МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

«Сибирский федеральный университет»

На правах рукописи

Шерстнев Павел Александрович

САМОКОНФИГУРИРУЕМЫЕ ЭВОЛЮЦИОННЫЕ АЛГОРИТМЫ С АДАПТАЦИЕЙ НА ОСНОВЕ ИСТОРИИ УСПЕХА ДЛЯ ПРОЕКТИРОВАНИЯ МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ

2.3.1 – Системный анализ, управление и обработка информации, статистика

Диссертация на соискание ученой степени

кандидата технических наук

Научный руководитель:
доктор технических наук,
профессор Семенкин Е.С.

Красноярск – 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность. В условиях интенсивного развития интеллектуальных информационных технологий (ИИТ) методы машинного обучения (МО) демонстрируют высокий потенциал, однако рост сложности моделей приводит к проблеме снижения их интерпретируемости, а также обостряет необходимость автоматизации процесса проектирования для обеспечения масштабируемости и надежности. Одним из эффективных решений является применение эволюционных алгоритмов (ЭА), которые зарекомендовали себя как мощные средства моделирования и оптимизации, способные работать с алгоритмически заданными функциями вещественных, булевых, целочисленных и разношкальных переменных. Однако эффективность ЭА напрямую зависит от выбора их внутренних настроек. Согласно «No Free Lunch» теореме Уолперта и Макреди, не существует единого набора настроек, который гарантированно работал бы оптимально на всех классах задач. Для решения этой проблемы требуется внедрение методов самоадаптации ЭА в процессе поиска решения. Выделяют два варианта такой настройки: самоконфигурирование — выбор типа генетического оператора, и самонастройка — адаптация численных параметров.

Несмотря на существование различных методов самоадаптации ЭА, задача выбора оптимальных настроек для конкретной задачи остается открытой. Это особенно критично при решении задач проектирования моделей МО, где необходимо учитывать ограниченность вычислительных ресурсов. В таких условиях особенно востребованы алгоритмы оптимизации, способные достигать более качественных решений при тех же ресурсах, либо обеспечивать сопоставимый результат с меньшими затратами.

Итогом становится то, что даже при разнообразии подходов проблема автоматизированного конструирования моделей МО, с минимальным участием

эксперта по-прежнему остается актуальной. Таким образом, разработка и исследование самоадаптивных методов моделирования и оптимизации для автоматизированного проектирования моделей МО представляет собой **актуальную научно-техническую задачу**.

Степень проработанности темы. В рамках формирования ИИТ с помощью ЭА успешно применяются для проектирования нечетких логических систем (НЛС) и нейронных сетей (ИНС) (F. Herrera, J. Del Ser, K. Stanley, P. Bonissone, B. B. Становов), автоматизированного проектирования ансамблей и коллективов моделей (P. N. Suganthan, Z. H. Zhou, S. Raschka, L. I. Kuncheva), а также для настройки и оптимизации других моделей МО и синтеза новых ЭА оптимизации сложных систем (G. Kendall, F. Hutter, B. Doerr, E. A. Сопов). В рамках направления самоадаптации разработано множество алгоритмов. К наиболее известным относятся самоконфигурируемые ЭА, корректирующие вероятность выбора генетических операторов в пользу наиболее эффективных (J. Niehaus, Семенкина М. Е.), коллективные алгоритмы, объединяющие несколько подходов (Kenneth A. De Jong, Ахмедова Ш. А.), и Success History-based Parameter Adaptation (SHA) — схема адаптации на основе истории успеха, ставшая основой многих современных методов самонастройки (R. Tanabe, T. Fukunaga, Становов В. В).

Целью диссертационной работы является повышение эффективности и интерпретируемости моделей машинного обучения за счет разработки и применения самоадаптивных эволюционных алгоритмов моделирования и оптимизации для автоматизированного формирования моделей.

Объектом исследования являются ЭА, используемые при автоматизированном проектировании моделей МО (ИНС, НЛС и их ансамблей). **Предмет исследования** составляют методы самоадаптации ЭА, направленные на повышение точности и интерпретируемости формируемых моделей.

Для достижения поставленной цели необходимо решить следующий комплекс **задач**:

1. Выполнить обзор современных подходов к проектированию ИИТ и методов самоадаптации ЭА с целью выявления наиболее перспективных решений.

2. Разработать метод самоадаптации ЭА, интегрирующий сильные стороны существующих подходов и обеспечивающий повышение эффективности алгоритмов.

3. На основе разработанного метода самоадаптации разработать генетический алгоритм (ГА) для задач оптимизации с разношкальными переменными и алгоритм генетического программирования (ГП) с динамической адаптацией операторов и параметров.

4. Разработать алгоритм автоматизированного формирования ансамблей нейронных сетей (АНС), оптимизирующий выбор их структуры и количества участников.

5. Разработать метод гибридизации ИИТ на основе ЭА, обеспечивающий высокую точность и интерпретируемость полученных решений и автоматизацию формирования моделей МО.

6. Реализовать предложенные методы в виде программных систем и провести сравнительный анализ с существующими подходами на репрезентативном множестве тестовых и практических задач.

Методы исследования. При выполнении диссертационного исследования применялись методы вычислительного интеллекта, эволюционных вычислений, оптимизации, теории вероятностей и математической статистики, теории обработки информации.

Научная новизна включает следующие пункты:

1. Разработан, реализован и исследован новый самоконфигурируемый ГА, отличающийся измененным циклом работы и комплексной модифицированной процедурой скрещивания, а также интеграцией механизма адаптации вероятностей операторов скрещивания и мутации на основе истории успеха, что обеспечивает повышение надежности по сравнению с известными аналогами.

2. Разработан, реализован и исследован новый самоконфигурируемый алгоритм ГП, отличающийся измененным циклом работы и комплексной модифицированной процедурой скрещивания, а также интеграцией механизма

адаптации вероятностей операторов скрещивания и мутации на основе истории успеха, что обеспечивает повышение надежности по сравнению с известными аналогами.

3. Разработан, реализован и исследован новый алгоритм автоматизированного формирования АНС на основе алгоритма ГП, отличающийся от известных способом совместного кодирования множества ИНС в одном бинарном дереве, что позволяет одновременно оптимизировать архитектуру участников ансамбля, их количество и параметры мета-модели, обеспечивая автоматизацию процесса формирования ИИТ.

4. Разработан, реализован и исследован новый метод гибридизации ИИТ на основе ЭА, отличающийся от известных автоматизированной интеграцией нейросетевых моделей и НЛС, что позволяет объединить высокую точность нейросетевой модели с возможностью логической интерпретации ее поведения, обеспечивая построение объяснимых и компактных моделей без необходимости ручной настройки.

Теоретическая значимость результатов диссертационного исследования состоит в том, что разработаны новые ЭА для автоматизированного формирования точных и интерпретируемых ИИТ. В частности, разработанные методы самоадаптации и генетические операторы расширяют существующие подходы к адаптации операторов и параметров ЭА. Кроме того, полученные в работе гибридные модели МО, объединяющие ИНС, их ансамбли и НЛС, способствуют развитию теории формирования интерпретируемых моделей МО, сочетающих преимущества высокой точности и интерпретируемости решений. Разработанный метод автоматизированного формирования АНС на основе предложенного метода кодирования структур ИНС бинарными деревьями расширяет теоретические основы эволюционного проектирования ансамблевых моделей, обеспечивая возможность одновременной оптимизации структуры участников ансамбля, их количества и структуры итоговой мета-модели.

Практическая значимость результатов исследования заключается в разработке и реализации программной библиотеки с открытым исходным кодом

«Thefittest» на языке Python, предназначенной для эффективного применения самоадаптивных ЭА в задачах оптимизации и проектирования ИИТ. Библиотека предоставляет удобный инструмент для пользователей, не обладающих специальными знаниями в области эволюционного моделирования, позволяя автоматизировать процесс формирования моделей, включая ИНС, АНС и НЛС.

Эффективность и работоспособность разработанных алгоритмов и программных систем подтверждена на различных задачах из открытых наборов данных, а также на ряде практических задач, включая прогнозирование акустических характеристик древесных панелей, краткосрочный прогноз силы ветра на морском побережье и моделирование процесса деградации солнечных батарей космического аппарата. Практическая значимость и качество разработанной библиотеки дополнительно подтверждены наградами, полученными на ряде профильных конкурсов: Samsung Innovation Campus 2024, Soft-Парад 2025, Премия «Гравитация 2025» (номинация «Алгоритмы и программные решения в области искусственного интеллекта и больших данных»).

