Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева»

На правах рукописи

Хритоненко Дмитрий Иванович

АДАПТИВНЫЕ КОЛЛЕКТИВНЫЕ НЕЙРО-ЭВОЛЮЦИОННЫЕ АЛГОРИТМЫ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

05.13.01 – Системный анализ, управление и обработка информации (космические и информационные технологии)

ДИССЕРТАЦИЯ
на соискание ученой степени кандидата технических наук

Научный руководитель
Семенкин Евгений Станиславович
доктор технических наук,
профессор

Красноярск – 2017

Оглавление

ВВЕДЕНИЕ

Актуальность. Век информационных технологий и большие объемы накопленной информации, требующие обработки, являются одной из причин возникновения такого направления как интеллектуальный анализ данных. Основная цель данного направления – поиск ранее неизвестных, нетривиальных, практически полезных и легко интерпретируемых закономерностей. Сегодня методы интеллектуального анализа данных развиваются достаточно динамично и используются во многих областях. Прогнозирование характеристик выпускаемого изделия, распознавание изображений, медицинская диагностика, банковский скоринг – лишь некоторые из задач, решаемых методами интеллектуального анализа данных.

Искусственные нейронные сети (ИНС) – один из распространенных сегодня методов интеллектуального анализа данных. Достоинствами метода является широкая область применения и точность получаемых моделей. К недостаткам ИНС стоит отнести высокую вычислительную сложность их формирования и обучения, а также их модель «черного ящика», которая делает интерпретацию результатов затруднительной. Формирование и обучение ИНС может быть сведено к оптимизационной задаче с переменными целочисленного, бинарного, вещественного и категориального типов. Целевая функция в такой задаче задается алгоритмически и, как правило, имеет большую размерность и пространство поиска, что говорит о высокой вычислительной сложности оптимизационной процедуры. Эволюционные алгоритмы (ЭА) – один из эффективных методов решения таких задач. Использование ЭА делает актуальным вопрос о выборе их настроек и параметров. Решение данного вопроса вызывает трудности даже у опытных исследователей и значительно отражается на качестве получаемой модели. Разработка методов, позволяющих в автоматическом режиме выбирать конфигурацию и настраивать параметры алгоритма ставит перед собой несколько целей: повысить эффективность получаемого решения, снизить вычислительные затраты, минимизировать требования к квалификации пользователя и тем самым

расширить область применимости. Сегодня уже разработано множество процедур настройки параметров эволюционных алгоритмов, однако их совместное применение при решении сложных задач оптимизации является не всегда успешным и недостаточно исследованным. Кроме того, из-за стремительного развития направления *Big Data* становится необходимым использование алгоритмов снижения размерности и объема данных для применения технологий анализа данных, основанных на ИНС. Поиск, разработка, реализация и практическое применение таких методов является предметом научного исследования.

Учитывая сказанное, можно утверждать, что разработка и исследование методов автоматизированного формирования искусственных нейронных сетей и их коллективов при помощи адаптивных эволюционных алгоритмов с применением методов отбора информативных признаков и обучающих примеров является **актуальной научно-технической задачей.**

Целью диссертационной работы является повышение качества нейросетевых моделей интеллектуального анализа данных и снижение вычислительных ресурсов, требуемых для их формирования, за счет использования адаптивных эволюционных алгоритмов оптимизации.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Выполнить обзор и исследовать эффективность методов самонастройки и самоконфигурирования эволюционных алгоритмов оптимизации на репрезентативном множестве тестовых задач.
2. Разработать адаптивный эволюционный алгоритм оптимизации, объединяющий в себе достоинства известных методов самонастройки и самоконфигурирования.
3. Выполнить обзор существующих методов формирования искусственных нейронных сетей с целью выявления наиболее эффективного их них.
4. Разработать эффективные адаптивные алгоритмы формирования искусственных нейронных сетей для решения задач интеллектуального анализа данных.

5. Выполнить обзор существующих методов и разработать эффективный алгоритм автоматического формирования коллективов нейронных сетей.

6. Реализовать разработанные подходы в виде программных систем и протестировать их эффективность на репрезентативном множестве тестовых и реальных задач.

Методы исследования. В процессе выполнения диссертационной работы использовались методы статистической обработки информации, теории вероятностей, эволюционных вычислений, оптимизации, нейросетевого моделирования, системного анализа, коллективного принятия решений, выявления закономерностей в массивах данных.

Научная новизна работы включает следующие пункты:

1. Разработан новый метод адаптивного управления уровнем мутации в эволюционных алгоритмах, отличающийся от известных способом расчета вероятности мутации гена.

2. Разработан новый метод адаптивного управления размером популяции в эволюционных алгоритмах, отличающийся от известных применением динамического показателя успешности подпопуляции.

3. Разработаны и реализованы эволюционные алгоритмы моделирования и оптимизации, отличающиеся от известных использованием эффективной комбинации модификаций самоконфигурирования, адаптации и селекции обучающих примеров.

4. Разработаны новые адаптивные эволюционные алгоритмы формирования искусственных нейронных сетей, отличающиеся от известных методом настройки конфигурации и параметров, а также применением процедур отбора информативных признаков и обучающих примеров.

5. Разработан новый метод построения коллективов искусственных нейронных сетей, отличающийся от известных применением адаптивного алгоритма генетического программирования с механизмом контроля разнообразия внутри коллектива и возможностью использования альтернативных методов анализа данных.

Теоретическая значимость результатов диссертационной работы состоит в разработке новых эволюционных алгоритмов моделирования и оптимизации, а также формирования коллективов искусственных нейронных сетей, позволяющих получать эффективные в смысле определенного исследователем критерия модели при помощи алгоритмов адаптации, что представляет собой вклад в теорию и практику интеллектуального анализа данных нейро-эволюционными алгоритмами.

Практическая ценность. Разработанные алгоритмы реализованы в виде программных систем, зарегистрированных в Роспатенте, и позволяют эффективно решать задачи классификации, регрессии и оптимизации. Программные системы в автоматическом режиме формируют эффективные структуры искусственных нейронных сетей и их коллективов. Они успешно применены при решении задач банковского скоринга, распознавания эмоций, медицинской диагностики, распознавания изображений и других задач анализа данных.

При помощи реализованных в виде программных систем алгоритмов были решены реальные практические задачи распознавания эмоций по речи и прогнозирования уровня заболеваемости населения. Сравнение с аналогами показало высокую эффективность предлагаемых подходов.

Реализация результатов работы. Разработанные в ходе выполнения диссертационного исследования алгоритмы успешно применены при выполнении работ в рамках проектного задания «Разработка теоретических основ автоматизации комплексного моделирования сложных систем методами вычислительного интеллекта» (2.1680.2017/ПЧ), гранта РФФИ № 14-06-00256 «Информационные технологии оценки и прогнозирования экологических рисков», а также в российско-германских проектах (совместно с университетом г. Ульм) «Распределенные интеллектуальные информационные системы обработки и анализа мультилингвистической информации в диалоговых информационно-коммуникационных системах» (ФЦП ИР, ГК №11.519.11.4002) и «Математическое и алгоритмическое обеспечение автоматизированного проектирования аппаратно-программных комплексов интеллектуальной

обработки мультилингвистической информации в распределенных высокопроизводительных системах космического назначения» (ФЦП НПК, ГК № 16.740.11.0742). Кроме того, отдельные решения использовались при выполнении проекта № 140/14 «Разработка теоретических основ эволюционного проектирования интеллектуальных информационных технологий анализа данных» тематического плана ЕЗН СибГАУ, а также гранта РФФИ № 16-41-243064 «Разработка алгоритмов проектирования кооперативных эволюционно-бионических технологий интеллектуального анализа данных с использованием систем на нечеткой логике». Диссертационная работа была поддержана Фондом содействия развитию малых форм предприятий в научно-технической сфере в рамках программы «У.М.Н.И.К» по проекту «Разработка нейро-эволюционных алгоритмов коллективного типа для решения задач интеллектуального анализа данных» в 2014-2016 гг., а также Красноярским Краевым фондом науки в рамках проекта «Распределенные самоконфигурируемые эволюционные алгоритмы автоматического формирования искусственных нейронных сетей».

Разработанные в ходе выполнения диссертации 7 программных систем зарегистрированы в Роспатенте. Две из них переданы в инновационные IT-компании г. Красноярска.

Основные защищаемые положения:

1. Разработанные методы адаптивного изменения размера популяции и величины мутации в эволюционных алгоритмах позволяют более эффективно использовать вычислительные ресурсы при решении задач моделирования и оптимизации этими алгоритмами.

2. Разработанные адаптивные эволюционные алгоритмы моделирования и оптимизации позволяют исследователю отказаться от настройки их основных параметров, что повышает эффективность применения методов, снижает вычислительные затраты и уменьшает влияние квалификации исследователя на ход эволюционного процесса.

3. Разработанные нейро-эволюционные алгоритмы решения задач восстановления регрессии, прогнозирования и классификации не уступают по эффективности известным аналогам.

4. Автоматический отбор информативных признаков внутри эволюционных алгоритма и селекция обучающих примеров динамическим перераспределением вероятности выбора измерения в обучающую выборку позволяют повысить точность получаемых моделей, снизить их сложность и количество вычислительных ресурсов, затрачиваемых на их формирование.

Публикации. По теме работы опубликовано более 25 печатных работ, в том числе 3 – в журналах из Перечня ВАК и 2 – в изданиях, индексируемых в международных базах цитирования Web of Science и/или Scopus. Получено 7 свидетельств о регистрации программных систем в Роспатенте.

Апробация работы. Результаты диссертационной работы докладывались на 11 Всероссийских и Международных научных конференциях, среди которых: Пятая Международная конференция «Системный анализ и информационные технологии» САИТ-2013 (Красноярск, 2013); 2nd, 4th and 5th International Workshops on Mathematical Models and their Applications (Красноярск, 2013, 2015, 2016), Четырнадцатая Национальная конференция по искусственному интеллекту с международным участием (КИИ, Казань, 2014); Всероссийская научно-практическая конференция «Информационно-телекоммуникационные системы и технологии» (ИТСиТ, Кемерово, 2014, 2015); Международная научно-практическая конференция «Решетневские чтения» (Красноярск, 2014, 2015); International Conference on Environmental Engineering and Computer Application (ICEECA, Hong Kong, China, 2014), Восьмой Всероссийский форум студентов, аспирантов и молодых ученных (Санкт-Петербург, 2014).

Структура работы. Диссертация состоит из введения, четырех глав, заключения, списка литературы и приложений.

ГЛАВА 1. ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ЭВОЛЮЦИОННЫХ АЛГОРИТМОВ МОДЕЛИРОВАНИЯ И ОПТИМИЗАЦИИ

1.1 История развития эволюционных алгоритмов

Появление эволюционных вычислений неразрывно связано с развитием ЭВМ и возможностью математического моделирования. Как и многие другие современные отрасли вычислительного интеллекта, такие как нейронные сети, эволюционные алгоритмы впервые появились в 50-х годах прошлого века [18, 21, 13], однако не получили широкого распространения ввиду недостаточной вычислительной мощи компьютеров того времени.

Существенное значение для распространения эволюционных алгоритмов сыграли работы Дж. Холланда, в частности, [51], в которой он исследовал клеточные автоматы. По настоящему широкое распространение эволюционные алгоритмы, и в частности генетический алгоритм, получили в конце 80-х и начале 90-х годов, когда они стали применяться для решения ряда практических задач. Значительный вклад в развитие эволюционных алгоритмов также был внесен работами Фогеля [35, 34, 36] и Швефеля [84, 85].

Основная идея всех эволюционных алгоритмов заключалась в следовании принципу выживания сильнейшего. Учитывая многообразие существующих сегодня вариантов эволюционных алгоритмов, можно сказать, что данный принцип является основной отличительной особенностью эволюционных алгоритмов. Кроме этого, подавляющее большинство эволюционных алгоритмов использует биологическую терминологию: множество решений называется популяцией, одно решение – индивидом, генетический код решения – хромосомой, итерация алгоритма – поколением и т.д.

Растущая популярность эволюционных алгоритмов в 80-е и 90-е годы привела к появлению множества различных направлений. В частности, помимо классических генетических алгоритмов, работающих с бинарной хромосомой, в которую могли быть закодированы сколь угодно сложные структуры, в 90-х были

предложены генетическое программирование [14, 63], эволюционное программирование, эволюционные стратегии [44], дифференциальная эволюция [110], а также многие другие.

Разнообразие эволюционных алгоритмов обусловлено множеством задач, для которых они применяются. В частности, генетическое программирование отличается представлением решений: вместо хромосомы фиксированной длины решения чаще всего представляют собой деревья, способные кодировать структуры произвольной сложности. При этом основная структура в каждом эволюционном алгоритме претерпевает незначительные изменения, в частности, большинство алгоритмов имеют следующие операции (операторы): селекция – следуя принципу выживания более приспособленных особей, выбирает их для следующих операторов, скрещивание – следует принципу наследственности, перемешивая генетическую информацию, но при этом не добавляя новую, и мутация – следует принципу изменчивости, внося небольшие случайные изменения в генетический код. Возможность кодирования решений различными способами привело к появлению множества вариантов каждого типа операторов в зависимости от решаемой задачи.

Среди областей, где эволюционные методы показали значительные успехи, можно привести не только задачи вещественной оптимизации, но также задачи комбинаторной оптимизации [41, 73, 60], многокритериальной оптимизации [29, 114], в которой эволюционные алгоритмы стабильно показывают существенно лучшие результаты, чем другие методы, а также условной оптимизации [117].

Кроме этого, эволюционные алгоритмы начали широко использоваться для проектирования систем на нечеткой логике и нейронных сетей. В частности, для нечетких систем эволюционные алгоритмы зачастую используются для нахождения нечетких правил и баз правил [54, 53], в то время как для нейронных сетей оптимизируются структуры сети и веса [107].

Современные направления исследований эволюционных алгоритмов и их разновидностей все чаще включают использование адаптации и самонастройки их параметров, так как эффективность эволюционных алгоритмов напрямую зависит

от их значений [72, 87]. Кроме этого, в большинстве случаев лучшие результаты для множества задач показывают алгоритмы, использующие множество популяций-агентов, в том числе так называемые коэволюционные подходы [78] и островные модели [93, 3].

1.2 Описание стандартных эволюционных алгоритмов

1.2.1 Общие определения и структурная схема алгоритма

В работе используются следующие разновидности эволюционных алгоритмов: генетический алгоритм (ГА), алгоритм генетического программирования (ГП). Эволюционные алгоритмы заимствуют терминологию из биологии. Для лучшего понимания механизма работы ЭА определим их:

- Индивид (фенотип) – некоторое решение, выраженное в терминах поставленной задачи;
- Хромосома (генотип) – закодированное в виде строки решение задачи, состоящее из цепочки генов;
- Ген – элемент хромосомы, являющийся символом некоторого алфавита, присущего конкретному эволюционному алгоритму;
- Популяция – конечное множество индивидов текущего поколения;
- Поколение – итерация алгоритма;
- Функция пригодности – преобразованная целевая функция (функционал), которая ставит в соответствие индивиду некоторую пригодность, которая в дальнейшем именуется пригодностью индивида.

Работа представленных эволюционных алгоритмов начинается с процедуры определения параметров. Исследователем определяется:

1. Способ представления решений. В случае использования ГА может быть использовано стандартное или Грей кодирование десятичного числа;

2. Размеры популяции и число поколений ЭА. Данные параметры влияют на эффективность работы алгоритма, а также на его вычислительную сложность;

3. Критерий останова;

4. Вид функции пригодности ЭА. Он зависит от типа решаемой задачи, а также от пожеланий исследователя относительно вида получаемого решения;

5. Способ инициализации популяции алгоритма;

6. Комбинацию генетических операторов, используемую в эволюционном алгоритме.

Структурная схема эволюционного алгоритма представлена на рисунке 1.1.

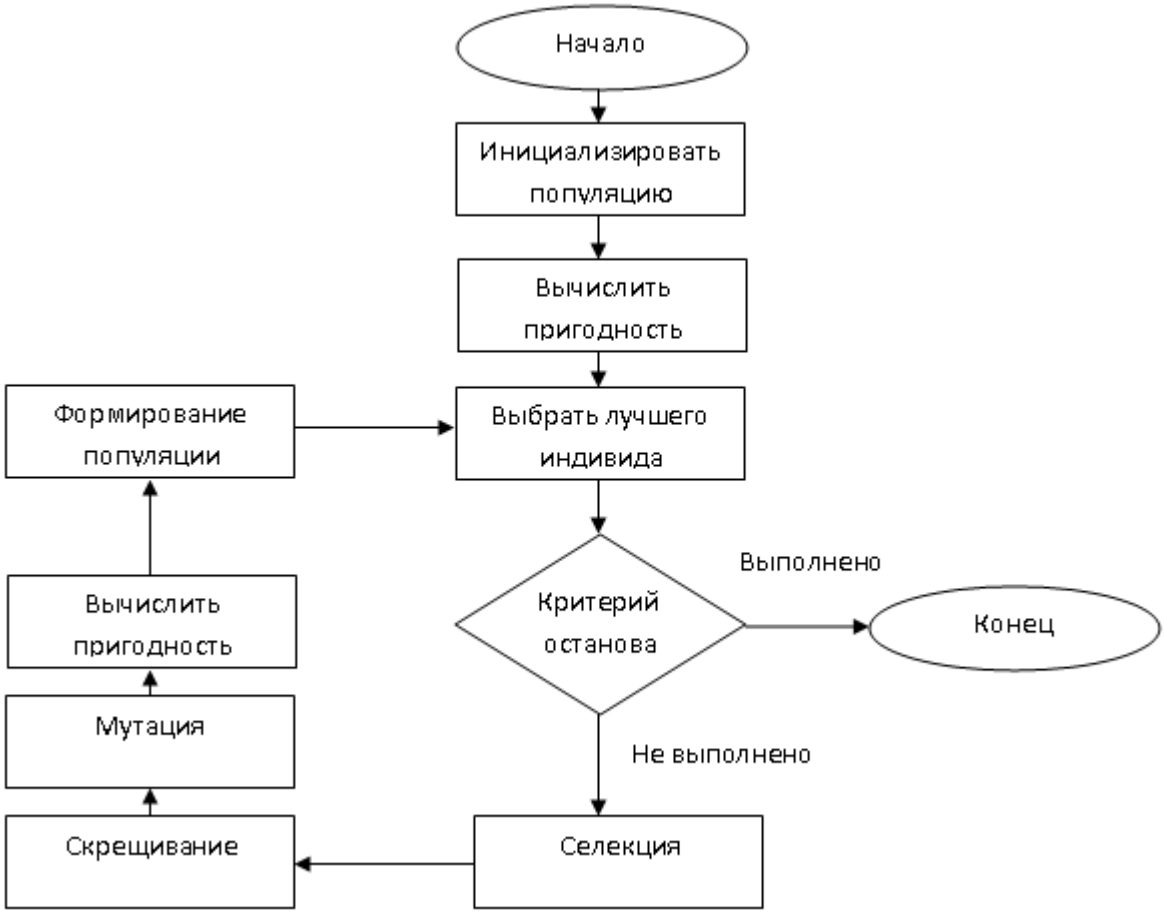

Рисунок 1.1 – Блок-схема стандартного генетического алгоритма

Представленные алгоритмы ГА и ГП имеют общую структуру (рисунок 1.1), однако функционирование некоторых из генетических операторов зависит от способа представления решения. Их подробное описание представлено в пункте 1.2.2. В этом описании используется генетический алгоритм однокритериальной

безусловной оптимизации и алгоритм генетического программирования для решения задач восстановления символьной регрессии.

В ГА используется классическая постановка задачи оптимизации. Термин «символьная регрессия» требует пояснения. Задача символьной регрессии заключается в нахождении математического выражения в символьной форме, аппроксимирующего зависимость между конечным набором значений независимых переменных и соответствующими значениями зависимых переменных. Постановка задачи может выглядеть следующим образом: Пусть имеется выборка $(\bar{x}_i, y_i), i = \overline{1,n}$, где n – объем выборки; $\bar{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})$ – набор значений независимых переменных, m – число переменных задачи; $y_i = f(\bar{x}_i)$ – соответствующая $\bar{x}_i$ зависимая переменная. Необходимо найти выражение $\hat{f}(\bar{x}_i)$ в символьном виде, которое аппроксимирует зависимость f между $\bar{x}_i$ и y_i соответственно.

Указанные алгоритмы ГА и ГП используют различные способы представления индивидов. В генетическом алгоритме индивид – бинарный вектор фиксированной длинны. Его размер не меняется на протяжении всего функционирования алгоритма. В ГП используется представления решений в виде дерева. Дерево – направленный граф, в котором каждая последующая вершина связана с одной и только одной предыдущей. Приведем пример дерева в ГП (рисунок 1.2).

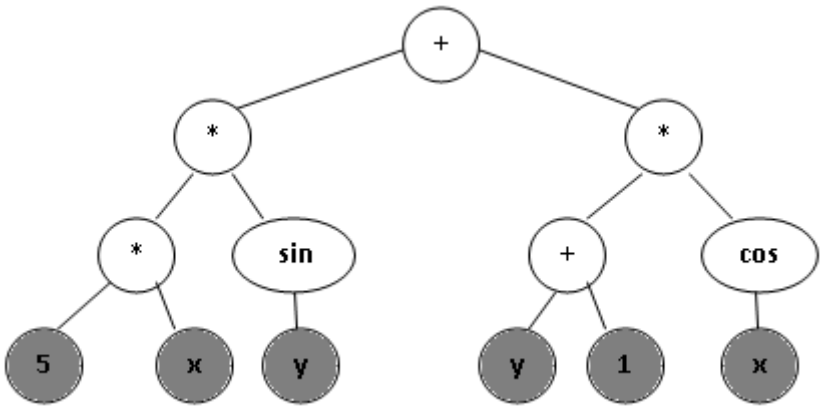

Рисунок 1.2 – Пример решения задачи символьной регрессии при помощи ГП

В данном случае деревом закодировано выражение $\hat{f}(x, y) = 5 \cdot x \cdot \sin(y) + (y + 1) \cdot \cos(x)$.

При работе с ГП определяются понятия функционального и терминального множества. Под функциональным множеством $\{F\}$ понимается множество всех возможных внутренних вершин дерева. Под терминальным $\{T\}$ – множество всех возможных внешних вершин дерева (представлены на рисунке 1.2 затемненными кругами). Объединение функционального и терминального множества – универсальное множество $U = F \cap T$. На функциональное и терминальное множество накладываются следующие условия:

- Замкнутость. Любой элемент множества *F* должен принимать в качестве аргумента любой элемент множества *U* (исключение деления на ноль и т.д.);
- Достаточность. Элементы множества *U* должны быть выбраны так, чтобы можно было эффективно решить поставленную задачу.

В задаче символьной регрессии терминальное множество определяется как множество констант и переменных-аргументов. Для эффективного функционирования алгоритма необходимо оценивание оптимального набора констант дерева (с точки зрения ошибки аппроксимации), т.к. в случае выбора «удачной» структуры (структуры, которая близка, совпадает с истинной или точно аппроксимирует истинную), существенно уменьшается ошибка. Пусть $Const_j$ – набор констант выражения, закодированного деревом T_j. Тогда оптимальный набор констант может быть определен следующим образом:

$$Const_j^{opt} = \arg \min_{Const \in R^k} \left(\frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{f}_j(\bar{x}_i, Const))^2\right) \quad (1.1)$$

В данной формуле *k* – число констант дерева T_j, *n* – объем выборки; $\bar{x}_i$ – набор значений независимых переменных; $\hat{f}_j(\bar{x}_i, Const)$ – вычисленное по дереву T_j значение выражения $\hat{f}_j$ в точке $\bar{x}_i$ при заданных дереву константах *Const*. В формуле 1.1. используется *MSE (mean square error)* критерий. В качестве

альтернативного критерия могут быть использованы *MAE (mean absolute error), RMSE* (*root mean square error*) и другие.

Начало работы ГП предполагает определение элементов терминального и функционального множества. Данные множества формируется пользователем на основе представлений о виде решения. Функциональное множество обычно включает в себя:

- Математические функции (sin, cos и т.д.);
- Арифметические операции ($^{+,-,\times,\div}$);
- Специальные пользовательские функции.

1.2.2 Операторы, зависящие от типа представления решения

Инициализация популяции генетического алгоритма. Чаще всего в ГА используется бинаризация пространства. В таком случае инициализация популяции заключается в случайном равновероятном заполнении генотипа нулями и единицами:

$$Genotype_{i,j} = rand(2), i = \overline{1, PopSize},\ j = \overline{1, GenSize}. \quad (1.2)$$

В данной формуле $Genotype_i$ – генотип *i*-го индивида (второй индекс формулы 1.2 – номер гена); *PopSize* – число индивидов в популяции; *GenSize* – общее количество генов индивида; *rand(t)* – функция, возвращающая случайно и равновероятно целое число $z \in [0, t)$.

Инициализация популяции в алгоритме генетического программирования. В ГП для инициализации популяции используются следующие методы выращивания деревьев:

- Полный. Задается максимальная глубина дерева *d*. Далее для вершин глубины $i = \overline{1, d-1}$ выбираются случайно и равновероятно элементы функционального множества, а для глубины *d* – терминального.

- Выращивание. Задается максимальная глубина дерева *d*. Далее для вершин глубины $i = \overline{1, d-1}$ случайно и равновероятно выбираются элементы из функционального и терминального множества. В случае выбора элемента из терминального множества рост текущей ветви заканчивается. Для глубины *d* выбираются элементы только из терминального множества.
- Комбинированный. Часть популяции выращивается полным методом, а часть – методом роста. Доля деревьев, выращенных полным методом – параметр, выбираемый исследователем.

Вычисление пригодности в генетическом алгоритме. Вычисление пригодности в ГА осуществляется в 2 этапа:

1. Переход от генотипа к фенотипу;
2. На основании полученного фенотипа вычисляется пригодность.

Для работы ГА необходима дискретизация пространства поиска. Для простоты изложения рассмотрим дискретизацию в одномерном случае. Задаются границы поиска по переменной *X*:

$$X \in [left, right]. \tag{1.3}$$

Параметры *left* и *right* определяют пространство поиска задачи оптимизации. Необходимым является определение шага дискретизации *step*. Данный параметр определяет требования к точности получаемого решения. Тогда число точек *NumPoints* находящихся на отрезке [*left,right*] определяется по следующей формуле:

$$NumPoints = \frac{right - left}{step} + 1. \tag{1.4}$$

Число генов *GenSize*, необходимых для кодирования *NumPoints* точек рассчитывается следующим образом:

$$GenSize = round(\log_2(NumPoints)) + 1. \tag{1.5}$$

Функция *round()* производит округление аргумента в меньшую сторону. Этим объясняется наличие слагаемого +1. Стоит заметить, что уравнение $2^{GenSize} = NumPoints$ в общем случае не выполняется. Поэтому после

определения числа генов в хромосоме индивида необходимо произвести пересчет шага дискретизации *step*:

$$step = \frac{right - left}{2^{GenSize} - 1}. \tag{1.6}$$

Далее вычисляется фенотип:

$$X_i = step \cdot BinToDec(Gen_i) + left, \tag{1.7}$$

где *BinToDec()* – функция, производящая перевод целого числа из бинарного представления в десятичное. При таком переводе нередко используется Грей-код. В случае многомерной оптимизации расчет по формулам (1.3-1.7) производится для каждой из переменных, а размер генотипа индивида вычисляется как сумма генов, используемых для кодирования каждой из переменных.

Итоговая формула вычисления пригодности индивида имеет вид:

$$fitness_i = \alpha \cdot F(X_i). \tag{1.8}$$

В данной формуле $fitness_i$ - пригодность i-го индивида с соответствующим фенотипом X_i. $\alpha = -1$ в случае задачи минимизации, $\alpha = 1$ – в случае задачи максимизации.

Вычисление пригодности в алгоритме генетического программирования. Для решения задач символьной регрессии при помощи ГП используется следующая функция пригодности:

$$fitness(T) = \frac{1}{1 + Error(T)}, \tag{1.9}$$

$$Error(T) = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - eval(\bar{x}_i, T))^2. \tag{1.10}$$

Здесь T – индивид (символьное выражение, представленное деревом); n – объем выборки задачи; $eval(\bar{x}_i, T)$ – функция, которая производит вычисление выражения T в точке $\bar{x}_i$.

Зачастую, необходимым является получение не только точного, но и по возможности простого выражения (в смысле числа операций). Поэтому предлагается следующий критерий:

$$fitness(T) = \frac{1}{1 + Error(T) + \alpha \cdot Lenght(T)}, \quad (1.11)$$

где *Error(T)* определяется по формуле (1.10); *Lenght(T)* – функция, возвращающая значение равное суммарному числу вершин дерева T (это значение также называют сложностью дерева)*;* α – некоторый коэффициент, задаваемый исследователем. Функция пригодности (1.11) позволяет бороться с чрезмерным разрастанием деревьев, а также позволяет в итоге получать некоторое компромиссное (между сложностью выражения и ошибкой аппроксимации) решение.

При решении задач больших размерностей нередко возникает проблема утери различных независимых переменных в решении. Поэтому для задач больших размерностей иногда используются следующие критерии:

$$fitness(T) = \frac{1}{1 + Error(T) + \alpha \cdot Lenght(T)} \cdot \frac{nv(T)}{N}, \quad (1.12)$$

$$fitness(T) = \frac{1}{1 + Error(T) + \alpha \cdot Lenght(T)} \cdot \beta^{nv(T)-N}, \quad (1.13)$$

где *nv(T)* – число различных переменных в выражении T, N – общее число переменных задачи, β – коэффициент, задаваемый исследователем. На практике такие методы расчета пригодности применяются довольно редко, т.к. формулы (1.12-1.13) основываются на гипотезе отсутствия зашумленности и взаимной корреляции атрибутов задачи. Они применяются при уверенности исследователя в информативности всех атрибутов, а также требовании их наличия в получаемой модели.

Скрещивание в ГА. Оператор скрещивания отвечает за получение новых потомков из родителей. При скрещивании потомок может унаследовать только те гены, которые принадлежали хотя бы одному из родителей. Наиболее распространенными являются следующие виды скрещивания:

- Одноточечное. Случайно выбирается точка разрыва хромосом. Обмениваясь частями хромосом, которые остались после точки разрыва, родители формируют новых потомков.

