, что уместно использование окрестностей радиусом $r = 2,...,7$. Значение $r = 1$ не рассматривается, т.к. в этом случае выигрыш по времени вычислений практически отсутствует и целесообразно использовать точные формулы (4.18) и (4.19).

График усредненного относительного выигрыша по времени $n = \dfrac{t(r)}{\tilde{t}(r)}$, где $t(r)$ и $\tilde{t}(r)$ соответственно время точного и приближенного вычисления

среднего и дисперсии в зависимости от радиуса r окрестности, показан на рис. 4.7.

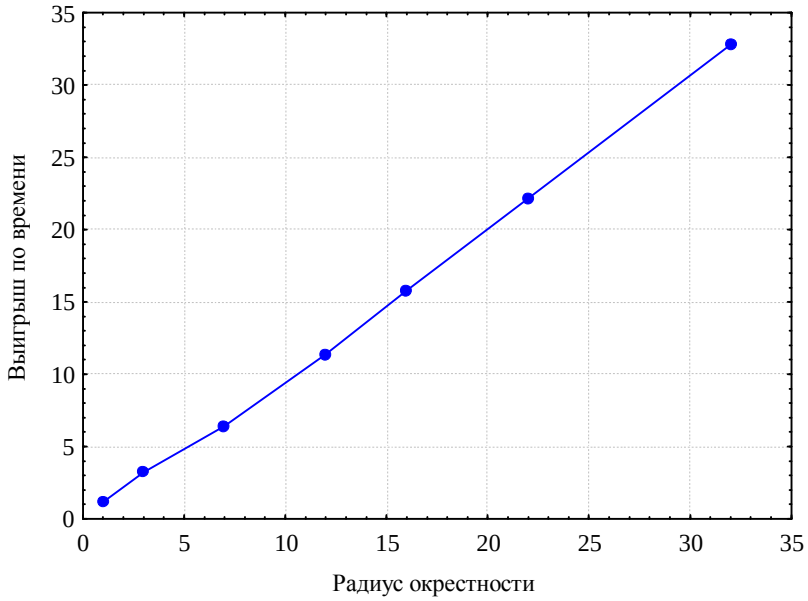


Рис. 4.7. Средний относительный выигрыш по времени $n = \frac{t(r)}{\tilde{t}(r)}$, где $t(r)$ и $\tilde{t}(r)$ соответственно время точного и приближенного вычисления локальных среднего и дисперсии в зависимости от радиуса r окрестности, при использовании формул (4.15) и (4.16) по сравнению с точными формулами (4.18) и (4.19) в зависимости от радиуса окрестности

В целом, по результатам экспериментов видно, что точность вычислений для приближенного среднего находится в большинстве случаев выше 20 дБ. Точность приближенного вычисления дисперсии более 30 дБ. Относительный выигрыш во времени вычислений по сравнению с использованием точных формул (4.18) и (4.19) линейно зависит от радиуса окрестности. Поскольку сложность точных вычислений имеет порядок $O(r^2)$, то сложность вычислений по приближенным формулам (4.15) и (4.16) можно оценить как $O(r)$. Последний вывод можно сделать также на основе сравнительного анализа формул (4.15) и (4.16) и формул (4.18) и (4.19), поскольку для точных формул количество операций сложения квадратично зависит от размеров ок-

рестности, а в случае использования приближенных формул эта зависимость линейна.

4.6 Тестирование трехэтапного способа обработки изображений

4.6.1 Описание экспериментов

Напомним, что в соответствии с локально-адаптивным подходом к обработке изображений рассматривается нейросетевая аппроксимация преобразования (4.2):

$$L^*(x,y) = T\left(L(x,y),\ \tilde{D}_{(x,y)},\ \tilde{m}_{(x,y)}\right),$$

Для уменьшения неопределенности, возникающей при обработке изображений, будем обучать ИНС аппроксимации преобразования (4.3):

$$L^*(x,y) = T\left(m_{(x,y)},\ D_{(x,y)},\ \langle L\rangle\right),$$

где характеристики $m_{(x,y)}$ и $D_{(x,y)}$ вычисляются точно для окрестности 3x3, имеющей наименьший радиус, что позволяет ускорить процесс обучения.

Для подсчета числа пикселей на границах областей различной яркости, необходимого для оценивания (4.8) изображения, обработанного ИНС, будем использовать ускоренный вариант детектора края Собеля, описанный в [19]. Поскольку выражение (4.8) дает приближенную оценку качества изображения, то нет необходимости минимизировать его до 0. В результате экспериментов установлено, что приемлемые результаты обработки достигаются, как правило, при значениях целевой функции f, лежащих в диапазоне [2,4; 2,85]. При меньших значениях f обработанные изображения становятся слишком контрастными с потерей деталей в светлых и темных областях, а при больших f получаются изображения со слабым контрастом. В качестве критерия останова будем использовать f_0 = 2,5.

Длительность эволюции составляет 25 поколений. Начальные значения остальных параметров алгоритма NEvA совпадают с использованными при тестировании эффективности алгоритма в Главе 3 и представлены в табл. 3.1. Напомним, что это сделано для экспериментальной проверки возможности адаптации алгоритма к решаемой задаче, что необходимо для уменьшения времени настройки алгоритма и повышения эффективности его использования. Для тестирования используется приложение QImager (см. п. 4.8).

Для обучения ИНС будем использовать изображение, показанное на рис. 4.8. Размеры изображения выбраны небольшими, чтобы повысить скорость обучения, и равны 128x128 пикселей.

Рис. 4.8. Изображение, использованное для обучения ИНС. Черная рамка вокруг изображения добавлена специально для облегчения визуального восприятия.

4.6.2 Результаты экспериментов

Время обучения составило около 80 секунд. Структура полученной ИНС представлена на рис. 4.9.

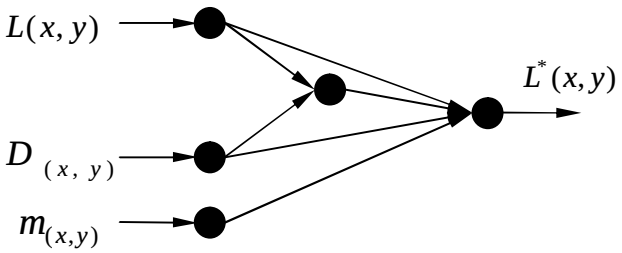
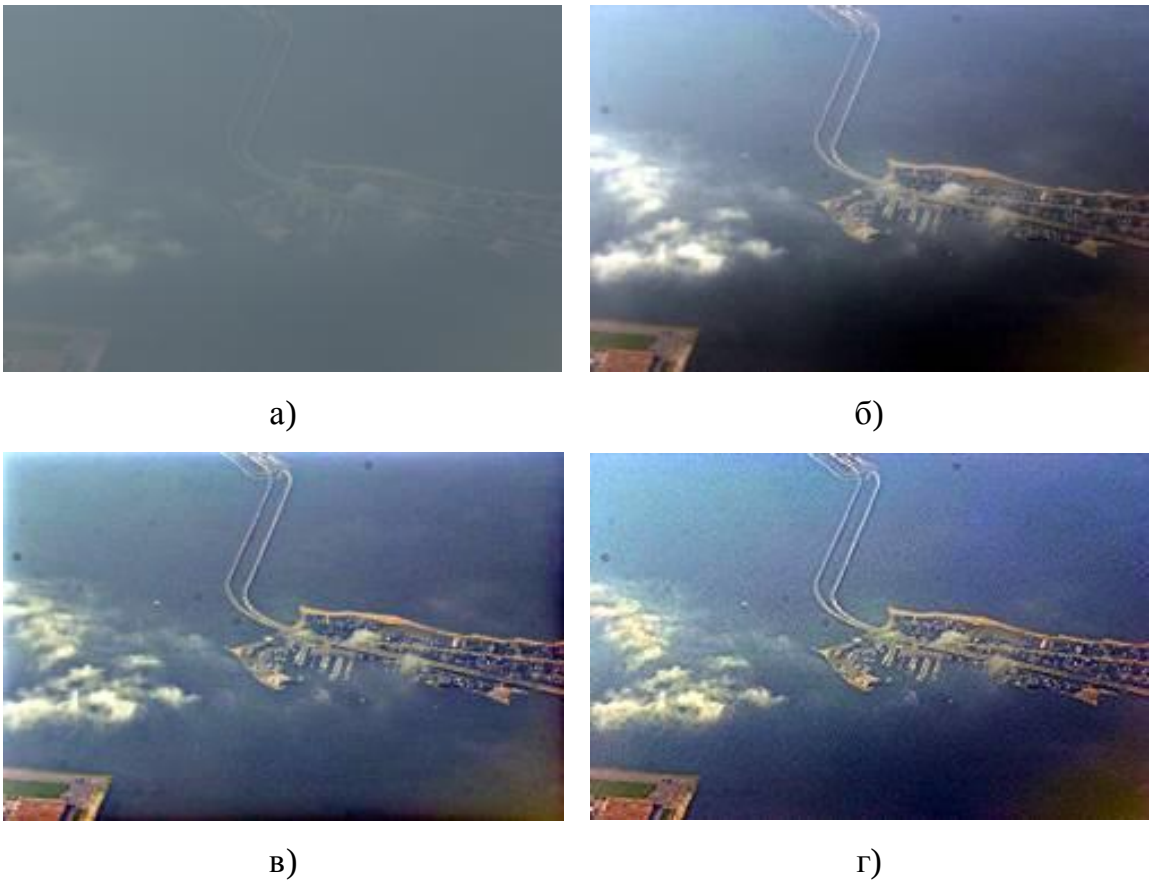


Рис. 4.9. Структура ИНС, полученной в результате эволюционного обучения с использованием изображения, представленного на рис. 4.1.

Скорость обработки изображений с использованием предлагаемого трехэтапного способа для окрестности 11x11 пикселей составляет около 0,9 мегапикселей/сек. при вычислениях на процессоре AMD Sempron 2500, работающего на частоте 1750 МГц. Примеры изображений, полученных с использованием трехэтапной обработки, и сравнение с алгоритмом автонастройки уровней яркости и Multi-Scale Retinex (MSR) представлены на рис. 4.10 и 4.11. Время обработки тестовых изображений составило около одной секунды. Заметим, что использование ИНС, отличной от показанной на рис. 4.9, скорее всего, даст другие результаты обработки ввиду различий реализуемого отображения «вход → выход».



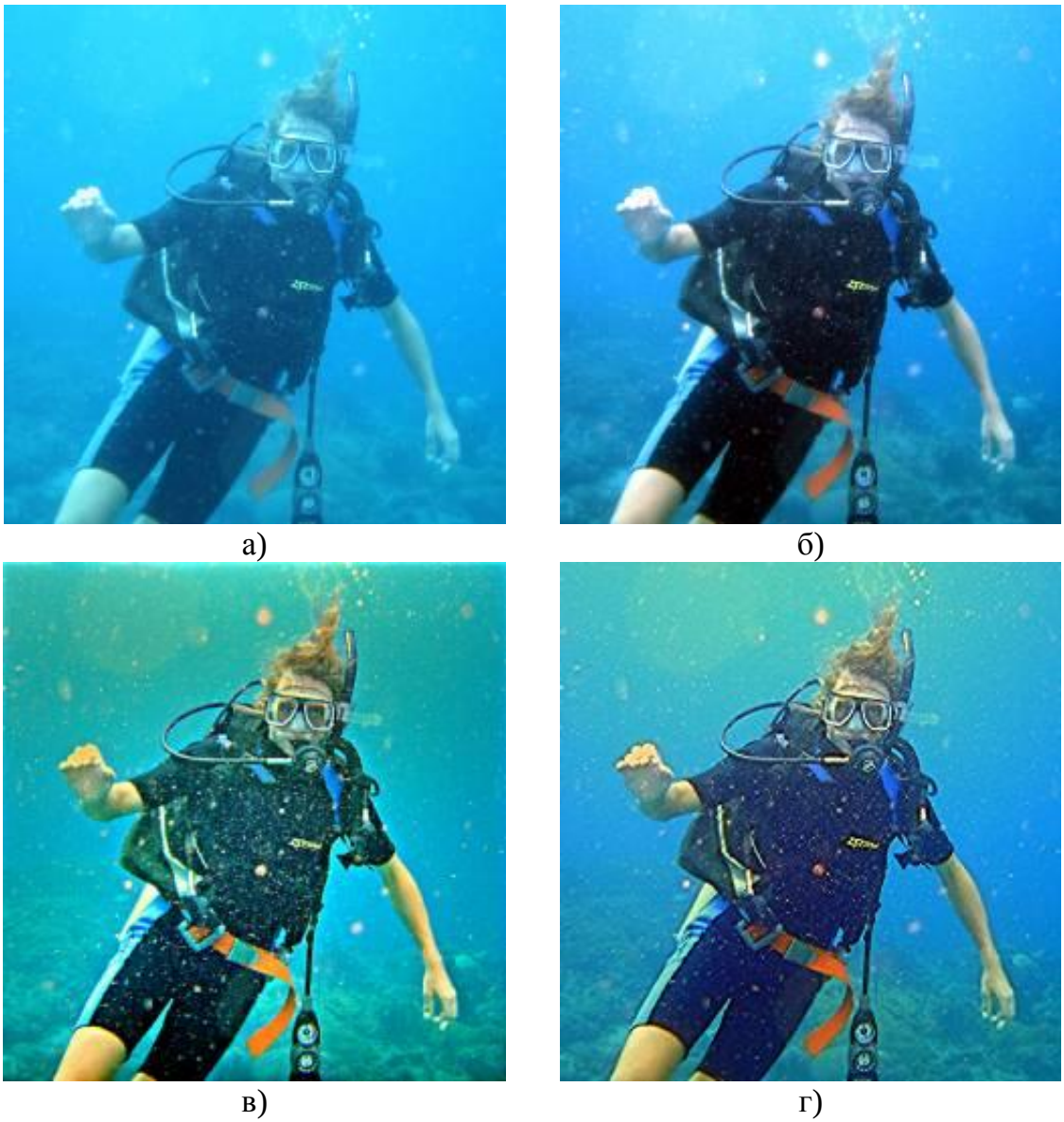
а) б)

в) г)

Рис. 4.10. Пример результата обработки изображения (а) [224] с использованием предлагаемого трехэтапного способа обработки (г) и сравнение с использованием только алгоритма автонастройки уровней яркости (б) и обработкой алгоритмом MSR (в).