Реализация результатов работы. В ходе диссертационного исследования была разработана и представлена в открытом доступе программная библиотека Thefittest, включающая программные реализации разработанных в диссертации алгоритмов, а также ряда аналогичных методов. С использованием данной библиотеки создано 6 программных систем, зарегистрированных в Федеральной службе по интеллектуальной собственности (Роспатент). Диссертационная работа выполнена в рамках проектов № 075-15-2022-1121 - мегагрант «Гибридные методы моделирования и оптимизации в сложных системах» и № FEFE-2023-0004 - государственное задание Министерства науки и высшего образования Российской Федерации «Адаптивные методы синтеза и управления проектированием компонентов сложных систем».

Разработанные в рамках диссертационного исследования программные системы внедрены в учебный процесс Института информатики и телекоммуникаций СибГУ им. Решетнева, а также Института математики и фундаментальной информатики Сибирского федерального университета.

Эффективность разработанных алгоритмов и реализующих их программ подтверждена также справками о передаче и использовании от трех научных организаций.

Основные защищаемые положения:

1. Самоконфигурируемый ГА с измененным циклом работы, комплексной модификацией процедуры скрещивания и адаптацией параметров на основе истории успеха обеспечивает повышение надежности решения задач разношкальной оптимизации по сравнению с известными аналогами.

2. Самоконфигурируемый алгоритм ГП с измененным циклом работы, комплексной модификацией процедуры скрещивания и адаптацией параметров на основе истории успеха обеспечивает повышение надежности решения задач символьной регрессии, построения моделей и предсказательного моделирования.

3. Метод кодирования АНС с помощью бинарных деревьев, обеспечивающий одновременную оптимизацию структуры отдельных сетей, количества участников ансамбля и структуры мета-модели, является эффективным средством автоматизированного построения ансамблей ИНС с возможностью адаптации структуры ансамбля и высокой точностью.

4. Метод гибридизации ИИТ на основе ЭА, обеспечивающий автоматизированную интеграцию ИНС, их ансамблей и НЛС, позволяет формировать модели с сохранением точности и повышенной объяснимостью принимаемых решений.

Соответствие диссертации паспорту научной специальности. Диссертационное исследование соответствует следующим пунктам паспорта специальности 2.3.1 «Системный анализ, управление и обработка информации, статистика»: 4 – Разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта, 5 – Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта.

Публикации. По теме диссертации опубликовано 22 работы, из них 5 статей в журналах Перечня ВАК РФ, в том числе одна в журнале из «Белого списка», 5 статей в изданиях, индексируемых в международных базах цитирования Web of Science и Scopus. Зарегистрированы 6 программных систем в Федеральной службе по интеллектуальной собственности (Роспатент).

Обоснованность и достоверность полученных результатов подтверждается многократными численными экспериментами с применением методов статистической обработки, использованием репрезентативного набора тестовых задач, сопоставлением с данными из научной литературы, а также успешным применением при решении практических задач и представлением результатов на профильных научных конференциях.

Апробация работы. Результаты диссертационного исследования и разработанные в его рамках алгоритмы были представлены и обсуждены на ряде международных и всероссийских научно-практических конференций, включая: Международную научно-практическую конференцию «Научные исследования – основа современной инновационной системы» (Саратов, 2025 г.), Международную научно-практическую конференцию «Фундаментальные и прикладные исследования в науке и образовании» (Ижевск, 2025 г.); Международную научно-практическую конференцию «Прорывные научные исследования как двигатель науки» (Пермь, 2025 г.); Международную ИТ-конференцию «Ключевые тренды развития искусственного интеллекта: наука и технологии» (МГТУ им. Баумана, 2023); Международную школу-семинар «Гибридные методы моделирования и оптимизации в сложных системах» (HMMOCS-2023 и HMMOCS-2022); III Международную научно-практическую конференцию «Современные достижения в области материаловедения и технологий» (CAMSTech-III 2022, Красноярск); VIII Международную научно-практическую конференцию «Актуальные проблемы авиации и космонавтики» (Красноярск, 2022 г.); Всероссийскую научную конференцию «Российская наука, инновации, образование — РОСНИО-2022» (Красноярск,); II Международную научно-практическую конференцию «Перспективы развития науки, инженерии, естественно-научного, технического и

цифрового образования» (ASEDU-II 2021, Красноярск); а также XXIV Международную научно-практическую конференцию «Решетнёвские чтения», посвященную памяти академика М. Ф. Решетнева (Красноярск, 2020 г.). Реализованная библиотека Thefittest была представлена в виде тьюториала на Международной конференции 13th International Workshop of Mathematical Models and their Applications (IWMMA’2024, Krasnoyarsk).

Диссертация была представлена и обсуждена на научно-технических семинарах кафедры программной инженерии Сибирского федерального университета и кафедры системного анализа и исследования операций СибГУ им. ак. М.Ф. Решетнева.

Структура работы. Диссертационная работа изложена на 148 страницах и состоит из введения, четырех глав, заключения, списка литературы из 181 источника и 7 приложений.

1 Эволюционные алгоритмы в интеллектуальных информационных технологиях и машинном обучении

1.1 Интеллектуальные информационные технологии и модели машинного обучения

Моделирование представляет собой процесс исследования объекта (системы), заключающийся в построении его модели, отображающей существенные характеристики, процессы и взаимосвязи. Математическое моделирование — это процесс создания и использования математического объекта (математической модели), находящегося в соответствии с целевыми характеристиками, процессами и взаимосвязями исследуемого объекта (системы). Выбор вида модели обусловлен природой объекта, целями исследования и требуемой точностью результатов моделирования. В силу принципа абстрагирования любая математическая модель представляет исследуемый объект с определенной степенью приближения. Далее рассматриваются математические модели (ММ), отличающиеся относительной простотой структуры и широким распространением в прикладных исследованиях. Модели такого вида содержат следующие элементы [1–3]:

1. Вектор x параметров, измеряемых на объекте $x = [x_1 \dots x_n]$, где x_i – значение i – го параметра. Можно называть x вектором состояния объекта. В случае, если изучается динамика объекта, то состояние описывается вектором $x(t) = [x_1(t) \dots x_n(t)]$;

2. Вектор $y(t)$ параметров, которые нельзя непосредственно измерить;

3. Неизвестная функциональная зависимость между входными и выходными переменными $y(t)$ и $x(t)$;

4. Наличие скрытых или нелинейных зависимостей внутри самого вектора входных или выходных переменных $x(t)$ и $y(t)$;

5. Математический аппарат исследования отношений (связей).

Математическая модель в таком случае определяется как (1.1):

$$y(t) = f(x(t), s), \tag{1.1}$$

где $x(t)$ — это вектор входных переменных; $y(t)$ — вектор выходных переменных; s — вектор внутренних параметров модели. Схематично математическая модель представлена на рисунке 1.1.

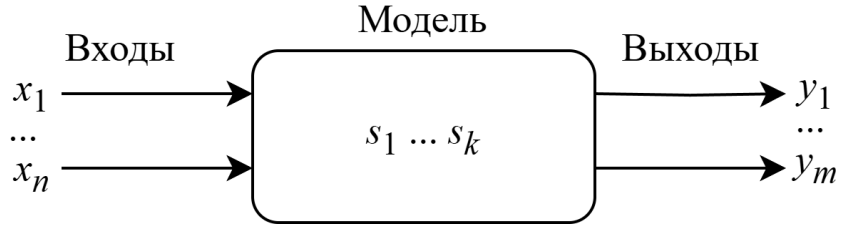

Рисунок 1.1 – Схема ММ объекта

Математическая модель играет ключевую роль в построении интеллектуальных информационных технологий (ИИТ) и моделей машинного обучения (МО). В контексте ИИТ математическая модель позволяет описывать как работу отдельных модулей, так и системы в целом. Это обеспечивает возможность анализа и оптимизации ее работы. В МО математическое моделирование выполняет две функции:

- *формализация зависимости между входными и выходными данными*. Любая модель МО представляет собой инструмент аппроксимации, восстанавливающий неизвестную функциональную зависимость по обучающим примерам;
- *оценка структуры и параметров моделей*. Формализация моделей через математический аппарат позволяет осуществлять подбор параметров, гиперпараметров и структуры моделей.