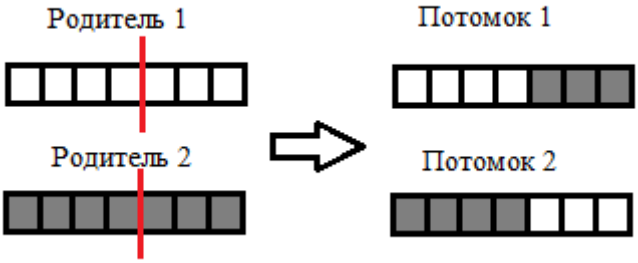

Рисунок 1.3 – Пример одноточечного скрещивания

- Двухточечное. Случайно выбираются две точки разрыва хромосом. Обмениваясь центральными частями хромосом, родители формируют новых потомков

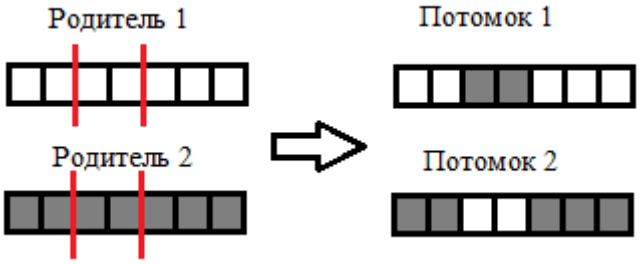

Рисунок 1.4 – Пример двухточечного скрещивания

- Равномерное. Генотип потомка формируется из родительских генотипов случайно и равновероятно.

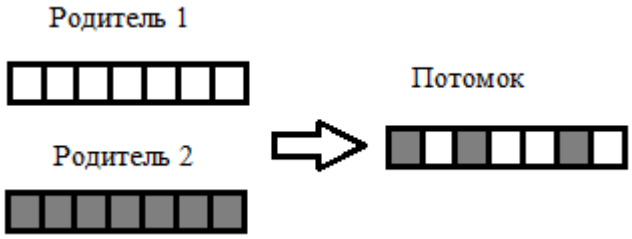

Рисунок 1.5 – Пример равномерного скрещивания

Мутация в генетическом алгоритме. Цель данного оператора – разнообразить генетический материал в популяции. Мутация позволяет избежать стагнации (выводит алгоритм из точек локального оптимума) и заключается в случайном измени генотипа индивида. Мутация – основной оператор поиска в ЭВ. Обычно вероятность мутации невелика. В генетическом алгоритме выделяют 3 уровня мутации:

Таблица 1.1 – Уровни мутации в генетическом алгоритме

Слабая	Средняя	Сильная
$p = \frac{1}{3 \cdot n}$	$p = \frac{1}{n}$	$p = \frac{3}{n}$

В таблице 1.1 p – вероятность мутации гена; n – общее число генов в генотипе. Пример мутации одного из генов при бинарном представлении решений показан на рисунке 1.6.

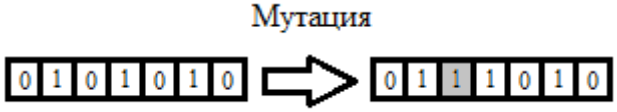

Рисунок 1.6 – Пример мутации в ГА

Скрещивание в алгоритме генетического программирования. В стандартном алгоритме ГП существуют две схемы скрещивания:

- Стандартное. Выбирается пара родителей. У каждого из них выбирается точка скрещивания (дуга в графе). Обмениваясь поддеревьями, находящимися ниже точки скрещивания, родители формируют пару потомков.

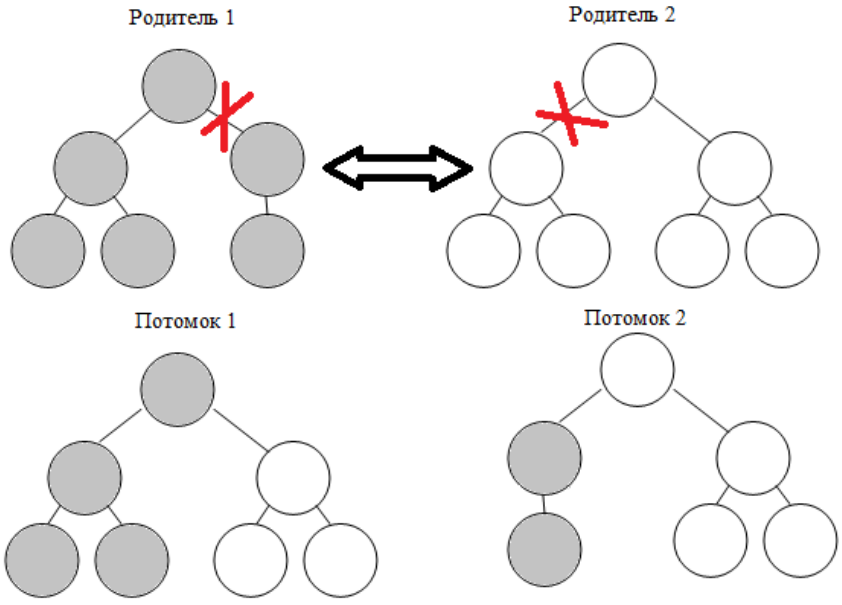

Рисунок 1.7 – Пример стандартного скрещивания

- Одноточечное. При одноточечном скрещивании у родительской пары выбирается общая точка скрещивания. Обмениваясь поддеревьями ниже данной точки, родители формируют пару индивидов. Приведем пример одноточечного

скрещивания. Жирными линями отмечены связи, которые могут быть выбраны в качестве общей точки скрещивания.

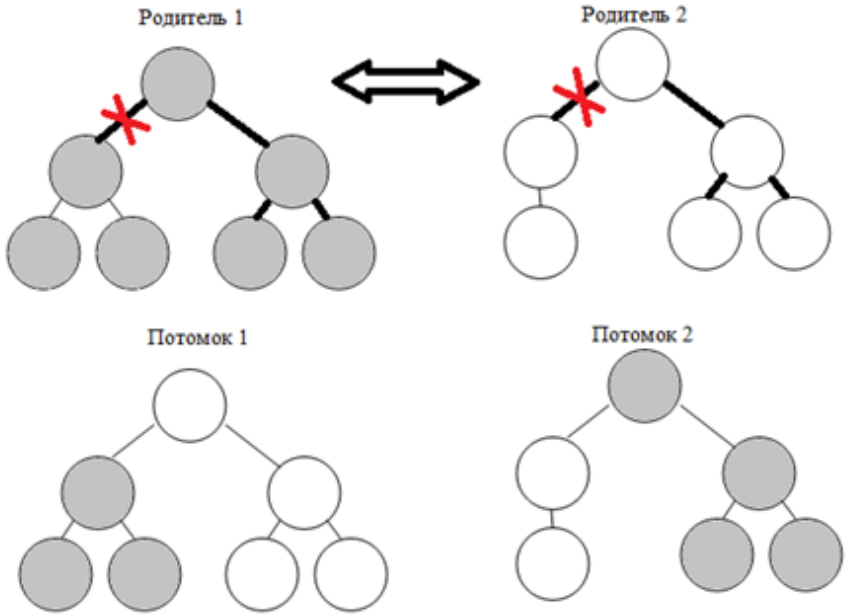

Рисунок 1.8 – Пример одноточечного скрещивания

Следует заметить, что схема стандартного скрещивания имеет склонность к разрастанию деревьев, что негативно сказывается на работе алгоритма. Эффект разрастания наблюдается даже в случае использования функции пригодности (1.11). Подстройка коэффициента α позволяет частично избежать этого, однако в таком случае теряется точность полученных алгоритмом ГП моделей.

Одноточечное скрещивание лишено этого недостатка, т.к. в процессе его работы получить дерево сложнее, чем оба из родителей невозможно. Однако для его эффективной работы необходимо достаточное количество уже выращенных деревьев. При этом они должны быть сложны по своей структуре и разнообразны.

Мутация в ГП. Мутация в алгоритме генетического программирования состоит в изменении одного или нескольких генов в хромосоме. В алгоритме ГП существуют две различных схемы мутации:

- Одноточечная. Заключается в изменении вершины дерева на случайно выбранный элемент того же типа.

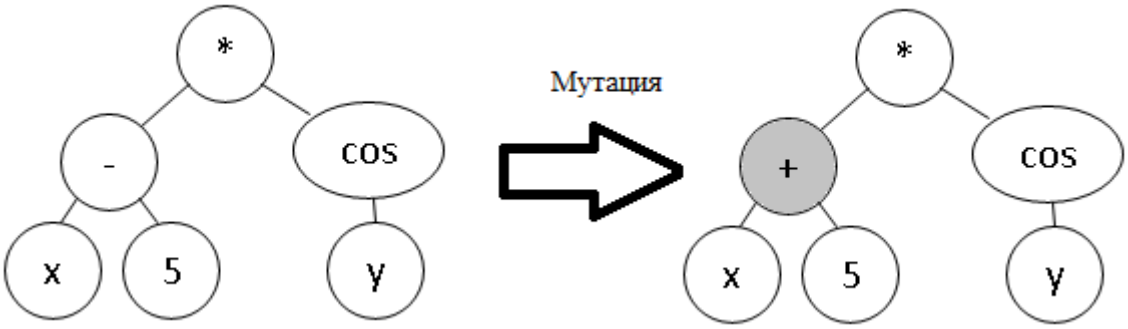

Рисунок 1.9 – Одноточечная мутация

- С заменой ветви. Выбранное поддерево заменяется деревом, полученным методом роста.

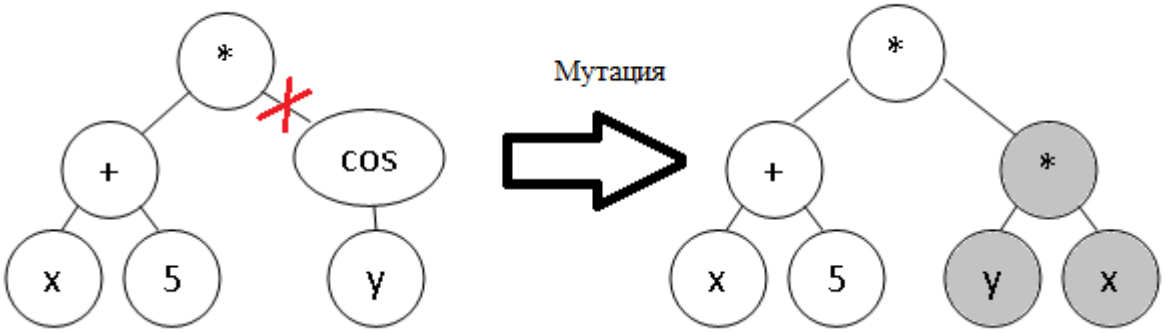

Рисунок 1.10 – Мутация с заменой ветви

1.2.3 Операторы, не зависящие от типа представления решения

Отбор лучшего индивида. ЭА – стохастические алгоритмы. Поэтому для того, чтобы решение улучшалось от поколения к поколению, необходимо сохранять лучший индивид (в смысле значения ЦФ или пригодности) на каждом поколении. Это дает нам гарантию того, что решение не может ухудшиться.

Проверка критерия останова. Критерий останова позволяет своевременно закончить работу алгоритма и не затрачивать лишние вычислительные ресурсы. Некоторые из используемых критериев останова:

- Достижение точки оптимума (возможно лишь на тестовых задачах) алгоритмом;
- Достижение заданной точности (в смысле значения ЦФ);

- Ограничение на число вычислений ЦФ;
- Ограничение на время работы ЭВМ;
- Стагнация алгоритма в точке локального оптимума заданное число поколений;

Селекция. Оператор селекции в эволюционном алгоритме осуществляет селективное давление и сходимость к точкам локального оптимума. Если взять 2 индивида (родителя) с высокой пригодностью и каким-либо образом получить из них потомков (новые решения), то высока вероятность того, что полученные решения также будут иметь высокую (а возможно и превосходящую родителей) пригодность. Основной идеей оператора селекции является следующая: чем выше пригодность индивида, тем больше шансов он имеет стать родителем.

В стандартном генетическом алгоритме выделяют 3 вида селекции: пропорциональную, ранговую, турнирную.

В пропорциональной селекции вероятность выбора индивида в качестве родителя зависит от пригодности:

$$p_i = \frac{fitness_i}{\sum_{j=1}^{n} fitness_j}, i = \overline{1,n} \tag{1.14}$$

Предполагается, что $fitness_i \geq 0 \; \forall \, i = \overline{1,n}$ (если неравенство не выполняется, то проблема легко решается сдвигом пригодностей). n – число индивидов в популяции;

В ранговой селекции производится сортировка по возрастанию пригодностей индивидов. Согласно полученной сортировке, индивидам приписываются ранги (наименьший ранг соответствует индивиду с худшей пригодностью). Если пригодности одинаковы, ранги вычисляются следующим образом:

$$r_i = \frac{\sum_{j=1}^{m} rank_j}{m}, i = \overline{1,n}, \tag{1.15}$$

где m – число индивидов, с одинаковыми пригодностями; $rank$ – ранги, полученные при сортировке; r – новые ранги. Затем, учитывая вновь полученные ранги, производится отбор по схеме пропорциональной селекции;

В турнирной селекции из популяции случайно и равновероятно отбирается некоторое число индивидов (это число задается пользователем) и между ними проводится турнир. Победителем турнира является тот индивид, пригодность которого выше чем у остальных. В случае одинаковых пригодностей право стать родителем разыгрывается случайно и равновероятно.

Формирование популяции. В результате каждой итерации эволюционного алгоритма возникает промежуточная популяция из родителей и потомков. Цель оператора формирования популяции – отобрать из промежуточной популяции индивидов среди родителей и потомков в следующее поколение. Данный оператор необходим для поддержания размера популяции. Известны два основных типа формирования популяции:

- Потомки замещают родителей;
- Новое поколение составляется путем отбора из родителей и потомков.

В эволюционном алгоритме нередко применяются стратегия элитизма: в текущее поколение переходит заданное число лучших особей предыдущего поколения. Использование любого типа формирования популяции и элитизма не допускает потери лучших достигнутых решений.

1.3 Исследование эффективности реализованных эволюционных алгоритмов

Описанные алгоритмы ГА и ГП были программно реализованы в виде программных систем на языке *C++* [122] с использованием среды разработки *CodeBlocks* [17] и компилятором *GCC* [42]. Эффективность алгоритмов проверялась на тестовых задачах конкурса конференции по эволюционным вычислениям (*Conference on Evolutionary Computation, CEC'2017*) [8]. В конкурсе

представлены 30 задач однокритериальной безусловной оптимизации функций 10, 30, 50 и 100 вещественных переменных. Тестовые задачи моделируют и комбинируют различные сложности, с которыми могут столкнуться алгоритмы на практике. Поворот осей, растяжение, сжатие, сдвиг, многоэкстремальность, пики и плато у целевой функции – некоторые из таких сложностей. Каждая из представленных в [8] функций имеет глобальный оптимум, положение которого определяется случайно на интервале [-80, 80] по каждой из переменных, а пространство поиска [-100, 100] по каждой переменной.

Известно, что генетический алгоритм в своей стандартной реализации, т.е. при бинарном представлении решений, плохо приспособлен для решения задач вещественной оптимизации, если требовать получения точки глобального оптимума. Поэтому для определения зависимостей эффективности генетического алгоритма от его параметров будет использована невязка между значением функции в точке глобального оптимума X^* и наилучшим значением $\overline{X}$, полученным генетическим алгоритмом:

$$Error(X) = F(X) - F(X^*) \tag{1.16}$$

Отметим, что все задачи конкурса *CEC'2017* являются задачами минимизации, поэтому $Error(X) \geq 0 \forall X \in [-100,100]^D$, где D – размерность решаемой задачи. Кроме того, длина хромосомы в генетическом алгоритме в 20 раз больше размерности решаемой задачи, т.к. для бинарного представления одной вещественной переменной с необходимой точностью требуется не менее 20 бит.

Эволюционные алгоритмы относится к классу стохастических, поэтому для статистического обоснования выводов в данной главе и в дальнейшем будет использован непараметрический критерий Вилкоксона [109]. Критерием останова в ГА является достижение максимального количества вычислений целевой функции $MaxFES = 10000 \cdot D$, где D – размерность решаемой задачи. Данный критерий предписан конкурсом *CEC'2017*. В случае тестирования алгоритма ГП

критерий останова оперирует числом вычислений функции пригодности (1.9 или 1.11). Критерий эффективности алгоритма ГП – *MAE* (*mean absolute error*):

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} | y_i - \bar{y}_i |, \tag{1.17}$$

где n – размер выборки задачи, y_i – истинное значение в i-ой точке, а $\bar{y}_i$ – значение полученной модели.

Для сравнения оптимизационных алгоритмов по всем из представленных функций *CEC'2017* организаторами был разработан агрегированный критерий:

$$Score = 50 \cdot (1 - \frac{SE - \min(SE)}{SE}) + 50 \cdot (1 - \frac{SR - \min(SR)}{SR}) \tag{1.18}$$

$$SE = 0.1 \cdot \sum_{i=1}^{30} ef_{10}^{i} + 0.2 \cdot \sum_{i=1}^{30} ef_{30}^{i} + 0.3 \cdot \sum_{i=1}^{30} ef_{50}^{i} + 0.4 \cdot \sum_{i=1}^{30} ef_{100}^{i} \tag{1.19}$$

$$SR = 0.1 \cdot \sum_{i=1}^{30} rank_{10}^{i} + 0.2 \cdot \sum_{i=1}^{30} rank_{30}^{i} + 0.3 \cdot \sum_{i=1}^{30} rank_{50}^{i} + 0.4 \cdot \sum_{i=1}^{30} rank_{100}^{i} \tag{1.20}$$

В формуле (1.19) ef_d^k – ошибка оптимизации для k-ой оптимизируемой функции размерности d (вычисляется согласно формуле 1.16). В (1.20) $rank_d^k$ – ранг алгоритма для функции k размерности d. Наименьший ранг соответствует алгоритму, с наименьшей ошибкой.

Представим результаты тестирования генетического алгоритма однокритериальной безусловной оптимизации. Набор статистики осуществлялся при проведении 10 серий по 100 независимых запусков. Для задач различных размерностей выделялись вычислительные ресурсы, согласно таблице 1.2.

Таблица 1.2 – Вычислительные ресурсы ГА

Размерность задачи	10	30	50	100
Размер популяции	250	375	500	625

Число поколений	400	800	1000	1600

Данные значения были получены как некоторое субоптимальное значение с точки зрения критерия (1.16) для большинства из тестовых функций.

Иллюстрация вида зависимости ошибки оптимизации (1.16) от конфигурации алгоритма (*Configuration*) для десятимерных функций Растригина и Розенброка представлена на рисунке 1.11.

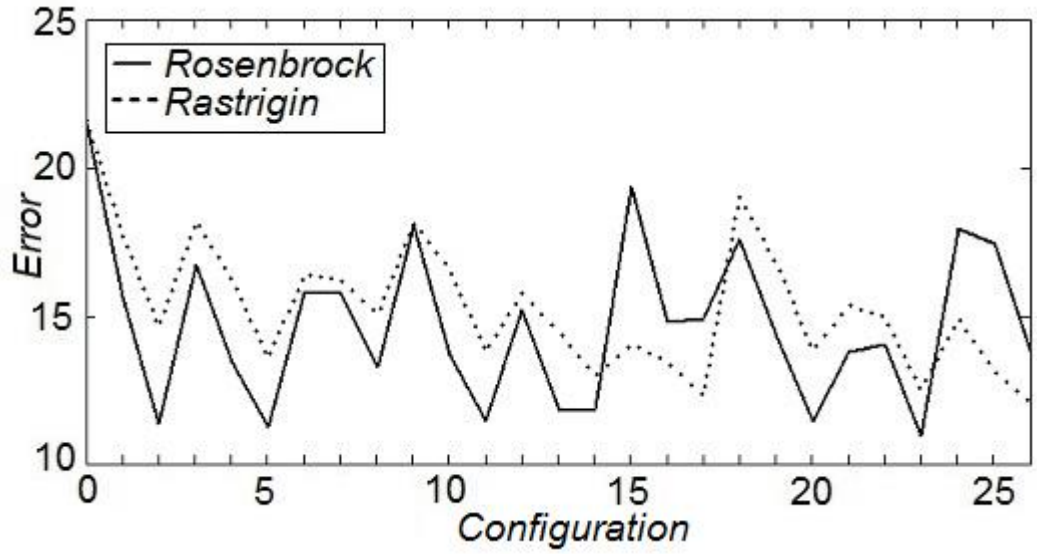

Рисунок 1.11 – Зависимость ошибки оптимизации от конфигурации ГА для функций Растригина и Розенброка

Рисунок 1.11 показывает существенную зависимость эффективности ГА от конфигурации для двух представленных функций. Аналогичные зависимости присутствуют и на остальных функциях. На функции Растригина оптимальной является конфигурация под номером 17, а для функции Розенброка – 23. Отличие конфигураций под номерами 23 и 17 на функции Растригина является статистически значимым по критерию Вилкоксона [109]. Таблица оптимальных номеров конфигураций ГА представлена в приложении А.

Опишем способ расшифровки конфигурации. Введем 3 величины $SelType = round(k/9)$, $RecType = round(k/3)\%3$, $MutType = k\%3$. Где ($k\%d$) – операция вычисления остатка от деления числа k на d, а $round(k)$ – функция округления до целого числа в меньшую сторону. После вычисления вспомогательных величин воспользуемся таблицей 1.3 и согласно нумерации определим используемые операторы. Пусть номер конфигурации k=17, тогда $SelType$=1, $RecType$=2, $MutType$=2. Соответственно алгоритм функционировал с ранговой селекцией равномерным скрещиванием и сильной мутацией.

Таблица 1.3 – Основные операторы ГА

Тип селекции (*SelType*)	Тип скрещивания (*RecType*)	Тип мутации (*MutType*)
0. Пропорциональная 1. Ранговая 2. Турнирная	0. Одноточечное 1. Двухточечное 2. Равномерное	0. Слабая 1. Средняя 2. Сильная

Проведенные вычислительные эксперименты показывают существенную зависимость эффективности ГА от его конфигурации, что обосновывает необходимость разработки и применения методов автоматического выбора конфигурации алгоритма под решаемую задачу в ходе ее решения.

ГА содержит ряд вещественных и целочисленных параметров: размер турнира, вероятность скрещивания, размер популяции, вероятность мутации. Среди перечисленных параметров для исследования наибольший интерес представляют вероятность мутации и размер популяции, т.к. их настройка позволяет добиваться наибольшего приращения эффективности [94].

На рисунке 1.12 проиллюстрирована зависимость эффективности ГА от размера популяции.

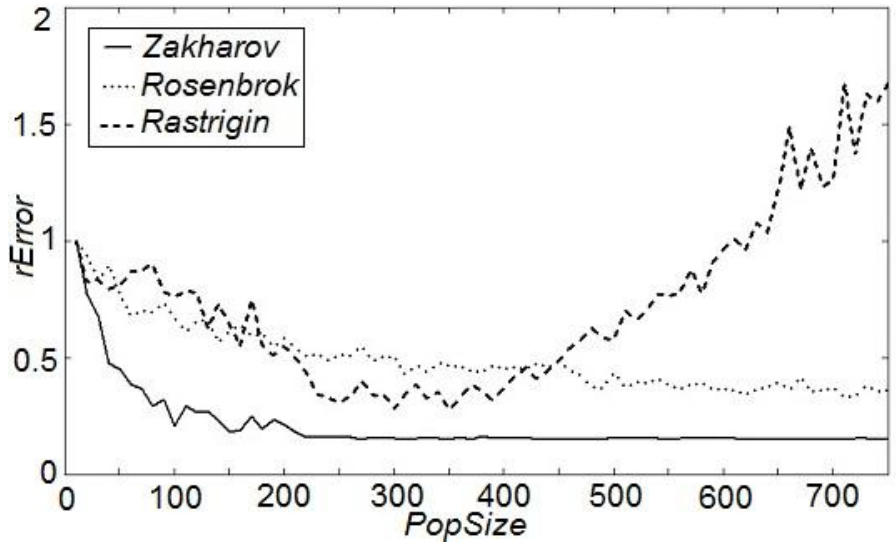

Рисунок 1.12 – Зависимость относительной ошибки оптимизации от размера популяции в ГА

Относительная ошибка *rError*, представленная на рисунке 1.12 рассчитывалась следующим образом:

$$rError(PopSize) = \frac{Error_{PopSize}}{Error_{ini}} \tag{1.21}$$

В формуле выше *PopSize* – текущий размер популяции, $Error_{PopSize}$ – ошибка, полученная алгоритмом с популяцией размера *PopSize*. $Error_{ini}$ – ошибка на популяции минимально заданного размера. Введение такого критерия обусловлено различными порядками ошибок для тестовых функций. Рисунок 1.12 иллюстрирует различные случаи поведения ГА на тестовых функциях, что обосновывает необходимость разработки и реализации методов автоматической настройки размера популяции.

На рисунке 1.13 представлены зависимости ошибки оптимизации (1.16) от вероятности мутации (*Mutation*). Отметим, что минимум ошибки достигается при значениях 0.035 для функции Розенброка и 0.0175 для функции Растригина, т.е. для разных целевых функции оптимальные величины мутации существенно отличаются. Тестирование стандартного ГА при установке высокого уровня мутации в задачах *CEC'2017* размерности 10 полагает вероятность мутации равной 0.0166, что является довольно близким значением к оптимальному параметру для функции Розенброка, однако для некоторых функций проигрыш

составлял до 25 %. Рисунок 1.13 подчеркивает необходимость более тонкой настройки такого параметра как вероятность мутации гена в индивиде.

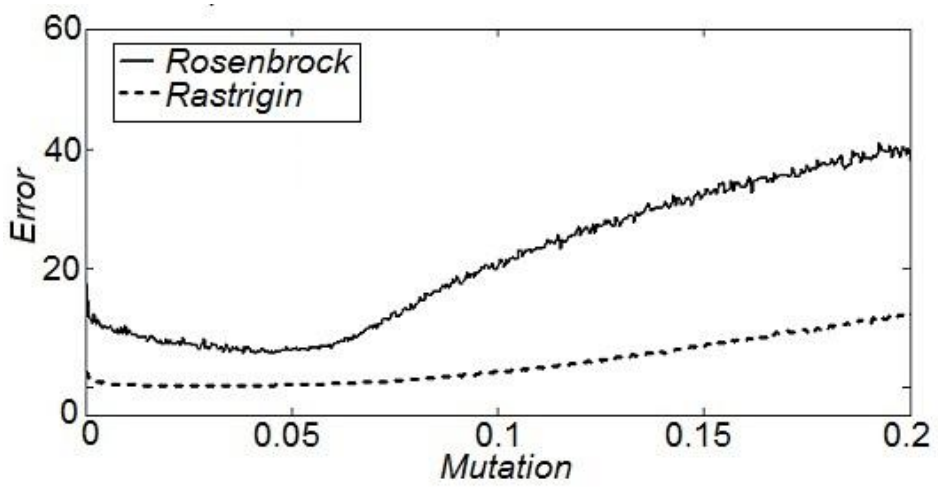

Рисунок 1.13 – Зависимость ошибки оптимизации от вероятности мутации в ГА

Алгоритм генетического программирования показал зависимости, схожие по поведению с ГА (рисунки 1.11-1.13). Как и в случае тестирования ГА было проведено 10 серий по 100 независимых запусков. Критерий эффективности *MAE* определялся по формуле (1.17). Точки выборки генерировались случайно и равновероятно на всем интервале оптимизации *CEC'2017*, а значения функции подвергались нормировке. Объем выборки n вычислялся как $n = 1000 \cdot D$, где D – размерность задачи аппроксимации. Для вычисления (1.17) выполнялась процедура 10-частной кросс-валидации.

Представим зависимость ошибки алгоритма ГП для функций Шэффера и Растригина. Расшифровка конфигурации алгоритма ГП производится по аналогии с ГА, при этом величины $SelType = round(k/4)$, $RecType = round(k/2)\%2$, $MutType = k\%2$. Для определения используемых генетических операторов воспользуемся таблицей 1.4 и согласно нумерации определим используемые операторы в комбинации. Пусть номер конфигурации k=7, тогда *SelType*=1, *RecType*=1, *MutType*=1. Соответственно алгоритм функционировал с ранговой селекцией стандартным скрещиванием и мутацией с заменой ветви.

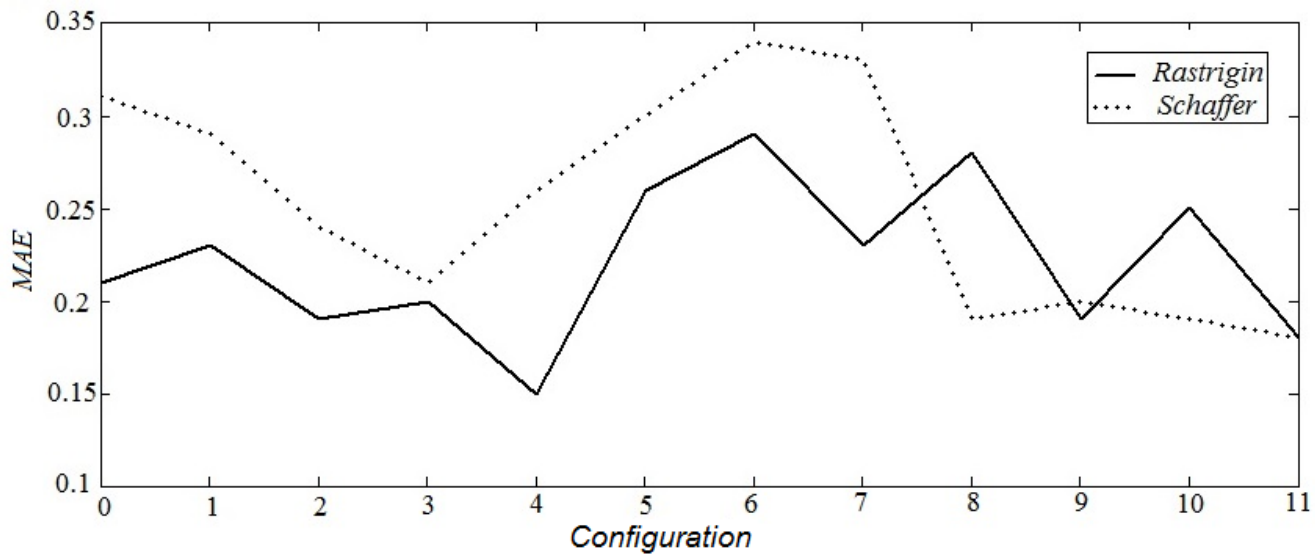

Рисунок 1.14 – Зависимость критерия *MAE* от параметров ГП

Таблица лучших комбинаций параметров ГП с точки зрения критерия (1.17) представлена в приложении Б.