Результаты трехэтапной обработки сравнивались с результатами работы алгоритмов Multi-Scale Retinex (MSR) [184] и известным алгоритмом автонастройки уровней [115]. Будем производить сравнение по следующим показателям:

1. Качество обработки.
2. Сохранение информации исходного изображения.



а) б)

в) г)

Рис. 4.11. Пример результата обработки изображения (а) [224] с использованием предлагаемого трехэтапного способа обработки (г) и сравнение с использованием только алгоритма автонастройки уровней яркости (б) и обработкой алгоритмом MSR (в).

Сравнение качества обработки осуществляется следующим образом. Пусть A – исходное изображение и B_{MSR}, B_{auto} и B_{ANN} – обработанные изображения, полученные с использованием соответственно MSR, алгоритма автонастройки уровней и предлагаемой трехэтапной обработки. Для каждого из изображений B_{MSR}, B_{auto} и B_{ANN} вычисляются значения визуальных оценок качества f_{rVIF} и коэффициентов f_{MR}^{*} и f_{mod}^{*}. После этого в соответствии со значениями каждой из оценок определяются ранги изображений, причем ранг 1 присваивается изображению с наилучшим качеством, а ранг 3 – изображению с наихудшим качеством. Таким образом, для обработанного изображения определяется набор из 3 рангов, по одному для каждой оценки. В результате обработки различных исходных изображений в соответствии со значениями используемых оценок качества подсчитывается сумма рангов для сравниваемых алгоритмов обработки изображений, и чем меньше сумма рангов алгоритма, тем лучше этот алгоритм обрабатывает изображения. Значения сумм рангов сравниваемых алгоритмов для набора из 53 исходных изображений приведены в табл. 4.2.

Табл. 4.2. Значения суммы рангов для алгоритмов обработки изображений по используемым оценкам визуального качества

Алгоритм	Сумма рангов по оценке f_{rVIF}	Сумма рангов по коэф. f_{MR}^{*}	Сумма рангов по коэф. f_{mod}^{*}	Сумма всех рангов
MSR	59	75	100	234
Автонастройка уровней	157	149	116	422
Трехэтапная обработка	102	94	102	298

По приведенным в табл. 4.2 данным видно, что по всем используемым оценкам визуального качества изображений алгоритм MSR обладает наи-

лучшими показателями, а алгоритм автонастройки уровней является наихудшим среди рассматриваемых. Предлагаемый трехэтапный способ обработки уступает алгоритму MSR, но превосходит алгоритм автонастройки уровней.

Анализ сохранения информации исходного изображения рассматриваемыми алгоритмами необходим для того, чтобы оценить насколько сравниваемые алгоритмы искажают в процессе обработки исходное изображение. Для этого будем использовать меру SSIM – Structure SIMilarity index [220], в которой сравнение двух изображений, исходного и измененного, производится на основе сравнения следующих компонент:

1. Распределение яркости.

2. Изменение контрастов.

3. Структурная информация.

Первые две компоненты характеризуют особенности человеческого восприятия изображения, а третья компонента необходима для учета пространственных зависимостей между соседними пикселями. Отмечено, что мера SSIM хорошо коррелирует с субъективными оценками и превосходит многие существующие подходы к оценке различия двух изображений [220].

Для оценки рассматриваемых алгоритмов MSR, автонастройки уровней и трехэтапной обработки с точки зрения сохранения информации исходного изображения также была проведена серия экспериментов и вычислены ранги для использованного ранее набора из 53 исходных изображений. Значения сумм рангов приведены в табл. 4.3. Видно, что, как и в случае оценки улучшения визуального качества изображений, алгоритм MSR показывает наилучший результат, алгоритм автонастройки уровней уступает всем алгоритмам, а способ трехэтапной обработки занимает «промежуточную» позицию.

Таким образом, по качеству обработки и по сохранению информации исходного изображения предлагаемый способ трехэтапной обработки превосходит алгоритм автонастройки уровней, но уступает алгоритму MSR. В то

же время отметим, что алгоритм MSR является вычислительно сложным [141], поскольку в нем используется преобразование Фурье и независимая обработка каждой цветовой плоскости при этом каждый пиксель обрабатывается с учетом окрестностей 3 радиусов, равных 15, 80 и 250 пикселей [184].

Табл. 4.3. Значения суммы рангов по мере SSIM

Алгоритм	Сумма рангов по мере SSIM
MSR	75
Автонастройка уровней	140
Трехэтапная обработка	104

4.7 Анализ работы ИНС

Рассмотрим более подробно преобразование яркости исходного изображения, осуществляемое обученной ИНС. Согласно предлагаемому способу обработки изображений (п. 4.1) новое значение яркости пикселя $L^*(x, y)$ определяется с использованием следующих трех параметров:

- яркость пикселя на исходном изображении $L(x,y)$;

- локальная дисперсия яркости $D_{(x,y)}$;

- локальная средняя яркость исходного изображения $m_{(x,y)}$.

Для анализа НС преобразования яркости рассмотрим три случая, соответствующие условно темной[1] ($m_{(x,y)} = 64$), условно яркой ($m_{(x,y)} = 192$) и условно нормальной по яркости ($m_{(x,y)} = 128$) окрестности пикселя. Будем анализировать зависимость величины выходного сигнала ИНС от различных значений $L(x, y)$ и $D_{(x,y)}$.

Типичные диаграммы, соответствующие результатам для различных

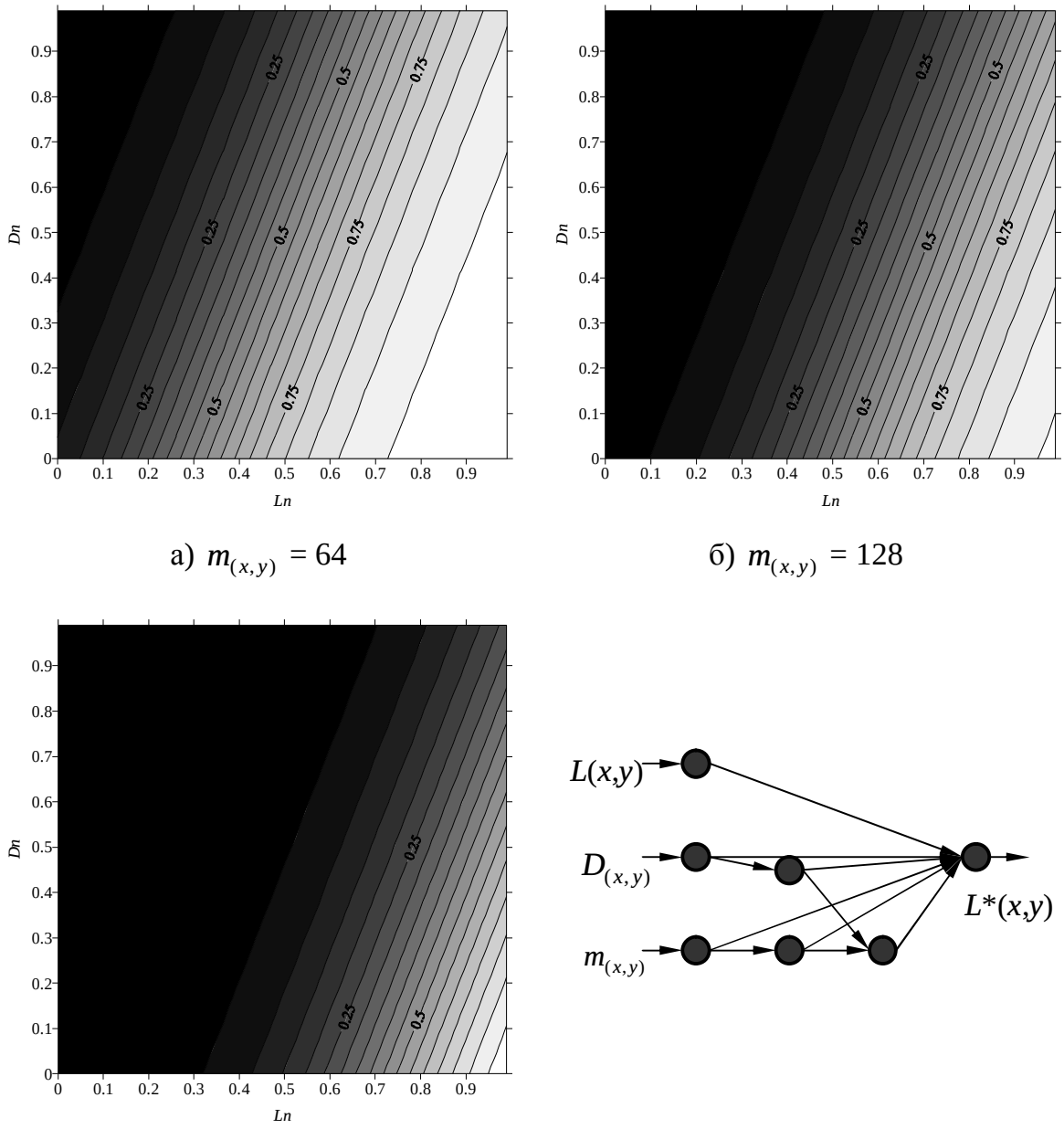
[1] Полагаем, что минимальная яркость пикселя равна 0, а максимальная яркость пикселя равна 255.

значений средней яркости исходных изображений приведены на рис. 4.12 и 4.13, где по оси абсцисс отложены нормированные значения исходной яркости L_n (0 – минимальная яркость, 1 – максимальная), а по оси ординат – локальная дисперсия яркости D_n (0 – минимальная дисперсия, 1 – максимальная). Новое нормированное значение яркости показано соответствующим цветом. Каждая изолиния соответствует значению нормированной яркости кратному 0,05.

По характеру изолиний на рис. 4.12 и 4.13 видно, что поверхность, описывающая изменение выходного сигнала ИНС, в целом имеет форму «сглаженной ступени». Сечение поверхности имеет форму графика логистической функции. Угол между проекциями изолиний на плоскость (L_n, D_n) и положительным направлением оси L_n определяет, будет ли пиксель затемняться или, наоборот, осветляться, а также зависимость величины изменения яркости от D_n.

Очевидно, что если проекции изолиний будут перпендикулярны оси абсцисс, то преобразование яркости пикселей не будет зависеть от дисперсии распределения яркости в окрестности этих пикселей. Кроме этого, средняя яркость всего изображения также задает характер изменения яркости исходного изображения, «сдвигая» поверхность вдоль оси L_n. Направление сдвига определяется величиной средней яркости исходного изображения: темному исходному изображению соответствует сдвиг влево (повышение яркости обработанного изображения), светлому – сдвиг вправо (понижение яркости обработанного изображения).

В случае фиксированного значения $m_{(x,y)}$ некоторые уровни яркости L_n* становятся либо практически недоступны в силу ограниченности величины D_n (рис. 4.12а, 4.13в), либо начинают доминировать в ярких (рис. 4.12а, 4.13а) или темных (рис. 4.12в, 4.13в) областях исходного изображения. Данные факторы приводят к потере деталей на изображении и общему ухудшению визуального качества обработанного изображения.