Таким образом, математические модели являются основой для построения ИИТ и систем МО, определяя логику обработки информации, структуру алгоритмов, методы обучения и интерпретации. Далее будут рассмотрены наиболее распространенные типы моделей, используемых в ИИТ и системах МО, включая нейронные сети, нечеткие логические системы, деревья решений, а также коллективные методы принятия решений.

Одним из наиболее распространенных и эффективных классов моделей, широко применяемых в ИИТ, являются искусственные нейронные сети (ИНС). Они реализуют сложные аппроксимации в рамках математических моделей и активно используются в задачах классификации, прогнозирования и распознавания [4–6]. Данный класс алгоритмов основан на биоинспирированной модели, аналогичной структуре человеческого мозга. ИНС в простом случае представляет собой последовательность связанных слоев, содержащих искусственные нейроны. В результате работы такой структуры формируется взвешенная сумма признаков, реализующая более сложную зависимость, чем в линейных моделях. Кроме этого, каждый нейрон имеет нелинейную функцию активации (ФА), которая является важной частью алгоритма [7, 8].

На вход нейрона поступает взвешенная сумма сигналов, после чего применяется ФА. Процесс расчета выхода нейрона выражается формулой (1.2):

$$y_i = f\left(\sum_{j=1}^{N} w_{ij}x_j + w_{i0}\right), \qquad (1.2)$$

где y_i — это выход нейрона, w_{ij} — веса, соответствующие связям между входами и нейроном, x_j — входные сигналы, w_{i0} — смещение, f — ФА.

Функция активации — ключевой элемент работы искусственного нейрона, определяющий зависимость между входным сигналом и выходным значением нейрона. Основное назначение ФА — добавление нелинейности в модель, с целью аппроксимировать сложные зависимости. Тип ФА выбирается с учетом свойств задачи и архитектуры ИНС. В таблице 1.1 приведены наиболее распространенные ФА, применяемые в современных ИНС [9, 10].

Таблица 1.1 – Популярные ФА

Название	**Формула**	**Область значений**
Тождественная	$f(x) = x$	$(-\infty, \infty)$
Пороговая	$f(x) = \{0 \;\; x < 0 \; 1 \;\; x \geq 0$	$(0,1)$
Логистическая	$f(x) = \frac{1}{1 + e^{-x}}$	$(0,1)$
Гиперболический тангенс	$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$	$(-1,1)$
Параметрический линейный выпрямитель (*PReLU*)	$f(x) = \{ax \;\; x < 0 \; x \;\; x \geq 0$	$[-\infty, \infty)$
Гаусс	$f(x) = e^{-x^2}$	$(0,1]$
Softmax	$f_i(x) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$	$(0,1)$

Важно отметить, что функция *Softmax* не является ФА одного нейрона, а применяется к слою нейронов. Эта функция применяется при формировании вероятностного распределения на выходе, когда выходные классы взаимоисключающие.

При объединении нейронов образуются ИНС, архитектура которых может различаться в зависимости от способа объединения, а также от вида решаемой задачи. В частности, ИНС можно разделить на полносвязные и неполносвязные (рисунок 1.2).

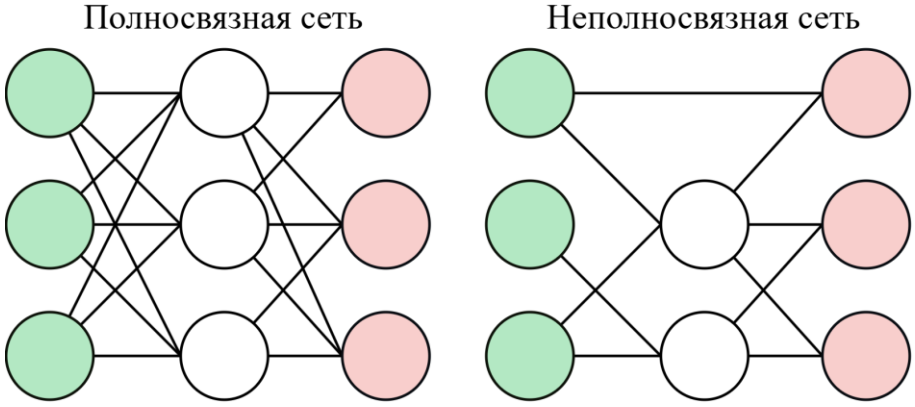

Рисунок 1.2 – Пример полносвязной и неполносвязной нейронной сети

ИНС подразделяются на архитектуры прямого распространения, в которых данные проходят от входа к выходу без обратных связей, и рекуррентные, способные учитывать временной контекст за счет циклических соединений. Рекуррентные нейронные сети предназначены для обработки последовательных данных. Для повышения эффективности были разработаны различные архитектурные модификации, направленные на улучшение работы с памятью. К ним относятся архитектуры LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Unit), использующие управляющие элементы для контроля сохранения и забывания информации, а также двунаправленные сети (BiLSTM/BiGRU), позволяющие учитывать как предыдущий, так и последующий контекст за счет параллельной прямой и обратной обработки последовательности [11–13]. Рекуррентные нейронные сети и их модификации широко используются в машинном переводе, анализе тональности, распознавании речи и прогнозировании временных рядов [14], а также в практических задачах, таких как прогнозирование гидрологических процессов [15], распознавание человеческой активности [16], диагностика состояния батарей [17] и предсказание сердечного ритма [18].

Еще одной распространенной архитектурой являются сверточные нейронные сети, которые эффективно извлекают пространственные признаки через фильтры и свертки, что позволяет обрабатывать данные с выраженной пространственной структурой. Сверточные нейронные сети эффективно применялись во многих задачах, включая задачи классификации и сегментации медицинских изображений [19], а также для анализа ЭКГ и диагностики сердечных нарушений [20]. Архитектура трансформеров, основанная на механизме внимания (attention) позволяет учитывать долгосрочные зависимости без рекуррентных связей [21]. Развитие архитектуры трансформер привело к появлению больших языковых моделей (large language model, LLM), таких как GPT. Эти модели обучены на масштабных корпусах текста и успешно применяются для генерации текста, суммаризации, анализа документов, а также в медицине и финансовом секторе [22].

Процесс обучения ИНС заключается в настройке весовых коэффициентов w

связей нейронов. Коэффициенты регулируются так, чтобы сеть обеспечивала наилучшее приближение к целевым выходам. Обучение ИНС может быть выполнено любым методом оптимизации, однако одним из наиболее эффективных способов обучения ИНС является метод обратного распространения ошибки [23–26]. Идея метода заключается в передаче ошибки от выхода к входу, причем на выходном слое ошибка вычисляется относительно целевых значений, а на последующих слоях – путем распространения сигнала по сети. (рисунок 1.3).

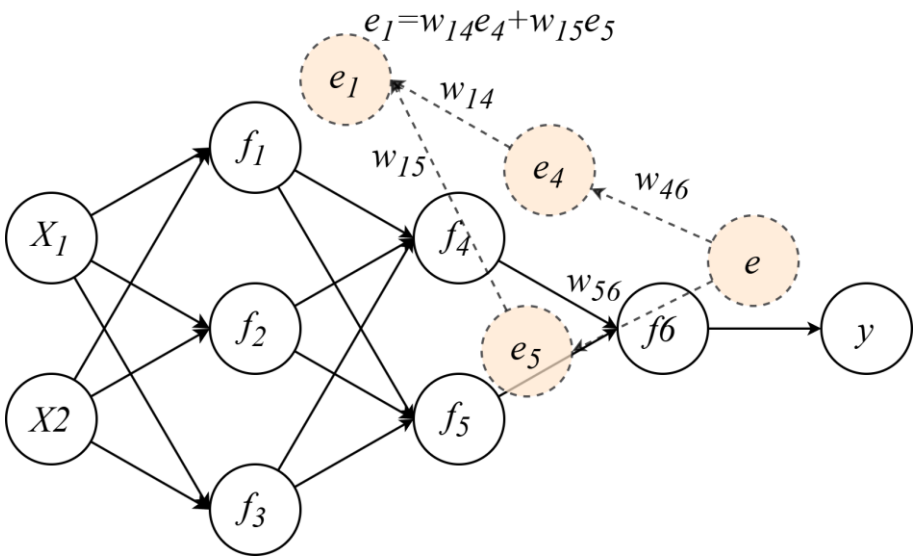

Рисунок 1.3 – Иллюстрация обратного распространения ошибки в нейронной сети

Веса сети корректируются с использованием вектора частных производных ошибки от весов. Коррекция вычисляется следующим образом (1.3):

$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}}, \tag{1.3}$$

где η — это шаг, задающий скорость движения в направлении антиградиента, E — ошибка сети. Метод обратного распространения ошибки является самым популярным и эффективным подходом для обучения ИНС, однако он обладает рядом ограничений, среди которых можно отметить проблемы «взрыва» и «затухания» градиента. Тем не менее существует множество модификаций, позволяющих справиться с некоторыми проблемами и ускорить сходимость алгоритма [27]. Кроме того, для полносвязных сетей данный подход обладает

высокой вычислительной эффективностью, поскольку в таком случае передача сигнала между слоями может быть представлена как умножение матриц, однако в случае неполносвязных сетей вычисление ошибок может представлять собой вычислительно сложную задачу.