Таблица 1.4 – Основные операторы ГП

Тип селекции (*SelType*)	Тип скрещивания (*RecType*)	Тип мутации (*MutType*)
0. Пропорциональная 1. Ранговая 2. Турнирная	0. Одноточечное 1. Стандартное	0. Одноточечная 1. С заменой ветви

Таблица 1.4 получена при вычислительных ресурсах, указанных в таблице 1.2. В тестировании использовалась вероятность мутации, равная 0.025. Как и в случае с ГА было показано различие оптимальных комбинаций генетических операторов в ГП для различных решаемых задач.

На рисунках 1.15 и 1.16 представлены зависимости критерия *MAE* от размера популяции и вероятности мутации. Для тестирования выбиралась наиболее эффективная комбинация генетических операторов, полученная полным перебором всех возможных вариантов на предыдущем этапе.

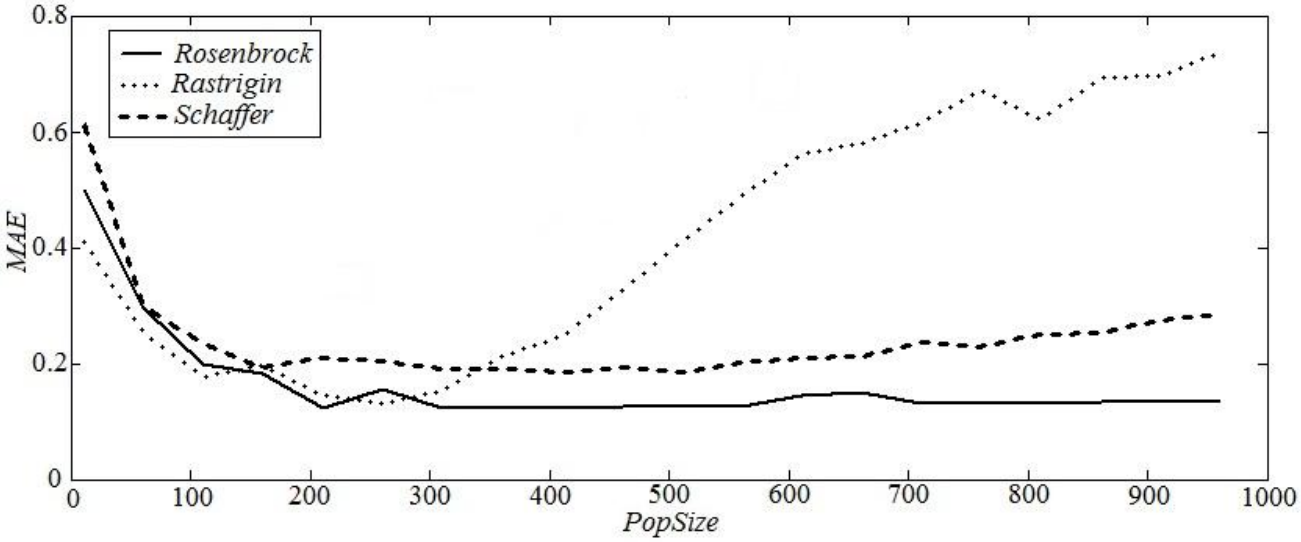

Рисунок 1.15 – Зависимость *MAE* от размера популяции ГП

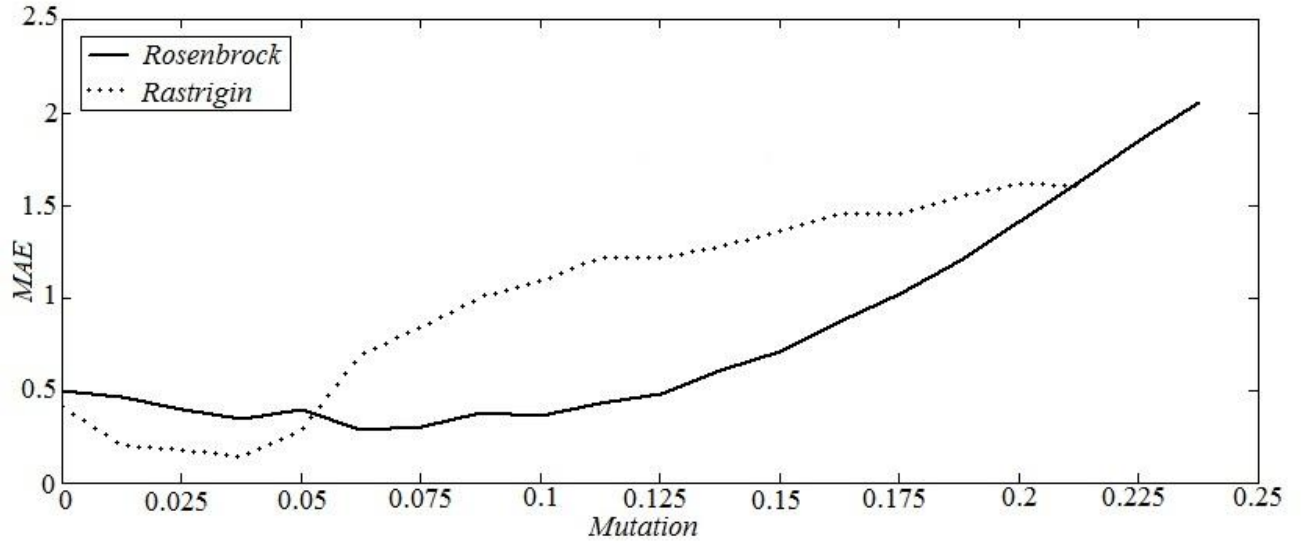

Рисунок 1.16 – Зависимость *MAE* от вероятности мутации индивида в ГП

Анализ представленных в (1.2.) генетических операторов, а также тестирование на репрезентативном множестве задач подтверждает гипотезу о существенной зависимости эффективности эволюционных алгоритмов от параметров, задаваемых пользователем. Численные эксперименты показывают, что эмпирическое определение оптимального набора параметров для всех алгоритмов и решаемых задач не имеет смысла, т.к. данный набор изменяется от задачи к задаче. Задача автоматического проектирования интеллектуальных технологий анализа данных (в том числе искусственных нейронных сетей) [90,

91] может быть сведена к оптимизационной со смешанными переменными (переменные вещественного, целочисленного и категориального типов), а ее решение требует привлечения большого количества вычислительных ресурсов. Игнорирование представленных особенностей ГА и ГП может привести к невозможности получения эффективного решения за адекватное время (или стоимость). Описывая особенности ЭА, необходимо упомянуть *NFL*-теорему, позволяющую определить путь дальнейшего решения проблемы автоматической настройки параметров.

1.4 *NFL*-теорема

Поиск оптимального набора параметров эволюционного алгоритма, обеспечивающего высокую эффективность, исследовался достаточно давно. В работах Вольперта и Макреди [110, 112] формулируется так называемая *NFL*-теорема (*No Free Lunch*). Данная теорема с момента публикации вызвала оживленную дискуссию в научном сообществе, т.к. ее следствием является невозможность подбора параметров генетического алгоритма, обеспечивающих лучшие результаты независимо от решаемой задачи. Теорема формулируется следующим образом:

Пусть $P(\mathrm{d}_m \mid \mathrm{F}, \mathrm{k}, \mathrm{alg})$ – условная вероятность получения частного решения d_m после *k* итераций алгоритма *alg* целевой функции *F*. Для любой пары алгоритмов alg_1 и alg_2 имеет место равенство:

$$\sum_F P(\mathrm{d}_m \mid \mathrm{F}, \mathrm{k}, \mathrm{alg}_1) = \sum_F P(\mathrm{d}_m \mid \mathrm{F}, \mathrm{k}, \mathrm{alg}_2) \qquad (1.22)$$

Формула (1.22) означает, что независимо от используемого алгоритма сумма условных вероятностей посещения решения d_m в пространстве решений одинакова на множестве *всех* целевых функций. Данная теорема означает, что не существует универсального алгоритма способного решать любые задачи (как эволюционного, так и любого другого). Поиски лучших параметров для

эволюционных алгоритмов не имеют смысла, так как их эффективность проверяется лишь на наборе тестовых задач и их репрезентативный набор не дает гарантии, что алгоритм покажет лучшие результаты на реальной задаче неизвестной структуры.

Согласно утверждению Вольперта и Макреди [111], «бесплатные обеды» возможны при использовании различных мета-алгоритмов. Мета-алгоритмы позволяют настраивать параметры и конфигурацию под решаемую задачу в ходе ее решения, используя информацию о предыдущих шагах поиска. Примером таких алгоритмов являются: коэволюционные алгоритмы, самоконфигурирующиеся, самонастраивающиеся (например, [3]) и другие модификации. Их описание будет представлено в следующей главе.

Эффективность методов настройки параметров и конфигурации также проверяется на ограниченном наборе тестовых функций, что в свою очередь согласно *NFL*-теореме не дает гарантии получения эффективных параметров. Однако использование мета-алгоритмов адаптации показывает их практическую полезность [1, 2], что позволяет надеяться на их успешное применение в дальнейших работах.

ВЫВОДЫ

В главе были представлены и программно реализованы два эволюционных алгоритма – алгоритм генетического программирования и генетический алгоритм. Данные алгоритмы были протестированы на актуальном репрезентативном множестве тестовых задач *CEC'2017*.

Реализация и тестирование алгоритмов, указанных в данной главе, преследовали несколько целей. Во-первых, была практически обоснована необходимость настройки размера популяции и уровня мутации в эволюционных алгоритмах. Во-вторых, показана зависимость эффективности алгоритмов от их конфигурации. В-третьих, была собрана статистика базовых версий ГА и ГП,

необходимая для разработки, реализации и сравнения эффективности методов самоконфигурирования и адаптации параметров алгоритма.

Достоинствами описанных алгоритмов являются: отсутствие требований к информации о целевой функции, устойчивость к разрывам и шумам, способность выходить из локальных оптимумов, низкая цена разработки и возможность интеграции с другими методами моделирования и оптимизации.

Среди недостатков можно выделить слабую теоретическую базу, отсутствие гарантий в получении приемлемого решения за конечное время, необходимость настройки параметров алгоритмов.

Автоматическая настройка параметров и конфигурации алгоритма в процессе функционирования является актуальной научно-технической задачей. Решению данной проблемы посвящена глава 2.

ГЛАВА 2. РАЗРАБОТКА АДАПТИВНЫХ ЭВОЛЮЦИОННЫХ АЛГОРИТМОВ МОДЕЛИРОВАНИЯ И ОПТИМИЗАЦИИ

2.1 Описание подходов к адаптации эволюционных алгоритмов

В главе 1 были представлены такие разновидности эволюционных алгоритмов как генетический алгоритм и алгоритм генетического программирования. Генетические операторы, представленные в главе, являются стандартными [128], однако существуют и другие их разновидности [32]. Решая задачу при помощи ЭА, исследователь в реальности сталкивается с двумя проблемами: необходимо найти оптимальное решение задачи, а также необходимо найти комбинацию параметров ЭА, которая позволит найти такое решение при минимальных затратах. Вторая проблема является, как правило, более сложной, чем первая. Этим обусловлено огромное количество статей, обсуждающих методы настройки ЭА.

В работах Айбена и др. [66, 28] приводится подробный обзор стратегий настройки параметров. Согласно данным работам, параметры ЭА можно разделить на 2 группы: локальные и глобальные. Если эффект от параметра отслеживается на уровне индивида, то такой параметр можно назвать локальным. Примером локальных параметров являются вероятности мутации и скрещивания. Под глобальными параметрами понимаются те из параметров ЭА, которые воздействуют сразу на всю популяцию: размер популяции, конфигурация алгоритма и др. Отмечается, что наиболее важными для настройки являются именно глобальные параметры.

В [66] приводится классификация методов настройки и отмечается различие в терминологии у исследователей в области ЭА. Выделяют 3 способа настройки параметров ЭА;

1. Детерминированный;
2. Адаптивный;

3. Самоадаптивный.

Детерминированные методы выполняют изменение параметров ЭА по заранее определенным правилам. Классическим примером детерминированного механизма настройки является расчет уровня мутации в ГА в зависимости от количества генов индивида.

Адаптивные методы используют принцип обратной связи. В процессе работы ЭА возможно вычисление большого количества вспомогательных величин (разброс точек в пространстве поиска, средняя пригодность, время стагнации в точке локального оптимума и др.). При помощи данных величин на основании множества эвристических методов выполняется подстройка управляющих параметров ЭА.

Третьим и наиболее сложным методом настройки является самоадаптивный алгоритм. В таких алгоритмах управляющие параметры кодируются внутри индивида и настраиваются при помощи эволюционного процесса.

В данной работе имеет место отличие в терминологии, используемой в [32]. Оно заключается в выделение из общего числа параметров ЭА тех, которые отвечают за конфигурацию алгоритма, т.е. тип селекции, тип скрещивания, тип мутации. Метод автоматической настройки конфигурации ЭА, используемый в данной диссертационной работе будем называть самоконфигурированием.

Методы вычисления размеров популяции. Появление такого параметра алгоритма как «размер популяции» ставит перед исследователем проблему выбора его значения. Ряд исследователей выбирали «стандартные» размеры, полагая их равными 50-100 индивидам. Другие вкладывали в расчет размера популяции понятие сложности задачи. При этом она оценивалась лишь экспертно и зачастую не отражала истинной картины. В особенности это было заметно при решении прикладных задач. Третьи проводили серию экспериментов, где оценивали оптимальное значение параметра. Однако довольно быстро была выяснена низкая эффективность таких подходов, что породило собой волну исследований в данной области, результатом которой стало множество теоретических работ [39, 40, 67] и эмпирических методов [83, 113] настройки

размеров популяции. Рассмотрим на примере генетического алгоритма некоторые из них.

Наиболее простыми являются методы, определяющие размер популяции (*PopSize*) через зависимости от номера текущего поколения: $PopSize(t) = f(t,\Theta) \cdot MaxPopSize$. Здесь *MaxPopSize* – максимально возможный размер популяции, а $f(t,\Theta)$ – некоторая зависимость, показывающая какой процент от максимального значения будет использован в момент времени t. Θ – вектор управляющих параметров. В случае гармонического закона изменения *f(t)* его компонентами могут выступать амплитуда и частота. Данные методы являются тривиальными, и ожидать от них большого прироста эффективности не стоит. Такие алгоритмы способны нивелировать слишком неудачные оценки исследователей относительно сложности решаемой задачи, но в свою очередь требуют выбора управляющих параметров. Стоит отметить, что некоторые из линейных модификаций успешно используются в алгоритме дифференциальной эволюции [102].

GAVaPS. Метод *GAVaPS* (*Genetic Algorithm with Varying Population Size*) был впервые предложен в [6]. Он основан на понятиях возраста и времени жизни индивида. При создании индивид имеет возраст равный нулю. При каждом следующем поколении, возраст индивида увеличивается на единицу. Если возраст индивида достигает некоторого порогового значения *LT* (времени жизни), он исключается из популяции. Генерация нового поколения в методе *GAVaPS* отличается от стандартной схемы генетического алгоритма [128]. Здесь каждый индивид имеет равную вероятность выбора стать потомком. Селективное давление в алгоритме осуществляется за счет «времени жизни», которое определяется исходя их пригодности индивида. Основная идея – чем выше пригодность индивида, тем больше время его жизни. Несложно заметить, что эффективность такого алгоритма будет зависеть от стратегии определения времени жизни. Существуют линейные, пропорциональные, и би-линейные стратегии [67]. Все из стратегий используют понятия *MinLT* и *MaxLT* –

максимальное и минимальное значение времени жизни. Как и в механизме селекции, пропорциональные стратегии представляют собой «рулетку». Линейные стратегии основываются на лучшем из найденных значений в популяции решений. Би-линейная стратегия является компромиссом и может быть представлена в виде следующей формулы:

$$LT(i) = \begin{cases} MinLT + \eta \cdot \dfrac{fit(i) - MinFit}{AvgFit - MinFit} & если \quad AvgFit \geq fit(i) \\ \dfrac{MaxLT - MinLT}{2} + \eta \cdot \dfrac{fit(i) - AvgFit}{MaxFit - AvgFit} & если \quad AvgFit < fit(i) \end{cases} \tag{2.1}$$

где $\eta = \dfrac{(MaxLT - MinLT)}{2}$, *MinFit*, *MaxFit*, *AvgFit* – минимальная, максимальная и средняя пригодности соответственно, а *fit*(*i*) – пригодность *i*-го индивида. Для начала работы алгоритма требуется выбор начального значения размера популяции, однако, как утверждают авторы, *GAVaPS* является робастным по данному параметру. Стоит отметить, что метод *GAVaPS* является очень чувствительным к коэффициенту скрещивания ρ, который определяет, какая доля индивидов может породить потомков на текущем поколении.

В дальнейшем алгоритм был доработан Бэком и Айбеном [11], получив название *APGA*. Основное отличие заключается в способе определения *LT*(*i*).

SAGA. Данный алгоритм был предложен в работе [48]. Он основан на одновременном функционировании трех независимых генетических алгоритмов. Размеры популяции в начале работы выбираются следующим образом: $PopSize_1 < PopSize_2 < PopSize_3$. Алгоритмы независимо функционируют в течение определенного периода времени *t* (параметр устанавливался пользователем). После чего происходит пересчет размеров популяции. Введем некоторую функцию $B(P_i)$, которая возвращает значение пригодности лучшего из индивидов *i*–ой популяции. Алгоритм вычисления размеров популяции опишем в виде таблицы 2.1:

Таблица 2.1 – Алгоритм вычисление размера популяции.

Условие	Событие
$B(P_1) < B(P_2) < B(P_3)$	$PopSize_1 = PopSize_2$ $PopSize_2 = PopSize_3$ $PopSize_3 = 2 \cdot PopSize_3$
$B(P_3) < B(P_2) < B(P_1)$	$PopSize_3 = PopSize_2$ $PopSize_2 = PopSize_1$ $PopSize_1 = 0.5 \cdot PopSize_1$
$B(P_2) < B(P_1) < B(P_3)$	$PopSize_1 = \frac{PopSize_1 + PopSize_2}{2}$
$B(P_1) < B(P_3) < B(P_2)$	
$B(P_2) < B(P_3) < B(P_1)$	$PopSize_3 = \frac{PopSize_2 + PopSize_3}{2}$
$B(P_3) < B(P_1) < B(P_2)$	

Для того чтобы сдержать избыточный рост размеров популяции алгоритмов, вводилось ограничение: $PopSize_2 \in [10,1000]$. Кроме того, размеры всех трех популяций отличались друг от друга минимум на 20 индивидов. В качестве недостатков алгоритма можно выделить необходимость настройки параметра t, а также указанного ограничения $PopSize_2$.

PRoFIGA. В отличие от стандартного генетического алгоритма [128], в *PRoFIGA* [105, 31] используется адаптивный размер популяции. Основная идея заключается в том, что популяция индивидов постепенно увеличивается в двух случаях. Во-первых, в случае роста пригодности лучшего из найденных индивидов. Во-вторых, в случае стагнации алгоритма в течение «долгого времени». В противном случае популяция уменьшается на небольшую величину (1-5 % от размеров популяции). *PRoFIGA* использует популяции большой размерности для исследования поискового пространства и малой для отыскания локальных оптимумов. Коэффициент роста X расчитывается следующим образом:

$$X = iFactor \cdot (MaxFES - FES) \cdot \frac{fitness_{new} - fitness_{old}}{fitness_0} \qquad (2.2)$$

В данной формуле *iFactor* – коэффициент увеличения ($iFactor \in [0,1]$), *FES* – число вычислений целевой функции алгоритмом на текущем поколении, *MaxFES*

– максимально возможное число вычислений целевой функции, $fitness_0$ – пригодность на нулевом поколении, $fitness_{new}$ и $fitness_{old}$ – пригодности на текущем и предыдущем поколении соответственно. Заметим, что данный алгоритм вводит ряд параметров, требующих настройки: начальный, максимальный и минимальный размеры популяции, коэффициенты увеличения и уменьшения популяции и, наконец, понятие «долгое время без улучшения». Авторы приводят некоторые эмпирические значения для этих коэффициентов, однако их настройка для конкретной задачи выглядит для конечного пользователя малоперспективной.

Методы вычисления вероятности мутации в эволюционных алгоритмах. Проблема настройки вероятности мутации в генетических и эволюционных алгоритмах не является тривиальной. С одной стороны, вероятность мутации – один из основных операторов поиска в пространстве решений и, следовательно, такой оператор должен довольно интенсивно использоваться. С другой стороны, большое значение вероятности мутации ослабляет селективное давление внутри популяции. Это нарушает основной принцип работы эволюционных алгоритмов – принцип наследования родительских генов (или цепочек таких генов).

В статических детерминированных подходах вероятность мутации определяется заранее и не изменяется в ходе решения задачи. Например, в стандартном генетическом алгоритме [128] существуют 3 уровня мутации (таблица 1.1). Для каждого из этих уровней вероятность P определяется по заранее установленным формулам, например: $P_{слаб.} = \frac{1}{3 \cdot n}$, $P_{сред.} = \frac{1}{n}$, $P_{сильн.} = \frac{3}{n}$, и не изменяется на протяжении всей работы алгоритма (n – число генов в индивиде-решении). Такой подход удобен с точки зрения простоты, однако не способен отследить моменты стагнации или сильного разрежения получаемых решений, что приводит к падению эффективности. Существуют и другие

эмпирические формулы, среди которых: $P = \frac{1}{m}$, $P = \frac{D}{m}$, а также различные нелинейные вариации. В данных формулах m – размер популяции, D – размерность решаемой задачи.

При использовании динамических детерминированных подходов вероятность мутации высчитывается через некоторые величины (как правило, это номер текущего поколения), изменяющиеся в ходе работы алгоритма. Примером такого подхода может служить [10], где вероятность мутации определяется как $P(t) = P_0 \quad e^{\frac{1}{t}}$. Здесь $P_0 \in [0,0.1]$ – начальная вероятность мутации, t – управляющий параметр (например, номер текущего поколения). Такой подход позволяет алгоритму на поздних этапах работы активнее использовать поисковое пространство с целью улучшения решения. Недостатком является отсутствие «обратной связи». Алгоритм, получивший улучшение целевой функции на поздних этапах, будет иметь высокий уровень мутации, что может помешать улучшению решения оператором скрещивания. Обратная ситуация, когда вероятность мутации убывает со временем, тоже имеет недостаток – преждевременная стагнация, т.к. оператор повышения разнообразия (мутация) на поздних этапах работы алгоритма не будет давать положительного эффекта.

Адаптивные алгоритмы позволяют более точно настраивать вероятность мутации. Они оперируют различными показателями, рассчитываемыми в ходе эволюционного поиска [62, 9], и на их основании по принципу обратной связи управляют уровнем мутации. В [10] была предложена идея поместить вероятность мутации в хромосому индивида, модифицировав подход до самоадаптивного. Получение нового индивида при этом проводилось в 2 этапа. На первом производилось скрещивание только тех генов, которые были ответственны за вероятность мутации. Результат скрещивания декодировался в вероятность мутации и полученная часть мутировала на основании новой вероятности. После мутации производился второй этап, на котором скрещиванию подвергался весь индивид, а для мутации вероятность рассчитывалась уже по обновленной части

генома. В дальнейших работах алгоритм был модифицирован и в [12] вероятность мутации кодировалась уже в виде вещественного параметра. В таком алгоритме для него применялись специфические методы скрещивания и мутации.

Некоторые из подходов адаптации предлагают использовать мета-алгоритмы для настройки параметров [19]. Данные алгоритмы являются весьма трудоемкими (с точки зрения количества вычислений значений целевой функции) и редко используются на практике.

2.2 Модификации эволюционных алгоритмов, используемые и предлагаемые в работе

На сегодняшний день различными учеными разработаны сотни методов, позволяющих выполнять автоматическую подстройку параметров ЭА. Такое количество методов неизбежно приводит к проблеме выбора наиболее эффективного из них. Разумеется, исследователи пытались создать некоторые универсальные алгоритмы, способные независимо от целевой функции подстраивать параметры ЭА обеспечивая максимальную эффективность. Учитывая *NFL*-теорему можно заключить, что разработка такого метода теоретически невозможна.

Целью данной главы является разработка комбинации модификаций ЭА, которая позволит впоследствии более эффективно решать задачу моделирования искусственных нейронных сетей и их коллективов. Приводятся лишь те модификации и разработки, которые позволили оказать улучшающее воздействие на реализуемые в дальнейшем алгоритмы.

Алгоритм самоконфигурирования в ЭА. Эффективность работы эволюционного алгоритма значительно зависит от выбора его конфигурации. Автоматическая процедура настройки данного параметра позволяет повысить эффективность ЭА, а также избавиться от влияния квалификации пользователя на ход эволюционного процесса. Предлагается использование метода самоконфигурирования, основанного на динамической адаптации вероятностей

выбора генетических операторов [86, 127]. Данный метод неоднократно был использован при теоретических исследованиях [24, 99] и при решении реальных практических задач [126, 116], где доказал свою эффективность. В указанном методе вероятность выбора оператора напрямую зависит от его успешности на предыдущих итерациях. В алгоритме ГА выделяется 3 группы операторов: селекции, скрещивания, мутации. Каждая группа представляет собой следующий набор:

Таблица 2.2 – Разбиение операторов ГА по группам и типам

Группы операторов	Селекция	Скрещивание	Мутация
Типы операторов	1. Пропорц. 2. Ранговая 3. Турнирная	1. Одноточечное 2. Двухточечное 3. Равномерное	1. Слабая 2. Средняя 3. Сильная

Алгоритм ГП представлен следующими наборами операторов:

Таблица 2.3 – Разбиение операторов ГП по группам и типам

Группы операторов	Селекция	Скрещивание	Интенсивность мутации	Тип мутации
Типы операторов	1.Пропорц. 2.Ранговая 3.Турнирная	1.Стандартное 2.Одноточечное 3.Равномерное	1.Слабая 2.Средняя. 3.Сильная	1.Одноточечное 2.Замена ветви

Алгоритм самоконфигурирования базируется на следующей идее: оператору, имеющему наибольшую эффективность на текущей популяции, увеличивают вероятность выбора на следующей итерации, в то время как остальным – уменьшают. При этом вероятности не могут стать ниже установленного порогового значения. В противном случае потенциально эффективным операторам на ранних этапах вероятность использования может быть уменьшена до нуля, что не даст возможности функционирования в дальнейшем, когда они были бы эффективнее сохранившихся.

Общий алгоритм настройки вероятностей выбора операторов внутри группы выглядит следующим образом:

1. В начале работы генетического алгоритма вероятности выбора всех операторов внутри групп одинаковы и равны: $P = \frac{1}{k}$, где k – число операторов внутри рассматриваемой группы;

2. На каждой итерации алгоритма происходит оценка качества работы каждого из операторов по средней пригодности потомков, полученных с его помощью: $fitness = \frac{f}{n}$. Здесь f – суммарная пригодность потомков, полученных рассматриваемым оператором, n – количество таких потомков;

3. Перераспределение вероятностей выбора операторов происходит следующим образом: наиболее эффективному из операторов увеличивают вероятность выбора на величину $((k\text{-}1)\cdot C)/(k\cdot N)$, в то время как остальным уменьшают вероятность на $C/(k\cdot N)$. Здесь k – число операторов внутри группы, C – некоторый коэффициент, N – число поколений алгоритма. При перераспределении сумма вероятностей всегда равна единице и вероятность выбора каждого из операторов не может быть ниже заданного порога.

4. Далее для каждого будущего потомка операторы, при помощи которых он будет порожден, выбираются случайным образом в соответствии с полученными распределениями вероятностей. Пока не выполнен критерий останова переходим к шагу 2.

Одноточечное скрещивание в ГП. Алгоритм ГП – эффективный метод моделирования различных структур. Одной из проблем, возникающих при его использовании, является чрезмерное разрастание деревьев. Данный эффект частично компенсируется наложением штрафа (1.11), однако настройка коэффициента штрафа α является нетривиальной. Следствием неверной настройки данного коэффициента является сильное падение точности ГП.

Существует модификация одноточечного скрещивания в алгоритме ГП, преобразующая его в равномерное [76, 77]. Такой подход позволяет придать

гибкость алгоритму, генерировать более разнообразные структуры и, как следствие, увеличить эффективность. Равномерное скрещивание выращивает деревья сложностью не больше, чем у родительских индивидов, следовательно, применение такой модификации на определенных этапах работы ЭА (определяются алгоритмом самоконфигурирования) позволит контролировать размеры (или сложность) получаемых решений.

Опишем данный алгоритм. Процедура равномерного скрещивания в ГП начинается также как и в одноточечном скрещивании – у родительских деревьев выбирается общая часть (совокупность связанных между собой вершин с одинаковым число исходящих связей). На рисунке ниже ребра такой общей части выделены жирными линиями.