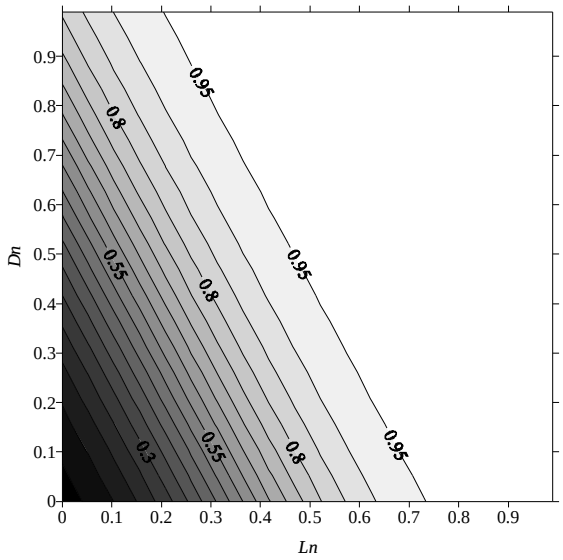
а) $m_{(x,y)}$ = 64

б) $m_{(x,y)}$ = 128

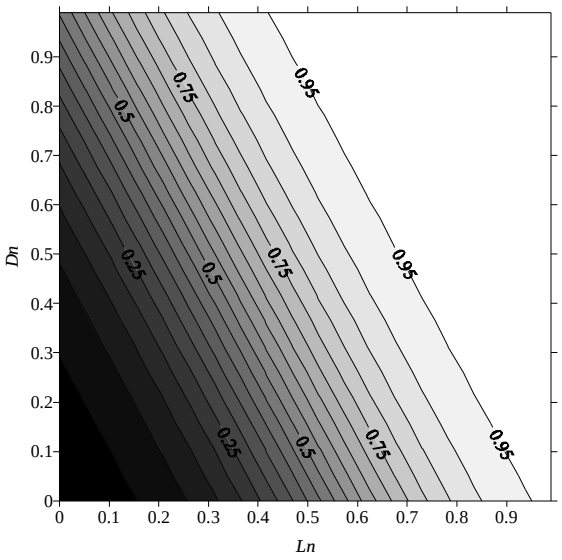
в) $m_{(x,y)}$ = 192

г) структура нейронной сети

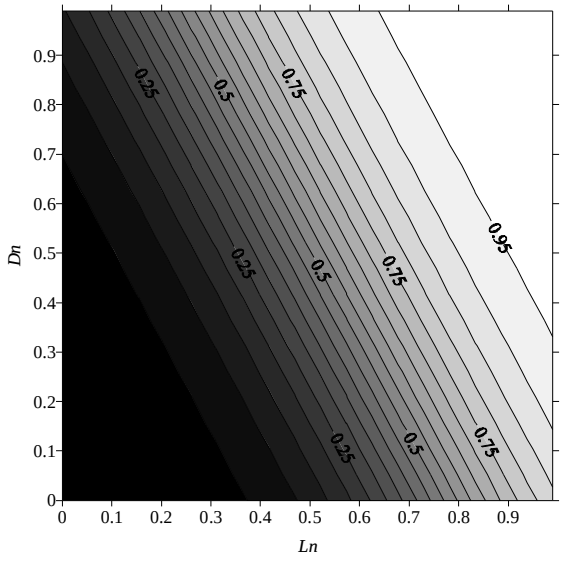
Рис. 4.12. Пример преобразования исходной яркости пикселя в зависимости от локальных средней L_n и дисперсии D_n яркости в окрестности пикселя. Новое значение яркости L^* показано соответствующим цветом. Каждая изолиния соответствует значению нормированной яркости кратному 0,05.



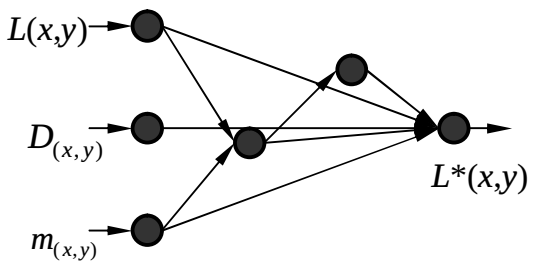
а) $m_{(x,y)}$ = 64



б) $m_{(x,y)}$ = 128



в) $m_{(x,y)}$ = 192



г) структура нейронной сети

Рис. 4.13. Пример преобразования исходной яркости пикселя в зависимости от локальных средней L_n и дисперсии D_n яркости в окрестности пикселя. Новое значение яркости Ln^* показано соответствующим цветом. Каждая изолиния соответствует значению нормированной яркости кратному 0,05.

Величина угла поворота проекций изолиний на плоскость (L_n, D_n) относительно оси L_n ограничивает изменение динамического диапазона яркостей пикселей исходного изображения. Таким образом, при обработке изображений со слабым контрастом динамический диапазон яркости обработанного

изображения будет, скорее всего, расширен, но для того, чтобы занять весь возможный спектр значений яркостей пикселей, необходима дополнительная обработка полученного изображения. В рамках предлагаемого трехэтапного способа обработки изображений, в качестве алгоритма постобработки, используется известный алгоритм автоматической настройки уровня яркости.

Отметим, что результаты анализа нейросетевого преобразования локальных характеристик для локально-адаптивной обработки изображений, позволяют предложить новый способ улучшения качества изображений. Основными особенностями этого способа являются следующие:

1. Преобразование яркости пикселя исходного изображения в зависимости от значений его локальных характеристик определяется формой и параметрами поверхности, характеризующей это преобразование (по аналогии с рис. 4.12 и 4.13).

2. Учет локальных характеристик, необходимый при использовании локально-адаптивного подхода, осуществляется за счет смещения (вертикального и горизонтального) поверхности преобразования.

4.8. Описание программ для обработки изображений

Для реализации трехэтапного способа обработки изображений разработано соответствующее программное обеспечение. Для разработки использован язык программирования C++ и среда программирования Microsoft Visual Studio .NET 2003. При реализации НЭ алгоритма NEvA использована разработанная инструментальная библиотека ECWorkshop, описанная в п. 3.7, и классы, реализующие возможность создания ИНС с произвольной топологией, представленные в п. 3.8.

4.8.1. Описание программы QImager

Программа QImager предназначена для обучения и тестирования ИНС для обработки изображений. Основное окно программы изображено на рис. 4.14. Описание элементов интерфейса приведено в табл. 4.4.

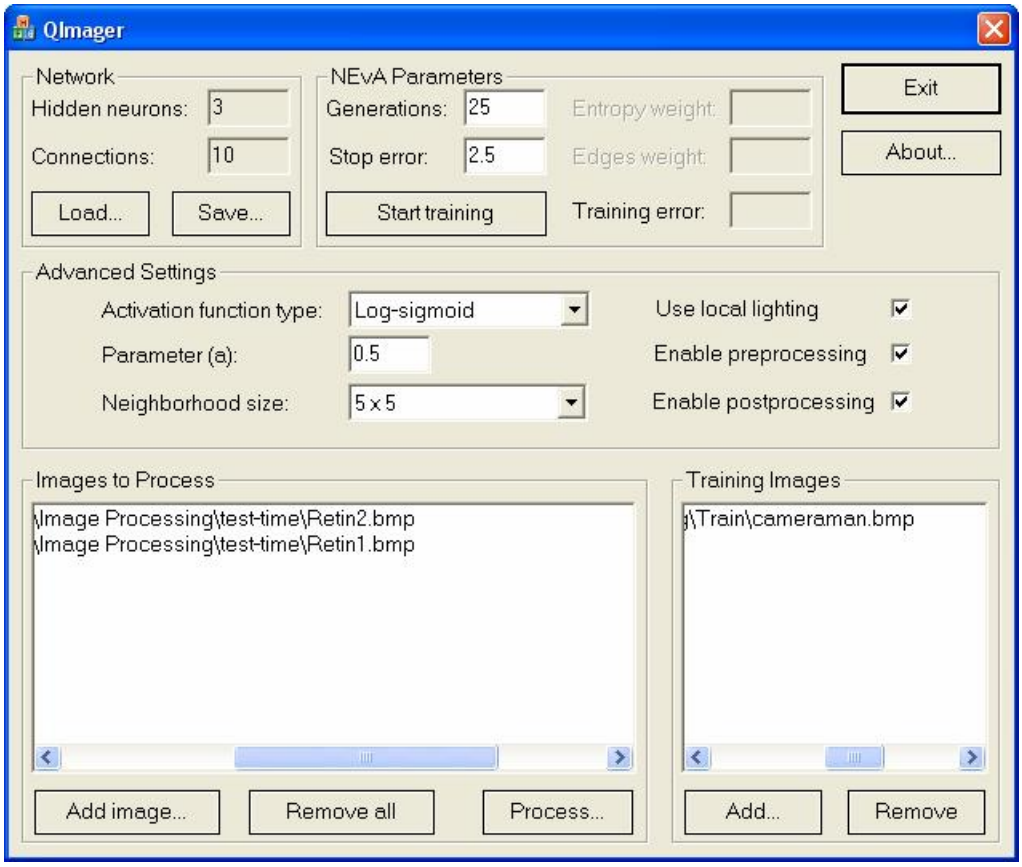


Рис. 4.14. Основное окно программы QImager.

Табл. 4.4. Описание функций интерфейса основного окна программы QImager

Название элемента интерфейса / группы элементов	**Назначение**
Network	
Hidden neurons	Количество скрытых нейронов в текущей (обученной / загруженной) ИНС
Connections	Количество межнейронных связей в текущей (обученной / загруженной) ИНС
Load…	Загрузка сохраненной ИНС из файла
Save…	Сохранение текущей ИНС
NEvA Parameters	

Название элемента интерфейса / группы элементов	**Назначение**
Generations	Количество поколений для обучения ИНС
Stop error	Целевое значение функции приспособленности
Start training…	Кнопка запуска обучения ИНС
Training error	Достигнутое значение функции приспособленности
Advanced Settings	
Activation function type	Тип функции активации нейронов
Parameter (a)	Значение параметра крутизны (наклона) функции активации
Neighborhood size	Размер локальной окрестности для локально-адаптивной обработки изображений
Use local lighting	Использование локально-адаптивной обработки
Enable preprocessing	Включение предобработки (предобработка яркости)
Enable postprocessing	Включение постобработки (автонастройка уровней)
Images to Process	
Add image…	Добавление изображения к списку изображений для обработки
Remove All	Очистка списка обрабатываемых изображений
Process…	Запуск обработки с использованием текущей (обученной / загруженной) ИНС
Training Images	
Add…	Добавление изображения к списку изображений для обучения
Remove	Удаление выбранного изображения из списка изображений для обучения

Для запуска обучения ИНС необходимо выбрать значения параметров обучения (количество поколений и целевое значение функции приспособленности), а также изображения, используемые для обучения и нажать кнопку «Start training». После этого появится окно обучения ИНС (рис. 4.15), в ко-

тором отображаются результаты обучения. Описание элементов интерфейса окна обучения приведено в табл. 4.5.

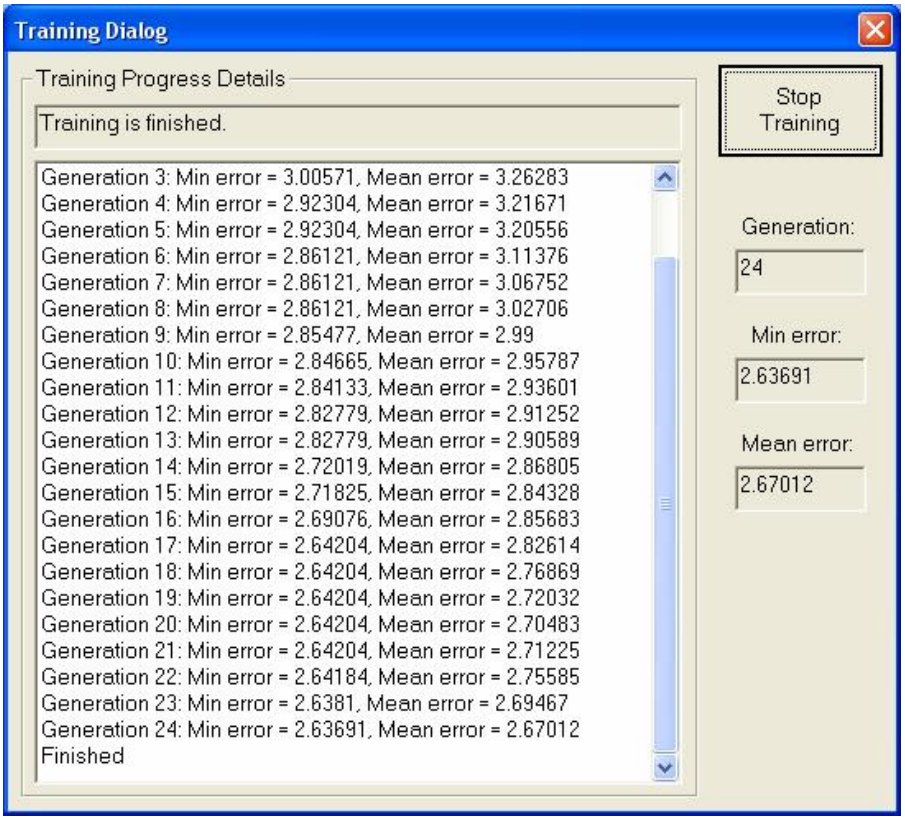


Рис. 4.15. Окно обучения ИНС

Табл. 4.5. Описание функций интерфейса окна обучения ИНС

Название элемента интерфейса / группы элементов	**Назначение**
Training Process Details	Группа элементов интерфейса (строка статуса и окно вывода) для отображения подробностей процесса обучения по поколениям.
Generation	Номер текущего поколения
Min error	Минимальное значение функции приспособленности в популяции текущего поколения
Mean error	Среднее значение функции приспособленности в популяции текущего поколения

Полученная в результате обучения ИНС автоматически сохраняется в файле bestnet.qnn, а также передается в основное окно программы и может быть сохранена в другой файл и/или использована для обработки произвольных изображений.