Другим популярным направлением обучения ИНС является использование эволюционных алгоритмов оптимизации. Совокупность подходов, использующих эволюционные методы для настройки ИНС, называется нейроэволюцией [28–31]. В данном случае, если задачей стоит настройка весовых коэффициентов сети, эволюционный алгоритм генерирует популяции различных весовых коэффициентов с фиксированной структурой сети. На рисунке 1.4 представлен пример представления весов ИНС в виде индивида – строки.

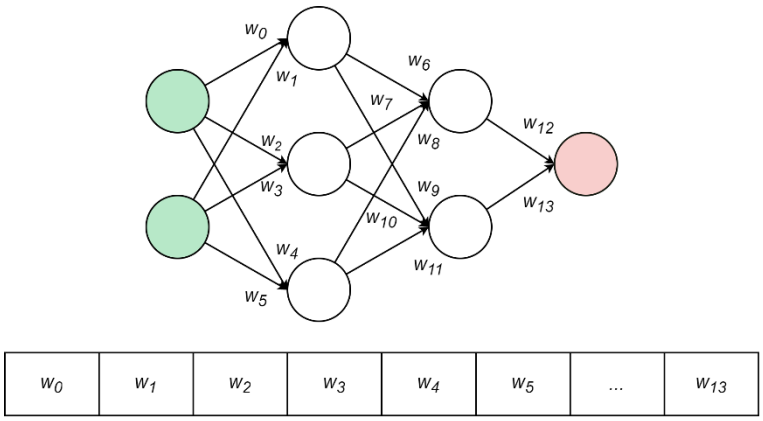

Рисунок 1.4 – Пример представления весов нейронной сети в виде строки

В процессе эволюции весовые коэффициенты изменяются таким образом, чтобы минимизировать ошибку ИНС на целевых значениях.

Другим актуальным и развивающимся направлением ИИТ являются нечеткие логически системы (НЛС). В 1965 г. профессором Калифорнийского университета Лотфи А. Заде была опубликована работа Fuzzy Sets, в которой было расширено классическое понятие множества. В частности, допускалось, что функция принадлежности элемента множеству может принимать любые значения в интервале [0;1], а не только 0 или 1. Такие множества получили название нечетких множеств (fuzzy sets) [32, 33]. Появление концепции нечетких множеств

было обусловлено необходимостью описания явлений, характеризующихся неточностью, неопределенностью или многозначностью. Использовавшиеся ранее математические методы, основанные на классической теории множеств, оказались недостаточными для моделирования подобных систем. На основе концепции нечетких множеств была сформирована теория нечеткой логики, позволившая строить логические рассуждения в условиях неопределенности. Это стало основой для разработки НЛС — моделей, способных интерпретировать и обрабатывать неточные, лингвистически выраженные знания. Уже в 1975 году Лотфи Заде предложил обобщение своей теории — нечеткую логику второго типа (Type-2 fuzzy logic), позволяющую моделировать не только нечеткие понятия, но и неопределенность самой функции принадлежности. Однако из-за высокой вычислительной сложности такие модели долго оставались теоретическими. Лишь с развитием вычислительных технологий и ростом интереса к интерпретируемым системам Type-2 fuzzy logic получила широкое применение в задачах, связанных с зашумленными или противоречивыми данными. НЛС используются для построения прогнозных моделей, особенно в условиях неполной или нечеткой информации [34], управления сложными системами [35], решения задач приближения сложных зависимостей [36], а также распознавания образов. Одним из ключевых компонентов НЛС является механизм нечеткого логического вывода, обеспечивающий переход от входных данных к интерпретируемым выходным результатам на основе набора правил. Рассмотрим структуру и этапы выполнения нечеткого вывода более подробно.

Нечеткий логический вывод. Основой для проведения операции нечеткого логического вывода является база правил, содержащая продукционные правила (нечеткие высказывания в форме "ЕСЛИ-ТО") и функции принадлежности для соответствующих лингвистических термов. При этом должны соблюдаться следующие условия:

1) Существует хотя бы одно правило для каждого лингвистического терма выходной переменной;

2) Для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки (левая часть правила).

Если какое-либо из условий не выполняется, то база правил является неполной. Пусть база правил состоит из *m* правил (1.4):

$$\begin{gathered} R_1\text{: Если } x_1 \text{ это } A_{11} \dots \text{И(ИЛИ)} \dots x_n \text{ это } A_{1n}, \text{ТО } y \text{ это } B_1 \\ R_i\text{: Если } x_1 \text{ это } A_{i1} \dots \text{И(ИЛИ)} \dots x_n \text{ это } A_{in}, \text{ТО } y \text{ это } B_i \\ R_m\text{: Если } x_1 \text{ это } A_{m1} \dots \text{И(ИЛИ)} \dots x_n \text{ это } A_{mn}, \text{ТО } y \text{ это } B \end{gathered} \quad (1.4)$$

где $x_k, k = 1 \dots n$ — это входные переменные; *y* — выходная переменная; A_{ik}, B_i — заданные нечеткие множества с функциями принадлежности. Результатом нечеткого вывода является четкое значение переменной *y* на основе заданных четких значений x_k.

В общем случае процедура нечеткого логического вывода включает три основных этапа:

1. *Фаззификация* — преобразование четких входных значений в нечеткие переменные с использованием соответствующих функций принадлежности.

2. *Нечеткий вывод* — применение базы правил для вычисления выходных нечетких множеств на основе предпосылок.

3. *Дефаззификация* — преобразование полученного нечеткого результата в четкое значение, пригодное для практического использования.

Алгоритмы нечеткого вывода различаются главным образом по типу используемых правил, логическим операциям и способу дефаззификации. Разработаны модели нечеткого вывода Мамдани [37], Сугено [38], Ларсена [39], Цукамото [40].

Таким образом, НЛС представляют собой формализованные модели, основанные на теории нечетких множеств и нечеткой логике — обобщениях классической булевой логики и теории множеств. Они позволяют эффективно моделировать процессы, описываемые лингвистически, с учетом неопределенности, неточности и субъективных факторов. НЛС являются мощным инструментом построения интерпретируемых моделей в условиях неполноты,

лингвистической неопределенности и отсутствия точных формальных зависимостей.

Помимо нейросетевых моделей и НЛС, в ИИТ широко используются и другие методы МО. Эти алгоритмы варьируются от простых, легко интерпретируемых моделей, таких как деревья решений, до более сложных ансамблевых методов, обеспечивающих высокую точность за счет объединения нескольких базовых моделей. Несмотря на меньшую архитектурную сложность по сравнению с ИНС, данные подходы часто демонстрируют высокую эффективность и устойчивость при решении широкого спектра практических задач. Ниже рассмотрены наиболее распространенные из них.

Дерево решений — один из методов поддержки принятия решений, широко применяемый в МО, анализе данных и статистике [41]. Структура дерева включает ветви и листья: ветви содержат предикаты (логические условия), а листья — выходные значения модели. Деревья решений часто применяются в интеллектуальном анализе данных (ИАД). Их цель — построение модели, способной предсказывать значение целевой переменной на основе набора входных признаков [42,43]. Предикат — это логическое выражение, принимающее значение ИСТИНА или ЛОЖЬ. Он разбивает исходное множество объектов на подмножества так, чтобы разделение было максимально информативным. В задачах регрессии вместо этих критериев применяются стандартные метрики ошибок, такие как среднеквадратичная ошибка (MSE) и средняя абсолютная ошибка (MAE).

Отдельное направление, связанное с алгоритмами мета-обучения, сформировалось в конце 1980-х годов, хотя многие подходы, восходящие к методам математической статистики, были известны и ранее. В простейшем случае коллективы алгоритмов МО принимают решения на основе голосования или усреднения предсказаний. Однако в стремлении повысить качество обучения в 1990-х годах были разработаны более сложные и эффективные методы построения коллективов. Наибольшее распространение получили три обобщенные схемы формирования коллективов алгоритмов: бустинг, бэггинг и стекинг. На рисунке 1.5

представлены схемы формирования коллективов алгоритмов МО по каждому из методов.