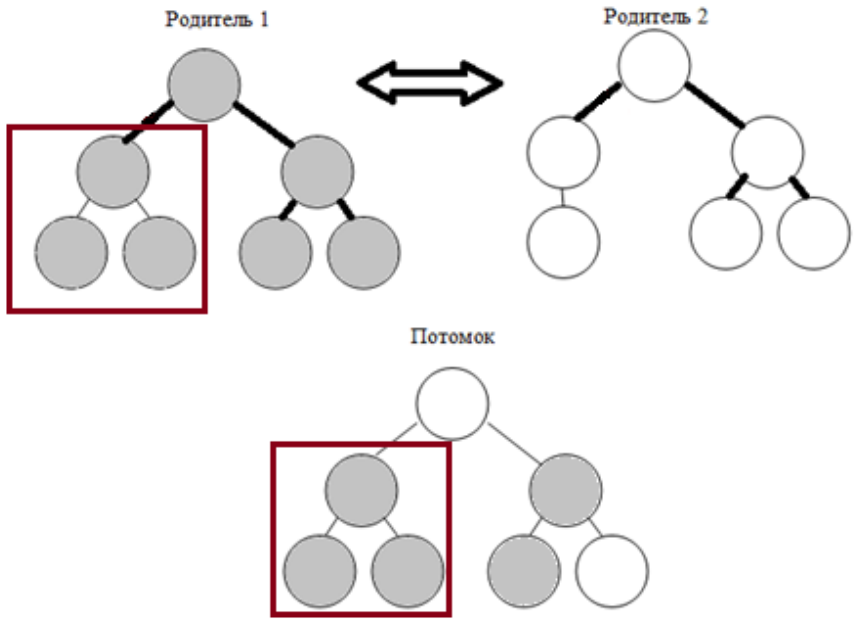

Рисунок 2.1 – Пример равномерного скрещивания

Внутри этой общей части деревья с вероятностью 0.5 обмениваются своими вершинами. Поддеревья, лежащие ниже общей части передаются потомку вместе со своим верхним узлом (также с вероятностью 0.5). На рисунке 2.1 они выделены прямоугольником. Полезной модификацией будет также возможность скрещивания и таких частей [88, 125]. Поддеревья вершин с различным числом исходящих связей будут конкурировать за право передачи своей вершины потомку:

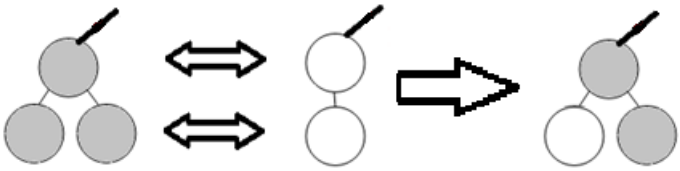

Рисунок 2.2 - Пример равномерного скрещивания поддеревьев

На рисунке 2.2 представлен пример для скрещивания поддеревьев. В первую очередь случайно и равновероятно выбирается одна из вершин, которая будет передана потомку. Затем считается число вершин на следующем уровне дерева. Для данного примера их число равно двум. Затем создается пул вершин, которые могут быть использованы для дальнейшей процедуры скрещивания (в примере их число равно трем). Каждая из вершин следующего уровня заполняется случайно и равновероятно из созданного пула. Процедура повторяется до тех пор, пока число вершин заполнения следующего уровня не будет равно нулю.

Алгоритм селекции обучающих примеров. Решение задачи интеллектуального анализа данных подразумевает существование некоторой выборки, из которой необходимо извлечь нетривиальные, ранее неизвестные закономерности о структуре данных. Задачи анализа данных сегодня оперируют сотнями тысяч и миллионами измерений большой размерности. В таких условиях становится необходимым не только снижение размерности выборки, но и отбор наиболее информативных обучающих примеров (или измерений) [79, 74]. Снижение объемов данных неизбежно приводит к потере информации, однако не всегда такая потеря негативно сказывается на качестве получаемых решений. Такой эффект становится возможным при исключении повторяющихся, близких и зашумленных измерений.

Используемый в работе подход селекции обучающих примеров был разработан в [96] и представляет собой итерационную процедуру оценки вероятности выбора измерения в обучающую выборку на основании ошибки проектируемой модели. На практике было показано, что такой алгоритм позволяет не только снизить вычислительную сложность алгоритма, но и повысить качество получаемых моделей [59]. Представленный метод не требует

предобработки данных, вычисления мер расстояния между объектами, не основан на методах классификации и кластеризации, что, безусловно, является его преимуществом.

Используемый метод селекции обучающих примеров состоит из следующих этапов:

1. Устанавливаются значения $U_i = 1 \forall i = \overline{1,n}$. n – количество измерений в выборке задачи. Задается параметр k, определяющий количество измерений, отбираемых в обучающую выборку.

2. Рассчитываются вероятности выбора измерений в обучающую выборку

$$p_i = \frac{1/U_i}{\sum_{i=1}^{n} 1/U_i} \tag{2.3}$$

Далее с их использованием отбирается k измерений. На сформированной подвыборке производится обучение и/или формирование модели.

3. Пересчитываются величины U_i. Если измерение под номером i было классифицировано верно, то $U_i = U_i + 1$. Иначе $U_i = 1$. Переходим к шагу 2 пока не выполнен критерий останова алгоритма.

Алгоритм адаптации размера популяции. Метод адаптации размера популяции, разработанный в рамках работы, имеет ряд отличий по сравнению с описанными в пункте (2.1) подходами. Во-первых, в качестве базовой модели разработки использовался самоконфигурируемый эволюционный алгоритм. Во-вторых, вводятся параметры, настройка которых является максимально упрощенной. В-третьих, оценка размера популяции происходит в рамках одного алгоритма. Данный алгоритм реализует адаптацию на основе истории «успешности» [103] подпопуляций определенного размера. Опишем алгоритм поэтапно:

1. В начале работы задаются параметры: *MaxPopSize* – максимально возможный размер популяции, k – число подпопуляций в алгоритме, *tOpt* – период работы алгоритма без изменения размера популяции, m и d – параметры, отвечающие за изменение размера популяции (коэффициенты сдвига и масштаба

в равномерном распределении). Вводятся вектора *S* и *P*. Компоненты вектора *S* приравниваются к нулю, а *P* задаются случайно и равновероятно на интервале [0.05,1]. Параметр времени *t*=0;

2. Проверяем критерий останова – достижение максимального числа вычислений целевой функции. Если *MaxFES* достигнуто, алгоритм завершает работу.

3. Положим $t = t + 1$. Если *t* кратно *tOpt* переходим к шагу 4, а в противном случае к шагу 5;

4. Находим $best = \arg\max_i(S_i)$. Случайным образом выбираем *j*-ю компоненту вектора *P*. Вычисляем $P_j = Rav(m,d) + P_{best}$. Здесь $Rav(m,d)$ - функция равномерного распределения с коэффициентом сдвига *m* и масштаба *d*. $P_i \in [0.05,1] \forall\, i = \overline{1,k}$. Отметим, что компоненты вектора *P*, отличные от *j*, не претерпевают никаких изменений. Вектор *S* в конце шага обнуляется;

5. Определим количество потомков, порождаемых в поколении *t:* $subPopSize = MaxPopSize \cdot \max_i(P_i)$. Для каждого потомка случайно выбирается компонента вектора P_j. Из всех индивидов случайным образом формируется подпопуляция размера $MaxPopSize \cdot P_j$. Далее к новой подпопуляции последовательно применяются стандартные операторы селекции, скрещивания и мутации [128] с учетом самоконфигурирования [86]. Если полученный потомок по пригодности превосходит родителей, увеличивается показатель «успешности» подпопуляции: $S_j = S_j + 1$. Как только было порождено *subPopSize* потомков, переходим к шагу 2.

Алгоритм расчета вероятности мутации эволюционного алгоритма.

Вероятность мутации в разработанном ЭА рассчитывается по следующей формуле:

$$P_j = P_j^o + \alpha \cdot \frac{Med(fitness_i)}{n} \qquad (2.4)$$

где P_j – вероятность мутации для j-го индивида; P_j^0 – вероятность мутации, которая была определена индивиду при помощи алгоритма самоконфигурирования; α – параметр, задаваемый пользователем; n – число генов в индивиде; $fitness_i$ – нормированный массив пригодностей на i-м поколении; $Med(fitness)$ – функция определения значения медианы массива $fitness$. Заметим, что при методе расчета (2.5) вероятность мутации может оказаться больше единицы (при малом числе n и большом значении параметра α, однако на практике такой случай встречается крайне редко из-за больших размерностей задач). При $P_j \geq 0.5$ устанавливается $P_j = 0.5$. Эффективность указанного способа оценки вероятности мутация была показана в [129].

2.3 Исследование эффективности адаптивных эволюционных алгоритмов на задачах восстановления символьной регрессии и оптимизации

Эффективность разработанных подходов адаптации эволюционных алгоритмов и их комбинированного использования с другими модификациями проверялась при условиях, описанных в главе 1. Для тестирования ГА использовались формулы (1.16) и (1.18), для алгоритма ГП использовался критерий (1.17), вычисленный по тестовой выборке. Для компактного изложения результатов будем представлять результаты на примере двух-трех широко известных функций конкурса *CEC'2017*. Общая тенденция при тестировании на других функциях сохранялась.

В первую очередь покажем полезность самоконфигурирования при использовании ГА на задачах *CEC'2017*.

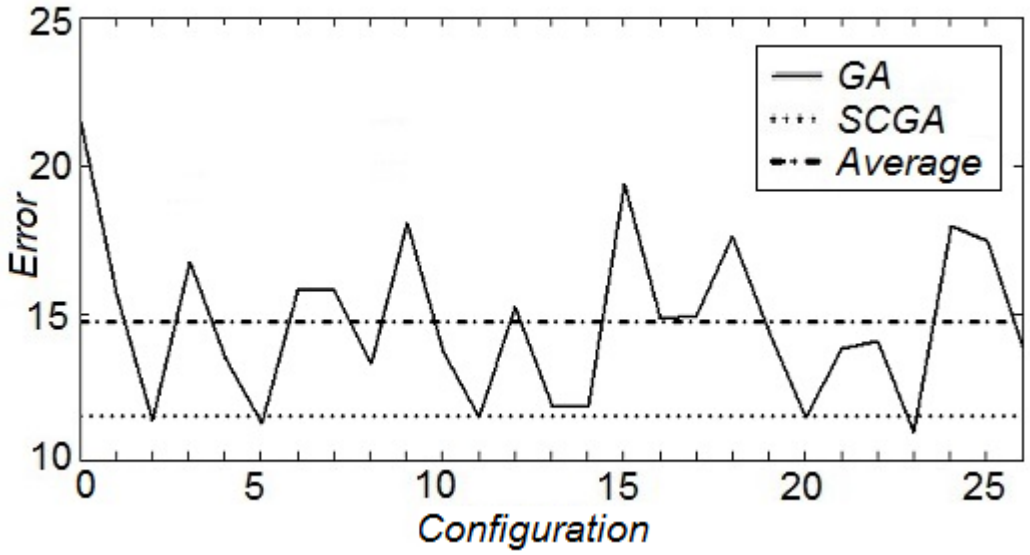

Рисунок 2.3 – Ошибки оптимизации на функции Растригина

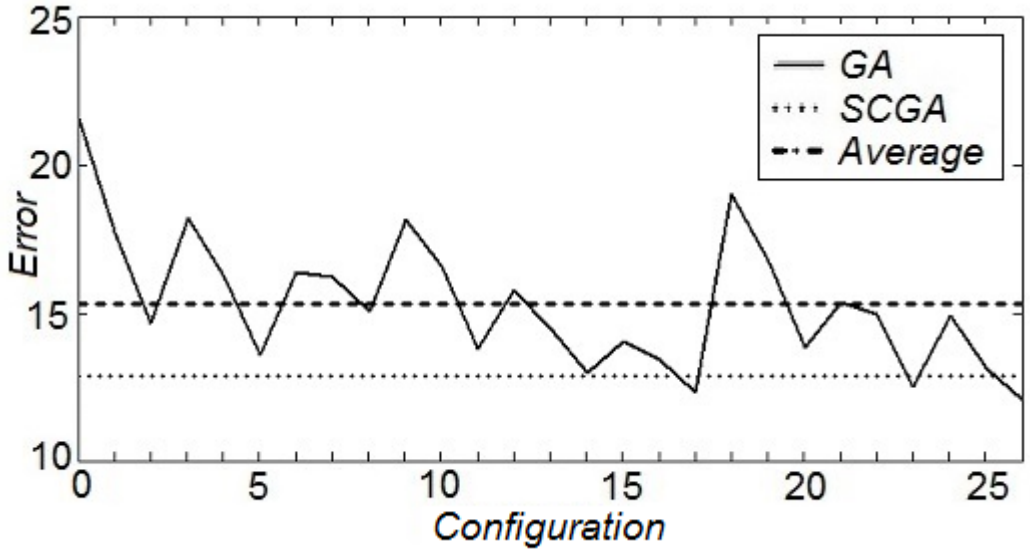

Рисунок 2.4 – Ошибки оптимизации на функции Швефеля

Представленные рисунки 2.3 и 2.4 показывают эффективность метода самоконфигурации (*SCGA*) в сравнении с базовой версией ГА (*GA*). Самоконфигурация существенно сокращает ошибку оптимизации (1.16), незначительно уступая только самой лучшей комбинации параметров.

Следующей реализованной модификацией является процедура расчета вероятности мутации. Т.к. эволюционные алгоритмы являются стохастическими, эффект от объединения нескольких модификаций в алгоритме может отрицательно сказаться на общей производительности. Поэтому требуется проведение всевозможных вариантов тестирования.

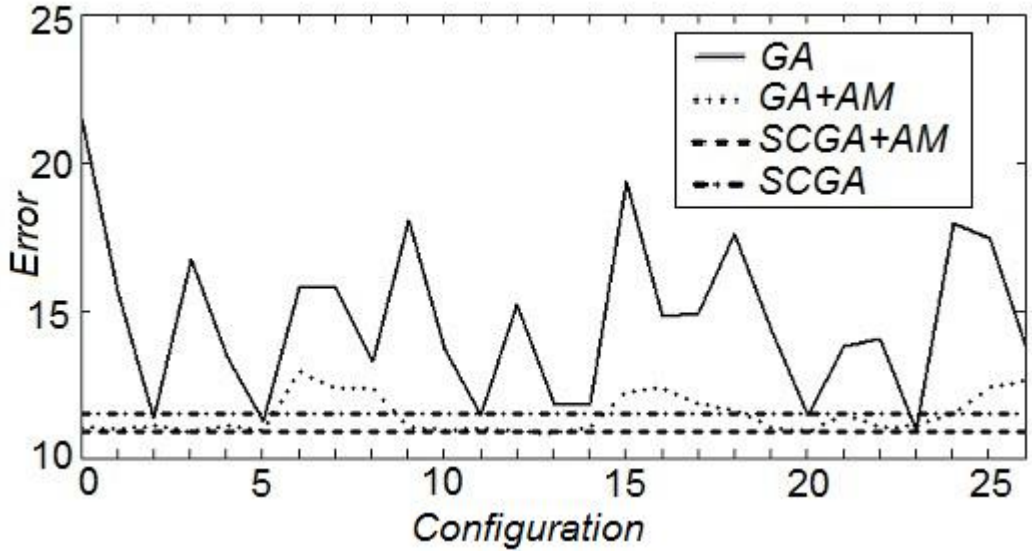

Рисунок 2.5 – Ошибки на функции Растригина

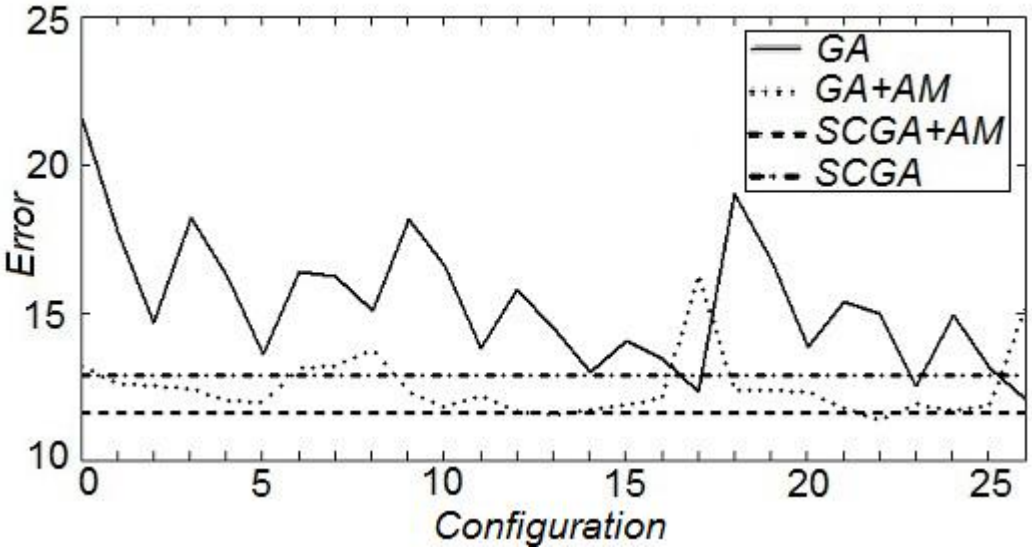

Рисунок 2.6 – Ошибки на функции Швефеля

Рисунки 2.5 и 2.6 показывают влияние адаптивной мутации (+*AM*) на базовую и самоконфигурируемую версию ГА. В обоих случаях наблюдается существенное падение ошибки оптимизации. Отметим, что адаптивная настройка вероятности мутации позволяет существенно снизить влияние конфигурации на эффективность ГА. Настройка коэффициента α выполнялась эмпирически. Значение $\alpha = 8 \pm 5$ позволяет достигать меньшей ошибки оптимизации по сравнению с базовой и самоконфигурируемой версией.

Реализация алгоритма адаптации размера популяции не принесла улучшения в смысле критерия (1.16). Однако характерным вкладом модификации

является снижение количества вычислений ЦФ (рисунок 2.7), требующегося для достижения идентичного с базовой версией уровня ошибки.

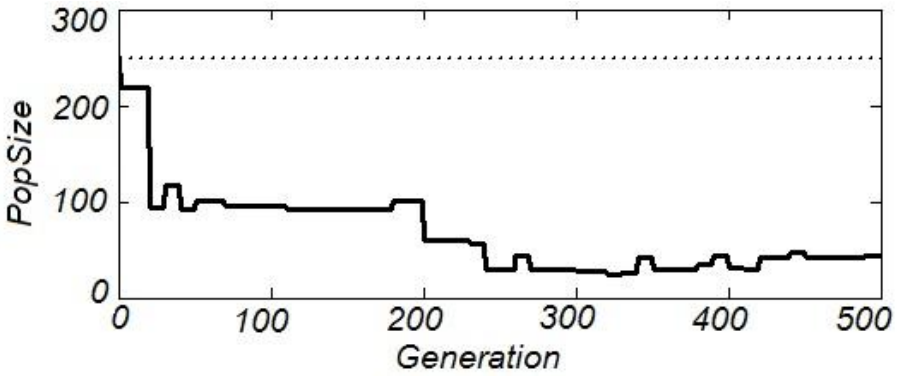

Рисунок 2.7 – Изменение размера популяции в ГА

Напомним, что в базовой модели для задачи десяти переменных задавался размер популяции $PopSize = 250$. Нетрудно заметить, что при одинаковом числе поколений (*Generation*=400) алгоритм с адаптивным размером популяции использует значительно меньше вычислительных ресурсов при той же эффективности получаемых решений.

Результаты совместного применения известных методов адаптации уровня мутации и размера популяции в самоконфигурируемом ГА представлены в таблице 2.4, где числа соответствуют критерию *Score* (1.18), максимальное (наилучшее) значение которого равно 100 и достигается в случае получения алгоритмом наилучшего известного решения.

Таблица 2.4 – Сравнение некоторых из известных подходов адаптации в ГА

	Алгоритм адаптации размера популяции					
		APS	*GAVAPS* [6]	*SAGA* [48]	*APGA* [11]	*PRoFIGA* [105]
Алгоритм адаптации мутации	*AM*	**100**	26	34	36	12
	CGA-AM [9]	5	2	7	9	2
	Self-AM [12]	61	2	27	25	5
	Meta-AM [19]	34	20	36	39	23

Из таблицы видно, что комбинация разработанных в диссертации методов адаптации (*AM* и *APS*) показала наилучший результат. Значительный проигрыш

некоторых вариантов объясняется большой областью значений тестовых функций. Нередко ошибка оптимизации могла достигать 10^3.

Заканчивая представление результатов тестирования генетического алгоритма приведем сравнение различных его модификаций, базовой версии и аналогов, известных из научной литературы [101] по критерию (1.18):

Таблица 2.5 – Сравнение эффективности разработанных модификаций с победителями конкурса

Победители *CEC'2017* [101]	*LSHADE*	*jSO*	*MM-OED*	*IDE best Nsize*	*DYYPO*	*PPSO*	*RB-IPOP-CMA-ES*	*TLBO-FL*
Значение критерия *Score*	**100**	38	35	24	13	13	12	12
Реализованные аналоги	С адаптивной мутацией				Без адаптивной мутации			
	SCGA +APS	*SCGA*	GA_{best}	GA_{aver}	*SCGA +APS*	*SCGA*	GA_{best}	GA_{aver}
Значение критерия *Score*	18	18	10	7	13	13	9	5

Расчет критерия проводился с использованием формулы (1.18) и предоставленными организаторами конкурса данными. *SCGA+APS* – самоконфигурирующийся ГА с адаптивным размером популяции. GA_{best} и GA_{aver} представляют стандартный генетический алгоритм с наилучшей и средней конфигурациями (в смысле усредненной ошибки оптимизации). Из представленной таблицы видно, что наибольшим значением критерия обладает алгоритм *LSHADE* [23]. Итоговый алгоритм, разработанный в диссертации, *SCGA+APS* занимает 5-е место среди представленных аналогов (восьми победителей конкурса), что для генетического алгоритма с бинарным представлением решений является хорошим результатом на тестовых задачах оптимизации с вещественными переменными. Наличие адаптивной настройки размера популяции не привело к улучшению критерия (1.18), однако позволило экономить в среднем до 25% вычислительных ресурсов.

Модификация алгоритма ГП до самоконфигурирующегося с адаптивным выбором размера популяции и уровня мутации также была протестирована на наборе задач *CEC'2017*. На графиках ниже представлены примеры получаемых результатов для функций Захарова и Растригина с использованием *MAE* критерия (1.17).

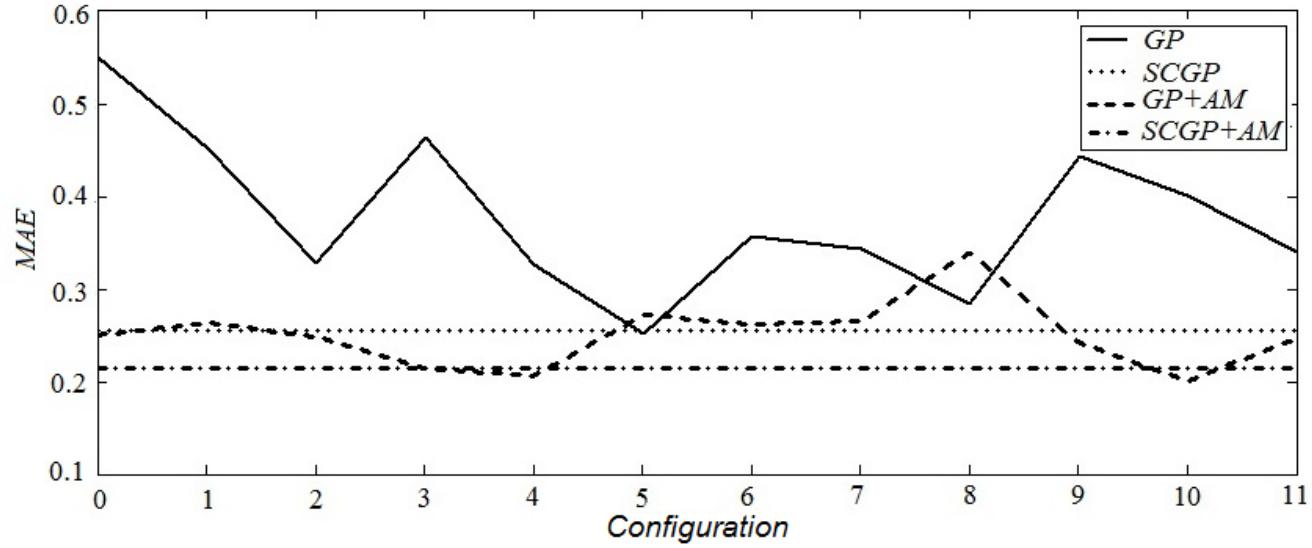

Рисунок 2.8 – Ошибка аппроксимации на функции Захарова

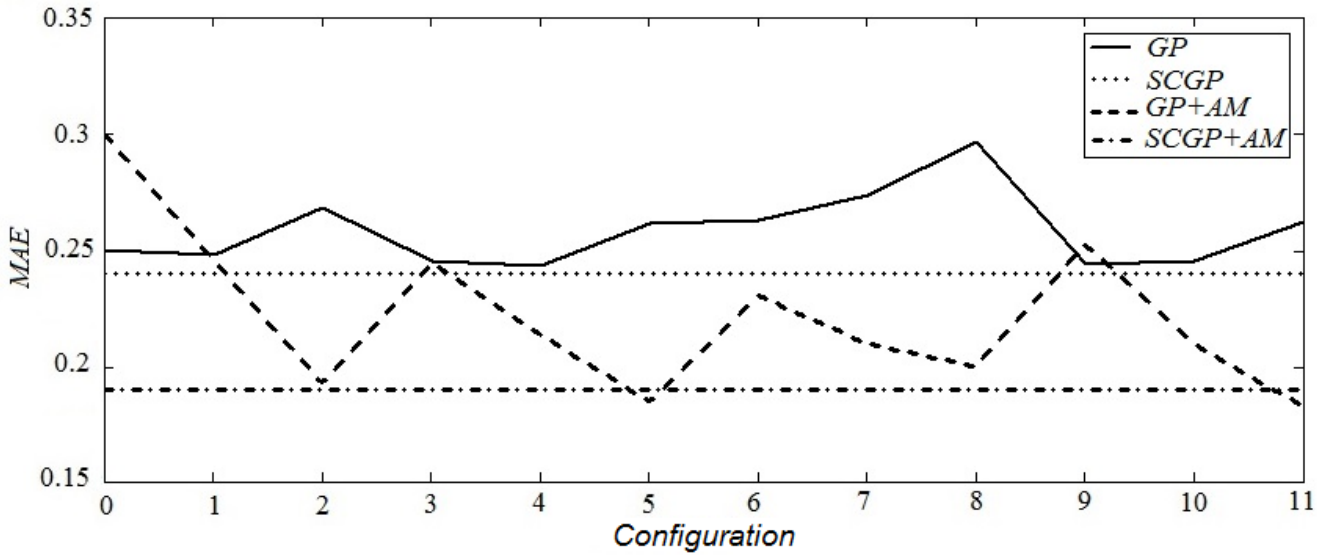

Рисунок 2.9 – Ошибка аппроксимации на функции Растригина

Из представленных графиков следует полезность использования адаптивной мутации при решении задач восстановления символьной регрессии алгоритмом

ГП. Алгоритм самоконфигурирования позволяет уменьшить ошибку (1.17) аппроксимации функции Захарова на 21% и функции Розенброка на 15%.

Аналогично таблице 2.4 таблица 2.6 показывает результаты тестирования реализованных алгоритмов адаптации в ГП с критерием (1.17).

Таблица 2.6 – Сравнение с известными подходами к адаптации в ГП

	Алгоритм адаптации размера популяции					
		APS	*GPVAPS* [6]	*SAGP* [48]	*APGP* [11]	*PRoFIGP* [105]
Алгоритм адаптации мутации	*AM*	**0.37**	0.5	0.45	**0.39**	0.44
	CGP-AM [9]	0.56	0.51	0.59	0.55	0.63
	Self-AM [12]	0.43	0.59	0.49	0.41	0.53
	Meta-AM [19]	0.49	0.46	0.52	0.51	0.41

Отметим, что разница между алгоритмами *APS* и *APGP* была статистически незначимой. Такой результат говорит о возможности применения альтернативных методов контроля размера популяции. Таким образом, согласно таблице 2.6, разработанные методы расчета вероятности мутации и размера популяции в ГП можно признать успешной модификацией.

Далее покажем полезность модификаций, которые были специально отобраны для повышения эффективности алгоритма генетического программирования. Такими модификациями являются: алгоритм селекции обучающих примеров и процедура равномерного скрещивания деревьев. В качестве базовой модели далее будем использовать самоконфигурирующийся алгоритм генетического программирования с разработанными процедурами адаптации.

Функционирование описанного в (2.2) алгоритма селекции обучающих примеров требует определения таких параметров как, период адаптации k и процент отбираемых измерений в обучающую выборку pN. Период адаптации k необходим для обучения модели на текущей подвыборке. Таблица 2.7 показывает

зависимости ошибки аппроксимации (1.6) от указанных параметров при использовании усреднения критерия по 30 функциям.

Таблица 2.7 – Зависимость эффективности алгоритма селекции обучающих примеров в ГП от параметров

pN \ k	25	50	75	100	125	150	175	200
10	0.72	0.75	0.7	0.7	0.69	0.71	0.73	0.76
20	0.5	0.5	0.45	0.46	0.45	0.46	0.45	0.5
30	0.45	0.43	0.42	0.42	0.41	0.42	0.45	0.47
40	0.31	0.29	0.25	0.23	0.23	0.24	0.24	0.25
50	0.29	0.23	0.23	0.21	0.21	0.23	0.22	0.3
60	0.29	0.2	**0.18**	**0.19**	**0.18**	**0.19**	0.24	0.28
70	0.33	0.31	0.26	0.27	0.26	0.27	0.3	0.37
80	0.44	0.34	0.3	0.3	0.29	0.31	0.32	0.33
90	0.45	0.39	0.34	0.35	0.34	0.36	0.37	0.36
100	0.38	0.37	0.37	0.37	0.37	0.38	0.38	0.37

Из представленных результатов следует, что для алгоритма селекции обучающих примеров субоптимальными значениями периода адаптации является $k \in [75,150]$, а для параметра $pN \in [50,60]$. Разница критерия эффективности в одну сотую признается незначимой статистически. Значение $k < 100$ является недостаточным для формирования модели. При $k > 100$ наблюдается эффект переобучения. Следствием такого эффекта является увеличение тестовой ошибки аппроксимации. Алгоритм позволяет использовать каждое второе измерение, что снижает общее время работы приблизительно в 2 раза, параллельно увеличивая точность. Если параметр pN становится меньше 40%, наблюдается резкое увеличение ошибки аппроксимации. Это объясняется избыточной потерей информации (измерений).

Добавление равномерного скрещивания в алгоритм ГП позволило снизить ошибку *MAE* и среднюю сложность дерева (таблица 2.8). При установленном

максимальном уровне размера деревьев равном 100 алгоритм *SCGP* без равномерной рекомбинации достигал сложности, приближенной к максимальной.