4.8.2. Описание программы QImagerLite

Возможность загрузки в программе QImager произвольной ИНС предъявляет требования к унификации алгоритма вычисления выходного сигнала ИНС, что отрицательно сказывается на быстродействии трехэтапного способа обработки. Отметим, что создание отдельного приложения, использующего конкретную (не произвольную) ИНС, позволяет значительно (до 3-4 раз) увеличить скорость обработки. Примером такого приложения является разработанная программа QImagerLite, основное окно которой изображено на рис. 4.16.

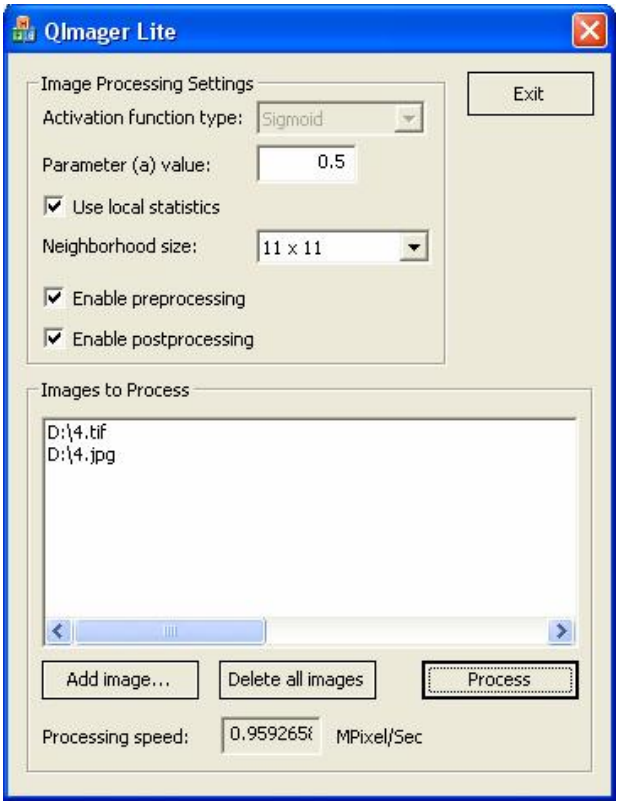


Рис. 4.16. Основное окно программы QImagerLite

Программой QImagerLite поддерживаются следующие графические форматы: BMP, TIFF, JPEG, GIF, PNG. Описание элементов интерфейса приведено в табл. 4.6.

Для запуска обработки изображений необходимо установить параметры обработки и выбрать изображения для обработки, отображающиеся в соответствующем списке. После этого нажатие на кнопку «Process» запускает обработку изображений и появляется окно статуса обработки с индикатором, который заполняется по мере обработки изображений из списка.

Обработанные изображения сохраняются в каталоге, в котором находится программа QImagerLite, в том же формате, в котором они поступили на обработку.

Табл. 4.6. Описание функций интерфейса программы QImagerLite

Название элемента интерфейса / группы элементов	**Назначение**
Image Processing Settings	
Parameter (a) value	Значение параметра крутизны (наклона) функции активации нейронов
Use local statistics	Использование локально-адаптивной обработки
Neighborhood size	Размер локальной окрестности для локально-адаптивной обработки изображений
Enable preprocessing	Включение предобработки (предобработка яркости)
Enable postprocessing	Включение постобработки (автонастройка уровней)
Images to Process	
Add image…	Добавление изображения к списку изображений для обработки
Delete all images	Очистка списка обрабатываемых изображений
Process…	Запуск обработки с использованием текущей (обученной / загруженной) ИНС

Название элемента интерфейса / группы элементов	Назначение
Processing speed	Отображение скорости обработки в 10^6 пикселях в секунду. Обновляется после завершения обработки изображений

4.9 Основные результаты и выводы по главе 4

1. Показано, что использование эволюционирующих нейронных сетей с достаточно грубой оценкой качества их функционирования представляет эффективный способ получения нейросетевого решения для быстрой и эффективной попиксельной обработки изображений.

2. Получены формулы для приближенного вычисления локальных характеристик изображений, что позволяет ускорить вычисления, необходимые для осуществления обработки изображений.

3. Исследование полученных формул для приближенного вычисления локальных характеристик показало существенное увеличение скорости вычислений по сравнению с точными формулами при сохранении приемлемой точности результатов Получены формулы для приближенного вычисления локальных характеристик (среднее и дисперсия) изображения, позволяющие быстро вычислять локальные среднее и дисперсию в прямоугольной окрестности с приемлемой точностью (более 20 дБ для среднего и более 30 дБ для дисперсии). Использование полученных формул позволяет существенно повысить скорость нейросетевой обработки изображений по локально-адаптивному методу по сравнению с использованием точных формул (свыше 30 раз для окрестности 65x65 пикселей).

4. Предложен трехэтапный способ улучшения качества изображений включающий в себя следующие этапы: (1) предобработка яркости исходного изображения; (2) обработка на локальном уровне с использованием ИНС; (3)

обработка на глобальном уровне с применением известного алгоритма автонастройки уровней яркости.

5. Результаты экспериментов показали высокую эффективность и качество предложенного трехэтапного способа обработки изображений по сравнению с рядом известных методов. Сравнение результатов разработанного способа трехэтапной нейросетевой обработки изображений с используемой в NASA технологией Multi-Scale Retinex (MSR) показало сопоставимость результатов при небольшом превосходстве MSR, однако скорость обработки с использованием предлагаемого способа выше за счет использования приближенных формул для вычисления локальных характеристик (среднее и дисперсия) с использованием окрестности меньшего размера и отказа от использования преобразования Фурье.

6. Разработано программное обеспечение для обучения и тестирования ИНС для обработки изображений.

ЗАКЛЮЧЕНИЕ

Диссертационная работа посвящена применению методов эволюционных вычислений для настройки и обучения искусственной нейронной сети и разработке программных средств для нейросетевой обработки изображений.

В результате выполнения диссертационной работы получены следующие основные научные и практические результаты и сделаны следующие выводы.

1. Разработан новый способ вычисления времени смешивания для генетических операторов кроссинговера для целочисленного кодирования. Особенностью предлагаемого способа является исключение из анализа динамики популяции, что значительно упрощает процесс вывода искомых оценок, которые согласуются как с уже имеющимися аналитическими оценками, так и с результатами экспериментов.

2. Предложена стратегия изменения размера популяции с использованием последовательности Фибоначчи, позволяющая адаптироваться к характеристикам эволюционного поиска. Экспериментально показано, что ее применение позволяет в большинстве случаев получить результаты, которые сопоставимы или превосходят результаты ГА с постоянным размером популяции.

3. Разработан НЭ алгоритм NEvA для одновременной эволюционной настройки структуры и весов связей ИНС. Экспериментальная проверка разработанного алгоритма на задачах классификации и адаптивного нейроуправления и сравнение с рядом известных алгоритмов и методов показали его высокое быстродействие и эффективность как с точки зрения выбранного критерия оценки (количество вычислений целевой функции), так и с точки зрения структуры получаемых ИНС.

4. Показана эффективность реализованных в алгоритме NEvA механизмов адаптации на примере тестовых задач, а также задачи улучшения визуального качества цифровых изображений за счет использования фиксирован-

ного набора начальных значений параметров алгоритма для всех этих задач.

5. Получены формулы для приближенного вычисления локальных характеристик изображений, что позволяет ускорить вычисления, необходимые для осуществления обработки изображений. Исследование полученных формул показало существенное увеличение скорости вычислений по сравнению с точными формулами при сохранении приемлемой точности результатов (более 20 дБ для среднего и более 30 дБ для дисперсии). Использование полученных формул позволяет существенно повысить скорость нейросетевой обработки изображений по локально-адаптивному методу по сравнению с использованием точных формул (свыше 30 раз для окрестности 65x65 пикселей).

6. Предложен трехэтапный способ улучшения качества цифровых изображений включающий в себя следующие этапы: (1) предобработка яркости исходного изображения; (2) обработка на локальном уровне с использованием ИНС; (3) обработка на глобальном уровне с применением известного алгоритма автонастройки уровней яркости. Сравнение результатов разработанного способа трехэтапной нейросетевой обработки изображений с используемой в NASA технологией Multi-Scale Retinex (MSR) показало сопоставимость результатов при небольшом превосходстве MSR, однако вычислительная сложность предлагаемого способа значительно меньше за счет использования приближенных формул для вычисления локальных характеристик (среднее и дисперсия) и отказа от использования преобразования Фурье.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **Арнольд, В.И.** О функциях трех переменных [Текст] / В.И. Арнольд // Докл. АН СССР. – 1957. – Т. 114, № 4. – С. 679-681.
2. **Бурцев, М.С.** Эволюция кооперации в многоагентной системе [Текст] / М.С. Бурцев // Научная сессия МИФИ-2005. VII Всероссийская научно-практическая конференция «Нейроинформатика-2005»: Сборник научных трудов. В 2-х частях. Ч.1. – М.: МИФИ, 2005. – С.217-224
3. **Вороновский, Г.К.** Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности [Текст] / Г.К. Вороновский, К.В. Махотило, С.Н. Петрашев, С.А. Сергеев. – Харьков: Основа, 1997 // http://neuroschool.narod.ru/books/gannvirt.html
4. **Выгодский, М.Я.** Справочник по высшей математике [Текст] / М.Я. Выгодский. – М: Джангар, 2001.
5. **Гамма, Э.** Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб: Питер, 2003.
6. **Глаз, А.Б.** Оценка вероятности образования оптимальной структуры перцептрона при ее оптимизации методами случайного поиска [Текст] / А.Б. Глаз, Л.А. Растригин // Задачи статистической оптимизации. – Рига: Зинатне, 1971. – С. 131-142.
7. **Глаз, А.Б.** Трехрядный статистический перцептрон со специальным устройством статистической адаптации структуры [Текст] / А.Б. Глаз, Л.А. Растригин // Перцептрон – система распознавания образов. – Киев: Наукова думка, 1975. – С. 334-385.
8. **Горбань, А.Н.** Нейронные сети на персональном компьютере [Текст] / А.Н. Горбань, Д.А. Россиев. – Новосибирск: Наука, Сиб. отделение, 1996.
9. **Горбань, А.Н.** Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей [Текст] / А.Н. Горбань // Сибирский

журнал вычислительной математики.– 1998. – Т. 1, № 1. – С. 12-24.

10. **Дунин-Барковский, В.Л.** Нейроинформатика [Текст] / А.Н. Горбань, В.Л. Дунин-Барковский, А.Н. Кирдин [и др.] – Новосибирск: Наука, Сиб. отделение, 1998.
11. **Журавель, И.М.** Краткий курс теории обработки изображений [Электронный ресурс] / И.М. Журавель // http://matlab.exponenta.ru/imageprocess/book2/index.php
12. **Карманов, В.Г.** Математическое программирование [Текст]: Учеб. пособие. – 5-е изд., стереотип. / В.Г. Карманов. – М.: Физматлит, 2001.
13. **Колмогоров, А.Н.** О представлении непрерывных функций нескольких переменных суперпозициями непрерывных функций меньшего числа переменных [Текст] / А.Н. Колмогоров // Докл. АН СССР. – 1956. – Т. 108, № 2. – С.179-182.
14. **Колмогоров, А.Н.** О представлении непрерывных функций нескольких переменных в виде суперпозиции непрерывных функций одного переменного [Текст] / А.Н. Колмогоров // Докл. АН СССР. – 1957. – Т. 114. № 5. –С. 953-956.
15. **Комарцова, Л.Г.** Исследование алгоритмов обучения многослойного персептрона [Текст] / Л.Г. Комарцова // Нейрокомпьютеры: Разработка и применение. – 2002. – № 12.
16. **Комарцова, Л.Г.** Нейрокомпьютеры [Текст] / Л.Г. Комарцова, А.В. Максимов. – М.: МГТУ им. Баумана, 2002.
17. **Курейчик, В.М.** Генетические алгоритмы [Текст] / Л.А. Гладков, В.М. Курейчик, В.В. Курейчик. – М.: Физматлит, 2006.
18. **Курейчик, В.М.** Теория и практика эволюционного моделирования [Текст] / В.В. Емельянов, В.М. Курейчик, В.В. Курейчик. – М.: ФИЗМАТЛИТ, 2003.
19. **Линдли, К.** Практическая обработка изображений на языке Си [Текст] / К. Линдли ; [пер. с англ. А.А. Брюзгина] – М.: Мир, 1996.

20. **Майерс, С.** Эффективное использование C++. 35 новых рекомендаций по улучшению ваших программ и проектов: Пер. с англ. [Текст] / С. Майерс. – М.: ДМК Пресс; Спб: Питер, 2006.