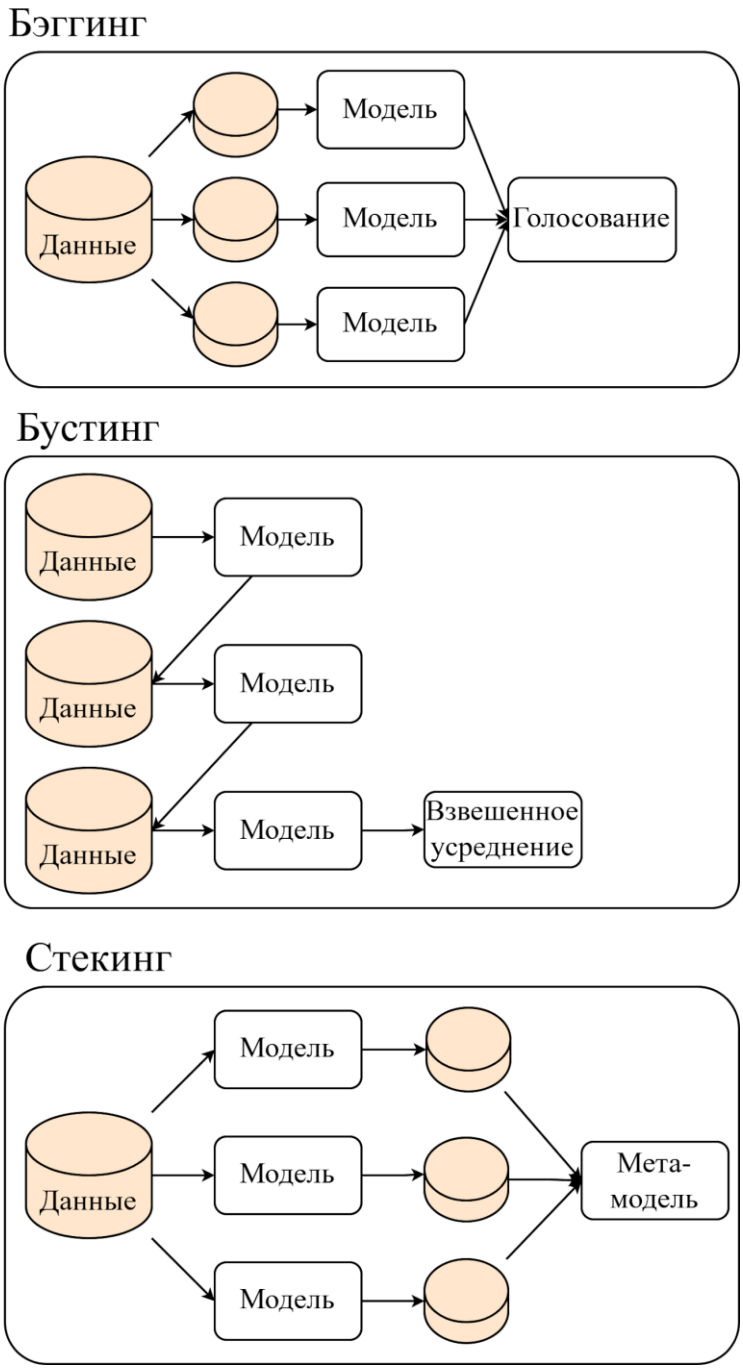

Рисунок 1.5 – Сравнительная схема коллективных методов машинного обучения: бэггинг, бустинг и стекинг

В 1989 году Роберт Шапир опубликовал в статье «Сила слабого обучения» алгоритм, основой которого было последовательное использование простых классификаторов с целью построить коллективный механизм классификации [44].

В 1995 году вместе с Йоавом Фройндом он модифицировал имеющиеся наработки в алгоритм AdaBoost [45]. Процедура, на которой основывались эти алгоритмы, сегодня называется бустингом. Каждый следующий классификатор в бустинге пытается исправить ошибки предыдущего. Это позволяет достигать высокой точности на обучающей выборке. Простота, гибкость и большой потенциал сделали бустинг одним из самых популярных инструментов МО на протяжении последующего десятилетия. Чаще всего в рамках бустинга в качестве базовых моделей используют деревья решений. На этой основе были разработаны современные и более эффективные алгоритмы, такие как XGBoost, LightGBM и CatBoost [46–48].

Другой популярный метод формирования ансамблей называется бэггинг. Его предложил американский математик и статистик Лео Брейман в 1994 году для улучшения точности классификации. Бэггинг основан на статистическом приеме бутстрэппинга, когда из выборки случайным образом формируются подвыборки с возвратом (элементы в подвыборках могут дублироваться) [49]. Наиболее распространенным частным случаем бэггинга является Random Forest [50], в котором в качестве базовых моделей выступают деревья решений, обучаемые на бутстрэп-выборках и случайно отобранных подмножеств признаков.

Третий популярный метод формирования коллектива часто «переоткрывали» различные исследователи в области анализа данных, но первые упоминания можно найти у английского математика Дэвида Уолперта в работе [51]. Метод называется стекинг, и его идея заключается в следующем: выходы множества базовых классификаторов используются в качестве мета-признаков, на которых затем обучается отдельная мета-модель. Таким образом задача решается несколькими методами, а итоговое предсказание формируется моделью МО на основе результатов всех участников коллектива.

Таким образом, математические модели играют фундаментальную роль в построении ИИТ. Простые и интерпретируемые алгоритмы, такие как деревья решений, остаются востребованными благодаря своей прозрачности и эффективности в типовых задачах. В частности, они широко применяются в таких

классах задач, как классификация, регрессия, кластеризация и выявление аномалий [43]. В то же время коллективные методы — бустинг, бэггинг и стекинг — позволяют существенно повысить качество предсказаний, хотя и сопровождаются ростом вычислительной сложности и снижением объяснимости. Эти ансамблевые подходы эффективны в задачах многоуровневой классификации, ранжирования, объединения гетерогенных моделей и обработки высокоразмерных данных [52–54]. Совокупность этих подходов обеспечивает широкий спектр инструментов для решения прикладных задач в ИИТ.

1.2 Эволюционные алгоритмы моделирования и оптимизации

Рассматривается задача однокритериальной безусловной глобальной оптимизации типа «черный ящик», где целевая функция не имеет явного аналитического выражения и ее свойства заранее неизвестны. Формально задача заключается в нахождении вектора параметров x^*, минимизирующего (или максимизирующего) целевую функцию (1.5):

$$x^* = argmin_{x\epsilon\Omega} f(x), \tag{1.5}$$

где Ω — это множество допустимых решений. В реальных сценариях невозможно точно определить глобальный минимум из-за численных ошибок и неопределенностей, поэтому критерий оптимальности ослабляется (1.6):

$$f(x^*) - \varepsilon = min_{x\epsilon\Omega} f(x), \tag{1.6}$$

где ε — это параметр, определяющий допустимый уровень погрешности при поиске оптимального решения. Для решения подобных задач традиционные методы оптимизации могут быть неэффективны, поскольку они зависят от свойств и структуры функции. В таких случаях ЭА предоставляют более гибкий подход.

Эволюционные алгоритмы представляют собой класс стохастических методов оптимизации, основанных на моделировании процессов биологической

эволюции, концептуально восходящих к эволюционной теории Ч. Дарвина. В основе таких алгоритмов лежат три фундаментальных принципа: изменчивость, наследственность и естественный отбор [55,56].

Ключевым достоинством ЭА, в отличие от градиентных и других строго математически обоснованных методов оптимизации, является то, что они не требуют информации о структуре целевой функции, ее непрерывности или дифференцируемости. Для реализации процесса оптимизации достаточно, чтобы каждая потенциальная точка в пространстве решений могла быть оценена с использованием числовой метрики — так называемой функции пригодности (ФП), отражающей степень соответствия решения поставленной задаче. Таким образом, роль целевой функции в ЭА опосредованно выполняется ФП, которая определяет, насколько «качественным» является то или иное решение относительно других членов популяции.