Таблица 2.8 – Эффективность применения оператора равномерного скрещивания в ГП

Алгоритм	*MAE*	Средняя сложность дерева
SCGP	0.21	95
SCGP+UR	0.18	67

В заключение представим сравнение базового алгоритма ГП (*Base GP*), с версией, включающей в себя все из перечисленных в главе модификаций (*Adapt-SCGP*), а именно:

1. Самоконфигурирование;
2. Адаптивная мутация;
3. Адаптивный размер популяции;
4. Генетический оператор равномерного скрещивания;
5. Процедура селекции обучающих примеров.

В таблицу 2.9 добавлено сравнение с существующими аналогами, реализованными в среде *RapidMiner*.

Таблица 2.9 – Сравнение эффективности различных модификаций ГП с известными аналогами

Алгоритм	*Mult. Regres.*	*Decision trees*	*Random forest*	*Neural net*	*SVM*	*Adapt-SCGP*	*Base GP*
Значение критерия *MAE*	0.51	0.59	0.53	0.39	0.43	**0.37**	0.52

Параметры альтернативных алгоритмов настраивались средствами системы *RapidMiner*, а в случае невозможности такой настройки – эмпирически. Разработанный алгоритм позволил достигнуть лучшей точности среди представленных аналогов.

ВЫВОДЫ

В данной главе разработаны, программно реализованы и исследованы адаптивные схемы настройки эволюционных алгоритмов, позволяющие в автоматическом режиме корректировать размер популяции и уровень мутации. В алгоритмы включены также полезные модификации, среди которых оператор равномерного скрещивания, самоконфигурирование, селекция обучающих примеров.

Исследование эффективности на репрезентативном множестве тестовых задача конкурса *CEC'2017* показало целесообразность применения, как отдельных модификаций, так и их наборов. Помимо ошибки оптимизации и аппроксимации были улучшены такие критерии как количество вычислений целевой функции, затрачиваемое на получение решения, трудоемкость и время работы алгоритма, а также сложность получаемых решений (для ГП). Сравнение с аналогами подтвердило полезность разработанных модификаций. Важной особенностью является снижение влияния квалификации пользователя на ход эволюционного процесса за счет адаптивной самонастройки алгоритмов в ходе решения задачи. Недостатком подхода является необходимость настройки новых числовых параметров, которая однако намного проще по сравнению с базовыми моделями эволюционных алгоритмов.

Проведенные исследования позволяют рассчитывать на повышение эффективности проектирования сложных технологий интеллектуального анализа данных, к которым относятся искусственные нейронные сети, за счет автоматизации процессов выбора их структур и настройки весовых коээфициентов. Следующая часть работы посвящена использованию разработанных алгоритмов в автоматическом проектировании нейронных сетей и их коллективов.

ГЛАВА 3. РАЗРАБОТКА АДАПТИВНЫХ НЕЙРО-ЭВОЛЮЦИОННЫХ АЛГОРИТМОВ

3.1 История развития нейросетевых технологий анализа данных

История развития нейронных сетей берет свое начало в 40-х годах прошлого века. В 1943 году нейрофизиолог В. МакКаллок и математик В. Питтс предложили формальное понятие нейронной сети и описание нервной активности [69]. В частности, в данной работе ими впервые была предложена модель искусственного нейрона. Несколько позже, в 1949 году Д. Хебб [47] предлагает первый алгоритм обучения, идея которого заключается в том, что связи между нейронами, которые активируются одновременно, усиливаются, данная идея является реализацией принципа обучения без учителя. В 1958 году Ф. Розенблатт предложил [80] однослойный персептрон для решения задач классификации. Несмотря на значительный успех данной модели в решении практических задач, позже выяснилось, что однослойный персептрон неспособен решать ряд важных задач, например т.н. задачу *XOR*. Самим Розенблаттом была предложена идея многослойных персептронов, но в то время она не получила существенного распространения из-за отсутствия подходящих методов обучения. В 1960 году Б. Видроу и М. Хофф разработали модели под названием *ADALINE* и *MADALINE* [108], которые нашли применение в телефонных линиях и используются до сегодняшнего дня. Позднее, в 1962 году ими же был предложен алгоритм обучения, в основе которого лежит дельта-правило, которое, по своей сути является реализацией градиентного спуска по поверхности ошибки. В 1969 году М. Мински и С. Паперт опубликовали работу [71] с описанием основных проблем известных на тот момент нейронных сетей. Данная работа существенно снизила интерес к ним до времен значительного увеличения вычислительной мощи ЭВМ. В 1972 году Т. Кохонен и Дж. Андерсон независимо друг от друга разработали схожие нейронные сети, которые, в своей основе, являлись реализацией

множества моделей типа *ADALINE*. Данные модели заложили основы понимания ассоциативной памяти.

Последующая история развития нейронных сетей неразрывно связана с алгоритмом обратного распространения ошибки, который впервые был предложен в 1974 году П. Дж. Вербосом [106] и А. И. Галушкиным [119]. Использования обратного распространения открыло дорогу к обучению многослойных сетей, однако в то время работы Вербоса и Галушкина не привлекли значительного внимания. К. Фукусима в 1980 году предложил [38] неокогнитрон, иерархичную многослойную нейронную сеть, способную эффективно распознавать рукописные символы, а также решать другие задачи распознавания образов. В 1982 году Дж. Хопфилд предложил названную в его честь сеть Хопфилда, состоящую из нескольких нейронов в одном слое, и способную самостоятельно корректировать ошибки и шумы. Также было показано, что данная система стремится к минимизации энергии. В том же году, Кохоненом была разработана т.н. самоорганизующаяся карта Кохонена [61], способная решать задачи кластеризации и визуализации. Данная нейронная сеть проецирует многомерное пространство в пространство меньшей размерности, и является методом обучения без учителя. В 1986 году независимо группой американских ученых Д. И. Румельхартом, Дж. Е. Хинтоном и Р. Дж. Вильямсом [81] и одновременно группой красноярских ученых С. И. Барцевым и В. А. Охониным [115] был заново открыт метод обратного распространения ошибки, что ознаменовало новую волну интереса к нейронным сетям. Дальнейшее развитие метода обратного распространения ошибки, по сути, свелось к поиску адаптивных методов настройки величины шага при обучении. В 90-е годы, с развитием вычислительных мощностей, распространение получили т.н. свёрточные нейронные сети, например, *LeNet*, предложенная Я. Лекуном в 1998 году [65]. Также, в 1997 году Хохрайтером и Шмидхубером была предложена рекуррентная нейронная сеть, получившая названия длительной краткосрочной памяти (*LSTM, Long Short-Term Memory*), позволившая добиться значительных успехов в распознавании речи и генерации текстов [50]. Дальнейшее развитие

включает появление эффективных методов обучения ограниченных машин Больцмана [49] и множества других архитектур, которые, однако, не будут рассматриваться в данной работе.

3.2 Общие понятия теории ИНС

Под нейронными сетями понимают вычислительные структуры, моделирующие простые биологические процессы человеческого мозга. Элементарный преобразователь в данных сетях – искусственный нейрон, названный так по аналогии с его биологическим прототипом. В состав нейрона входят умножители (синапсы или весовые коэффициенты), сумматор и нелинейный преобразователь (активационная функция). Сумматор отвечает за сложение всей поступающей по синаптическим связям информации, как от других нейронов, так и от внешних входных сигналов. Схематически искусственный нейрон можно представить следующим образом:

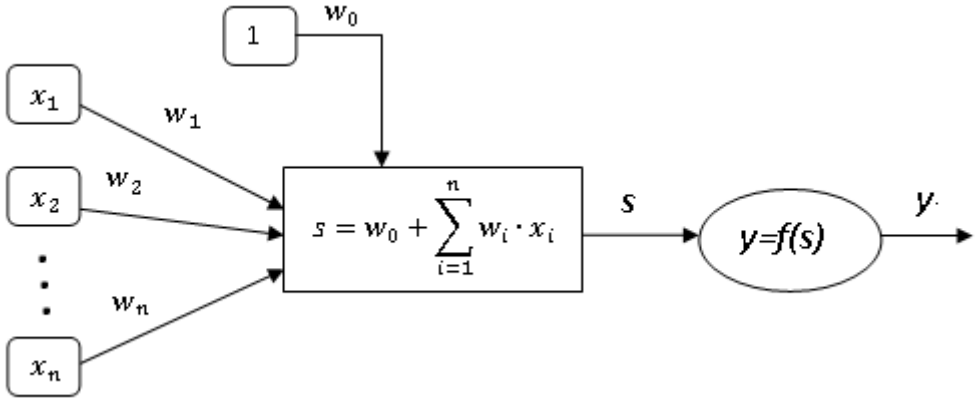

Рисунок 3.1 – Искусственный нейрон.

Здесь n – число входов нейрона, w_i – весовые коэффициенты связи соответствующих значений x_i, $f(s)$ – активационная функция, y – выход нейрона. Из представленного рисунка видно, что выход нейрона зависит от активационной функции. В общем случае активационная функция – нелинейная, однако существуют и линейные варианты. Представим наиболее известные из них:

Таблица 3.1 – Виды активационных функций искусственных нейронных сетей

Название	Формула	Область значений
Пороговая	$f(s)=\begin{cases}0, s<\alpha\\1, s\geq\alpha\end{cases}$	$[0,1]$
Знаковая	$f(s)=\begin{cases}-1, s<\alpha\\1, s\geq\alpha\end{cases}$	$[-1,1]$
Сигмоидальная	$f(s)=\frac{1}{1+e^{-\alpha\cdot s}}$	$(0,1)$
Полулинейная	$f(s)=\begin{cases}0, s<\alpha\\s, s\geq\alpha\end{cases}$	$[0,\infty)$
Линейная	$f(s)=\alpha\cdot s$	$[-\infty,\infty)$
Радиальная базисная	$f(s)=e^{-s^2}$	$(0,1)$
Треугольная	$f(s)=\begin{cases}\alpha-\lvert s\rvert, \lvert s\rvert<\alpha\\0, \lvert s\rvert\geq\alpha\end{cases}$	$[0,\alpha]$
Гиперболический тангенс	$f(s)=\frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$	$(-1,1)$
Полулинейная с насыщением	$f(s)=\begin{cases}0, s<0\\s, 0\leq s\leq\alpha\\\alpha, s>\alpha\end{cases}$	$[0,\alpha]$
Линейная с насыщением	$f(s)=\begin{cases}-\alpha, s<-\alpha\\s, -\alpha\leq s\leq\alpha\\\alpha, s>\alpha\end{cases}$	$[-\alpha,\alpha]$

Объединенные каким-либо образом нейроны образуют нейронную сеть. В зависимости от способа объединения нейронов выделяют множество различных типов нейронных сетей, среди которых: персептрон Розенблатта, сети Хемминга, Ворда, Кохонена, Элмана, свёрточные нейронные сети и д.р. Искусственная нейронная сеть (ИНС) – совокупность моделей биологических нейронов, связанных между собой в единую систему.

Обычно ИНС состоят из входного, выходного и скрытых слоев. При этом, число нейронов входного слоя, обычно, равно числу входов решаемой задачи, а число нейронов выходного – количеству выходов задачи. Скрытых слоев может быть несколько, и они могут содержать разное число нейронов. Приведем стандартный вид ИНС, который зачастую используется для решения задач:

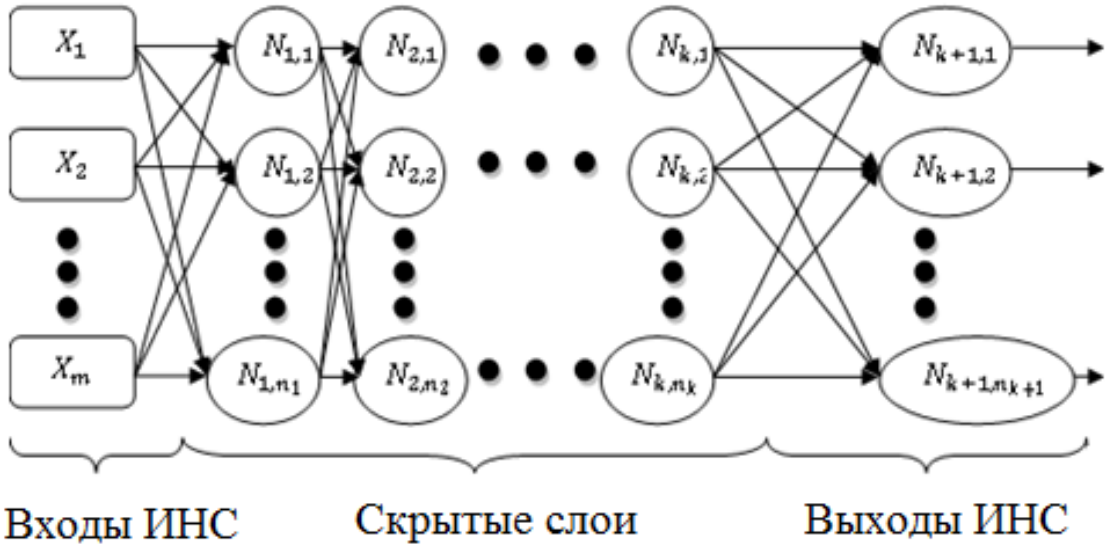

Рисунок 3.2 – Общий вид ИНС.

На данном рисунке $\overline{X}$ – вектор входов нейросети (или атрибутов решаемой задачи), m – общее количество входов, k – количество скрытых слоев в ИНС, $N_{i,j}$ – j-й нейрон i-го слоя сети. В общем случае количество нейронов на каждом из скрытых слоёв является различным и обозначается вектором $\overline{N}$. $k+1$ – выходной слой сети.

Для решения задач при помощи таких ИНС пользователь задает ее структуру, а именно выбирает число скрытых слоев и определяет количество нейронов на каждом из них, а также типы функций активации; затем при помощи методов оптимизации проводится обучение выбранной сети. Если эффективность работы устраивает пользователя, полученная модель в дальнейшем используется для решения задачи, в противном же случае изменяется структура и процедура повторяется снова. Стоит заметить, что, как правило, используются сигмоидальные функции, указанные в таблице 3.1.

Существуют два подхода к выбору структуры НС.

Первый подход заключается в том, что изначально задается сеть большего объема, чем необходимо для решения поставленной задачи. Затем сеть проходит процедуру обучения при помощи какого-либо метода оптимизации (ГА, алгоритм сопряженных градиентов и т.д.). После этого, основываясь на некоторых методиках, удаляются незначимые нейроны и связи. Это происходит до тех пор, пока ошибка несущественно изменяет свое значение.

Второй подход – противоположность первому. К сетям (обученным) с изначально малой архитектурой при помощи определенных правил добавляются нейроны и связи. Это происходит до тех пор, пока НС не обучается на поставленную задачу.

Данные подходы имеют один большой недостаток: они являются локальными методами поиска. Каждый из предложенных подходов может прекратить свою работу при достижении некоторого локального оптимума.

Такие подходы позволяют строить относительно простые модели нейронных сетей, однако их точность зачастую является недостаточной и напрямую зависит от опыта и удачливости пользователя при выборе структуры.

Разработка и реализация программных систем, реализующих автоматическое генерирование нейросетевых моделей, позволит избежать этого недостатка. Для придания гибкости таким программным системам необходимо заложить в них некоторые модификации стандартной структуры, показанной на рисунке 3.2.

Известно, что зачастую часть атрибутов при решении различных задач является зашумленными, коррелирующими друг с другом, малоинформативными. Их использование при построении модели может заметно ухудшить ее качество. Потому в процессе функционирования программной системы необходимо находить и отбрасывать такие атрибуты.

Полезными модификациями также являются процедуры настройки внутренней структуры скрытых слоев ИНС. Перечислим их:

1. Автоматический выбор количества скрытых слоев в ИНС, а также выбор количества нейронов на каждом из них;

2. Предыдущий вариант с добавлением возможности настройки типов активационных функций (указанных в таблице 3.1) для каждого нейрона сети;

3. Вариант номер 2 с возможностью добавления межслойных связей, удалением некоторых связей между нейронами.

Механизмы, позволяющие производить отбор информативных признаков и настраивать внутреннюю структуру скрытых слоев ИНС в реализованных эволюционных алгоритмах, будут описаны ниже.

Для того чтобы ИНС с выбранной структурой корректно решала поставленную задачу, необходима настройка всех весовых коэффициентов каждого нейрона сети, а также некоторых числовых параметров (параметры активационных функций). Существуют 2 вида обучения НС:

- Обучение с учителем. Данный вид обучения происходит на основании обучающей выборки.
- Обучение без учителя. Данный метод предполагает обучение сети без наличия обучающей выборки [120].

Наиболее известным методом обучения ИНС является метод обратного распространения ошибки [121]. Данный метод основан на корректировке весов от выхода к входу таким образом, чтобы ошибка уменьшалась. В данном алгоритме градиент вычисляется по рекуррентным формулам. Однако данный подход может быть использован только для полносвязных сетей с послойным распространением сигнала, что делает его непригодным для использования в общем случае. Для обучения ИНС в данной работе используется алгоритм *CMA-ES* [52] и *GA* с локальным спуском. Оптимизируемым критерием является следующий:

$$F(\mathrm{W}) = \frac{1}{n} \cdot \sum_{i=1}^{n} (eval(x_i, W) - y_i)^2 . \qquad (3.1)$$

Здесь W – набор всех вещественных параметров НС (включает веса и параметры активационных функций; $eval(x_i, W)$ функция вычисления значения на выходе ИНС, соответствующего вектору входных параметров x_i; y_i – истинное значение прогнозируемой величины, n – объем выборки.

3.3 Разработка адаптивных нейро-эволюционных алгоритмов

Для того чтобы решать задачу автоматического формирования НС при помощи ГП, необходимо выбрать способ кодирования НС деревом, а также выбрать функцию пригодности. Остальной же алгоритм ГП останется без изменений. В качестве функции пригодности можно выбрать (1.12), т.к. она учитывает (помимо ошибки) сложность полученной сети, а также число переменных, входящих в задачу. Данная формула легко приводится к (1.9) и (1.11) при помощи соответствующих значений коэффициентов. В данной работе кодирование НС будет производиться при помощи бинарных деревьев (т.е. функциональное множество – только бинарные операции) [123]. Необходимо определить функциональное и терминальное множества для алгоритма, учитывая условия замкнутости и достаточности. В данном алгоритме терминальное множество будут составлять все переменные-входы задачи раздельно (что позволит производить отбор информативных признаков), а также нейроны со всеми типами активационных функций, указанных в таблице 3.1. Функциональное множество будет наполнено следующими операторами:

- + – это оператор объединения нейронов в слой;
- > – оператор упорядочивания, означающий, что «левое» поддерево является входом в «правое»;
- < – аналог предыдущего оператора (происходит лишь инверсия операндов).

Проиллюстрируем сказанное. На рисунке 3.3 представлен пример кодирования простой НС при помощи дерева с использованием описанных выше элементов. Здесь N_1 – нейрон первого типа (таблица 3.1); In_j – j-ый атрибут задачи.

Стоит заметить, что при таком кодировании может сложиться ситуация, когда нейрон не будет иметь входов. Тогда он будет усложнять структуру НС, не неся никакого полезного действия. На такие нейроны будут подаваться все переменные-входы задачи.

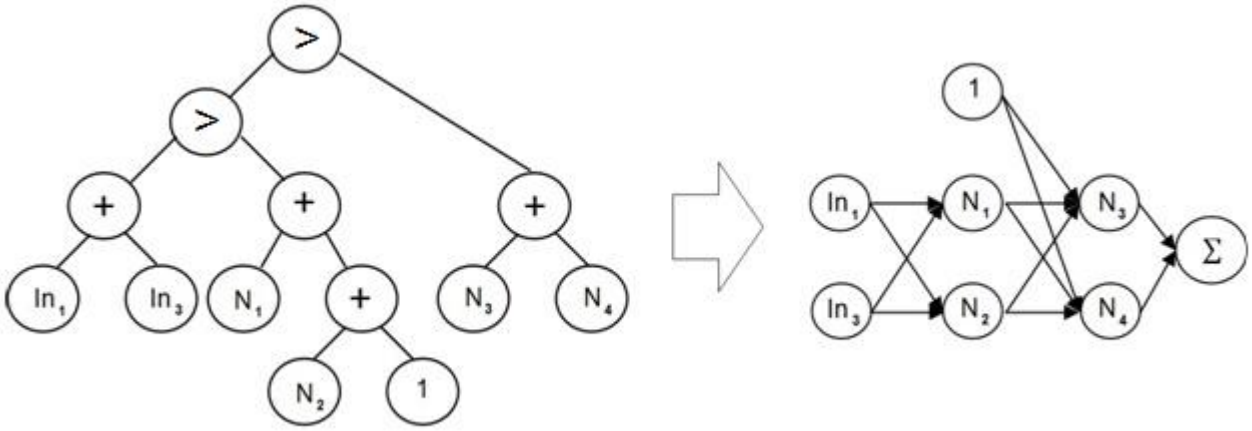

Рисунок 3.3 – Пример кодирования НС.

Однако стоит заметить, что данный подход нельзя использовать без одной очень важной модификации. В общем случае, функциональное и терминальное множества не удовлетворяют условиям замкнутости. А именно, возможны случаи, когда, например, на переменную-вход будет подаваться некоторое значение. Тогда необходимо провести адаптацию дерева:

Рисунок 3.4 – Пример адаптации.

При реализации алгоритма в виде программной системы используется представление деревьев в виде ОПЗ (Обратная Польская Запись, *Reverse Polish Notation*) [30]:

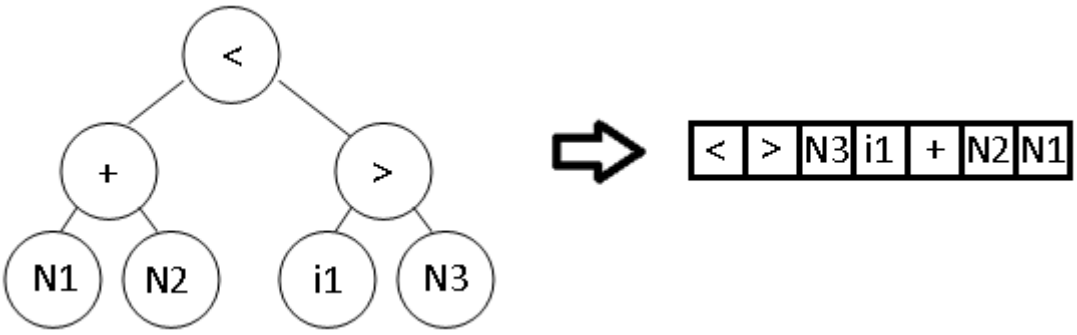

Рисунок 3.5 – Представление дерева в виде ОПЗ.

Использование указанного функционального множества подразумевает вычисление по дереву в направлении корня. Для того чтобы избежать рекурсивных вычислений по дереву (что может в разы увеличить быстродействие итоговой программной системы), необходимо проводить процедуры адаптации некоторых деревьев. Пример одного из таких деревьев представлен на рисунке ниже.

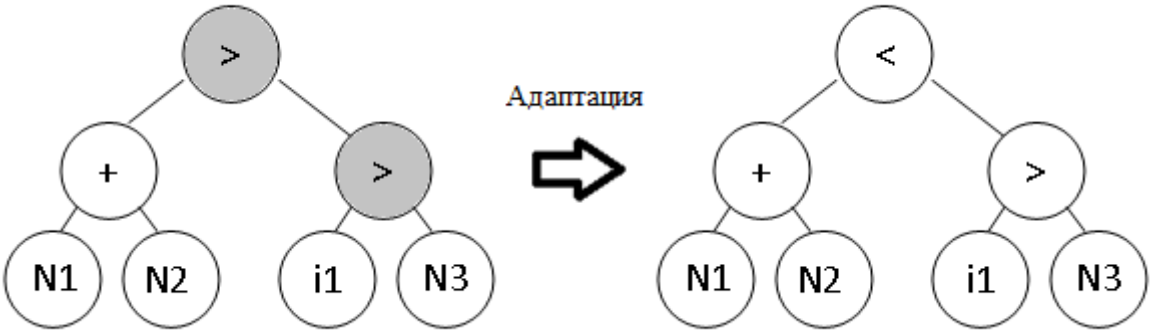

Рисунок 3.6 – Пример адаптации.

Появление связей, обозначенных затемненными кругами, будет требовать другого метода чтения дерева, что значительно усложнит вычисление по последнему. Поэтому предлагаются различные варианты адаптации для этого случая (один из возможных вариантов представлен выше). Функция адаптации дерева играет важную роль в представленном алгоритме. Она обеспечивает условие замкнутости для терминального и функционального множеств.

Генетический алгоритм кодирует ИНС при помощи матрицы смежности $M_{n\times n}$ нейронов. В работе используется треугольная матрица, т.к. рекуррентные нейронные сети не рассматриваются. При необходимости добавляются две матрицы, отвечающие за отбор информативных признаков $I_{k\times n}$ и кодирования типа активационных функций $A_{3\times n}$. k – количество атрибутов задачи, n – максимальное число нейронов сети. Тип каждой активационной функции нейрона кодируется при помощи 3 бит информации. Далее из трех представленных матриц формируется бинарная строка, позволяющая генетическому алгоритму производить процедуру оптимизации структуры. Параметр n задается пользователем.

Предложенные подходы допускают:

- Наличие межслойных связей;
- Отсутствие некоторых связей между слоями;
- Использование различных видов нейронов;
- Отбор информативных признаков в процессе функционирования;
- Выращивание (автоматическая настройка) скрытых слоев.

3.4 Описание решаемых практических задач классификации, регрессии, прогнозирования. Используемые критерии эффективности.

Проверка эффективности разработанных адаптивных эволюционных алгоритмов проектирования искусственных нейронных сетей производилась на тестовых задачах с репозиториев машинного обучения *KEEL* [5] и *UCI* [7]. В таблице ниже представлены основные числовые характеристики данных задач.

Таблица 3.2 – Тестовые задачи из репозиториев машинного обучения

Задача	Число измерений	Количество атрибутов	Число классов
Классификация			
Magic	19020	10	2
Page-Blocks	5472	10	5
Australian credit	690	14	2
German credit	1000	20	2
Texture	5500	40	11
Twonorm	7400	20	2
Segment	2310	19	7
Ring	7400	20	2
Penbased	10992	16	10
Satimage	6435	36	6
Регрессия			
Forest Fires	517	12	-
Weather	1609	9	-
Concrete	1030	8	-
Pole Telecom.	14998	26	-
House	22784	16	-

В таблице 3.2 представлены задачи из различных областей. Так, задача *German credit* оценивает по различным характеристикам (кредитная история, объем требуемых кредитных средств, возраст и др.) возможность выдачи кредита обратившемуся в кредитную организацию лицу. Задача *Australian credit* выявляет подозрительные операции с банковскими картами, используя историю транзакций. *Weather* является задачей прогнозирования погоды в зависимости от таких характеристик, как скорость ветра, давление, уровень моря, максимальная и минимальная температура. *Concrete* – задача прогнозирования прочности бетона на сжатие в зависимости от физико-химического состава. Задача *Pole Telecom* является коммерческой, значение ее атрибутов скрыто от исследователей, однако она обладает большой размерностью (среди представленных на сайте *KEEL*) и полезна для тестирования разрабатываемых алгоритмов.

Разработанными алгоритмами решались также две реальные практические задачи, а именно: задача распознавания эмоций по звуковому сигналу, задача прогнозирования уровня заболеваемости населения в зависимости от химического состава воздуха г. Красноярска.

Решение задачи распознавания эмоций по звуковому сигналу проводится на основании выборки полученной в Техническом университете Берлина, которая включает в себя набор высказываний, на немецком языке записанный десятью актерами разного пола [25, 46]. В оригинале выборка представляет набор коротких звуковых файлов, однако имеется также и её оцифрованный вариант, что приводит ее к задаче классификации. Всем высказываниям в выборке сопоставлен один из семи классов: нейтральный, гнев, страх, радость, скука, тоска, отвращение. Высказывания характеризуется сотнями различных характеристик: амплитуда, частота, спектральные и кепстральные характеристики. Общее число атрибутов – 384.

В задаче прогнозирования уровня заболеваемости населения выборка представлена массивом с 39 столбцами, 28 из которых являются атрибутами задачи, а 10 – прогнозируемые значения. Задача имеет всего 32 измерения. Один из столбцов – год, в котором были сделаны замеры. А таблице данных имеются

пропуски. Выборка получена из отчетов организации «КрасСтат». Атрибуты показывают содержание различных вредных веществ в воздухе: медь, цинк, железо, хлор и т.д. Остальные измерения показывают различные возникшие заболевания и смертность от них. В приложении В показаны основные числовые характеристики задачи, а также нумерация прогнозируемых значений, которые будут использованы в дальнейших исследованиях.

Для оценки эффективности алгоритмов машинного обучения существует множество критериев. Их вид, как правило, зависит от предпочтений пользователя относительно качества получаемой модели. Представим некоторые из них.

В задачах классификации наиболее простым и интуитивно понятным является такой критерий как точность:

$$Accuracy = \frac{P}{N} \tag{3.1}$$

где P количество верно классифицированных объектов, N – общее число объектов. Производным критерием является ошибка классификации:

$$Error = 1 - Accuracy \tag{3.2}$$

На практике нередко возникают случаи, когда количество объектов, принадлежащих одному из классов (мажоритарный класс), значительно превосходит количество объектов другого класса (миноритарный класс). Такие задачи встречаются в медицинской диагностике [75, 26], в задачах выявления подозрительных банковских операций [27], в распознавании нештатных ситуаций различных технологических процессов [130]. Из формулы (3.1) видно, что классификаторы, построенные с использованием такого критерия, будут смещаться в сторону мажоритарного класса. Однако наибольший интерес, как правило, представляют объекты миноритарного класса. Например, в задачах медицинской диагностики простейшее правило определения всех пациентов как здоровых может соответствовать точности 0.9 и выше. При этом правило абсолютно не способно выявить больных пациентов и для практики является бессмысленным. Для задач с несбалансированными данными (с существующими

мажоритарными и миноритарными классами) были разработаны альтернативные критерии [95]. Данные критерии определяются с использованием матрицы ошибок A:

Таблица 3.3 – Пример матрицы ошибок в задаче классификации

		Прогноз классификатора		
	№ класса	1	2	3
Истинные значения	1	57	0	2
	2	1	17	3
	3	0	1	29

Значения таблицы указывают число верно и ошибочно классифицированных объектов по каждому из классов. В примере из таблицы 3.3 количество верно классифицированных объектов класса 1 равно 57. При этом два раза объект первого класса был ошибочно отнесен к третьему. Теперь величину (3.1) можно пересчитать как:

$$Accuracy = \frac{\sum_{i=1}^{n} A_{i,i}}{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}} \tag{3.3}$$

В формуле (3.3) n – количество классов в задаче. По матрице A становится возможным вычисление точности и полноты для каждого из классов:

$$Precision_{cl} = \frac{A_{cl,cl}}{\sum_{i=1}^{n} A_{cl,i}} \tag{3.4}$$

$$Recall_{cl} = \frac{A_{cl,cl}}{\sum_{i=1}^{n} A_{i,cl}} \tag{3.5}$$

Результирующая точность и полнота рассчитывается как среднее арифметическое по всем классам (3.4): $Precision = \frac{1}{n} \cdot \sum_{i=1}^{n} Precision_i$, $Recall = \frac{1}{n} \cdot \sum_{i=1}^{n} Recall_i$. Такое усреднение называется макроусреднением [95].