21. **Мосалов, О.П.** Модель эволюционной ассимиляции приобретенных навыков в нейросетевых системах управления адаптивных агентов [Текст] / О.П. Мосалов, В.Г. Редько // Научная сессия МИФИ-2005. VII Всероссийская научно-практическая конференция «Нейроинформатика-2005»: Сборник научных трудов. В 2-х частях. Часть 1. – М.: МИФИ, 2005. – С.210-217. // http://www.iont.ru/projects/rfbr/90197/

22. Нейрокомпьютеры в задачах обработки изображений [Текст] / Под ред. Ю.В. Гуляева, А.И. Галушкина. – М.: Радиотехника, 2003.

23. Нейроматематика [Текст]. Серия «Нейрокомпьютеры и их применение», книга 6 / под ред. А.И. Галушкина. – М.: ИПРЖР, 2002.

24. **Паклин, Н.Б.** Обучаем нейронную сеть генетическим алгоритмом [Текст] / Н.Б. Паклин. – 2003. // http://paklin.newmail.ru

25. **Редько, В.Г.** К теории эволюции. Модель возникновения "программ жизнедеятельности" [Текст] / Редько В.Г. // Журнал общей биологии. – 1991. – № 3. – С. 334-342.

26. **Редько, В.Г.** Эволюция, нейронные сети, интеллект: Модели и концепции эволюционной кибернетики [Текст] – 3-е изд. / В.Г. Редько. – М.: КомКнига, 2005.

27. **Редько, В.Г.** Оценка эффективности эволюционных алгоритмов [Текст] / В.Г. Редько, Ю.Р. Цой // Доклады АН. – 2005. – Т. 404, №3. – С. 312-315.

28. **Розенблат, Ф.** Принципы нейродинамики. Перцептроны и теория механизмов мозга [Текст] / Ф. Розенблат. – М.: Мир, 1965.

29. **Сергин, А.В.** Биологически правдоподобный нейросетевой детектор края [Текст] / А.В. Сергин // Научная сессия МИФИ-2005. VII Всероссийская

научно-практическая конференция «Нейроинформатика-2005»: Сборник научных трудов. В 2-х частях. Ч.1. – М.: МИФИ, 2005. – С. 249-256.

30. **Спицын, В.Г.** Эволюционирующие искусственные нейронные сети [Текст] / В.Г. Спицын, Ю.Р. Цой // Молодежь и современные информационные технологии. Сборник трудов IV Всероссийской конференции студентов, аспирантов и молодых ученых. – Томск: Изд-во ТПУ, 2006. – С. 411-413.
31. **Тарасов, В.Б.** От многоагентных систем к интеллектуальным организациям: философия, психология, информатика [Текст] / В.Б. Тарасов. – М.: Эдиториал УРСС, 2002.
32. **Хайкин, С.** Нейронные сети: полный курс [Текст] – 2-е издание / С. Хайкин ; под ред. Н.Н. Куссуль. – М.: Издательский дом «Вильямс», 2006.
33. **Хомич, А.В.** Анализ и оптимизация операций мутации и кроссовера в генетических алгоритмах [Текст] / А.В. Хомич, Л.А. Жуков // Материалы Докладов V Всероссийской конференции «Новые информационные технологии в исследовании сложных структур». – Томск, Вестник ТГУ. – 2004. – С. 111-114.
34. **Хомич, А.В.** Эволюционный метод оптимизации структуры нейронной сети с учителем [Текст] / А.В. Хомич, Л.А. Жуков // Научная сессия МИФИ-2005. VII Всероссийская научно-практическая конференция "Нейроинформатика-2005": Сборник научных трудов. В 2-х частях. Ч. 1. – М: МИФИ, 2005. – С. 11-18.
35. **Царегородцев, В.Г.** Редукция размеров нейросети не приводит к повышению обобщающих способностей [Текст] / В.Г. Царегородцев // Материалы XII Всероссийского семинара «Нейроинформатика и ее приложения». – Красноярск, 2004. –С. 163-165. // http://www.neuropro.ru/
36. **Цой, Ю.Р.** Генетический алгоритм настройки искусственной нейронной сети [Текст] / Ю.Р. Цой, В.Г. Спицын // Тезисы докладов конференции-

конкурса студентов, аспирантов и молодых ученых «Технологии Microsoft в информатике и программировании». – Новосибирск: НГУ, 2004. – С. 131-133.

37. **Цой, Ю.Р.** Использование генетического алгоритма для настройки весов и структуры искусственной нейронной сети [Текст] / Ю.Р. Цой, В.Г. Спицын // Молодежь и современные информационные технологии. Сборник трудов II Всероссийской научно-практической конференции студентов. – Томск: Изд-во ТПУ, 2004. – С. 221-223.
38. **Цой, Ю.Р.** Многоагентный нейроэволюционный подход к адаптивному управлению [Текст] / Ю.Р. Цой // Труды 10-й Юбилейной международной научно-практической конференции студентов и молодых ученых «Современные техника и технологии». – Томск: Изд-во ТПУ, 2004. – с. 219-220.
39. **Цой, Ю.Р.** К выбору размера популяции [Текст] / Ю.Р. Цой, В.Г. Спицын // Труды международных научно-технических конференций «Интеллектуальные системы (IEEE AIS'04)» и «Интеллектуальные САПР (CAD-2004)». Научное издание в 3-х томах. Т.1. – М.: Физматлит, 2004. – С. 90-96.
40. **Цой, Ю.Р.** Применение генетического алгоритма для решения задачи адаптивного нейроуправления [Текст] / Ю.Р. Цой, В.Г. Спицын // Труды VII Всероссийской научно-технической конференции «Нейроинформатика-2005». В 2-х частях. Ч.1. – М.: МИФИ, 2005. – С. 35-43.
41. **Цой, Ю.Р.** Адаптивный оператор мутации для нейроэволюционного алгоритма [Текст] / Ю.Р. Цой, В.Г. Спицын // XI Международная научно-практическая конференция студентов, аспирантов и молодых ученых «Современные техника и технологии». Труды в 2-х томах. Т.2. – Томск: Изд-во ТПУ, 2005. – С. 262-264.

42. **Цой, Ю.Р.** Нейроэволюционный подход [Текст] / Ю.Р. Цой, В.Г. Спицын // Нейрокомпьютеры: разработка и применение. – 2005. – №6. – С. 15-25.

43. **Цой, Ю.Р.** Исследование генетического алгоритма с динамически изменяемым размером популяции [Текст] / Ю.Р. Цой, В.Г. Спицын // Труды международных научно-технических конференций «Интеллектуальные системы (IEEE AIS'05)» и «Интеллектуальные САПР (CAD-2005)». – М.: Физматлит, 2005. – С. 241-246.

44. **Цой, Ю.Р.** Применение нейроэволюционного подхода для решения задач классификации и аппроксимации [Текст] / Ю.Р. Цой, В.Г. Спицын // Нейроинформатика и ее приложения: Материалы XIII Всероссийского семинара. – Красноярск: ИВМ СО РАН, 2005. – С. 123-124.

45. **Цой, Ю.Р.** Нейроэволюционное улучшение качества изображений [Текст] / Ю.Р. Цой, В.Г. Спицын, А.В. Чернявский // Научная сессия МИФИ - 2006. VIII Всероссийская научно-техническая конференция "Нейроинформатика-2006": Сборник трудов. В 3-х частях. Ч.1. М.: МИФИ, 2006. – С. 181-189.

46. **Цой, Ю.Р.** Настройка клеточных автоматов с помощью искусственных нейронных сетей [Текст] / Ю.Р. Цой // Научная сессия МИФИ - 2006. VIII Всероссийская научно-техническая конференция "Нейроинформатика-2006": Сборник трудов. В 3-х частях. Ч.3. М.: МИФИ, 2006. – С. 49-55.

47. **Цой, Ю.Р.** Эволюционный подход к настройке и обучению искусственных нейронных сетей [Электронный ресурс] / Ю.Р. Цой, В.Г. Спицын // Электронный рецензируемый журнал «Нейроинформатика». – 2006. – Т. 1, № 1. – С. 34-61 // http://ni.iont.ru/Journal/N1

48. **Цой, Ю.Р.** Способ улучшения качества монохромных и цветных изображений, основанный на применении эволюционирующей нейронной сети

[Текст] / Ю.Р. Цой, В.Г. Спицын, А.В. Чернявский // Информационные технологии. – 2006. – № 7. – С. 27-33.

49. **Цой, Ю.Р.** О математических моделях эволюционных алгоритмов [Текст] / Ю.Р. Цой // Перспективные информационные технологии и системы. – 2006. – № 2 (26). – С. 42-47. http://pitis.tsure.ru/
50. **Цой, Ю.Р.** Трехэтапная обработка цифровых изображений с использованием эволюционирующих искусственных нейронных сетей [Текст] / Ю.Р. Цой, В.Г. Спицын // Всероссийская научная конференция по нечетким системам и мягким вычислениям НСМВ-2006 (20-22 сентября 2006г., Тверь): Труды конференции. – М.: Физматлит, 2006. – С. 231-244.
51. **Цой, Ю.Р.** Один способ вычисления времени смешивания для генетических операторов скрещивания [Текст] / Ю.Р. Цой // Десятая национальная конференция по искусственному интеллекту с международным участием КИИ-2006 (25-31 сентября 2006 г., Обнинск): Труды конференции. В 3-х т. Т.3. – М.: Физматлит, 2006. С. 1047-1054.
52. **Цой, Ю.Р.** К применению нейронных сетей для аппроксимации таблицы правил клеточного автомата [Текст] / Ю.Р. Цой // Нейроинформатика и ее приложения: Материалы XIV Всероссийского семинара. – Красноярск: ИВМ СО РАН, 2006. – С. 129-130.
53. **Цой, Ю.Р.** Введение в нейроэволюционный подход: основные концепции и приложения [Текст] / Ю.Р. Цой // Научная сессия МИФИ - 2007. IX Всероссийская научно-техническая конференция "Нейроинформатика-2007": Лекции по нейроинформатике. Часть 2. –М.:МИФИ, 2007. – С. 43-76.
54. **Шукович, Г.** Применение генетических алгоритмов и систем генерирующих графов для создания модулярных нейросетей [Текст] / Г. Шукович // Программирование. – 2002. – № 1. – С. 13-20.

55. **Angeline, P.J.** An evolutionary algorithm that constructs recurrent neural networks [Текст] / P.J. Angeline, G.M. Saunders, J.B. Pollack // IEEE Transactions on Neural Networks. – 1993. – No. 5. – P. 54-65.

56. **Arabas, J.** GAVAPS—a genetic algorithm with varying population size [Текст] / J. Arabas, Z. Michalewicz, J. Mulawka // Proceedings of the First IEEE International Conference on Evolutionary Computation. – New York: IEEE Press, 1994. – P. 73-78.

57. **Arnold, D.V.** Performance analysis of evolution strategies with multi-recombination in high-dimensional R^N-search spaces disturbed by noise [Текст] : Technical report no. CI 94/00 / D.V. Arnold, H.-G. Beyer. – University of Dortmund, Germany, 2000 // http://sfbci.cs.uni-dortmund.de

58. **Baeck, T.** Self-adaptation in genetic algorithms [Текст] / T. Baeck // Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life / eds. F.J. Varela, P. Bourgine. – Cambridge, MA: MIT Press, 1992. – P.263-271.

59. **Baeck, T.** Self-adaptation [Текст] / T. Baeck // Handboook of Evolutionary Computation. – Bristol: Institute of Physics Publishing; New York: Oxford University Press, 1997. – C7.1:1-15.

60. **Baeck, T.** An overview of parameter control methods by self-adaptation in evolutionary algorithms [Текст] / T. Baeck // Fund. Inform. – 1998. – Vol. 35, no. 1-4. – P. 51-66.

61. **Baeck, T.** An empirical study on GAs "without parameters" [Текст] / T. Baeck, A.E. Eiben, N.A.L. van der Vaart // Proceedings of the 6th Conference on Parallel Problem Solving from Nature. LNCS no.1917. – Berlin: Springer, 2000. – P. 315-324.

62. **Baeck, T.** Evolutionary computation [Текст] / T. Baeck, D. Fogel, Z. Michalewicz. – Berlin Heidelberg: Springer-Verlag, 2000.

63. **Baker, J.E.** Adaptive selection methods for genetic algorithms [Текст] / J.E. Baker // Proceedings of the International Conference on Genetic Algorithms and Their Applications. – 1985. – P.101-111.

64. **Banzhaf, W.** Interactive evolution [Текст] / W. Banzhaf // Evolutionary Computation I: Basic Algorithms and Operators / eds. T. Baeck [et al.]. – Bristol and Philadelphia: IOP Publishing, 2000. – P. 228-234.

65. **Barnett, L.** Evolutionary search on fitness landscapes with neutral networks [Текст] : Unpublished PhD thesis / L. Barnett. – The University of Sussex, England, 2003. // http://www.cogs.susx.ac.uk/users/lionelb/

66. **Barron, A.R.** Universal approximation bounds for superposition of a sigmoidal function [Текст] / A.R. Barron // IEEE Transactions on Information Theory. – 1993. – Vol. 93. – P. 930-945.