ЭА эффективно работают даже в условиях высокой размерности пространства решений, поскольку способны обнаруживать приближенные экстремумы, исследуя ограниченную часть допустимого множества. Это выгодно отличает ЭА от классических эвристических методов, таких как, например, метод деления отрезка пополам, эффективность которого существенно снижается при переходе к многомерным задачам. Благодаря универсальности применяемых представлений ЭА способны эффективно работать с вещественными, булевыми, целочисленными, а также разношкальными переменными, что делает их особенно востребованными при решении сложных задач оптимизации.

На практике ЭА, как правило, применяются в задачах, где использование традиционных методов затруднено вследствие отсутствия аналитического описания, высокой размерности, нестандартных ограничений или иных факторов, существенно осложняющих прямое математическое моделирование и анализ.

В таких ситуациях особую ценность приобретает универсальный эволюционный механизм, лежащий в основе работы ЭА. Каждый из индивидов популяции представляет собой отдельную точку в пространстве допустимых решений. Первая популяция формируется случайным образом, равномерно

покрывая все пространство поиска. Далее, с использованием операторов селекции, скрещивания и мутации, осуществляется процесс эволюции популяции

Оператор селекции выполняет отбор индивидов на основе значения ФП, определяя вероятность их участия в формировании следующего поколения. Целью селекции является повышение общей пригодности членов популяции за счет сохранения и передачи наиболее успешных генетических признаков.

Оператор скрещивания (рекомбинации) комбинирует генетический материал двух или более выбранных индивидов, формируя новое потомство. Данный оператор обеспечивает перераспределение информации и способствует передаче потенциально полезных свойств к следующим поколениям.

Оператор мутации вносит случайные изменения в генетическую структуру отдельных особей, как правило, с низкой вероятностью. Он играет ключевую роль в поддержании генетического разнообразия популяции и снижении риска преждевременной сходимости алгоритма к локальному экстремуму.

Таким образом, эти три оператора совместно реализуют стохастический поисковый процесс, направленный на приближение к оптимальному решению. Проектирование ЭА включает ряд этапов, каждый из которых влияет на эффективность оптимизации:

- *выбор представления решения*. Определяется способ кодирования решений: бинарный, вещественный, целочисленный, древовидный или смешанный. Представление должно соответствовать специфике задачи и быть совместимо с генетическими операторами;
- *инициализация начальной популяции*. Начальная совокупность решений формируется случайно или с использованием эвристической информации. Разнообразная популяция повышает вероятность успешной сходимости;
- *определение операторов эволюции*. Определяются методы селекции, скрещивания и мутации. От их выбора зависит баланс между поиском новых областей пространства решений (exploration) и углубленной проработкой уже выявленных областей (exploitation);

- *задание функции пригодности (ФП)*. Обеспечивает количественную оценку качества решений. ФП должна адекватно отражать цели задачи и быть применима в процедуре селекции;
- *определение условий останова*. Устанавливаются критерии завершения алгоритма: достижение требуемого качества решения, лимит поколений, отсутствие прогресса, временные или ресурсные ограничения.

Генетический алгоритм (ГА) — это семейство ЭА оптимизации, которые ищут решение в гиперкубе, определенном бинарной строкой, в которой закодирован индивид (рисунок 1.6) [21, 56, 57].

1	0	1	1	1	0	0	1

Рисунок 1.6 – Пример генотипа индивида в ГА

Цель ГА — определить, какая точка заданного гиперкуба обеспечивает экстремум целевой функции. ГА работает с популяцией решений, которая итеративно улучшается путем отбора и комбинаций наиболее перспективных из них.

Далее в цикле работы ГА (рисунок 1.7) выполняются три ключевых шага: селекция, скрещивание и мутация. Ниже описывается принцип и этапы работы ГА.

Рисунок 1.7 – Общая схема цикла работы генетического алгоритма

ГА нашли широкое применение в инженерной оптимизации, оптимизации расписаний, маршрутизации, обучении моделей МО, автоматическом проектировании, а также в задачах физики и биологии [59–61]. Современные ГА включают широкий спектр модификаций, направленных на повышение эффективности поиска в сложных пространствах. Одним из передовых подходов является P3 (Probabilistic Prototype-based Population) — это пирамидальный ГА, в котором каждый индивид улучшает себя итеративно на основе вероятностной модели, формируемой без явного задания операторов скрещивания и мутации [62]. Основой пирамидальных ЭА послужили подходы, основанные на выявлении зависимостей между переменными. Hierarchical BOA (hBOA), использующий байесовские сети для выявления зависимостей между переменными [63]; Linkage Tree Genetic Algorithm (LTGA), строящий иерархии взаимозависимых групп генов [64]; и LEA (Linkage Evolutionary Algorithm), алгоритм, использующий информацию о взаимозависимостях между переменными [65].

В задачах многокритериальной оптимизации применяются специализированные ЭА. Классическим решением является NSGA-II, основанный

на Парето-ранжировании [66]. Современные модификации, такие как NSGA-III, ориентированный на задачи с большим числом целей [67], и MOEA/D, использующий декомпозицию задачи, демонстрируют высокую эффективность в прикладных областях — от логистики до биоинформатики [68].

Алгоритм генетического программирования (ГП) — это семейство алгоритмов оптимизации, эволюционирующих программы, представленные в виде древовидных структур, каждый внутренний узел в которых является операцией, а конечный узел - операндом [69]. Благодаря гибкости такого способа кодирования, с помощью ГП могут решаться задачи, где структура решения заранее неизвестна или сложна для аналитического описания. Наиболее часто ГП используется для решения задач символьной регрессии [70], классификации [71], формирования моделей МО и алгоритмов оптимизации, синтеза программ и оптимизации сложных систем [72,73]. Этапы работы ГП обычно аналогичны этапам большинства ЭА и включают следующие шаги: инициализацию начальной популяции; оценку индивидов; отбор индивидов, которые будут формировать новое поколение с помощью генетического оператора селекции; рекомбинация выбранных индивидов для создания потомков путем применения генетического оператора скрещивания; мутация индивидов-потомков путем применения генетического оператора мутации; замещение предыдущего поколения потомками. Затем, происходит переход на этап оценки индивидов и цикл повторяется [74]. Наиболее широко распространенными является древовидное представление, в котором решения моделируются в виде синтаксических деревьев, составленных из функциональных и терминальных элементов. Тем не менее, в практике ГП применяются и альтернативные формы представления, включая линейные хромосомы (Linear Genetic Programming, LGP), представление в виде графов (например, Cartesian Genetic Programming, CGP) [69,75–78]. Выбор формы представления определяется спецификой задачи, требуемыми операциями над программами и желаемыми свойствами пространства решений.

Другим широко распространенным ЭА оптимизации является дифференциальная эволюция (ДЭ). ДЭ представляет собой один из эффективных

методов популяционной глобальной оптимизации, ориентированный на непрерывные пространства поиска [79]. Как и в случае с ГА, целью алгоритма является поиск глобального экстремума заданной целевой функции. Однако ключевым отличием ДЭ от ГА является непосредственное использование вещественных векторов без промежуточного этапа кодирования: каждый индивид в популяции представляется в виде вещественного вектора фиксированной длины, непосредственно соответствующего параметрам задачи (рисунок 1.8).

1.2	0.7	1.3	-5.0	0.16	-0.17	1.1	3.2

Рисунок 1.8 – Пример представления решения в ДЭ

Общий цикл работы ДЭ во многом схож с циклом работы в ГА, поскольку алгоритм также основан на популяционном подходе и применении генетических операторов. Однако принцип формирования новых решений и порядок применения операторов в ДЭ существенно отличаются. Основной акцент в данном методе делается на оператор мутации с разностной формулой, за которым следует скрещивание (кроссовер), формирующий пробный вектор, и селекция, реализуемая по принципу замещения.

На каждом шаге ДЭ для каждого индивида популяции формируется мутантный вектор, на основе которого в дальнейшем строится пробный кандидат решения. Мутация, определяет направление поиска и степень разнообразия в популяции. В общем случае новый вектор вычисляется с использованием линейной комбинации других особей текущей популяции [80]. Наиболее распространенные схемы мутации представлены в таблице 1.2. Существуют операторы мутации, в которых выбирается не лучший индивид из популяции, а случайный из доли лучших индивидов (*pbest*). Применение таких операторов уменьшает шансы алгоритма застрять в локальном оптимуме [81].