Одновременная максимизация точности и полноты на практике редко возможна, поэтому вводится F-мера:

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.6}$$

F-мера является компромиссом между точностью и полнотой. При необходимости можно изменить веса точности и полноты в формуле (3.6):

$$F = (\beta^2 + 1) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall} \quad (3.7)$$

Полагая $\beta \in (0,1)$ приоритет отдается точности классификации, при $\beta > 1$ - полноте. Значение $\beta = 1$ приводит формулу (3.7) к (3.6).

Для задач бинарной классификации (число классов равно двум) является распространенным *AUC* (*area under ROC curve*) [64] критерий.

В задачах восстановления регрессии и прогнозирования используется множество критериев [33]. Определим величины $E_i = X_i - F_i$ и $PE_i = \frac{X_i - F_i}{X_i}$, где X_i – истинное значение, а F_i – прогнозируемое. При объеме выборки равном *n* критерии имеют следующий вид (таблица 3.4):

Таблица 3.4 – Критерии эффективности задачи регрессии и прогнозирования

Название	Формула критерия
ME (mean error)	$ME = \frac{1}{n} \cdot \sum_{i=1}^{n} E_i$
MAD (mean absolute deviation) *MAE (mean absolute error)*	$MAE = MAD = \frac{1}{n} \cdot \sum_{i=1}^{n} \lvert E_i \rvert$
MSE (mean of squared errors)	$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (E_i)^2$
RMSE (root mean squared errors)	$RMSE = (\frac{1}{n} \cdot \sum_{i=1}^{n} (E_i)^2)^{\frac{1}{2}}$
MPE (mean percentage error)	$MPE = \frac{100}{n} \cdot \sum_{i=1}^{n} PE_i$
MAPE (mean absolute percentage error)	$MAPE = \frac{100}{n} \cdot \sum_{i=1}^{n} \lvert PE_i \rvert$

Для объективной оценки качества получаемых моделей необходимо разделять выборку задачи на тестовую и обучающую. Среди существующих подходов наиболее известными являются:

1. Случайное разбиение;
2. Скользящий экзамен;
3. Кросс-валидация.

В случайном разбиении с заданной пользователем вероятностью объект относится к обучающей выборке. Объекты, не попавшие в обучающую выборку, относятся к тестовой. Обычно в качестве данной вероятности выбирают $p_{train} \in [0.6,0.9]$. Качество модели вычисляется на основании тестовой выборки.

При проведении процедуры скользящего экзамена в обучающую выборку попадают все измерения, кроме *i*-го. На обучающей выборке объема *n*-1 выполняется тренировка модели, а ее эффективность оценивается по тестовому *i*-му измерению. Процедура скользящего экзамена выполняется *n* раз при $i = \overline{1,n}$. Критерий эффективности усредняется по всем итерациям процедуры. Очевидным недостатком такого подхода является высокая вычислительная сложность, т.к. при относительно небольших объемах данных время, затрачиваемое на тренировку *n* моделей, достаточно велико.

Метод кросс-валидации можно назвать общим случаем скользящего экзамена. А данном алгоритме разбиения выбирается число *k* – количество частей, на которые разбивается исходная выборка задачи. Одна из частей относится к тестовой выборке (под номером *j*), остальные – к обучающей. Далее согласно процедуре скользящего экзамена выполняется серия из *k* экспериментов при $j = \overline{1,k}$ и вычисляется критерий эффективности. Отметим, что в некоторых случаях кросс-валидации к тестовой выборке относят сразу несколько частей из исходной.

3.5 Исследование эффективности предлагаемых адаптивных эволюционных алгоритмов нейросетевого моделирования

Разработанные в главе нейро-эволюционные алгоритмы (ГА и ГП) были программно реализованы и протестированы на задачах, описанных в предыдущем пункте. К алгоритмам применены различные, в том числе разработанные в рамках диссертационного исследования, модификации. Данные модификации обладают параметрами, исчерпывающее тестирование по настройке которых не представляется возможным. Ниже будут приведены те значения (таблица 3.5), которые рекомендуются для использования в задачах нейросетевого моделирования.

Таблица 3.5 – Рекомендуемые значения параметров модификаций ГА и ГП

	Модификация			
	Самоконфигу-рирование	Адаптивная мутации	Адаптивный размер популяции	Равномерное скрещивание
Значение в ГА	$C \in [0.5,2]$	$\alpha \in [4,12]$	$k \in [3,7]$ $tOpt \in [10,50]$ $m \in [-0.2,0.2]$ $d \in [0.25,2]$	-
Значение в ГП		$\alpha \in [2,6]$		-

В таблице 3.5 отсутствуют параметры селекции обучающих примеров, т.к. они зависят от типа решаемой задачи. Для задачи восстановления регрессии это период адаптации $k \in [100,150]$ и процент обучающей подвыборки $pN \in [40,60]$. В задачах классификации $k \in [50,200]$, $pN \in [25,40]$.

В первую очередь тестировались базовые и самоконфигурирующиеся версии алгоритмов ГА и ГП. Дополнительно для сравнения добавлен алгоритм с оператором равномерного скрещивания (*SCGP+UR*). При тестировании использовалась процедура 10-частной кросс-валидации. В таблице 3.6 представлена ошибка классификации (3.2) выраженная в процентах, а также *MSE* критерий таблицы 3.4.

Таблица 3.6 – Сравнение эффективности самоконфигурирующихся и базовых версий алгоритмов

	GA_{aver}	GP_{aver}	GA_{best}	GP_{best}	*SCGA*	*SCGP*	*SCGP+UR*
Классификация							
Magic	16.66	17.54	15.78	15.7	15.42	15.74	15.46
Page-Blocks	4.64	4.65	4.35	4.58	4.35	4.39	4.34
Australian credit	12.57	12.23	11.77	11.74	11.28	11.71	11.46
German credit	27.89	27.67	24.6	25.91	24.3	24.51	24.35
Texture	9.37	9.46	8.62	8.59	8.21	8.26	8.25
Twonorm	6.32	6.97	6.28	6.52	6.16	6.17	6.12
Segment	6.74	7.05	6.52	6.4	6.38	6.29	6.29
Ring	7.12	7.29	7.1	6.89	6.74	6.71	6.7
Penbased	9.13	9.04	8.7	8.13	8.37	8.27	8.11
Satimage	15.36	15.4	15	14.87	14.43	15.12	15.04
Регрессия							
Forest Fires	389.5	391.55	363.87	355.67	351.58	358.5	353.8
Weather	1.17	1.2	1.16	1.16	1.17	1.16	1.15
Concrete	33.52	33.1	33.33	32.86	32.09	31.47	30.65
Pole Telecom.	84.58	84.37	84.18	82.94	82.36	82.53	80.26
House	9.04	9.06	8.91	8.62	8.94	8.63	8.51

В представленной таблице необходимо выделить 2 вариации нейэро-эволюционного алгоритма, задающие минимум ошибки классификации и восстановления регрессии: *SCGA*, *SCGP+UR*. Добавление равномерного скрещивания в алгоритм генетического программирования позволило существенно увеличить точность моделей, получаемых алгоритмом *SCGP*. Такой эффект возникает за счет большего разнообразия генерируемых в ходе эволюционного процесса деревьев. Эволюция ИНС в алгоритмах ГА и ГП значительно отличается. ГА на ранних этапах является более трудоемким, т.к. изначально генерирует более сложные структуры. В некоторых случаях самоконфигурирующийся алгоритм проигрывает лучшей комбинации параметров (*best*), однако последняя была выявлена полным перебором, что является недопустимым при решении реальных задач. Если сравнивать со средними

значениями (GA_{aver}, GP_{aver}), то становится очевидным преимущества разрабатываемого подхода.

Использование адаптивной мутации в реализованных алгоритмах позволило снизить ошибку классификации и регрессии (таблица 3.7). В процессе тестирования было выявлено, что алгоритмы ГА и ГП с модификациями самоконфигурирования, равномерного скрещивания, адаптивными мутацией и размером популяции имеют равную эффективность с точки зрения непараметрического критерия Вилкоксона. В дальнейшей работе для компактности представления результатов их эффективности будут представлены среднеарифметическим значением. Алгоритму присвоим аббревиатуру *ANNEA*. Модификация, позволяющая в автоматическом режиме настраивать размер популяции (*+APS*), позволила уменьшить общее число вычислений целевой функции в среднем на 25%. При этом эффективность алгоритмов осталась неизменной.

Таблица 3.7. Исследование эффективности алгоритмов адаптивной мутации и размера популяции

	SCGA	*SCGP+UR*	*ANNEA*	*ANNEA+APS*
Классификация				
Magic	15.42	15.46	15.2	15.08
Page-Blocks	4.35	4.34	4.13	4.23
Australian credit	11.28	11.46	11.16	10.99
German credit	24.3	24.35	23.84	23.76
Texture	8.21	8.25	7.89	8.05
Twonorm	6.16	6.12	5.9	5.97
Segment	6.38	6.29	6.19	6.14
Ring	6.74	6.7	6.67	6.54
Penbased	8.37	8.11	8.05	7.91
Satimage	14.43	15.04	14.01	14.09
Регрессия				
Forest Fires	351.58	353.8	339.41	341.7
Weather	1.17	1.15	1.12	1.09
Concrete	32.09	30.65	27.9	28.4
Pole Telecom.	82.36	80.26	74.34	73.5
House	8.94	8.51	8.3	8.37

На всех представленных задачах алгоритм (*ANNEA*) смог найти более удачную конфигурацию искусственной нейронной сети. Комбинированное использование процедур адаптации в 8 случаях из 15 позволило увеличить точность. При более

тщательном подборе условий эксперимента (число поколений и эпох обучения) для каждой из задач возможно показать статистическую значимость превосходства комбинированного метода. В условиях недостаточного количества вычислительных ресурсов алгоритм адаптации размера популяции однозначно является полезной модификацией. Алгоритм адаптации вероятности мутации полезен при любых условиях.

Подход селекции обучающих примеров позволил в главе 2 значительно снизить трудоемкость алгоритма. Ранее было описано, что в реальных задачах анализа данных нередко встречаются дублирующиеся, зашумленные, плохо распознаваемые измерения. В таких условиях оправдано применение описанного в пункте (2.2) диссертации алгоритма селекции обучающих примеров. Результаты его применения в нейросетевом моделировании указаны в таблице 3.8. Алгоритм с отбором измерений обозначен приставкой (+*IS*).

Таблица 3.8 –Эффективность нейросетевого моделирования с алгоритмом селекции обучающих примеров

	ANNEA+APS	*ANNEA+APS+IS*
Классификация		
Magic	15.08	14.72
Page-Blocks	4.23	4.23
Australian credit	10.99	9.97
German credit	23.76	21.70
Texture	8.05	7.27
Twonorm	5.97	4.24
Segment	6.14	5.10
Ring	6.54	4.99
Penbased	7.91	3.55
Satimage	14.09	13.82
Регрессия		
Forest Fires	341.7	317
Weather	1.09	1.06
Concrete	28.4	26.9
Pole Telecom.	73.5	67.4
House	8.37	7.9

Ошибка нейросетевого моделирования была уменьшена на большинстве из представленных задач. Кроме того, время работы алгоритмы было снижено в 3 раза в случае классификации и в 2 раза при аппроксимации.

Не все представленные задачи классификации являются сбалансированными (таблица 3.9.). Следовательно, точность и ошибка классификации могут не объективно отражать полезность получаемых моделей.

Таблица 3.9 – Распределение объектов по классам

	1	2	3	4	5	6	7	8	9	10	11
Magic	12332	6688	-	-	-	-	-	-	-	-	-
Page-Blocks	4913	329	-	-	-	-	-	-	-	-	-
Australian credit	300	390	-	-	-	-	-	-	-	-	-
German credit	300	700	-	-	-	-	-	-	-	-	-
Texture	500	500	500	500	500	500	500	500	500	500	500
Twonorm	3703	3697	-	-	-	-	-	-	-	-	-
Segment	330	330	330	330	330	330	330	-	-	-	-
Ring	3664	3736	-	-	-	-	-	-	-	-	-
Penbased	1143	1143	1144	1055	1144	1055	1056	1142	1055	1055	-
Satimage	1533	703	1358	626	707	1508	-	-	-	-	-

Задачи *Page-Blocks*, *German Credit*, *Setimage* и *Magic* имеют большой дисбаланс распределения объектов по классам. Сравнение *F*-мер, полученных в результате тестирования разработанных алгоритмов, приведено в таблице 3.10.

Таблица 3.10 – Сравнение *F*-мер

	SCGP+UR	*ANNEA+APS*	*ANNEA+APS+IS*
Magic	0.822	0.843	0.858
Page-Blocks	0.795	0.831	0.853
Australian credit	0.852	0.894	0.915
German credit	0.805	0.813	0.827
Texture	0.931	0.982	0.987
Twonorm	0.829	0.827	0.832
Segment	0.915	0.96	0.98
Ring	0.916	0.968	0.97
Penbased	0.92	0.972	0.978
Satimage	0.942	0.976	0.984

Разработанный эволюционный алгоритм автоматического формирования ИНС позволяет оказывать улучшающее воздействие на *F*-меру в задачах классификации. Алгоритм является инвариантным в смысле оптимизируемого критерия, что, безусловно, является полезной модификацией, т.к. зачастую

конечный исследователь желает использовать различные метрики для оценки эффективности модели.

Представим сравнение алгоритма *ANNEA+APS+IS* с известными из научной литературы аналогами [97, 16, 82, 4] решения задач классификации. Критерий эффективности $(1-F)\cdot 100$, где F рассчитывается по формуле (3.7).

Таблица 3.11 – Сравнение разработанного алгоритма с аналогами на задачах классификации

Задача	*ANNEA+IS*	*HEFCA_IS*	*GP-Coach*	*IVFS-Amp*	*FARC-HD*
Magic	**14.72**	15.08	20.18	20.82	15.49
Page-Blocks	4.23	**3.25**	8.77	5.84	4.99
Texture	7.27	**4.45**	-	-	7.11
Twonorm	**4.24**	4.81	15.17	-	4.72
Segment	**5.10**	5.19	24.04	-	-
Ring	**4.99**	5.08	-	16.89	5.92
Penbased	**3.55**	3.81	17.80	21.73	3.96
Satimage	13.82	12.93	27.50	-	**12.68**

Сравнение на задачах регрессии производится с использованием *MSE* критерия. Результаты аналогов взяты с репозитория *KEEL* [5].

Таблица 3.12 – Сравнение разработанного алгоритма с аналогами на задачах регрессии

Задача	*ANNEA+IS*	*TSK-IRL*	*Linear-LMS*	*ANFIS-SUB*	*LEL-TSK*	*METSK-HD*
Forest Fires	317	-	2013	**204.8**	1407	5587
Weather	1.06	-	1.2	**0.8**	1.6	1.2
Concrete	26.9	**19.1**	54.7	188.2	31.4	23.8
Pole Telecom.	67.4	-	465	131.7	-	**61**
House	7.9	-	10.4	**7.6**	-	8.6

Разработанный алгоритм решения задач классификации выигрывает у аналогов на большинстве из представленных задач. На задачах регрессии показатели не столь высоки, однако дальнейшее применение в коллективных нейросетвых моделях позволит превзойти указанные специализированные алгоритмы по точности. На настройку алгоритма не затрачиваются дополнительные вычислительные ресурсы, что является его достоинством.

В заключение главы покажем результаты сравнения разработанного алгоритма с аналогами на описанной задаче распознавания эмоций по звуковому сигналу. В таблице 3.13 представлены результаты сравнения известных аналогов, а также базовой версии алгоритма (*SCGP+UR*) и разработанной (*ANNEA+APS+IS*). Более подробное сравнение представлено в приложении Г.

Таблица 3.13 – Сравнение с известными аналогами на задаче распознавания эмоций

Алгоритм	Точность	Алгоритм	Точность
ANNEA+APS+IS	0.716	*SCGP+UR*	0.710
Naive_Bayes	0.608	*Linear Regression*	0.640
Fast_Large_Margin	0.610	*W-PLSClassifier*	0.650
Linear_Discriminant_Analysis	0.620	*Logistic_Regression*	0.670
W-MultilayerPerceptron	0.660	*W-LMT*	0.670
W-FT	0.670	*W-LogisticBase*	0.670
Neural Net	0.701	*W-GaussianProcesses*	0.650

Применение комбинации разработанных и реализованных модификаций позволило повысить точность решения реальной задачи классификации.

Решение задачи экологического прогнозирования представлено в приложении Д. Ниже приведена свободная таблица 3.14, позволяющая оценить качество полученного решения базовой и модифицированной версии алгоритма, а также произвести сравнение с аналогами, реализованными в среде *RapidMiner*. Критерий эффективности – *MPE*.

Таблица 3.14 – Сводная таблица результатов экологического прогнозирования

	Out 1	*Out 2*	*Out 3*	*Out 4*	*Out 5*	*Out 6*	*Out 7*	*Out 8*	*Out 9*	*Out 10*
Базовое решение	15.7	**2.7**	**0.92**	**1.76**	**2.9**	1.65	10.7	10.7	12.5	15.7
SCGP+UR	9.77	12.69	6.08	4.18	6.28	1.50	0.50	5.29	4.68	10.5
ANNEA+APS	**8.82**	12.19	5.38	2.34	5.27	**0.95**	**0.20**	**4.75**	**0.75**	**8.10**
Mult. regression	9.84	12.78	6.29	4.27	6.58	1.51	0.52	5.34	4.87	10.58
Decision tree	10.23	13.76	6.36	4.59	6.74	1.53	0.54	5.55	4.69	11.1
Random forest	10.68	13.83	6.22	4.35	6.67	1.57	0.53	5.74	4.97	11.09
Neural net	10.25	12.21	5.78	4.19	6.34	1.45	0.5	5.06	4.81	10.52
SVM	9.92	12.92	6.17	4.2	6.35	1.52	0.51	5.31	4.74	10.68

Задача экологического прогнозирования является зависимой от времени, поэтому для каждой из прогнозируемых величин был подобран оптимальный

размер временного лага. Далее задача решалась как статическая. Базовое решение предполагает приравнивание прогнозируемому значению в момент времени *t* предыдущее (в момент времени *t*-1).

Таким образом, при помощи разработанного алгоритма (*ANNEA+APS*) удалось прогнозировать 6 из 10 представленных характеристик. В таблице 3.14 отслеживается полезность разработанных в работе модификаций, т.к. они позволили улучшить критерий *MPE* базовой модели *SCGP+UR*. Неудовлетворительное прогнозирование четырех характеристик может быть следствием малого объема выборки. Для получения более достоверных моделей прогноза были сделаны рекомендации по сбору более подробной статистики.

ВЫВОДЫ

В главе было рассмотрено применение нейросетевых технологий анализа данных для решения задач прогнозирования, аппроксимации и классификации.

Были представлены и программно реализованы 2 эволюционных алгоритма автоматического проектирования искусственных нейронных сетей, а именно алгоритм генетического программирования и генетический алгоритм. К реализованным алгоритмам были применены полезные модификации, позволившие на тестовых и практических задачах снизить ошибку моделирования и затрачиваемые вычислительные ресурсы. Выявлены полезные комбинации модификаций, а также указаны рекомендуемые параметры алгоритмов, позволяющие исследователю получать точные модели без ручной настройки управляющих параметров. Показана инвариантность алгоритмов адаптации относительно формы критерия эффективности. Сравнение с аналогами показало полезность реализованного алгоритма.

Разработанный алгоритм способен более точно решать задачи интеллектуального анализа данных большой размерности и требует меньшего количества вычислительных ресурсов.

ГЛАВА 4. РАЗРАБОТКА КОЛЛЕКТИВНОГО НЕЙРО-ЭВОЛЮЦИОННОГО АЛГОРИТМА ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

4.1 История развития и методы формирования коллективов интеллектуальных информационных технологий

В подавляющем большинстве случаев при решении задач анализа данных, классификации или регрессии используется какая-то одна технология анализа данных. Однако достаточно давно, ещё в работе [45] было замечено, что так называемые ансамбли нейронных сетей могут показывать лучшие результаты. В частности, решение задачи несколькими отдельными моделями, одного или разных типов, позволяет повысить обобщающую способность.

Простейшими методами формирования коллектива являются методы голосования и взвешенного голосования. Простое голосование иногда используется для задач классификации, в этом случае решение принимается по принципу большинства. Взвешенное голосование также учитывает степень доверия каждой части ансамбля при решении задачи. Более сложными методами являются комитетные системы [125], а также метод, предложенный в [55] Джорданом и Якобсом, основанный на идее определения областей компетенции.

В 1988 году Кернсом и Валиантом [70, 58] была предложена основная идея бустинга (boosting), которая заключается в объединении нескольких слабых моделей в более сильную коллективную модель. В последующем данная идея значительно изменила область машинного обучения. Большинство алгоритмов бустинга использует итеративное добавление новых слабых членов комитета для повышения качества классификации. Известными разновидностями бустинга являются *AdaBoost* [37] и *xgboost*.

Другой популярный метод построения коллективов или ансамблей для решения задач анализа данных является т.н. бэггинг (*bagging, bootstrap*

aggregating). Впервые данный метод был предложен Брейманом в 1994 году, опубликован в 1996 [20]. Идея бэггинга заключается в создании подвыборок из начальной выборки, измерения в которых выбраны случайным образом с последующим обучением множества моделей, каждая из которых строится на отдельной выборке. Для большинства методов машинного обучения, за исключением метода *K* ближайших соседей, бэггинг позволяет существенно повысить обобщающие способности алгоритмов.

Кроме того, существуют другие методы повышения качества решения задач анализа данных, например, в случае использования нейронных сетей, предлагается использование различных целевых функций [43], или же специфические методы инициализации моделей [68].

В общем и целом, большинство методов построения коллективов нацелены на то, чтобы члены коллектива исправляли ошибки друг друга. Это отражено в [56, 104]. При этом коллективы алгоритмов решения задачи объединяют различные технологии, которые самостоятельно способны решать задачу (в этом смысле база нечетких правил коллективом не является).

Важным моментом при формировании коллектива является не только выбор его членов и их обучение, но также и способ объединения в коллектив. Бустинг и бэггинг являются методами, которые обучают членов коллектива независимо друг от друга, в этом смысле они не производят искусственную специализацию членов коллектива [15]. Альтернативой является, например, перераспределение обучающих примеров между обучающими методами для членов коллектива, приводящие к специализации их на определенной части задачи (например, иерархичные смеси экспертов [57]). Кроме того, возможен вариант, при котором члены коллектива и способ их комбинирования настраивается одновременно как, например, в [118].

Одним из современных методов построения коллективов является использование эволюционных алгоритмов, в частности, алгоритма генетического программирования [89, 92]. Данный метод комбинирования решений отдельных компонентов отличается тем, что при его использовании компоненты могут

объединяться в коллектив и использованием всевозможных математических операций, взвешиваться или же исключаться из коллектива при необходимости. Данный подход позволяет производить автоматическую настройку коллектива без привлечения значительных знаний со стороны исследователя.

4.2 Адаптивный эволюционный алгоритм формирования коллективов искусственных нейронных сетей

В работе предлагается проектирование различных коллективов с использованием разработанного в главе 2 алгоритма ГП. Терминальное множество данного алгоритма представляет набор искусственных нейронных сетей, а также различных настраиваемых констант. Функциональное множество – арифметические операции (+,–,×,÷) и математические функции (*sin*, *cos*, *ln* и др). Обязательным условием является замкнутость, а значит в алгоритме предусмотрено отсутствие деления на ноль, взятие логарифма от отрицательного числа и пр. Степенные функции не используются в алгоритме, т.к. влекут слишком стремительное увеличение значений, провоцируя ошибку программной системы. Пример коллектива из трех нейронных сетей, закодированного деревом, представлен на рисунке 4.1.

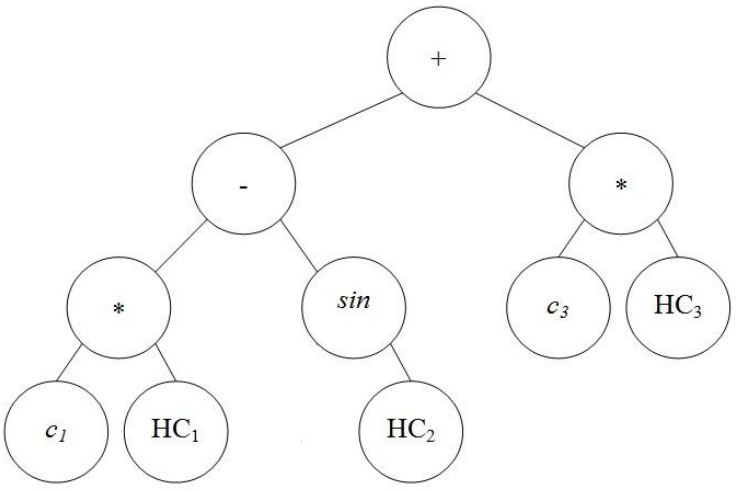

Рисунок 4.1 – Пример кодирования коллектива ИНС в ГП

На рисунке 4.1 итоговое выражение вычисляется по формуле $C_1 \cdot HC_1 - \sin(HC_2) + C_3 \cdot HC_3$. Здесь HC_i означает выход i-го члена коллектива. C_i – некоторые вещественные константы, настраивающиеся оптимизационными процедурами. Описанный алгоритм позволяет легко генерировать известные виды коллективов. На рисунке 4.2 представлен метод взвешенного голосования.

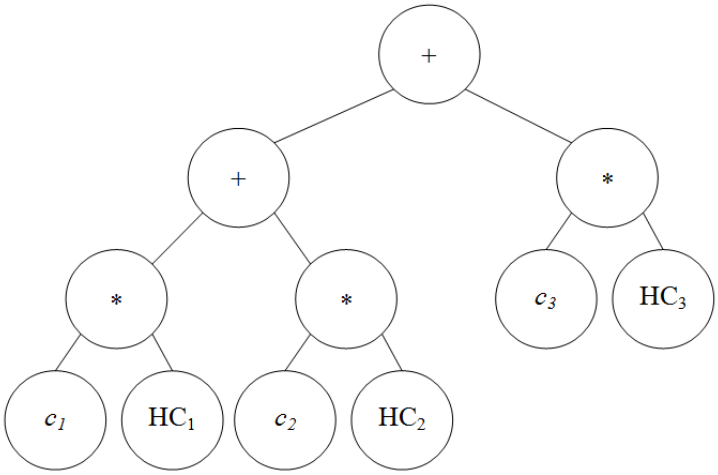

Рисунок 4.2. – Пример кодирования метода взвешенного голосования в ГП

Для создания эффективных коллективов необходимо поддерживать разнообразие внутри множества сгенерированных ИНС. Поддержание разнообразия позволяет снизить эффект переобучения итоговой модели. В работе предлагается следующий алгоритм отбора нейронных сетей в коллектив:

1. Генерируется k нейросетей с помощью как ГА так и ГП;

2. Каждой нейросети ставится в соответствие вектор $\overline{X} = \{x_1, x_2, ..., x_m\}$, показывающий ошибку на каждом из измерений выборки данных; m – число измерений в выборке. Так, если мы имеем дело с задачей классификации единица на i-ой позиции данного вектора может говорить об ошибочно принятом решении текущей ИНС, а ноль – о правильном. В случае решения задачи регрессии в таких позициях может стоять, например, разница между истинным и полученным при помощи ИНС значением.

3. Выбирается ИНС с минимальной суммой по вектору $\overline{X}$ (в случае регрессии – сумма по модулю);

4. Далее, руководствуясь заранее определенной метрикой и максиминным критерием $\max_i(\min_j(d(\overline{X}_i, \overline{X}_j)))$ выбираются остальные ИНС. Здесь $i \in ENS, j \in PL$. ENS – множество уже отобранных в коллектив ИНС, PL – множество претендентов в коллектив. Примером выбираемых метрик могут служить метрика Хемминга для задачи классификации, а также критерий *MSE* для решения задач восстановления регрессии;

5. Пункт 4 выполняется до тех пор, пока не отобрано необходимое количество ИНС, либо метрика $d(\overline{X}_i, \overline{X}_j)$ перестала превышать некоторый порог «разнообразия» B для любых i и j.

Указанная процедура позволяет производить отбор сетей для последующего формирования коллективов.

Необходимость применения описанного алгоритма дополнительно обусловлена проблемой «перестановок» в эволюционных алгоритмах. Проблема заключается в существовании индивидов с одинаковым фенотипом и разным генотипом. Простейшим примером является перестановка двух ветвей дерева:

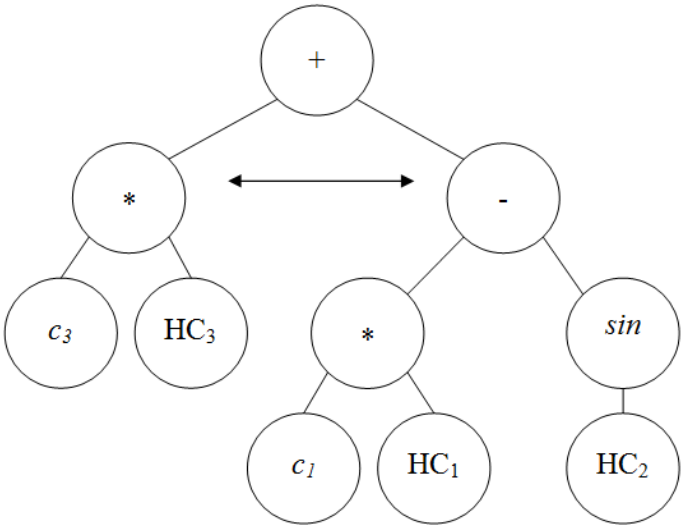

Рисунок 4.3 – Проблема «перестановок» в ГП

Если сравнивать рисунок 4.1 и 4.3 можно заметить, что в деревьях закодирован один и тот же коллектив. Однако с точки зрения генотипа данные деревья являются разными. В описанной процедуре метрика $d(\overline{X}_i, \overline{X}_j)$ между такими

деревьями равна нулю, а следовательно одно из них не будет отобрано в коллектив.