67. **Belew, R.K.** Evolving networks: Using the genetic algorithm with connectionist learning [Текст] / R.K. Belew, J. McInerney, N.N. Schraudolph // Artificial Life II. – 1992. – P. 511-547.

68. **Beyer, H.-G.** An alternative explanation for the manner in which genetic algorithms operate [Текст] / H.-G. Beyer // BioSystems. – 1997. – No. 41. – P. 1-15.

69. **Beyer, H.-G.** On self-adaptive features in real-parameter evolutionary algorithms [Текст] / H.-G. Beyer, K. Deb // IEEE Transactions on Evolutionary Computation. – 2001. – Vol. 5, no. 3. – P. 250-270.

70. **Beyer, H.-G.** How to analyse evolutionary algorithms [Текст] : Technical report no. CI-139/02 / H.-G. Beyer, H.-P. Schwefel, I. Wegener. – University of Dortmund, Germany, 2002. // http://sfbci.cs.uni-dortmund.de

71. **Caprari, G.** Fascination of downscaling – Alice the sugarcube robot [Текст] / G. Caprari, T. Estier, R. Siegwart // Journal of Micro-Mechatronics. – 2002. – Vol. 3, no. 1. – P. 177- 189.

72. **Caruana, R.A.** Benefiting from the variables that variable selection discards [Текст] / R.A. Caruana, V.R. de Sa // Journal of Machine Learning Research. – 2003. – Vol. 3. – P. 1245-1264.

73. **Chester, D.L.** Why two hidden layers are better than one [Текст] / D.L. Chester // Proceedings of International Joint Conference on Neural Networks. Vol. 1. – 1990. – P. 265-268.

74. **Chen, C.-C.J.** Creating melodies with evolving recurrent neural networks [Текст] / C.-C.J. Chen, R. Miikkulainen // Proceedings of the 2001 International Joint Conference on Neural Networks (IJCNN-01). – IEEE, 2001.

75. **Chernyavskii, A.V.** Image processing using evolving neural network [Текст] / A.V. Chernyavskii, Yu.R. Tsoy, V.G. Spitsyn // XIII International Symposium "Atmospheric and Ocean Optics. Atmospheric Physics", Tomsk, July 2-6, 2006. – P. 104.

76. **Cliff, D.** Explorations in evolutionary robotics [Текст] / D. Cliff, I. Harvey, P. Husbands // Adaptive Behavior. – 1993. – No. 2. – P. 73-110.

77. **Crutchfield, J.P.** Dynamics of evolutionary processes [Текст] / J.P. Crutchfield, P. Schuster // Evolutionary Dynamics – Exploring the Interplay of Selection, Neutrality, Accident, and Function / Eds. J.P. Crutchfield, P. Schuster. – New York: Oxford University Press, 2002.

78. **Cvetkovic, D.** The optimal population size for uniform crossover and truncation selection [Текст] : Technical report no. GMD AS GA 94-11 / D. Cvetkovic, H. Muhlenbein. – German National Research Center for Computer Science (GMD), Germany, 1994.

79. **Cybenko, G.** Approximation by superposition of a sigmoidal functions [Текст] / G. Cybenko. – University of Illinois, 1988.

80. **Davis, L.** Adapting operator probabilities in genetic algorithms [Текст] / L. Davis // Proceedings of Third International Conference on Genetic Algorithms and their Applications. – 1989. – P. 61–69.

81. **De Jong, K.A.** An analysis of the behavior of a class of genetic adaptive systems [Текст] : Unpublished PhD thesis / K. De Jong. – University of Michigan, Ann Arbor, 1975. – Also University microfilms No. 76-9381 // http://www.cs.gmu.edu/~eclab

82. **De Jong, K.A.** Generation gaps revisited [Текст] / K.A. De Jong, Sarma J. // Foundations of Genetic Algotihms 2. – 1993. – P.19-28.

83. **De Jong, K.A.** An Analysis of the Interacting Roles of Population Size and Crossover [Текст] / K.A. De Jong, W.M. Spears // Proceedings of the International Workshop Parallel Problem Solving from Nature. – 1990. – P. 38-47.

84. **De Jong, K.A.** A formal analysis of the role of multi-point crossover in genetic algorithms [Текст] / K.A. De Jong, W.M. Spears // Annals of Mathematics and Artificial Intelligence. – 1992. – Vol. 5, no.5. – P. 1-26.

85. **Deb, K.** Simulated binary crossover for continuous search space [Текст] / K. Deb, R.B. Agrawal // Complex Systems. – 1995. – No. 9. – P. 115-148.

86. **Deb, K.** Self-adaptive genetic algorithms with simulated binary crossover [Текст] / K. Deb, H.-G. Beyer // Evolutionary Computation. – 2000. – Vol. 9, no. 2. – P. 197-221.

87. **Derakhshani, R.** GETnet: A general framework for evolutionary temporal neural networks [Текст] / R. Derakhshani // Proceedings of International Joint Conference on Neural Networks. – Montreal, Canada, 2005. – P. 3150-3155.

88. “Eden” Project // www.csse.monash.edu.au/~jonmc/projects/eden/

89. **Egmont-Petersen, M.** Image processing with neural networks – a review [Текст] / M. Egmont-Petersen, D. deRidder, H. Handels // Pattern Recognition. – 2002. – Vol. 35. – P. 2279–2301.

90. **Eiben, A.E.** Evolutionary algorithms with on-the-fly population size adjustment [Текст] / A.E. Eiben, E. Marchiori, V.A. Valko // Parallel Problem Solving from Nature, PPSN VIII, LNCS Vol. 3242 / eds. X. Yao [et al.]. – Berlin: Springer-Verlag, 2004. – P. 41-50.

91. **Eshelman, L.J.** Biases in the crossover landscape [Текст] / L.J. Eshelman, R.A. Caruana, J.D. Schaffer // Proceedings of the Third International Conference on Genetic Algorithms. – 1989. – P.10-19.

92. **Fahlman, S.E.** The cascade-correlation learning architecture [Текст] / S.E. Fahlman, C. Lebiere // Advances in Neural Information Processing Systems 2 / ed. Touretzky D. S. – San Mateo, CA: Morgan Kaufman Publishers, 1990. – P. 524-532.

93. **Fernandes, C.** Self-regulated population size in evolutionary algorithms [Текст] / C. Fernandes, A. Rosa // Proceedings of the 9-th International Conference Parallel Problem Solving from Nature PPSN IX. Lecture Notes in Computer Science no. 4193 / eds. T.P. Runarsson [et al.]. – Berlin Heidelberg: Springer-Verlag, 2006. – P. 920-929.

94. **Ferreira, C.** Gene expression programming: Mathematical modeling by an artificial intelligence [Электронный ресурс] / C. Ferreira. – Angra do Heroismo, Portugal, 2002. // http://gene-expression-programming.com/ GepBook/Introduction.htm

95. **Fiszelew, A.** Automatic generation of neural networks based on genetic algorithms [Текст] / A. Fiszelew, P. Britos, G. Perichisky, R. Garcia-Martinez // Revista Eletronica de Sistemas de Informacao. – 2003. – Vol. 2, no. 1. // http://citeseer.ist.psu.edu/

96. **Floreano, D.** Evolution of spiking neural circuits in autonomous mobile robots [Текст] / D. Floreano, Y. Epars, J.C. Zufferey, C. Mattiussi // International Journal of Intelligent Systems. – 2006. – Vol. 21, no. 9. – P. 1005-10024.

97. **Fogarty, T.** Varying the probability of mutation in the genetic algorithm [Текст] / T. Fogarty // Proceedings of the 3rd International Conference on Genetic Algorithms / ed. J.D. Schaffer. – Morgan Kaufmann, 1989. – P.104-109.

98. **Fogel, L.J.** An evolutionary programming approach to self-adaption on finite state machines [Текст] / L.J. Fogel, P.J. Angeline, D.B. Fogel // Proceedings of the Forth Annual Conference on Evolutionary Programming / eds. J.R.

McDonnell, R.G. Reynolds, D.B. Fogel. – Massachusetts, MA: MIT Press, 1995. – P.355-365.

99. **Fogel, D.B.** Meta-evolutionary programming [Текст] / D.B. Fogel, L.J. Fogel, J.W. Atmar // Proceedings of 25th Asilomar Conference on Signals, Systems and Computers / ed. R.R. Chen. – Pacific Grove, CA, 1991. – P. 540-545.
100. **Fukushima, K.** Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position [Текст] / K. Fukushima, S. Miyake // Pattern Recognition. – 1982. – Vol. 15, no. 6. – P. 455–469.
101. **Fullmer, B.** Using marker-based genetic encoding of neural networks to evolve finite state behavior [Текст] / B. Fullmer, R. Miikkulainen // Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life (ECAL-91). – Paris, 1991. – P.255-262. // http://www.nn.cs.utexas.edu/
102. **Funanashi, K.** On the approximate realization of continuous mappings by neural networks [Текст] / K. Funanashi // Neural Networks. – 1989. – Vol. 2. – P. 183-192.
103. **Gagne, C.** Genericity in evolutionary computation software tools: principles and case-study [Текст] / C. Gagne, M. Parizeau // International Journal on Artificial Intelligence Tools. – 2006. – Vol. 15, no. 2. – P. 173–174.
104. **Gallant, A.R.** There exists a neural network that does not make avoidable mistakes [Текст] / A.R. Gallant, H. White // Proceedings of IEEE International Conference on Neural Networks. Vol. 1. – San Diego, CA: Lawrens Erlbaum Accociates, 1988. – P. 657-664.
105. **Gilev, S.E.** On completeness of the class of functions computable by neural networks [Текст] / S.E. Gilev, A.N. Gorban // Proc. of the World Congress on Neural Networks (WCNN'96). – San Diego, CA: Lawrens Erlbaum Accociates, 1996. – P. 984-991.

106. GNUGO [Электронный ресурс] // http://www.gnu.org/software/gnugo/gnugo.html

107. **Goldberg, D.E.** Genetic algorithms in search, optimization and machine learning [Текст] / D.E. Goldberg. – Addison-Wesley, 1989.

108. **Goldberg, D.E.** A comparative analysis of selection schemes used in genetic algorithms [Текст] / D. Goldberg, K. Deb // Foundations of Genetic Algorithms. – 1991. – P.69–93.

109. **Goldberg, D.E.** Genetic algorithms, noise, and the sizing of populations [Текст] / D.E. Goldberg, K. Deb, J.H. Clark // Complex Systems. – 1992. – No.6. – P. 333-362.

110. **Gomez, F.** Incremental evolution of complex general behavior [Текст] / F. Gomez, R. Miikkulainen // Adaptive Behavior. – 1997. – No. 5. – P. 317-342. // http://www.nn.cs.utexas.edu/

111. **Gomez, F.** 2D pole balancing with recurrent evolutionary networks [Текст] / F. Gomez, R. Miikkulainen // Proceedings of International Conference on Artificial Neural Networks (ICANN-98). – New-York: Elsevier, 1998. // http://www.nn.cs.utexas.edu/

112. **Gomez, F.** Solving non-markovian control tasks with neuroevolution [Текст] / F. Gomez, R. Miikkulainen // Proceedings of the International Joint Conference on Artificial Intelligence. – San Francisco, CA, 1999. – P.1356-1361. // http://www.nn.cs.utexas.edu/

113. **Gomez, F.** Active guidance for a finless rocket using neuroevolution [Текст] / F. Gomez, R. Miikkulainen // Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO 2003). – Washington, DC, 2003. // http://www.nn.cs.utexas.edu/

114. **Gomez, J.** Self Adaptation of operator rates in evolutionary algorithms [Текст] / J. Gomez // Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO 2004). LNCS no. 3102. – Berlin: Springer-Verlag, 2004. – P.1162-1173.

115. **Gonzalez, R.C.** Digital image processing [Текст] / R.C. Gonzalez, R.E. Woods. – Reading MA: Addison-Wesley, 1992.

116. **Grefenstette, J.J.** Optimization of control parameters for genetic algorithms [Текст] / Grefenstette J.J. // IEEE Transactions on Systems, Man, and Cybernetics. – 1986. – No. 1 (16) – P.122-128.

117. **Gruau, F.** Neural network synthesis using cellular encoding and the genetic algorithm [Текст] : Unpublished PhD thesis / F. Gruau. – l'Universite Claude Bernard, Lyon, 1994. // http://citeseer.ist.psu.edu/

118. Handbook of genetic algorithms [Текст] / ed. L. Davis. – New York: Van Nostrand Reinhold, 1991.

119. **Hansen, N.** Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation [Текст] / N. Hansen, A. Ostermeier // Proceedings of the IEEE Conference on Evolutionary Computation. – 1996. – P.312-317.

120. **Hansen, N.** Completely derandomized self-adaptation in evolution strategies [Текст] / N. Hansen, A. Ostermeier // Evolutionary Computation. – 2000. – Vol. 9, no. 2. – P. 159-195.

121. **Harik, G.R.** The gambler's ruin problem, genetic algorithms, and the sizing of populations [Текст] / G.R. Harik, E. Cantu-Paz, D.E. Goldberg, B.L. Miller // Proceedings of the 4th IEEE Conference on Evolutionary Computation. – IEEE Press, 1997. – P.7-12.