Таблица 1.2 – Наиболее распространенные операторы мутации в ДЭ

Название	Формула
rand/1	$v_{i,G} = x_{r_1,G} + F\left(x_{r_2,G} - x_{r_3,G}\right)$
rand/2	$v_{i,G} = x_{r_1,G} + F\left(x_{r_2,G} - x_{r_3,G}\right) + F\left(x_{r_4,G} - x_{r_5,G}\right)$
best/1	$v_{i,G} = x_{best,G} + F\left(x_{r_2,G} - x_{r_3,G}\right)$
best/2	$v_{i,G} = x_{best,G} + F\left(x_{r_2,G} - x_{r_3,G}\right) + F\left(x_{r_4,G} - x_{r_5,G}\right)$
current-to-rand/1	$v_{i,G} = x_{r_1,G} + +F\left(x_{r_2,G} - x_{r_3,G}\right)$
current-to-best/1	$v_{i,G} = x_{i,G} + F\left(x_{best,G} - x_{i,G}\right) + F\left(x_{r_1,G} - x_{r_2,G}\right)$

где $F \in [0,1]$ — это параметр алгоритма, регулирующий силу мутации, индексы r_i — случайно выбранные порядковые номера индивидов, i — номер текущего индивида, $best$ – номер лучшего в смысле ФП индивида в популяции.

После выполнения мутации сгенерированный мутантный вектор подвергается скрещиванию (recombination) с текущей особью, в результате чего формируется пробный вектор, для которого будет оцениваться пригодность. Скрещивание производится поэлементно, и для каждого компонента j-го измерения применяется следующая схема (1.7):

$$u_{j,i,G} = \{v_{j,i,G} \text{ если } rand(0,1) \le CR \; x_{j,i,G} \text{ иначе,} \qquad (1.7)$$

где $rand(0,1)$ – это случайное число, равномерно распределенное на интервале [0,1], $CR \in [0,1]$ – параметр вероятности кроссовера (crossover rate), регулирующий степень заимствования компонентов из мутантного вектора.

На финальном этапе каждого шага алгоритма ДЭ применяется селекция, определяющая, какая особь будет включена в следующее поколение.

$$x_{i,G+1} = \{u_{i,G} \text{ если } f\left(u_{i,G}\right) \le f\left(x_{i,G}\right) \; x_{i,G} \text{ иначе} \qquad (1.8)$$

Это реализуется по принципу сравнения значений ФП: если пробный вектор демонстрирует не худшее (или лучшее) значение ФП по сравнению с текущим вектором, то он сохраняется. В противном случае в популяции остается исходный родитель (1.8). ДЭ часто используется в составе гибридных алгоритмов, совмещающих ее с методами локального поиска. Такой подход позволяет повысить

точность и надежность алгоритма, особенно в задачах со сложными ландшафтами [82–84].

Несмотря на широкую применимость ЭА, одной из ключевых проблем является выбор подходящих настроек алгоритма. Эффективность конкретной конфигурации зачастую существенно зависит от особенностей решаемой задачи. Результаты, полученные на тестовых функциях, не всегда гарантируют аналогичную эффективность на реальных прикладных задачах. Данная проблема тесно связана с теоремой No Free Lunch (NFL), сформулированной Д. Вольпертом и У. Макреди [85]. Согласно этой теореме, при усреднении по всем возможным задачам оптимизации ни один алгоритм не превосходит другие по эффективности. Иными словами, универсального алгоритма, одинаково эффективного для всех задач, не существует. Одним из решений данной проблемы является внедрение методов самоадаптации ЭА в процессе поиска решения. Выделяют два варианта самоадаптации:

- *Самонастройка (self-tuning)* — это автоматизированная корректировка числовых параметров алгоритма, таких как коэффициент масштабирования, вероятность мутации или вероятность скрещивания. В рамках самонастройки значения параметров динамически изменяются в процессе оптимизации на основе информации, полученной в ходе работы алгоритма;
- *Самоконфигурация (self-configuration)* — это выбор или адаптация структурных компонентов ЭА в процессе его работы. К таким компонентам относятся, в частности, типы операторов селекции, скрещивания и мутации.

В литературе предложено множество подходов к самоадаптации. Ниже рассмотрим некоторые из наиболее известных и часто используемых методов.

Первый метод называется *PDP (Population-level Dynamic Probabilities)* и впервые упоминается в работе [86]. В этом методе каждому оператору присваивается минимальная вероятность применения (нижний порог), рассчитываемая по формуле (1.9):

$$p_{all} = \frac{0.2}{n}, \tag{1.9}$$

где n — это количество различных операторов заданного типа. Для каждого оператора также вычисляется мера его относительной эффективности (1.10):

$$r_i = \frac{success_i^2}{used_i}, \tag{1.10}$$

где $success_i$ — это число успешных применений i-го оператора (то есть случаев, когда потомок оказался более приспособленным, чем родитель); $used_i$ — общее число применений этого оператора.

На основе этой информации новая вероятность применения оператора рассчитывается следующим образом (1.11):

$$p_i = p_{all} + \left[r^i \frac{1 - np_{all}}{\sum_{j=1}^{n} r_j} \right] \tag{1.11}$$

Таким образом, более успешные операторы получают большую вероятность применения в последующих поколениях, а менее эффективные — меньшую. Метод предполагает сравнение потомка с родителем, однако конкретный способ выбора родителя (например, случайный выбор) может повлиять на точность оценки успешности, что является одним из ограничений данного подхода.

Другой метод *IDP (Individual-Level Dynamic Probabilities)*. В отличие от PDP, метод IDP предполагает адаптацию вероятностей применения операторов на уровне отдельных особей, а не всей популяции [86]. Основная идея заключается в том, чтобы для каждого индивида отслеживать число неудачных применений каждого оператора. Эти значения затем используются для вычисления индивидуализированных вероятностей применения операторов. Для каждого индивида и оператора i рассчитывается новая вероятность применения по формуле (1.12):

$$p_i = p_{all} + \left[\frac{\left(cnt_j^k + 1 - cnt_j^i \right)(100 - np_{all})}{n\left(cnt_j^k + 1 \right) - \sum_{k=1}^{n} cnt_j^k} \right], \tag{1.12}$$

где n — это число различных операторов определенного типа; $cnt_j(i)$ — число неудачных применений оператора i к индивиду j; p_{all} — нижний порог вероятности.

Таким образом, чем чаще оператор демонстрирует неэффективность при применении к конкретному индивиду, тем ниже становится вероятность его последующего использования. Метод имеет одно существенное ограничение: он предполагает наличие хотя бы одного родителя у каждой особи (для оценки успешности применения оператора). Это делает IDP неприменимым для самоконфигурирования операторов селекции, так как в момент выбора родителя потомок не существует, и, соответственно, оценка «успешности» невозможна.

Третий метод Self-Configuring Evolutionary Algorithm [87] основан на увеличении вероятности выбора оператора, который доставил наибольшую среднюю пригодность на данном поколении. Пусть z_k — число различных операторов k — го типа. Начальная вероятность выбора каждого из операторов определяется как $p_i = \frac{1}{z}$. Для каждого оператора и каждого типа вычисляется среднее значение ФП (1.13)

$$AvgFit_i = \frac{\sum_{j=1}^{n_i} f_{ij}}{n_i}, i = 1,2, \dots, z, \tag{1.13}$$

где n_i — это количество индивидов, при создании которых использовался i-й оператор. f_{ij} — пригодность j — го потомка, для которого использовался i-й оператор. На каждом поколении вероятность применения оператора, который доставил максимальное значение $AvgFit_i$ увеличивается (1.14):

$$p_i = p_i + \frac{(z-1)K}{zN}, i = 1,2, \dots, z, \tag{1.14}$$

где N — это число поколений алгоритма, K — константа, регулирующая скорость изменения вероятности. Вероятности остальных операторов пересчитываются следующим образом (1.15):

$$p_i = p_i - \frac{K}{zN}, i = 1,2, \dots, z \tag{1.15}$$

Такая корректировка необходима для сохранения нормированного

распределения вероятностей. Метод может применяться к любому типу операторов в составе ЭА, где возможен выбор из нескольких альтернативных вариантов.

Метод *SHADE (Success-History Based Parameter Adaptation for Differential Evolution)*. Алгоритм SHADE представляет собой одну из наиболее известных адаптивных модификаций ДЭ, в которой параметры масштабирования F и вероятности кроссовера CR адаптируются на основе накопленной истории успешных применений операторов (истории успеха) [81].