Опишем итоговый алгоритм при помощи блок-схемы (рисунок 4.4):

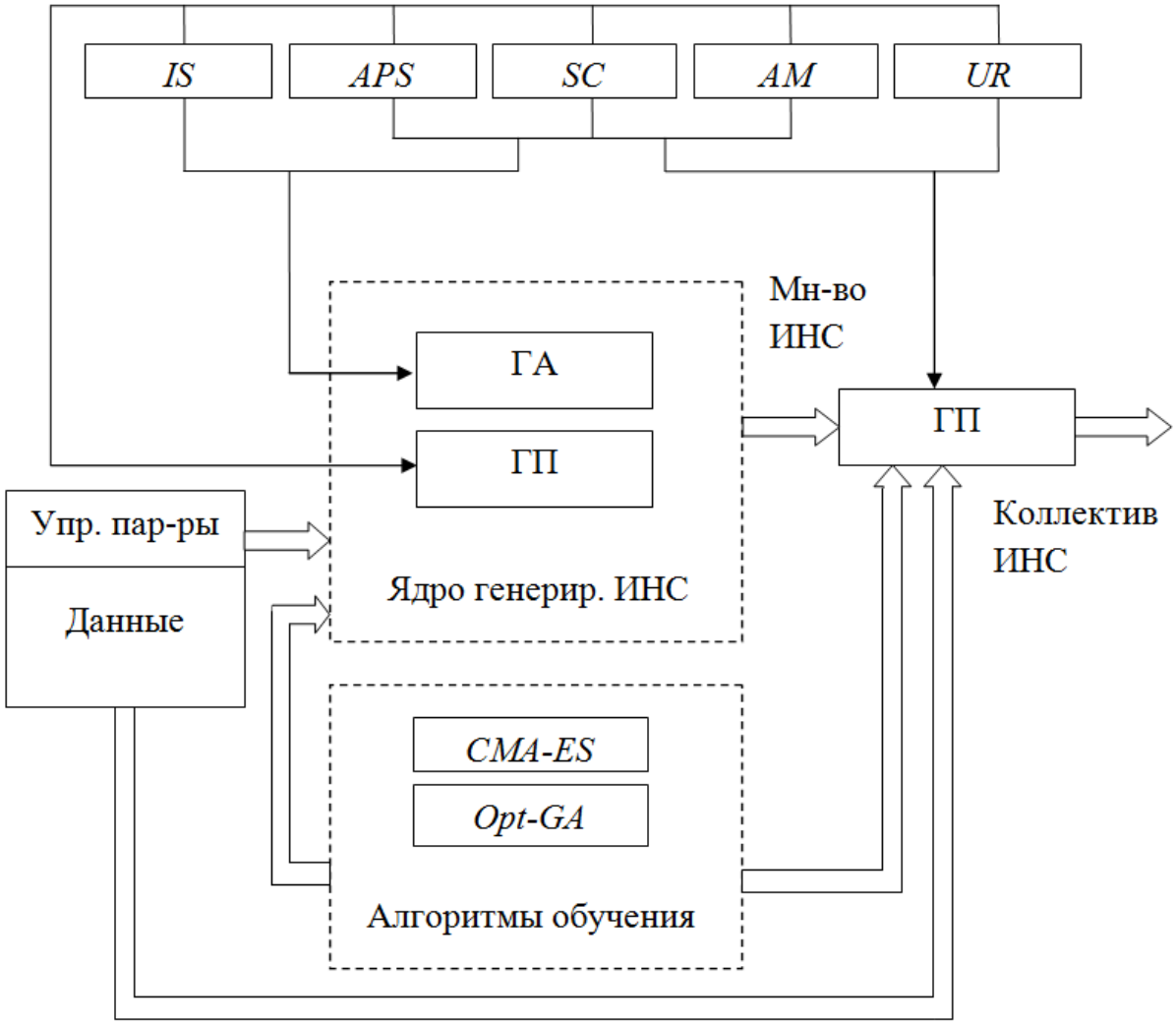

Рисунок 4.4 – Блок схема разработанного коллективного нейро-эволюционного алгоритма

В рисунке 4.4 используются следующие сокращения: *IS* – селекция обучающих примеров, *APS* – адаптация размера популяции, *SC* – самоконфигурирование, *AM* – адаптивная мутация, *UR* – равномерное скрещивание. Наиболее важным структурным элементом является блок формирования ИНС. Он включает в себя 2 алгоритма, разработанных в главе 3. Все реализованные и разработанные модификации (указаны в верхней части диаграммы) применяются к алгоритму ГП. Аналогично для ГА, отличие составляет отсутствие необходимости в алгоритме равномерного скрещивания (*UR*). Обучение ИНС выполняется

алгоритмом *CMA-ES* [52], а в случае большой размерности разработанным в главе 2 ГА. Проектирование ИНС начинается с загрузки данных и управляющих параметров. Вычислительные ресурсы внутри блока генерирования распределяются равномерно среди ГА и ГП. Так, если пользователь установил максимальное число вычислений целевой функции равное 100, то оба алгоритма смогут вычислить пригодность сети ровно 50 раз. При формировании достаточного количества нейронных сетей к решению задачи подключается алгоритм ГП проектирующий коллектив.

4.3 Исследование эффективности разработанного коллективного нейро-эволюционного алгоритма

Исследование эффективности реализованного алгоритма проводилось на задачах, описанных в главе 3. При тестировании использовалась процедура 10-частной кросс-валидации с 20 запусками на каждой из частей. В таблице 4.1 представлены различные вариации алгоритма ГП, использующегося для автоматического формирования коллективов. Отметим, что для отбора нейронных сетей использовался алгоритм поддержания разнообразия, описанный в пункте 4.2 главы. В таблице представлена ошибка классификации и *MSE* критерий.

Разработанные алгоритмы (*SCGP+UR AM+APS)* и (*SCGP+UR AM)* имеют наибольшую эффективность среди представленных. Далее при сравнении с аналогами, будем использовать сокращенную аббревиатуру *ENS_ANN*, под которой будем понимать алгоритм *SCGP+UR AM+APS*. Его выбор обусловлен точностью моделирования и сниженными вычислительными ресурсами на формирование модели. Из таблицы 4.1 очевидна полезность использования самоконфигурирования совместно с генетическим оператором равномерного скрещивания. GP_{best} и *SCGP* имеют равную ошибку классификации, однако в реальных задачах лучшая конфигурация неизвестна. Алгоритм GP_{aver} проигрывает всем из представленных модификаций.

Таблица 4.1 – Сравнение эффективности самоконфигурирующихся и базовых версий алгоритмов

	GP_{aver}	GP_{best}	*SCGP*	*SCGP+UR*	*SCGP+UR AM*	*SCGP+UR AM+APS*
Классификация						
Magic	14.76	14.366	14.325	14.251	14.04	13.5
Page-Blocks	4.21	3.99	4.18	4.08	4.06	4.01
Australian credit	11.16	10.66	11.03	10.62	9.54	9.86
German credit	23.12	21.80	21.81	20.71	20.55	19.97
Texture	8.05	7.88	8.00	8.03	7.68	7.91
Twonorm	5.92	5.74	5.66	5.59	5.35	5.49
Segment	6.14	5.96	6.00	5.84	5.66	5.61
Ring	6.53	6.48	6.37	6.40	6.22	6.23
Penbased	8.1	8.08	8.024	8.03	8.03	7.87
Satimage	14.94	14.84	14.63	14.64	14.53	14.12
Регрессия						
Forest Fires	311	307	299	302	269	272
Weather	1.04	0.97	0.94	0.84	0.87	0.83
Concrete	24.9	23.5	22.7	22.3	21	21.7
Pole Telecom.	68.5	63.1	62.9	62.4	60.1	58.1
House	8.28	8.19	8.21	8.23	8.17	8.2

Приведем в качестве примера коллективы, сгенерированные для задачи прогнозирования прочности бетона на сжатие (*Concrete*). Всего для формирования коллективов использовалось 20 ИНС, из которых процедурой поддержания разнообразия удалялись в среднем 4. Отметим, что на количество оставшихся нейронных сетей влияет выбор порога *B*. Настройка данного параметра проста, т.к. при выборе значения исследователю сразу становится известным количество ИНС, попадающих в коллектив.

$$Ensemble_1 = C_1 \cdot ANN_1 + C_2 \cdot ANN_4 - C_3 \cdot ANN_7 \quad (4.1)$$

$$Ensemble_2 = C_1 \cdot ANN_1 + C_2 \cdot \frac{ANN_5 \cdot ANN_6}{ANN_2} \quad (4.2)$$

Формулы 4.1 и 4.2 являются различными вариациями решения задачи (*Concrete*). Если формула 4.1 является вариацией взвешенного голосования, то формула 4.2.

имеет более сложную структуру. При этом среднеквадратичная ошибка первого коллектива приблизительно равна 22, а второго 21.5. Это говорит о практической полезности более сложных структур коллективов. Алгоритм ГП генерирует разнообразные коллективы, среди которых, хоть и очень редко, можно выделить стандартные структуры типа 4.1.

Произведем сравнение с аналогами, используя критерий, рассчитывающийся через *F*-меру: $I = (1 - F) \cdot 100$.

Таблица 4.2 – Сравнение эффективности с известными классификационными алгоритмами

Задача	*ANNEA+IS*	*ENS+ANN*	*HEFCA_IS*	*GP-Coach*	*IVFS-Amp*	*FARC-HD*
Magic	14.72	**14.3**	15.08	20.18	20.82	15.49
Page-Blocks	4.23	4.07	**3.25**	8.77	5.84	4.99
Texture	7.27	7.29	**4.45**	-	-	7.11
Twonorm	4.24	**4.21**	4.81	15.17	-	4.72
Segment	**5.10**	5.13	5.19	24.04	-	-
Ring	4.99	**4.73**	**5.08**	-	16.89	5.92
Penbased	3.55	**3.19**	3.81	17.80	21.73	3.96
Satimage	13.82	13.46	12.93	27.50	-	**12.68**

Таблица 4.3 – Сравнение с аналогами на задачах банковского скоринга

Название алгоритма	*Australian credit*	*German credit*	Название алгоритма	*Australian credit*	*German credit*
SC_GP	0.9022	0.7950	*Bayesian approach*	0.8470	0.6790
MGP	0.8985	0.7875	*Boosting*	0.7600	0.7000
2SGP	0.9027	0.8015	*Bagging*	0.8470	0.6840
GP	0.8889	0.7834	*RSM*	0.8520	0.6770
Fuzzy classifier	0.8910	0.7940	*CCEL*	0.8660	0.7460
C4.5	0.8986	0.7773	*CART*	0.8744	0.7565
LR	0.8696	0.7837	*MLP*	0.8986	0.7618
k-NN	0.7150	0.7151	***ENS_GP***	0.9014	0.8003

Результаты таблицы 4.2 говорят о полезности коллективного подхода к решению задач классификации. Разработанный алгоритм превзошел по эффективности представленные аналоги на 5 задачах из 8. Задачи банковского скоринга оценивались по такому критерию как точность классификации. В таблице 4.3 показано, что коллективный алгоритм занимает 3 место на задаче *Australian credit*

и 2-е место на задаче *German credit*, проигрывая лишь узкоспециализированному алгоритму *2SGP*. Если сравнивать с результатами, представленными в главе 3, можно заметить существенное увеличение точности классификации.

Коллективный подход позволил приблизиться к лидирующим алгоритмам на задачах восстановления регрессии. В таблице ниже приведены значения *MSE* критерия. Алгоритм *ANFIS-SUB* является гибридным, объединяющим в себе достоинства ИНС и систем на нечеткой логике. Данный алгоритм весьма сложен в эксплуатации, что нельзя сказать про разработанный.

Таблица 4.4 – Сравнение эффективности на задачах регрессии

Задача	*ANNEA+IS*	*ENS+ANN*	*TSK-IRL*	*Linear-LMS*	*ANFIS-SUB*	*LEL-TSK*	*METSK-HD*
Forest Fires	317	272	-	2013	**204.8**	1407	5587
Weather	1.06	0.83	-	1.2	**0.8**	1.6	1.2
Concrete	26.9	21.7	**19.1**	54.7	188.2	31.4	23.8
Pole Telecom.	67.4	**58.1**	-	465	131.7	-	61
House	7.9	8.2	-	10.4	**7.6**	-	8.6

Коллективный подход позволил увеличить точность при решении задачи распознавания эмоций.

Таблица 4.5 – Решение задачи распознавания эмоций по звуковому сигналу

Алгоритм	Точность	Алгоритм	Точность
ENS+ANN	**0.725**	*W-LinearRegression*	0.632
Naive_Bayes	0.608	*Linear Regression*	0.640
Fast_Large_Margin	0.610	*W-PLSClassifier*	0.650
Linear_Discriminant_Analysis	0.620	*Logistic_Regression*	0.670
W-MultilayerPerceptron	0.660	*W-LMT*	0.670
W-FT	0.670	*W-LogisticBase*	0.670
Neural Net	0.701	*W-GaussianProcesses*	0.650

Увеличение эффективности по сравнению с нейронной сетью, реализованной в среде разработки *RapidMiner*, составило 2.5%. Сравнение с подходами главы 3 показало прирост точности в 0.9%. Такой результат можно признать

удовлетворительным лишь в том случае, когда точность является основополагающим критерием.

Результат решения задачи экологического прогнозирования также незначительно изменился:

Таблица 4.6 – Сводная таблица результатов экологического прогнозирования

	Out 1	*Out 2*	*Out 3*	*Out 4*	*Out 5*	*Out 6*	*Out 7*	*Out 8*	*Out 9*	*Out 10*
Базовое решение	15.7	**2.7**	**0.92**	**1.76**	**2.9**	1.65	10.7	10.7	12.5	15.7
SCGP+UR	9.77	12.69	6.08	4.18	6.28	1.50	0.50	5.29	4.68	10.5
ANNEA+APS	8.82	12.19	5.38	2.34	5.27	0.95	0.20	4.75	0.75	8.10
ENS_ANN	**8.53**	12.2	5.05	2.32	5.14	**0.94**	**0.20**	**4.56**	**0.79**	**7.94**
Mult. regression	9.84	12.78	6.29	4.27	6.58	1.51	0.52	5.34	4.87	10.58
Decision tree	10.23	13.76	6.36	4.59	6.74	1.53	0.54	5.55	4.69	11.1
Random forest	10.68	13.83	6.22	4.35	6.67	1.57	0.53	5.74	4.97	11.09
Neural net	10.25	12.21	5.78	4.19	6.34	1.45	0.5	5.06	4.81	10.52
SVM	9.92	12.92	6.17	4.2	6.35	1.52	0.51	5.31	4.74	10.68

В таблице 4.6 можно наблюдать небольшой прирост точности, который не меняет общей картины решения задачи. Характеристики *Out2-Out4* по-прежнему плохо поддаются прогнозу. Уточнение выборки данных, ее расширение и доработка технологии может стать ключом разрешения указанной проблемы.

ВЫВОДЫ

Представлен и программно реализован эволюционный алгоритм автоматического генерирования коллективов искусственных нейронных сетей. Его отличительной особенностью является использование самоконфигурирования, адаптации уровня мутации и размера популяции, равномерного скрещивания. Указанные модификации позволили существенно повысить точность моделирования на задачах классификации и регрессии. Улучшение было достигнуто в сравнении с базовой моделью и известными аналогами.

Решение практических задач анализа данных позволило незначительно увеличить точность моделирования. Достоинством разработанного алгоритма является минимизация настраиваемых параметров, что позволяет, не обладая экспертными знаниями в области эволюционных вычислений, строить эффективные нейросетевые модели, что позволяет существенно расширить область применения соответствующих интеллектуальных информационных технологий.

Дальнейшее развитие разработанного подхода предполагает проектирование более сложных структур анализа данных, таких как *lstm* сети, и привлечение в формируемый коллектив альтернативных технологий, что нетрудно сделать, благодаря специальным свойствам разработанных подходов.

ЗАКЛЮЧЕНИЕ

Цель диссертационной работы достигнута, поставленные задачи решены. Получены следующие результаты:

1. Разработан новый метод адаптивного управления уровнем мутации в эволюционных алгоритмах, отличающийся от известных способом расчета вероятности мутации гена.

2. Разработан новый метод адаптивного управления размером популяции в эволюционных алгоритмах, отличающийся от известных применением динамического показателя успешности подпопуляции.

3. Разработаны и реализованы эволюционные алгоритмы моделирования и оптимизации, отличающиеся от известных использованием эффективной комбинации модификаций самоконфигурирования, адаптации и селекции обучающих примеров.

4. Разработаны новые адаптивные эволюционные алгоритмы формирования искусственных нейронных сетей, отличающиеся от известных методом настройки конфигурации и параметров, а также применением процедур отбора информативных признаков и обучающих примеров.

5. Разработан новый метод построения коллективов искусственных нейронных сетей, отличающийся от известных применением адаптивного алгоритма генетического программирования с механизмом контроля разнообразия внутри коллектива и возможностью использования альтернативных методов анализа данных.

6. Эффективность реализованных в виде программных систем алгоритмов продемонстрирована на репрезентативных множествах тестовых задач и подтверждена успешным применением при решении ряда практических задач.

Таким образом, в диссертационной работе разработаны, исследованы и апробированы новые самоконфигурируемые адаптивные эволюционные алгоритмы формирования искусственных нейронных сетей и их коллективов, что

является существенным вкладом в теорию и практику обработки информации и интеллектуального анализа данных.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Akhmedova Sh., Semenkin E., Gasanova T., Minker W. Co-Operation of Biology Related Algorithms for Support Vector Machine Automated Design. – In proceedings of the International Conference on Engineering and Applied Sciences Optimization (OPT-i'2014), pp. 1831-1837.

2. Akhmedova, Sh., Semenkin, E. Co-Operation of Biology Related Algorithms Meta-Heuristic in ANN-Based Classifiers Design. – In proceedings of the World Congress on Computational Intelligence (WCCI'14), pp. 867-873. – 2014.

3. Akhmedova, Sh., Semenkin, E. Co-Operation of Biology Related Algorithms. – In proceedings of 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 2207-2214. – 2013.

4. Alcala-Fdez J., Alcala R., Herrera F., A fuzzy association rulebased classification model for high-dimensional problems with genetic rule selection and lateral tuning. – IEEE Trans. Fuzzy Syst., vol. 19, no. 5, pp. 857–872. – Oct. 2011.

5. Alcalá-Fdez J., Sánchez L., Garcia S., Jesus M. J., Ventura S., Garrell J.M., Otero J., Romero C., Bacardit J., Rivas V.M., Fernández J. C., Herrera F. KEEL: A software tool to assess evolutionary algorithms for data mining problems. – Soft Comput., vol. 13, no. 3. – p. 307–318. – Feb. 2009.

6. Arabas J., Michalewicz Z., and Mulawka J. GAVaPS – a genetic algorithm with varying population size // Proc. of the First IEEE Conf. on Evolutionary Computation, pp. 73–78, NJ, 1994. IEEE Press.

7. Asuncion A., Newman D. UCI machine learning repository. – University of California, Irvine, School of Information and Computer Sciences [Электронный ресурс]. Режим доступа: http://www.ics.uci.edu/~mlearn/MLRepository.html – 2007

8. Awad N. H., Ali M. Z., Liang J. J., Qu B. Y., Suganthan P. N. Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization // Technical Report of

Nanyang Technological University, Jordan University of Science and Technology, Zhengzhou University.

9. Back T. and Schutz M. Intelligent Mutation Rate Control in Canonical Genetic Algorithms // Proc. of the International Symposium on Methodologies for Intelligent Systems, pp. 158-167, 1996.

10. Back T. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm // In R. Manner and B. Manderick editors, Parallel Problem Solving from Nature, pp. 85–94. North Holland, Amsterdam, 1992.

11. Back T., Eiben A. E., and van der Vaart N. A. L. An empirical study on GAs “without parameters” // Parallel Problem Solving from Nature, PPSN VI, pp. 315–324, 2000.

12. Back Th. and Schutz M. Intelligent mutation rate control in canonical genetic algorithms // In ISMIS, pages 158–167, 1996.

13. Baeck T., Hammel U., and Schwefel H.-P. Evolutionary Computation: Comments on the History and Current State / IEEE Transactions On Evolutionary Computation, Vol. 1, No. 1. – 1997.

14. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D. (1997), Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications, Morgan Kaufmann

15. Bauer, E., Kohavi, R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants // Machine Learning. – 36 (1999) pp. 105–142.

16. Berlanga F. J., Rivera A. J., del Jesus M. J., Herrera F., GP-COACH: Genetic programming-based learning of compact and accurate fuzzy rulebased classification systems for high-dimensional problems. – Inf. Sci., vol. 180, no. 8, pp. 1183–1200. – Apr. 2010.

17. Bjorklund A.F., Modak B.K., Jiminez P.A., and etc. CodeBlocks Manual. [Электронный ресурс] – 2017. Режим доступа: www.codeblocks.org/docs/manual_en.pdf

18. Box, G. E. P. Evolutionary operation: A method for increasing industrial productivity / Appl. Statistics, vol. VI, no. 2, pp. 81–101, 1957.

19. Brain Z. and Addicoat M. Using Meta-Genetic Algorithms to tune parameters of Genetic Algorithms to find lowest energy Molecular Conformers // Proc. of the Alife XII Conference, Odense, Denmark, 2010

20. Breiman, Leo (1996). "Bagging predictors". Machine Learning. 24 (2): 123–140.

21. Bremermann, H. J. Optimization through evolution and recombination // Self-Organizing Systems, M. C. Yovits et al., Eds. Washington, DC: Spartan, 1962.

22. Brester Ch.Yu., Semenkin E.S. Development of adaptive genetic algorithms for neural network models multicriteria design // Вестник Сибирского государственного аэрокосмического университета им. академика М.Ф. Решетнева. 2013. № 4 (50). С. 99-103.

23. Bujok P., Tvrdík J. Adaptive Differential Evolution: SHADE with Competing Crossover Strategies // Artificial Intelligence and Soft Computing, pp 329-339, 2015

24. Burakov S.V., Semenkin E.S. Solving variational and Cauchy problems with self-configuring genetic programming algorithm // International Journal of Innovative Computing and Applications. 2013. Т. 5. № 3. С. 152-162.

25. Burkhardt F., Paeschke A., Rolfes M., Sendlmeier W.F., and Weiss B. A Database of german emotional speech. In Interspeech, pages 1517-1520, 2005.

26. Celebi M. E., Kingravi H. A., Uddin B., Iyatomi H., Aslandogan Y. A., Stoecker W. V., Moss R. H. A methodological approach to the classification of dermoscopy images. – Comput.Med. Imag. Grap., vol. 31, no. 6, pp. 362–373. – 2007.

27. Cieslak D., Chawla N., Striegel A., Combating imbalance in network intrusion datasets. – in IEEE Int. Conf. Granular Comput., pp. 732– 737. – 2006.

28. de Jong K.A.D. An analysis of the behavior of a class of genetic adaptive systems // Ph.D. thesis, University of Michigan, 1975

29. Deb K., Pratap A., Agarwal S., Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2), 2002, pp. 182-197.

30. Dhenakaran Dr.S.S. EMPLOYING REVERSE POLISH NOTATION IN ENCRYPTION // Advanced Computing: An International Journal (ACIJ), Vol.2, No.2, March 2011

31. Eiben A. E., Marchiori E., and Valko V. A. Evolutionary algorithms with onthe-fly population size adjustment // Parallel Problem Solving from Nature PPSN VIII, pp. 41–50. Springer, 2004.

32. Eiben A.E., Smith J.E. Introduction to evolutionary computing // Springer, Berlin Heidelberg, 2003

33. Flores B. A pragmatic view of accuracy measurement in forecasting // Omega, Vol. 14, Issue 2. - Elsevier 1986. P. 93-98.

34. Fogel, D. B. An evolutionary approach to the traveling salesman problem // Biological Cybern., vol. 60, pp. 139–144, 1988.

35. Fogel, D. B. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence / Piscataway, NJ: IEEE Press, 1995.

36. Fogel, D. B. Evolving artificial intelligence // Ph.D. dissertation, Univ. of California, San Diego, 1992.

37. Freund, Yoav; Schapire, Robert E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting". Journal of Computer and System Sciences. 55: 119.

38. Fukushima, Neocognitron (1980). "A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". Biological Cybernetics. 36 (4): 93–202.

39. Goldberg D. E. The Design of Innovation – Lessons from and for Competent Genetic Algorithms // Kluwer Academic Publishers, Norwell, MA, 2002.

40. Goldberg D. E., Deb K., and Clark J. H. Genetic algorithms, noise, and the sizing of populations // Complex Systems, № 6, pp 333–362, 1992.

41. Gordon, V. S., Boehm, A. P. W., Whitley, L. D. A note on the performance of genetic algorithms on zero-one knapsack problems. In: Proc. 1994 ACM Symp. Applied Computing, E. Deaton, D. Oppenheim, J. Urban, and H. Berghel, Eds. New York: ACM, 1994, pp. 194–195.

42. Gough B. An Introduction to GCC. [Электронный ресурс] – 2004. Режим доступа: http://www.network-theory.co.uk/docs/gccintro/

43. Hampshire, J. A novel objective function for improved phoneme recognition using time-delay neural networks / J. Hampshire, A. Waibel // IEEE Transactions on Neural Networks 1 (2) (1990) pp. 216-228.

44. Hansen N, Ostermeier A (2001). Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation, 9(2) pp. 159–195.

45. Hansen, L.K. Neural network ensembles / L.K. Hansen, P. Salamon IEEE Trans. Pattern Analysis and Machine Intelligence 12 (10) (1990) pp.993-1001.

46. Haq S. and Jackson P.J.B. Machine Audition: Principles, Algorithms and Systems, chapter Multimodal Emotion Recognition, pages 398-423. IGI Global, Hershey PA, 2010.

47. Hebb, Donald (1949). The Organization of Behavior. New York: Wiley. ISBN 978-1-135-63190-1.

48. Hinterding R., Michalewicz Z., and Peachey T. C. Self-adaptive genetic algorithm for numeric functions // Parallel Problem Solving from Nature, PPSN IV, pp. 420–429, 1996.

49. Hinton, G. E. (2010). "A Practical Guide to Training Restricted Boltzmann Machines". Tech. Rep. UTML TR 2010-003.

50. Hochreiter, Sepp; Schmidhuber, Jürgen (1997-11-01). "Long Short-Term Memory". Neural Computation. 9 (8): 1735–1780.\

51. Holland, J. H. Adaptation in Natural and Artificial Systems. – Ann Arbor, MI: Univ. of Michigan Press, 1975.

52. Igel C., Wiegand S., Friedrichs F. Evolutionary Optimization of Neural Systems: The Use of Strategy Adaptation // Trends and Applications in Constructive Approximation pp 103-123. – 2005

53. Ishibuchi H. et al. Hybridization of fuzzy GBML approaches for pattern classification problems. – IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics. – 2005.

54. Ishibuchi H. et al. Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. – Fuzzy Sets and Systems. – 1997.

55. Jacobs, R. A. Adaptive mixtures of local experts / R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton // Neural Computation. – 1991. – no. 3. – Pp. 79–87.

56. Jain, A. K., Duin, R. P. W., Mao, J. Statistical pattern recognition: A review // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2000. – Vol. 22, no. 1. – Pp. 4–37.

57. Jordan, M. I., Jacobs, R. A. Hierarchical mixtures of experts and the EM algorithm // Neural Computation. – 1994. – no. 6. – Pp. 181–214.

58. Kearns, M., Valiant, L. G. Cryptographic limitations on learning Boolean formulae and finite automata // Proc. of the 21st Annual ACM Symposium on Theory of Computing. – 1989. – Pp. 433–444.

59. Khritonenko D., Stanovov V., Semenkin E. Applying an instance selection method to an evolutionary neural classifier design // IOP Conference Series: Materials Science and Engineering. V International Workshop on Mathematical Models and their Applications. – 2017. - Vol. 173.

60. Khuri, S., Baeck, T., Heitkoetter, J. An evolutionary approach to combinatorial optimization problems. In: Proc. 22nd Annu. ACM Computer Science Conf., D. Cizmar, Ed. New York: ACM, 1994, pp. 66–73.

61. Kohonen, Teuvo (1982). "Self-Organized Formation of Topologically Correct Feature Maps". Biological Cybernetics. 43 (1): 59–69.

62. Korejo I., Yang S., Li C. A Comparative Study of Adaptive Mutation Operators for Genetic Algorithms // Proc. of the VIII Metaheuristics International Conference, 2009. Hamburg, Germany.

63. Koza, J.R. (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press

64. Lachiche, N., & Flach, P. A. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. – In Proceedings of ICML’2003 pp. 416–423. – 2003.

65. LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition" (PDF). Proceedings of the IEEE. 86 (11): 2278–2324.

66. Lin L., Gen M. Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation. Soft Computing 13(2): Pp. 157–168, 2009

67. Lobo F.G. and Lima C.F.. Adaptive Population Sizing Schemes in Genetic Algorithms // Studies in Computational Intelligence (SCI) № 54, 185 –204, 2007.

68. Maclin, R. Combining the predictions of multiple classifiers: using competitive learning to initialize neural networks / R. Maclin, J.W. Shavlik // in: Proc. IJCAI-95, Montreal, Canada, Morgan Kaufmann, San Mateo, CA, 1995, pp. 524-530.

69. McCulloch, Warren; Walter Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". Bulletin of Mathematical Biophysics. 5 (4): 115–133.