122. **Harik, G.R.** A parameter-less genetic algorithm [Текст] / G.R. Harik, F.G. Lobo // Proceedings of the Genetic and Evolutionary Computation Conference. Vol.1. – Morgan Kaufmann, 1999. – P.258-265.

123. **Hassibi, B.** Second order derivatives for network pruning: Optimal brain surgeon [Текст] / B. Hassibi, D.G. Stork // Advances in Neural Information Processing Systems 5 / eds. S.J. Hanson, J.D. Cowan, C.L. Giles. – San Mateo, CA: Morgan Kaufman Publishers, 1993. – P. 164-171.

124. **Hecht-Nielsen, R.** Kolmogorov's mapping neural network existing theorem [Текст] / R. Hecht-Nielsen // Proceedings of the First IEEE International Conference on Neural Networks. – San Diego, CA, 1987. – Vol. 3. – P. 11-14.

125. **Herrera, F.** Hybrid crossover operators for real-coded genetic algorithms: An experimental study [Текст] / F. Herrera, M. Lozano, A. M. Sanchez. – 2002.

126. **Hesser, J.** Towards an optimal mutation probability for genetic algorithms [Текст] / J. Hesser, R. Manner // Proceedings of the 1st Conference on Parallel Problem Solving from Nature. LNCS No. 496 / eds. H.-P. Schwefel, R. Manner. – Berlin: Springer-Verlag, 1990. – P.23-32.

127. **Heistermann, J.** Different learning algorithms for neural networks – a comparative study [Текст] / J. Heistermann // Parallel Problems Solving from Nature, Workshop Proceedings / eds. Y. Davidor, H.P. Schwefel, R. Manner. – Springer-Verlag, 1994. – P. 368-396.

128. **Hinterding, R.** Gaussian mutation and self-adaptation in numeric genetic algorithms [Текст] / R. Hinterding // IEEE International Conference on Evolutionary Computation. – IEEE Press, 1995. – P. 384-389.

129. **Hinterding, R.** Adaptation in evolutionary computation: A survey [Текст] / R. Hinterding, Z. Michalewicz, A.E. Eiben // Proceedings of the 4th IEEE International Conference on Evolutionary Computation. – Indianapolis, 1997. – P.65-69.

130. **Hinterding, R.** Self-adaptive genetic algorithm for numeric functions [Текст] / R. Hinterding, Z. Michalewicz, T. Peachey // Parallel Problem Solving from Nature PPSN IV, LNCS vol. 1141 / eds. H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel. – Berlin: Springer-Verag, 1996. – P. 420-429.

131. **Holland, J.H.** Adaptation in natural and artificial systems [Текст] / J.H. Holland. – Michigan: The University of Michigan Press, 1975.

132. **Holland, J.H.** Building blocks, cohort genetic algorithms, and hyperplane-defined functions [Текст] / J.H. Holland // Evolutionary Computation. – 2000. – Vol. 8, no. 4. – P. 373-391.

133. **Hopfield, J.J.** Neural networks and physical systems with emergent collective computational abilities [Текст] / J.J. Hopfield // Proc. Natl. Acad. Sci. USA. – 1982. – Vol. 79, no. 8. – P. 2554-2558.

134. **Hornik, K.M.** Multilayer feedforward networks are universal approximators [Текст] / K.M. Hornik, M. Stinchcombe, H. White // Neural Networks. – 1989. – Vol. 2. – P. 359-366.

135. **Igel, C.** Neuroevolution for reinforcement learning using evolution strategies [Текст] / C. Igel // Proceedings of Congress on Evolutionary Computation 2003 (CEC 2003), Vol. 4. – IEEE Press, 2003. – P. 2588-2595. // http://www.neuroinformatik.ruhr-uni-bochum.de/PEOPLE/igel/ publications.html

136. **Igel, C.** Operator adaptation in evolutionary computation and its application to structure optimization of neural networks [Текст] / C. Igel, M. Kreutz // Neurocomputing. – 2003. – Vol. 55, no. 1-2. – P. 347-361.

137. **Igel, C.** Evolutionary optimization of neural systems: The use of strategy adaptation [Текст] / C. Igel, S. Wiegand, F. Friedrichs // Trends and Applications in Constructive Approximation. International Series of Numerical Mathematics / eds. M.G. de Bruin, D.H. Mache, J. Szabados. – Birkhauser Verlag, 2005.

138. **Jahne, B.** Digital Image Processing [Текст] / B. Jahne. – Berlin: Springer-Verlag, 2002.

139. **James, D.** A comparative analysis of simplification and complexification in the evolution of neural network Topologies [Текст] / D. James, P. Tucker // Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2004). – New York, NY: Springer-Verlag, 2004. // http://citeseer.ist.psu.edu/

140. **Julstrom, B.** What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm [Текст] / B. Julstrom // Proceedings of Sixth International Conference on Genetic Algorithms. – 1995. – P.81-87.

141. **Elad, M.** Reduced complexity Retinex algorithm via the variational approach [Текст] / M. Elad, R.Kimmel, D. Shaked, R. Keshet // J. Vis. Commun. Image R. – 2003. – No. 14. – P. 369–388.

142. **Keijzer, M.** Evolving objects: a general purpose evolutionary computation library [Текст] / M. Keijzer, J.J. Merelo, G. Romero, M. Schoenauer // In EA-01, Evolution Artificielle, 5th International Conference in Evolutionary Algorithms, 2001.

143. **Kita, H.** A comparison study of self-adaptation in evolution strategies and real-coded genetic algorithms [Текст] / H. Kita // Evolutionary Computation. – 2000. – Vol. 9, no. 2. – P. 223-241.

144. **Kitano, H.** Designing neural network using genetic algorithm with graph generation system [Текст] / H. Kitano // Complex Systems. – 1990. – No. 4. – P. 461–476.

145. **Kochenov, D.A.** Approximations of functions of C[A,B] class by neuralnet predictors (architectures and results) [Текст] / D.A. Kochenov, D.A. Rossiev // AMSE Transaction, Scientific Siberian, A. – 1993. – Vol. 6. Neurocomputing. – PP. 189-203.

146. **Kohonen, T.** The self organizing map [Текст] / T. Kohonen // Proc. of IEEE. – 1990. – Vol. 78. – P. 1464-1479.

147. **Kohonen, T.** Self-organizing maps [Текст] / T. Kohonen. – Berlin Heidelberg: Springer-Verlag, 1995.

148. **Korning, P.G.** Training of neural networks by means of genetic algorithms working on very long chromosomes [Текст] / P.G. Korning // International Journal of Neural Systems. – 1995. – Vol. 6, no. 3. – P. 299-316. // http://citeseer.ist.psu.edu/

149. **Koza, J.** Genetic programming: a paradigm for genetically breeding computer population of computer programs to solve problems [Текст] / J. Koza. – Cambridge, MA: MIT Press, 1992.

150. **Kreinovich, V.Y.** Arbitrary nonlinearity is sufficient to represent all functions

by neural networks: A theorem [Текст] / V.Y. Kreinovich // Neural Networks. – 1991. – Vol.4. – P.381-383.

151. **Land, E.** Recent advances in retinex theory [Текст] / E. Land // Vision Research. – 1986. – Vol. 26, no. 1. – P. 7-21.
152. **Law, D.** Grounding robotic control with genetic neural networks [Текст] : Technical report no. AI94-223 / D. Law, R. Miikkulainen. – The University of Texas at Austin, 1994. // http://www.nn.cs.utexas.edu/
153. **LeCun, Y.** Optimal brain damage [Текст] / Y. LeCun, J.S. Denker, S.A. Solla // Advances in Neural Information Processing Systems 2 / ed. D.S. Touretzky. – San Mateo, CA: Morgan Kaufman Publishers, 1990. – P. 598-605. // http://yann.lecun.com/
154. **Light, W.A.** Ridge functions, sigmoidal functions and neural networks [Текст] / W.A. Light / In E.W. Cheney, C.K. Chui, L.L. Schumacher (eds.): Approximation theory VII. – Boston: Academic Press, 1992. – P. 163-206.
155. **Lippmann, R.** An introduction to computing with neural nets [Текст] / R. Lippmann // IEEE ASSP Magazine. – 1987.– April. – P. 4-22.
156. **Lobo, F.** The parameter-less genetic algorithm: rational and automated parameter selection for simplified genetic algorithm operation [Текст] : Unpublished PhD thesis / F. Lobo. – The University of Lisbon, Portugal, 2000.
157. **Lubberts, A.** Co-evolving a go-playing neural network [Текст] / A. Lubberts, R. Miikkulainen // Proceedings of Coevolution: Turning Adaptive Algorithms upon Themselves, Birds-of-a-Feather Workshop, Genetic and Evolutionary Computation Conference (GECCO-2001). – San Francisco, CA: Kaufmann, 2001. – P. 14-19. // http://www.nn.cs.utexas.edu/
158. **Luke, S.** ECJ13: A java-based evolutionary computation and genetic programming research system [Электронный ресурс] / S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, J. Bassett, R. Hubley, A. Chircop // 2005 // http://cs.gmu.edu/?eclab/projects/ecj.
159. **Lund, H.H.** Specialization in populations of artificial neural networks [Текст

] / H.H. Lund, B.H. Mayoh // Proceedings of the 5-th Scandinavian Conference on Artificial Intelligence (SCAI'95). – Amsterdam: IOS Press, 1995. // http://citeseer.ist.psu.edu/

160. **Maniezo, V.** Genetic evolution of the topology and weight distribution of neural networks [Текст] / V. Maniezo // IEEE Transactions of on Neural Networks. – 1994. – Vol. 5, no. 1.
161. **Marquardt, D.** An algorithm for least squares estimation of nonlinear parameters [Текст] / D. Marquardt // SIAM. – 1963. – P. 431-442.
162. **McCormack, J.** New challenges for evolutionary music and art [Электронный ресурс] / J. McCormack // SIGEVOlution newsletter. – 2006. – Vol. 1, no.1. – P. 5-11.
163. **McCulloch, W.S.** A logical calculus of ideas immanent in nervous activity [Текст] / W.S. McCulloch, W.H. Pitts // Bulletin of Mathematical Biophysics. – 1943. – Vol.5. – P. 115-133.
164. **Michalewicz, Z.** Genetic algorithms + data structures = Evolution Programs [Текст] / Z. Michalewicz. – Berlin: Springer-Verlag, 1992.
165. **Minsky, M.L.** Perceptrons [Текст] / M.L. Minsky, S.A. Papert – Cambridge, MA: MIT Press, 1969.
166. **Montana, D.** Training feedforward neural networks using genetic algorithms [Текст] / D. Montana, L. Davis // Proceedings of the 11th International Joint Conference on Artificial Intelligence. – San Mateo, CA: Morgan Kaufmann, 1989. – P. 762–767.
167. **Moriarty, D.E.** Efficient reinforcement learning through symbiotic evolution [Текст] / D.E. Moriarty, R. Miikkulainen // Machine Learning. – 1996. – No. 22. – P. 11-33. // http://www.nn.cs.utexas.edu/
168. **Moriarty, D.E.** Forming neural networks through efficient and adaptive coevolution [Текст] / D.E. Moriarty, R. Miikkulainen // Evolutionary Computation. – 1998. – Vol. 5, no. 4. // http://www.nn.cs.utexas.edu/
169. **Munteanu, C.** Gray-scale image enhancement as an automatic process driven

by evolution [Текст] / C. Munteanu, A. Rosa // IEEE Trans. on Systems, Man, and Cybernetics – part B: Cybernetics. – 2004. – Vol. 34, no. 2.

170. **Littmann, E.** Cascade network architectures / E. Littmann, H. Ritter // Proceedings of International Joint Conference on Neural Networks. – 1992. – Vol. 2. – P. 398-404.
171. **Nguyen, T.C.** Evolvable 3D modeling for model-based object recognition systems [Текст] / T.C. Nguyen, T.S. Huang // Advances in Genetic Programming / ed. K. Kinnear. – Cambridge, MA: MIT Press, 1994.– P. 459-475.
172. **Nolfi, S.** Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines [Текст] / S. Nolfi, D. Floreano. – Cambridge, MA: MIT Press/Bradford Books, 2000.
173. **Nowak, M.** Error thresholds of replication in finite populations: Mutation frequencies and the onset of Muller's ratchet [Текст] / M. Nowak, P. Schuster // Theoretical Biology. – 1989. – No. 137 – P.375-395.
174. **Ostermeier, A.** A derandomized approach to self-adaptation of evolution strategies [Текст] / A. Ostermeier, A. Gawelczyk, N. Hansen // Evolutionary Computation. – 1995. – Vol. 2, no. 4. – P. 369-380.
175. **Pan, Z.** Evolving both the topology and weights of neural networks [Текст] / Z. Pan, L. Kang, S. Nie // Parallel Algorithms and Applications. – 1996. – No. 9. – P. 299-307. // http://citeseer.ist.psu.edu/
176. **Perez-Bergquist, A.S.** Applying ESP and region specialists to neuro-evolution for go [Текст] : Technical report no. CSTR01-24 / A.S. Perez-Bergquist. – The University of Texas at Austin, 2001. // http://www.nn.cs.utexas.edu/
177. **Prechelt, L.** An empirical comparison of seven programming languages [Текст] / L. Prechelt // IEEE Computer. – 2000. – Vol. 33, no. 10. – P. 23–29.
178. **Polani, D.** Eugenic neuro-evolution for reinforcement learning [Текст] / D. Polani, R. Miikkulainen // Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000). – San Francisco, CA: Kaufmann, 2000.