70. Michael Kearns(1988); Thoughts on Hypothesis Boosting, Unpublished manuscript (Machine Learning class project, December 1988)

71. Minsky, Marvin; Papert, Seymour (1969). Perceptrons: An Introduction to Computational Geometry. MIT Press.

72. Niehaus, J., Banzhaf, W. Adaption of Operator Probabilities in Genetic Programming. – EuroGP 2001, LNCS 2038. – p. 325-336. – 2001.

73. Olsen, A. Penalty functions and the knapsack problem. In: Proc. 1st IEEE Conf. on Evolutionary Computation. Piscataway, NJ: IEEE Press, 1994, pp. 554–558.

74. Olvera-López J.A., Carrasco-Ochoa J.A., Martínez-Trinidad J.F. Prototype selection via prototype relevance. – In:Ruiz-Shulcloper J., Kropatsch W.G. CIARP 2008, LNCS5197. Habana, Cuba, pp. 153–160. – 2008.

75. Peng X. King I., Robust BMPM training based on second-order cone programming and its application in medical diagnosis. – Neural Netw., vol. 21, no. 2–3, pp. 450–457. – 2008.

76. Poli, R., Langdon, W.B. On the search properties of different crossover operators in genetic programming. In J. R. Koza, et al., editors, Genetic Programming 1998:Proceedings of the Third Annual Conference, pages 293–301, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998.

77. Poli, R., Page, J., Langdon, W.B. Smooth uniform crossover, sub-machine code GP and demes: A recipe for solving high-order boolean parity problems. In W. Banzhaf, et al., editors, Proceedings of the Genetic and Evolutionary Computation Conference, volume 2, pages 1162-1169, Orlando, Florida, USA, 1999.

78. Potter, M. A., De Jong, K. A. Cooperative coevolution: An architecture for evolving co-adapted subcomponents // Evolutionary Computation, 2000, no. 8, pp. 1–29.

79. Raicharoen T., Lursinsap C. A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm. – Pattern Recognit Lett 26(10): pp. 1554–1567. – 2005.

80. Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain". Psychological Review. 65 (6): 386–408.

81. Rumelhart D. E., Hinton G. E., Williams R. J. Learning Internal Representations by Error Propagation // Parallel Distributed Processing. Vol. 1. — Cambridge, MA: MIT Press, 1986. P. 318—362.

82. Sanz J. A., Fernandez A., Bustince H., Herrera F., Improving the performance of fuzzy rule-based classification systems with interval-valued fuzzy sets and genetic amplitude tuning. – Inf. Sci., vol. 180, no. 19, pp. 3674–3685. – Oct. 2010.

83. Schlierkamp-Voosen D. and Muhlenbein H. Adaptation of population sizes by competing subpopulations // Proceedings of 1996 IEEE International Conference on Evolutionary Computation, pages 330–335, 1996.

84. Schwefel, H.-P. Evolution and Optimum Seeking. – New York: Wiley, 1995 (Sixth-Generation Computer Technology Series).

85. Schwefel, H.-P., Rudolph, G. Contemporary evolution strategies. In: Advances in Artificial Life. 3rd Int. Conf. on Artificial Life (Lecture Notes in Artificial Intelligence, vol. 929), F. Mor´an, A. Moreno, J. J. Merelo, and P. Chac´on, Eds. Berlin, Germany: Springer, 1995, pp. 893–907.

86. Semenkin E., Semenkina M. Self-configuring genetic algorithm with modified uniform crossover operator // Lecture Notes in Computer Science. 2012. T. 7331 LNCS. № PART 1. C. 414-421.

87. Semenkin E., Semenkina M. Self-configuring genetic algorithm with modified uniform crossover operator. – Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7331 LNCS (PART 1). – p. 414-421. – 2012.

88. Semenkin E.S., Semenkina M.E. Integration of Intelligent Information Technologies Ensembles with Self-Configuring Genetic Programming Algorithm // Vestnik. Scientific Journal of Siberian State Aerospace University named after academician M. F. Reshetnev. – № 4 (44).–2012.

89. Semenkin E.S., Semenkina M.E. Integration of Intelligent Information Technologies Ensembles with Self-Configuring Genetic Programming Algorithm // Vestnik. Scientific Journal of Siberian State Aerospace University named after academician M. F. Reshetnev. – № 4 (44).–2012.

90. Shabalov A., Semenkin E., Galushin P. Automatized Design Application of Intelligent Information Technologies for Data Mining Problems // Joint IEEE Conference "The 7th International Conference on Natural Computation & The 8th International Conference on Fuzzy Systems and Knowledge Discovery" – FSKD’2011: pp. 2596-2599.

91. Shabalov A., Semenkin E., Galushin P. Integration of Intelligent Information Technologies Ensembles for Modeling and Classification // Hybrid Artificial Intelligence Systems. Lecture Notes in Computer Science, Volume 7208/2012, pp. 365-374.

92. Shabalov A., Semenkin E., Galushin P. Integration of Intelligent Information Technologies Ensembles for Modeling and Classification // Hybrid Artificial Intelligence Systems. Lecture Notes in Computer Science, Volume 7208/2012, pp. 365-374.

93. Skolicki, Z. An Analysis of Island Models in Evolutionary Computation, GECCO'05, June 25–29, 2005, Washington, DC, USA. Copyright 2005 ACM 1-59593-010-8/05/0006.

94. Smith S.K., Eiben A.E. Comparing Parameter Tuning Methods for Evolutionary Algorithms // Proc. 2009 IEEE Congress on Evolutionary Computation, 2009

95. Sokolova M., Lapalme G. A systematic analysis of performance measures for classification tasks. – Information Processing and Management 45, pp. 427–437. – 2009.

96. Stanovov V., Semenkin E., Semenkina O. Instance Selection Approach for Self-configuring Hybrid Fuzzy Evolutionary Algorithm for Imbalanced Datasets // Advances in Swarm and Computational Intelligence, LNCS 9140, 2015, pp. 451–459.

97. Stanovov V., Semenkin E., Semenkina O. Instance Selection Approach for Self-configuring Hybrid Fuzzy Evolutionary Algorithm for Imbalanced Datasets // Advances in Swarm and Computational Intelligence, LNCS 9140, 2015, pp. 451–459.

98. Stanovov V.V., Semenkin E.S., Semenkina O.E. Self-configuring hybrid evolutionary algorithm for fuzzy imbalanced classification with adaptive instance selection // Journal of Artificial Intelligence and Soft Computing Research. 2016. Т. 6. № 3. С. 173-188.

99. Stanovov V.V., Semenkin E.S., Semenkina O.E. Self-configuring hybrid evolutionary algorithm for fuzzy imbalanced classification with adaptive instance selection // Journal of Artificial Intelligence and Soft Computing Research. 2016. Т. 6. № 3. С. 173-188.

100. Storn, Rainer and Price, Kenneth. Differential Evolution — A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. (1997)

101. Suganthan P. N., Ali M. Z., Wu G., Mallipeddi R., Liang J. J., Qu B. Y. Special Session & Competitions on Real-Parameter Single Objective Optimization: [Электронный ресурс] – 2017. Режим доступа: http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2017/CEC2017.htm

102. Tanabe R. and Fukunaga A. S. Improving the Search Performance of SHADE Using Linear Population Size Reduction // Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1658-1665, 2014.

103. Tanabe R. and Fukunaga A. S. Improving the Search Performance of SHADE Using Linear Population Size Reduction // Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1658-1665, 2014.

104. Tresp, V. Committee machines // Handbook for Neural Network Signal Processing / Ed. by Y. H. Hu, J.-N. Hwang. – CRC Press, 2001.

105. Valko V. A. Self-calibrating Evolutionary Algorithms: Adaptive Population Size // Master's thesis, Free University Amsterdam, 2003.

106. Werbos P. J. Beyond regression: New tools for prediction and analysis in the behavioral sciences. — Ph. D. thesis, Harvard University, Cambridge, MA, 1974.

107. Whitley, L. D. Genetic algorithms and neural networks. In: Genetic Algorithms in Engineering and Computer Science, G. Winter, J. P´eriaux, M. Gal´an, and P. Cuesta, Eds. Chichester, UK: Wiley, 1995, ch. 11, pp. 203–216.

108. Widrow B., Hoff T., 1960: An adaptive "ADALINE" neuron using chemical "memistors", technical report

109. Wilcoxon F. Individual Comparisons by Ranking Methods. // Biometrics Bulletin 1. — 1945. — P. 80-83.

110. Wolpert, D. The Lack of A Priori Distinctions between Learning Algorithms. – Neural Computation, pp. 1341–1390. – 1996.

111. Wolpert, D.H., Macready, W.G. Coevolutionary free lunches. – IEEE Transactions on Evolutionary Computation, 9(6): pp. 721–735. – 2005.

112. Wolpert, D.H., Macready, W.G., No Free Lunch Theorems for Optimization, – IEEE Transactions on Evolutionary Computation 1, 67. – 1997.

113. Yu T.-L., Sastry K., and Goldberg D. E. Online population size adjusting using noise and substructural measurements // Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 3, pp. 2491–2498. IEEE Press, 2005.

114. Zitzler E., Laumanns M., Thiele L. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Proceedings of evolutionary

methods for design, optimization and control with applications to industrial problems (EUROGEN2001). Barcelona, 2001, pp. 95-100.

115. Барцев С. И., Охонин В. А. Адаптивные сети обработки информации. — Красноярск: Ин-т физики СО АН СССР, 1986. Препринт N 59Б. — 20 с.

116. Бураков С.В., Залога А.Н., Панькин С.И., Семенкин Е.С., Якимов И.С. Применение самоконфигурируемого генетического алгоритма для моделирования атомной кристаллической структуры химических соединений по данным рентгеновской дифракции // Программные системы и вычислительные методы. 2014. № 4. С. 500-512.

117. Ворожейкин А.Ю. Разработка и исследование адаптивного вероятностного генетического алгоритма для многокритериальных задач условной оптимизации / А.Ю. Ворожейкин, Е.С. Семенкин // Труды международных научно-технических конференций «Интеллектуальные системы» (AIS’08) и «Интеллектуальные САПР» (CAD-2008). – М.: Физматлит, 2008, Т.1. – С. 15-21

118. Воронцов, К. В., Каневский, Д. Ю. Коэволюционный метод обучения алгоритмических композиций // Таврический вестник информатики и математики. – 2005. – № 2. – С. 51–66.

119. Галушкин А. И. Синтез многослойных систем распознавания образов. — М.: Энергия, 1974.

120. Заенцев И.В. Нейронные сети: основные модели. – Воронеж 1999 г. 76 стр.

121. Короткий С., Нейронные сети: алгоритм обратного распространения: [Электронный ресурс] – URL: http://www.neurok.ru

122. Либерти Д., Джонс Б. Освой самостоятельно C++ за 21 день, 5-е издание. : Пер. с англ. – М.: Вильямс, 2008. – 768 с.: ил.

123. Липинский Л.В., Семенкин Е.С. Применение алгоритма генетического программирования в задачах автоматизации проектирования интеллектуальных информационных технологий // Сибирский журнал науки и технологий. – №3 (10). – 2006. – С. 22-26.

124. Мазуров, В. Д. Метод комитетов в задачах оптимизации и классификации. – М.: Наука, 1990.

125. Семенкин Е.С., Семенкина М.Е. Применение генетического алгоритма с модифицированным оператором равномерной рекомбинации при автоматизированном формировании интеллектуальных информационных технологий // Вестник Сибирского государственного аэрокосмического университета имени академика М. Ф. Решетнева. – № 3 (16). – 2007. – С. 27-33.

126. Семенкин Е.С., Семенкина М.Е., Попов Е.А. Проектирование ансамблей классификаторов самоконфигурируемыми эволюционными алгоритмами // Системы управления и информационные технологии. 2016. Т. 66. № 4. С. 38-43.

127. Семенкина М.Е Самоадаптивные эволюционные алгоритмы проектирования информационных технологий интеллектуального анализа данных. // Искусственный интеллект и принятие решений. № 1. – 2013. С. 13-23.

128. Сергиенко, А.Б, Галушин, П.В, Бухтояров, В.В., Сергиенко, Р.Б., Сопов, Е.А., Сопов, С.А. Генетический алгоритм. Стандарт: [Электронный ресурс] – 2015. Режим доступа: https://github.com/Harrix/Standard-Genetic-Algorithm

129. Хритоненко Д.И., Семенкин Е.С. Адаптивная мутация в самоконфигурируемых эволюционных алгоритмах // Системы управления и информационные технологии, №3(69), 2017. – С. 37-42

130. Шаханов Н.В., Варфоломеев И.А., Ершов Е.В., Юдина О.В. Прогнозирование отказов оборудования в условиях малого количества поломок – Вестник ЧГУ. № 6 (75). С. 36–41. – 2016.

ПУБЛИКАЦИИ ПО ТЕМЕ РАБОТЫ

Статьи в ведущих рецензируемых научных журналах и изданиях

1. Хритоненко Д.И., Семенкин Е.С. Адаптивная мутация в самоконфигурируемых эволюционных алгоритмах // Системы управления и информационные технологии, №3(69), 2017. – С. 37-42

2. Khritonenko D., Semenkin E., Sugak E., Potilicina E. Solving the problem of city ecology forecasting with neuro-evolutionary algorithms // Вестник СибГАУ. 2015. Т. 16, № 1. С. 137–143.

3. Хритоненко Д.И., Семенкин Е.С. Distributed self-configuring evolutionary algorithms for artificial neural networks design // Вестник СибГАУ. 2013. № 4 (50), С. 112-116.

Публикации в изданиях, индексируемых в международных базах

4. Khritonenko D., Stanovov V., Semenkin E. Applying an instance selection method to an evolutionary neural classifier design // IOP Conference Series: Materials Science and Engineering. V International Workshop on Mathematical Models and their Applications. – 2017. - Vol. 173. **(Web of Science, Scopus).**

5. Khritonenko D., Semenkin E. Application of artificial neural network ensembles for city ecology forecasting using air chemical composition information // Proceedings of the 2014 International Conference on Environmental Engineering and Computer Application (ICEECA 2014, Hong Kong). **(Scopus).**

Публикации в сборниках трудов конференций

6. Становов В. В., Хритоненко Д. И., Шкраба А. Метод селекции обучающих примеров в нейросетевых классификаторах // Материалы XIX междунар. науч.-практ. конф., «Решетневские чтения». СибГАУ, 2015. – Красноярск. Т. 2. С. 100-102.

7. Хритоненко Д. И. Самоконфигурируемый эволюционный алгоритм автоматического проектирования рекуррентных нейронных сетей // Материалы Всероссийской научно-практической конференции «Информационно-

телекоммуникационные системы и технологии» (ИТСиТ-2015), 2015. – Кемерово. С. 488-489.

8. Хритоненко Д. И., Семенкин Е. С., Сугак Е. В., Потылицина Е. Н. Автоматическое генерирование нейросетевых моделей в задаче прогнозирования уровня заболеваемости населения // Труды четырнадцатой национальной конференции по искусственному интеллекту с международным участием (КИИ-2014), 2014. – Казань. Т. 1, С. 276-285.

9. Хритоненко Д. И. Автоматическое проектирование коллективов искусственных нейронных сетей в задачах анализа данных // Материалы Восьмого Всероссийского форума студентов, аспирантов и молодых ученых «Наука и инновации в технических университетах», 2014. – Санкт-Петербург. С. 60-61.

10. Хритоненко Д. И., Становов В. В. Comparing intelligence information technologies efficiency for solving classification problem // Материалы XIII междунар. науч. конф. бакалавров, магистрантов и аспирантов «Молодежь. Общество. Современная наука, техника и инновации», 2014. – Красноярск, СибГАУ. С. 217–219.

11. Хритоненко Д. И. Программная система автоматического генерирования нейросетевых классификаторов эволюционными алгоритмами // Материалы XVIII Международ. науч. конф. «Решетневские чтения», 2014. – Красноярск, СибГАУ. Ч. 2. С. 290-291

12. Хритоненко Д. И., Семенкин Е.С. Применение распределенных самоконфигурируемых эволюционных алгоритмов в задачах нейросетевого моделирования // Труды V Международной конференции «Системный анализ и интеллектуальные технологии» (САИТ-2013), 2013. – Красноярск. Т. 2, С. 108-114.

13. Хритоненко Д. И. Самоконфигурируемый распределенный генетический алгоритм //: Труды XIII Междунар. науч. Конференции «Интеллект и наука», 2013. – Железногорск. – С. 134-135.

14. Хритоненко Д. И. Применение самонастраивающегося алгоритма генетического программирования в задачах нейросетевого моделирования // Сборник материалов победителей и лауреатов Всероссийского конкурса научно-исследовательских работ студентов и аспирантов в области технических наук, 2012. – Санкт-Петербург. С. 39–40.

Зарегистрированные программные системы

15. Семенкин Е.С., Панфилов И.А., Сопов Е.А., Становов В.В., Хритоненко Д.И., Брестер К. Ю., Семенкина О.Е. Программная система для автоматизированного управления интеллектуальными информационными сетями. Свидетельство № 2015662581 о гос. регистрации в Реестре программ для ЭВМ от 26.11.2015.

16. Семенкин Е.С., Панфилов И.А., Сопов Е.А., Становов В.В., Хритоненко Д.И., Брестер К. Ю., Семенкина О.Е. Программная система для автоматизированной генерации моделей и алгоритмов решения задач анализа активности пользователя. Свидетельство № 2015662579 о гос. регистрации в Реестре программ для ЭВМ от 26.11.2015.

17. Семенкин Е.С., Панфилов И.А., Сопов Е.А., Становов В.В., Хритоненко Д.И., Брестер К. Ю., Семенкина О.Е. Программная система автоматизированного проектирования интеллектуальных информационных сетей. Свидетельство № 2015662501 о гос. регистрации в Реестре программ для ЭВМ от 25.11.2015.

18. Хритоненко Д. И., Панфилов И.А., Сопов Е.А. Система автоматического генерирования коллективов нейросетевых моделей распределенными самоконфигурируемыми эволюционными алгоритмами. Свидетельство № 2014612727 о гос. регистрации в Реестре программ для ЭВМ от 05.03.2014.

19. Брестер К. Ю., Панфилов И.А., Семенкин Е.С., Семенкина О.Е., Сопов Е.А., Становов В.В., Хритоненко Д.И. Система автоматизированной классификации и категоризации мультилингвистических документов по содержанию. Свидетельство № 2013660992 о гос. регистрации в Реестре программ для ЭВМ от 26.11.2013.

20. Хритоненко Д. И., Семенкин Е. С. Система автоматического генерирования нейросетевых моделей самоконфигурируемыми эволюционными алгоритмами. Свидетельство № 2013619021 о гос. регистрации в Реестре программ для ЭВМ от 24.09.2013.

21. Хритоненко Д. И., Сергиенко Р. Б. Распределенная программная система автоматического формирования нейронных сетей эволюционными алгоритмами. Свидетельство № 2013611034 о гос. регистрации в Реестре программ для ЭВМ от 9.01.2013.

ПРИЛОЖЕНИЕ А. ОПТИМАЛЬНЫЕ НОМЕРА КОНФИГУРАЦИЙ ГА ДЛЯ РЕШЕНИЯ ЗАДАЧ *CEC'2017*

Номер функции	Размерность задачи			
	10	30	50	100
1	7	7	7	8
2	13	16	16	16
3	7	8	8	8
4	23	23	23	23
5	17	17	23	17
6	9	15	9	15
7	26	26	26	26
8	23	23	23	23
9	25	25	25	25
10	16	17	17	17
11	2	2	2	2
12	23	23	23	23
13	10	10	11	11
14	25	25	25	26
15	8	7	7	8
16	9	9	10	11
17	17	17	17	17
18	17	16	17	17
19	26	26	26	26
20	25	25	26	26
21	13	13	13	14
22	23	23	23	23
23	14	14	14	14
24	25	26	26	26
25	23	22	22	23

26	23	23	26	26
27	24	26	25	26
28	17	17	16	17
29	6	3	6	6
30	3	6	6	6

Расшифровка таблицы представлена в пункте 1.3 диссертации.

ПРИЛОЖЕНИЕ Б. ОПТИМАЛЬНЫЕ НОМЕРА КОНФИГУРАЦИЙ ГП ДЛЯ АППРОКСИМАЦИИ ЗАДАЧ *CEC'2017*

Номер функции	Размерность задачи			
	10	30	50	100
1	8	4	6	4
2	8	9	8	8
3	4	0	6	10
4	4	9	11	4
5	4	9	8	4
6	11	10	10	8
7	4	8	7	8
8	8	2	4	10
9	11	1	8	0
10	6	9	8	9
11	4	10	9	10
12	11	9	10	8
13	6	0	10	8
14	4	8	11	8
15	8	4	8	8
16	4	7	8	2
17	1	10	10	8
18	8	6	4	6
19	8	0	4	1
20	6	4	8	6
21	8	4	8	0
22	11	5	9	6
23	4	1	4	9
24	2	0	9	10
25	10	10	8	4

26	8	11	6	10
27	8	0	6	6
28	0	0	8	9
29	9	6	10	0
30	0	8	0	0

Расшифровка таблицы представлена в пункте 1.3 диссертации.

ПРИЛОЖЕНИЕ В. ЧИСЛОВЫЕ ХАРАКТЕРИСТИКИ ЗАДАЧИ ПРОГНОЗИРОВАНИЯ УРОВНЯ ЗАБОЛЕВАЕМОСТИ НАСЕЛЕНИЯ

Прогнозируемая величина	Рождаемость (на 1 тыс. чел.)	Смертность (на 1 тыс. чел.)	Смертность детей до 1 года (на 1 тыс. чел.)	Больных злокач. новообразованиями (на 100 тыс чел.)	Смертность от новообразований (на 100 тыс чел.)
Нумерация	Out1	Out2	Out3	Out4	Out5
Мин.	8.3	8.9	4.8	1042	180.26
Макс.	14.3	15.5	21	2483.8	211.9
Среднее	10.98	12.41	14.26	1822.65	202.44
Стандарт. укл.	1.89	1.80	5.91	429.42	8.75
Прогнозируемая величина	Смертность Система кровообращения на 100 тыс. чел.	Смертность Органы дыхания (на 100 тыс ч-к)	Больных психич. расстройствами (100 тыс. чел.)	Смертность от психических расстройств (100 тыс. чел.)	Смертность от болезней нервной системы (100 тыс. чел.)
Нумерация	Out6	Out7	Out8	Out9	Out10
Мин.	441.44	38.69	686.7	0.91	5.31
Макс.	726.05	72.16	14074	11.42	8.98
Среднее	594.03	55.93	931.07	5.83	6.79
Стандарт. укл.	76.16	8.70	198.96	3.91	1.13

ПРИЛОЖЕНИЕ Г. СРАВНИТЕЛЬНЫЕ РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ РАСПОЗНАВАНИЯ ЭМОЦИЙ ПО ЗВУКОВОМУ СИГНАЛУ.

В данном приложении представлен критерий точности классификации:

$$Accuracy = \frac{P}{N},$$

где P – количество верно классифицированных объектов, N – общее число объектов. Результаты аналогов получены при использовании сред *R* и *RapidMiner*.

Алгоритм	Точность	Алгоритм	Точность
Decision Tree	0.42	*Local Polynomial Regression*	0.54
Support Vector Machine	0.36	*Support Vector Machine (LibSVM)*	0.55
Support Vector Machine (Linear)	0.36	*W-BayesianLogisticRegression*	0.57
W-NB Tree	0.41	*W-PaceRegression*	0.56
Naive Bayes	0.608	*W-RandomForest*	0.57
W-KStar	0.39	*W-ADTree*	0.55
W-PART	0.42	*K-FuzzyRules*	0.61
W-BayesNetGenerator	0.48	*W-M5P*	0.61
W-REPTree	0.43	*Gaussian Process*	0.61
Rule Induction	0.44	*AutoMLP*	0.61
W-J48graft	0.45	*Fast Large Margin*	0.61
W-BFTree	0.39	*Linear Discriminant Analysis*	0.62
W-JRip	0.45	*W-MultilayerPerceptron*	0.66
Naive Bayes Kernel	0.53	*W-FT*	0.67
W-lBk	0.52	*Neural Net*	0.701
W-DTNB	0.47	*W-LinearRegression*	0.632
W-HyperPipes	0.52	*Linear Regression*	0.64
W-IB1	0.54	*W-PLSClassifier*	0.65

W-J48	0.54	*Logistic Regression*	0.67
W-M5Rules	0.501	*W-LMT*	0.67
W-Logistic	0.57	*W-LogisticBase*	0.67
Perceptron	0.6	*W-GaussianProcesses*	0.65
ENS_ANN	**0.725**	***ANNEA+APS+IS***	**0.716**

ПРИЛОЖЕНИЕ Д. РЕШЕНИЕ ЗАДАЧИ ЭКОЛОГИЧЕСКОГО ПРОГНОЗИРОВАНИЯ НЕЙРОЭВОЛЮЦИОННЫМИ АЛГОРИТМАМИ

В представленных ниже таблицах k – величина временного лага. Нумерация прогнозируемых характеристик описывается в приложении 3. Критерий эффективности – $MAPE = \frac{100}{n} \cdot \sum_{i=1}^{n} | PE_i |$, $PE_i = \frac{X_i - F_i}{X_i}$, где X_i – истинное значение, а F_i – прогнозируемое, объем выборки равен n

Таблица Д1 – Эффективность базовой версии *SCGP+UR*

k	*Out1*	*Out2*	*Out3*	*Out4*	*Out5*	*Out6*	*Out7*	*Out8*	*Out9*	*Out10*
1	12.67	**12.69**	**6.077**	4.481	8.562	2.696	0.799	8.267	**4.686**	**10.496**
2	**9.768**	13.793	6.992	7.067	**6.279**	3.092	**0.499**	7.481	5.979	10.668
3	10.4	14.293	7.667	**4.184**	7.486	3.59	1.791	5.395	5.596	11.445
4	11.198	13.496	8.594	8.662	7.373	1.8	1.993	**5.294**	5.495	11.78
5	10.98	15.438	8.677	7.968	7.994	3.293	2.798	6.998	5.783	11.967
6	11.183	15.039	7.989	5.674	7.173	**1.497**	3.196	7.192	5.784	12.097
7	11.57	16.192	9.059	5.796	9.356	2.399	3.394	8.483	6.469	12.467

SCGP+UR – алгоритм автоматического проектирования ИНС с использованием ГП модифицированного самоконфигурированием, равномерным скрещиванием, отбором информативных признаков.

Таблица Д2 – Эффективность версии *ANNEA+APS*

k	*Out1*	*Out2*	*Out3*	*Out4*	*Out5*	*Out6*	*Out7*	*Out8*	*Out9*	*Out10*
1	11.804	15.974	6.745	2.996	6.368	**0.953**	0.76	6.075	1.991	9.162
2	**8.819**	12.337	5.518	2.884	**5.266**	0.995	0.481	5.175	0.951	8.619
3	8.89	**12.19**	**5.375**	3.077	5.356	1.024	**0.197**	**4.752**	0.955	**8.098**
4	9.208	12.535	5.506	2.673	5.685	0.97	0.69	4.793	**0.754**	8.365

5	10.145	13.351	6.185	**2.34**	6.091	1.254	0.673	5.699	1.279	8.571
6	10.175	14.292	5.694	3.596	6.122	1.203	0.374	5.849	2.181	9.077
7	10.623	14.304	6.481	2.99	6.633	1.364	0.466	5.897	2.248	8.846

ANNEA+APS – усредненный показатель для ГА и ГП, используемых в автоматическом проектировании ИНС. По сравнению с предыдущим алгоритмом добавлены модификации адаптации уровня мутации и размера популяции.

Таблица Д3 – Эффективность коллективного нейроэволюционного алгоритма *ENS_ANN*

k	*Out1*	*Out2*	*Out3*	*Out4*	*Out5*	*Out6*	*Out7*	*Out8*	*Out9*	*Out10*
1	11.671	15.77	6.385	2.854	6.711	0.979	0.75	6.125	2.063	9.531
2	**8.53**	12.678	5.536	2.898	**5.139**	0.969	0.497	5.212	0.951	8.543
3	9.462	**12.256**	**5.05**	3.179	5.302	1.029	**0.196**	4.853	0.938	**7.944**
4	9.129	12.668	5.646	2.613	5.7	**0.942**	0.695	**4.554**	**0.791**	8.569
5	10.585	13.51	5.868	**2.322**	6.185	1.236	0.699	5.839	1.246	8.784
6	10.263	14.127	5.631	3.334	6.13	1.231	0.377	5.816	2.181	9.096
7	10.739	15.017	6.452	3.039	6.387	1.316	0.463	5.812	3.058	8.851

ENS_ANN – Итоговый алгоритм проектирования коллективов ИНС, описанный в главе 4.

В таблице Д4 представлено сравнение указанных в данном приложении алгоритмов с аналогами, реализованными в среде *RapidMiner*. Производится сравнение с «базовым решением» с целью оценки практической полезности получаемых моделей. В базовом решение $F_i = X_{i-1}$, т.е. базовое решение оценивает эффективность простейшего правила прогноза (полагать прогнозируемое значение в момент времени *i* равным значению в предыдущий период времени *i*-1).

Таблица Д4 – Сводная таблица результатов экологического прогнозирования

	Out 1	*Out 2*	*Out 3*	*Out 4*	*Out 5*	*Out 6*	*Out 7*	*Out 8*	*Out 9*	*Out 10*
Базовое решение	15.7	**2.7**	**0.92**	**1.76**	**2.9**	1.65	10.7	10.7	12.5	15.7
SCGP+UR	9.77	12.69	6.08	4.18	6.28	1.50	0.50	5.29	4.68	10.5
ANNEA+APS	8.82	12.19	5.38	2.34	5.27	0.95	0.20	4.75	0.75	8.10
ENS_ANN	**8.53**	12.2	5.05	2.32	5.14	**0.94**	**0.20**	**4.56**	**0.79**	**7.94**
Mult. regression	9.84	12.78	6.29	4.27	6.58	1.51	0.52	5.34	4.87	10.58
Decision tree	10.23	13.76	6.36	4.59	6.74	1.53	0.54	5.55	4.69	11.1
Random forest	10.68	13.83	6.22	4.35	6.67	1.57	0.53	5.74	4.97	11.09
Neural net	10.25	12.21	5.78	4.19	6.34	1.45	0.5	5.06	4.81	10.52
SVM	9.92	12.92	6.17	4.2	6.35	1.52	0.51	5.31	4.74	10.68