– P. 1041-1046. // http://www.nn.cs.utexas.edu/

179. **Prugel-Bennett, A.** The mixing rate of different crossover operators / A. Prugel-Bennett // Foundations of Genetic Algorithms 6. – 2001. – P. 261–274. // http://citeseer.ist.psu.edu/

180. Quality Assessment Research page [Электронный ресурс] // http://live.ece.utexas.edu/research/quality

181. **Rabani, Y.** A computational view of population genetics / Y. Rabani, Y. Rabinovich, A. Sinclair // Random Structures & Algorithms. 1998. № 12(4). P. 313–334.

182. **Radcliffe, N.** Genetic set recombination and its application to neural network topology optimization [Текст] / N. Radcliffe // Neural Computing and Applications. – 1993. – Vol. 1. – P. 67-90.

183. **Ragg, T.** A comparative study of neural network optimization techniques [Текст] / T. Ragg, H. Braun, H. Landsberg // In Proceedings of the ICANNGA 97. – Berlin Heidelberg: Springer-Verlag, 1997.

184. **Rahman, Z.** Image enhancement, image quality and noise [Текст] / Z. Rahman, D.J. Jobson, G.A. Woodell, G.D. Hines // Proc. of SPIE Photonic Devices and Algorithms for Computing VII. – 2005.

185. **Rechenberg, I.** Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution [Текст] / I. Rechenberg // Werkstatt Bionik und Evolutionstechnik. – Stuttgart: Frommann-Holzboog, 1973.

186. **Rudolph, G.** Self-adaptive mutations may lead to premature convergence [Текст] / G. Rudolph // IEEE Transactions on Evolutionary Computation. – 2001. – Vol. 5, no. 4. – P. 410-414.

187. **Rumelhart, D.E.** Learning representations of back-propagation errors [Текст] / D.E. Rumelhart, G.E. Hinton, R.J. Williams // Nature (London). – 1986. – Vol. 323. – P. 533-536.

188. **Saravanan, N.** Evolving neural control systems [Текст] / N. Saravanan, D.B. Fogel // IEEE Expert. – 1995. – P. 23-27.

189. **Schaffer, J.D.** An adaptive crossover distribution mechanism for genetic algorithms [Текст] / J.D. Schaffer, A. Morishima // Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications / ed. J.J. Grefenstette. – Lawrence Erlbaum Associates, 1987. – P.36-40.

190. **Schlierkamp-Voosen, D.** Adaptation of population sizes by competing subpopulations [Текст] / D. Schlierkamp-Voosen, H. Muhlenbein // Proceedings of the 1996 IEEE Conference on Evolutionary Computation. – Piscataway: IEEE Press, 1996. – P.330-335.

191. **Schmidhuber, J.** Evolino: Hybrid neuroevolution [Текст] / J. Schmidhuber, D. Wierstra, F.J. Gomez // Optimal Linear Search for Sequence Learning. Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI). – Edinburgh, 2005. – P. 853-858.

192. **Schwefel, H.-P.** Numerische optimierung von computer-modellen mittels der evolutionsstrategie [Текст] / H.-P. Schwefel // Interdisciplinary Systems Research. – 1977. Vol. 26.

193. **Shewchuk, J.R.** An introduction to the conjugated gradient without the agonizing pain [Текст] / J.R. Shewchuk – Carnegie-Melon University, Pittsburg, PA, 1994.

194. **Sheikh, H.R.** An Information Fidelity Criterion for Image Quality Assessment Using Natural Scene Statistics [Текст] / H.R. Sheikh, A. C. Bovik, and G. de Veciana // IEEE Transactions on Image Processing. – May 2005.

195. **Smith, J.** Recombination strategy adaptation via evolution of gene linkage [Текст] / J. Smith, T. Fogarty // Proceedings of the IEEE International Conference on Evolutionary Computation. – Piscataway: IEEE Press, 1996. – P. 826-831.

196. **Smith, J.** Self-adaptation of mutation rates in a steady-state genetic algorithm [Текст] / J. Smith, T. Fogarty // Proceedings of the 3rd IEEE International Conference on Evolutionary Computation. – Piscataway: IEEE Press, 1996. – P.318-323.

197. **Smith, R.E.** Adaptively resizing populations: An algorithm and analysis [Текст] / R.E. Smith // Proceedings of the 5th International Conference on Genetic Algorithms / ed. S. Forrest. – San Francisco: Morgan Kaufmann, 1993. – P. 653.

198. **Sontag, E.D.** Feedback stabilization using two-hidden-layer nets [Текст] / E.D. Sontag // IEEE Transactions on Neural Networks. – 1992. – Vol. 3. – P. 981-990.

199. **Spears, W.M.** Adapting crossover in evolutionary algorithms [Текст] / W.M. Spears // Proceedings of the Evolutionary Programming Conference. – Cambridge, MA: The MIT Press, 1995. – P. 367-384.

200. **Spears, W.M.** Evolutionary algorithms: the role of mutation and recombination [Текст] / W.M. Spears. – Berlin Heidelberg: Springer-Verlag, 2000.

201. **Spitsyn, V.G.** Application of Evolving Artificial Neural Network for Image Processing [Текст] / V.G. Spitsyn, Y.R. Tsoy // USNC/URSI National Radio Science and AMEREM Meetings. 9-14 July 2006, Albuquerque, New Mexico, USA. – P. 745.

202. **Sprecher, D.A.** On the structure of continuous functions of several variables [Текст] / D.A. Sprecher // Transactions of the American Mathematical Society. – 1965. – Vol. 115. – P. 340-355.

203. **Srinivas, M.** Adaptive probabilities of crossover and mutation in genetic algorithms [Текст] / M. Srinivas, L.M. Patnaik // Transactions on Systems, Man and Cybernetics. – 1994. – Vol. 24, no. 4. – P. 656-667.

204. **Stanley, K.O.** Evolving neural networks through augmenting topologies [Текст] / K.O. Stanley, R. Miikkulainen // Evolutionary Computation. – 2002. – Vol. 10, no. 2. – P. 99-127. // http://www.nn.cs.utexas.edu/

205. **Stanley, K.O.** Evolving a roving eye for go [Текст] / K.O. Stanley, R. Miikkulainen // Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2004). – New York, NY: Springer-Verlag, 2004. // http://www.nn.cs.utexas.edu/

206. **Stanley, K.O.** The NERO real-time video game [Текст] : Technical report AI-TR-04-312 / K.O. Stanley, B.D. Bryant, R. Miikkulainen. – The University of Texas at Austin, 2004. // http://www.nn.cs.utexas.edu/

207. **Stanley, K.O.** Patterns without development [Текст] : Technical report CS-TR-06-01 / K.O. Stanley. – University of Central Florida, 2006.

208. **Syswerda, G.** Uniform crossover in genetic algorithms [Текст] / G. Syswerda // Proceedings of Third International Conference on Genetic Algorithms. – San Mateo, CA: Morgan Kaufmann, 1989. – P.2-9.

209. **Thierens, D.** Non-redundant genetic coding for neural networks [Текст] : Technical report no.UU-CS-1998-46 / D. Thierens. – Utrecht University, Netherlands, 1998. // http://www.cs.uu.nl/research/techreps/aut/dirk.html

210. **Thierens, D.** Genetic weight optimization of feedforward neural network controller [Текст] / D. Thierens, J. Suykens, J. Vanderwalle, B. De Moor // Proceedings of International Conference on Artificial Neural Networks and Genetic Algorithms. – Berlin Heidelberg: Springer-Verlag, 1993. – P. 658-663.

211. **Todd, S.** Evolutionary art and computers [Текст] / S. Todd, W. Latham. – London: Academic, 1992.

212. **Tsoy, Y.R.** Using genetic algorithm with adaptive mutation mechanism for neural networks design and training [Текст] / Y.R. Tsoy, V.G. Spitsyn // Optical memory and neural networks, 2004. – Vol. 13, № 4. – Pp. 225-232.

213. **Tsoy, Y.R.** Use of Design Patterns for Design of the Software Environment for Researches in Genetic Algorithms [Текст] / Y.R. Tsoy, V.G. Spitsyn // Proceedings of 8-th Korea-Russia International Symposium on Science and Technology KORUS-2004. – Tomsk, 2004. – Pp. 166-168.

214. **Tsoy, Y.R.** Using Genetic Algorithm with Adaptive Mutation Mechanism for Neural Networks Design and Training [Текст] / Y.R. Tsoy, V.G. Spitsyn // Proceedings of 9-th Korea-Russia International Symposium on Science and Technology KORUS-2005. – Novosibirsk, 2005. – Pp. 237-241.

215. **Tsoy, Y.R.** Digital Images Enhancement with Use of Evolving Neural Net-

works [Текст] / Y.R. Tsoy, V.G. Spitsyn // Proceedings of the 9-th International Conference Parallel Problem Solving from Nature – PPSN IX / Runarsson T.P. et al. (Eds.). – Lecture Notes in Computer Science no. 4193. – Berlin Heidelberg: Springer-Verlag, 2006. – P. 593-602.

216. **Tsoy, Y.R.** Evolutionary Algorithms Design: State of the Art and Future Perspectives [Текст] / Y.R. Tsoy // Proceedings of IEEE East-West Design and Test Workshop (EWDTW'06). – Sochi, Russia, September 15-19, 2006. – P. 375-379.
217. **Turing, A.M.** Computing machinery and intelligence [Текст] / A.M. Turing // Mind. – 1950. – Vol. 236, no. 59.
218. **Tuson, A.** Adapting operator settings in genetic algorithms [Текст] / A. Tuson, P. Ross // Evolutionary Computation. – 1998.
219. **Valsalam, V.** Constructing good learners using evolved pattern generators [Текст] / V. Valsalam, J. Bednar, R. Miikkulainen // Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005). – New York: ACM, 2005. – P. 11-18. // http://www.nn.cs.utexas.edu/
220. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, Apr. 2004.
221. **Whitley, D.** Genitor: A different genetic algorithm [Текст] / D. Whitley, J. Kauth // Proceedings of the Rocky Mountain Conference on Artificial Intelligence. – Denver, CO, 1988. – P. 118-130.
222. **Whitley, D.** Genetic Algorithms and Neural Networks: Optimizing Connections and Connectivity / D. Whitley, T. Starkweather, C. Bogart // Parallel Computing, 1990, Vol. 14, pp. 341-361. http://www.cs.colostate.edu/~genitor/
223. **Wieland, A.** Evolving neural network controllers for unstable systems [Текст] / A. Wieland // Proceedings of the International Joint Conference on Neural Networks. – Piscataway, New Jersey, 1991. – P. 667–673.
224. **Woodell, G.A.** Enhancement of imagery in poor visibility conditions [Текст

] / G.A. Woodell, D.J. Jobson, Z. Rahman, G.D. Hines // Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IV. Proc. SPIE 5778. – 2005.

225. **Xiao, J.** Evolutionary planner/navigator: Operator performance and self-tuning [Текст] / J. Xiao, Z. Michalewicz, L. Zhang // Proceedings of the 3rd IEEE International Conference on Evolutionary Computation. – IEEE Press, 1996. – P.366-371.
226. **Yamamichi, T.** Synthesis of binary cellular automata based on binary neural networks [Текст] / T. Yamamichi, T. Saito, K. Taguchi, H. Torikai // Proceedings of International Joint Conference on Neural Networks. Montreal, Canada. – 2005. – P. 1361-1365.
227. **Yang, J.** Feature subset selection using genetic algorithm [Текст] / J. Yang, V. Honavar // Genetic Programming 97. – San Francisco, Morgan Kaufmann Publishers, 1997. – P. 380-385.
228. **Yao, X.** Evolving artificial neural networks / X. Yao // Proceedings of the IEEE. – 1999. – Vol. 87, no. 9. – P. 1423-1447. // http://www.cs.bham.ac.uk/~xin/
229. **Yao, X.** Making Use of population information in evolutionary artificial neural networks [Текст] / X. Yao, Y. Liu // IEEE Transactions on systems, man, and cybernetics - part B: Cybernetics. – 1998. – Vol. 28, no. 3. – P. 417-425. // http://www.cs.bham.ac.uk/~xin/
230. **Yong, C.H.** Cooperative coevolution of multi-agent systems [Текст] : Technical report AI01-287 / C.H. Yong, R. Miikkulainen. – The University of Texas at Austin, 2000. // http://www.nn.cs.utexas.edu/
231. **Yu, T.-L.** Online population size adjusting using noise and substructural measurements [Текст] : IlliGAL report no. 2005017 / T.-L. Yu, K. Sastry, D.E. Goldberg. – The University of Illinois, 2005.

ПРИЛОЖЕНИЕ 1.