В алгоритме используется память фиксированной длины H, хранящая наиболее успешные значения параметров из прошлых поколений: $M_F = \{M_F^1, \ldots, M_F^H\}$ и $M_{CR} = \{M_{CR}^1, \ldots, M_{CR}^H\}$. Для каждого индивида параметры F_i и CR_i генерируются с использованием стохастических процедур (1.16):

$$CR_i = randn(M_{CR}^{r_i}, 0.1), F_i = randn(M_F^{r_i}, 0.1), \tag{1.16}$$

где $r_i \in \{1, \ldots H\}$ — это случайный индекс, а *randn*, *randc* — генераторы на основе нормального и Коши распределений соответственно. После выполнения отбора все успешные пары (F_i, CR_i), приведшие к улучшению пригодности, сохраняются в списки S_F и S_{CR}. Затем средние значения, формируемые из этих списков, обновляют элементы памяти M_F^k и M_{CR}^k по взвешенному среднему Лемера (1.17) и (1.18):

$$M_F^k = mean_{WL}(S_F); M_F^k = mean_{WL}(S_{CR}); \tag{1.17}$$

$$mean_{WL}(S) = \frac{\sum_j w_j \cdot s_j^2}{\sum_j w_j \cdot s_j};\ w_j = \frac{\Delta f_i}{\sum \Delta f_l}, \tag{1.18}$$

где Δf_i — это улучшение ФП для j-го успешного потомка. Дополнительно алгоритм SHADE получил ряд расширений, направленных на повышение эффективности при решении сложных задач. Одним из таких расширений является L-SHADE, в котором внедрен механизм линейного уменьшения размера популяции по ходу оптимизации [81]. Это позволяет на ранних этапах исследования использовать большую популяцию, а затем — постепенно концентрировать вычисления на доработке найденных решений, снижая нагрузку на вычисления.

Success History-based Adaptive Genetic Algorithm (SHAGA). SHAGA адаптирует вероятности мутации M_r и скрещивания C_r на основе истории успеха. Первоначально подход Success History-based Parameter Adaptation (SHA) был предложен для ДЭ, а позже успешно адаптирован для ГА [88]. В SHAGA текущий индивид фиксируется как первый родитель, второй выбирается с помощью турнирной селекции (размер турнира равен 2). Оператор скрещивания применяется к каждому гену отдельно, определяя, наследуется ли он от второго родителя с вероятностью, вычисляемой по формуле (1.19):

$$CR_i = randn(M_{CR}^{r_i}, 0.1), \tag{1.19}$$

где *randn* — это случайное число с нормальным распределением, а $M_{CR}^{r_i}$ — запоминаемое значение вероятности скрещивания, обновляемое на основе успешности применения на предыдущих поколениях. Оператор мутации инвертирует биты строки с вероятностью *Mr*, которая динамически обновляется по формуле (1.20):

$$MR_i = randn(M_{MR}^{r_i}, 0.1), \tag{1.20}$$

где *randc* — это генератор случайных чисел по распределению Коши, а $M_{MR}^{r_i}$ — запоминаемое значение вероятности мутации. Эти параметры адаптируются в ходе работы алгоритма: если применение оператора улучшает решение, соответствующие значения запоминаются и используются для генерации последующих. Такой механизм позволяет SHAGA гибко настраивать интенсивность мутации и скрещивания в зависимости от задачи оптимизации.

Также различные методы самоадаптации ЭА были предложены в рамках гибридных и кооперативных алгоритмов. В кооперативной коэволюции (Cooperative Coevolution) популяция разделяется на подвиды, которые эволюционируют параллельно и обмениваются лучшими решениями (или их компонентами) [89–91]. Ансамблевые подходы к ДЭ (например, EPSDE) используют пул различных стратегий мутации и кроссовера, конкурирующих в процессе поиска [92]. Метод COBRA (Collective Bionic Algorithm with Biogeography

Based Migration Operator) объединяет несколько биоинспирированных алгоритмов, адаптируя вероятности их применения в зависимости от эффективности каждого оператора [93]. Существуют также подходы, нацеленные на адаптацию отдельных операторов внутри ЭА. Так, в методе [94] вероятность мутации корректируется в зависимости от ранга индивида в популяции, что способствует улучшению надежности. Также разработаны методы самоконфигурирования, в которых механизмы селекции применяются не только для отбора родителей, но и для выбора генетических операторов, повышая адаптивность алгоритма к текущей задаче [95].

Особое внимание в области численной оптимизации заслуживают алгоритмы, продемонстрировавшие высокую эффективность в рамках международных соревнований, проводимых на ежегодной конференции IEEE Congress on Evolutionary Computation (CEC) [96]. Одним из таких является NL-SHADE-LBC (Non-Linear population size reduction SHADE with Linear Bias Change) [97]. В нем используется нелинейное уменьшение популяции, а также линейная адаптация параметров с изменяемым смещением. Дополнительно применяется усиленное ранговое давление и модифицированная стратегия работы с архивом. Другой эффективный метод — L-SRTDE (Linear Success Rule-based Topology DE), в котором коэффициент масштабирования определяется на основе доли улучшенных решений в каждом поколении, то есть уровня успешности (success rate) [98].

Помимо методов самонастройки и самоконфигурирования, существует отдельное направление — гиперэвристика. Она предполагает автоматизированное проектирование алгоритмов, ориентированных на конкретные классы задач. Это может быть достигнуто как через офлайн-обучение, так и через онлайн-адаптацию компонентов в процессе работы. Гиперэвристика применяется, например, для автоматического создания операторов селекции, мутации и декодирования, адаптированных к структуре задачи [99–101]. Эти методы позволяют создавать специализированные конфигурации, превосходящие универсальные алгоритмы по эффективности.

1.3 Применение эволюционных алгоритмов при проектировании интеллектуальных информационных технологий

Развитие ИИТ сопровождается растущей потребностью в автоматическом проектировании эффективных моделей и их параметров. В этом контексте ЭА, рассмотренные ранее как мощные методы глобальной оптимизации, находят широкое применение в задачах МО и построения интеллектуальных систем. ЭА позволяют формировать архитектуру и структуру моделей в условиях высокой размерности и отсутствия аналитических градиентов, что особенно актуально для ИНС, НЛС и ансамблей алгоритмов. В данном разделе рассматриваются существующие подходы к использованию ЭА для автоматического проектирования и адаптации структур ИИТ, включая синтез ИНС, построение правил НЛС, а также гибридизацию с другими методами МО.

Эффективно использовать ЭА для реализации процедуры автоматической настройки структуры ИНС. В таких методах структура сети закодирована в индивиде одним из двух способов [102]:

− *Прямое кодирование*. При таком подходе каждый нейрон и соединение явно указаны в генотипе индивида;

− *Косвенное кодирование*. В генотипе указаны различные операции и процедуры, с помощью которых генерируется определенная архитектура.

Одним из наиболее известных подходов, использующих прямое кодирование, является метод NEAT (NeuroEvolution of Augmenting Topologies) [29]. Этот алгоритм обеспечивает одновременную эволюцию структуры и весов сети. Эволюционный процесс начинается с минимальных топологий и постепенно усложняет их путем добавления новых нейронов и связей. Ключевая особенность NEAT — использование механизма номеров инноваций, который позволяет отслеживать изменения структуры и корректно осуществлять скрещивание между

сетями. Для защиты структурных инноваций применяется техника видообразования, разбиения популяции на виды по схожести, что способствует стабильному развитию различных архитектур. Позднее на основе NEAT были разработаны его расширения, такие как HyperNEAT, позволяющий кодировать крупные регулярные архитектуры с помощью косвенного кодирования [30], и CoDeepNEAT, в котором реализована коэволюция модулей и их соединений для глубоких ИНС. Эти методы продемонстрировали высокую эффективность в задачах автоматического проектирования ИНС для сложных задач, включая обработку изображений и управление в средах с непрерывным пространством состояний [103]. Позже нейроэволюция получила развитие в рамках более широкого направления автоматизированного поиска архитектур ИНС (Neural Architecture Search, NAS). Современные методы NAS позволяют подбирать не только весовые коэффициенты ИНС, но и структуру сети, включая количество слоев, их типы, типы ФА и связи между ними. Одним из ключевых инструментов в NAS остаются ЭА. Примером является метод Regularized Evolution, продемонстрировавший конкурентноспособную точность в задачах классификации изображений [104]. В работе [105] предлагается гибридный подход, объединяющий сверточные нейронные сети и трансформеры в рамках эволюционного поиска, что позволило достичь высокой точности (97%) на наборе данных CIFAR-10 при умеренной вычислительной нагрузке. Отдельного внимания заслуживают подходы, в которых архитектура сети формируется путем постепенного роста из минимальных примитивов в более сложные конструкции [105].

При использовании косвенного кодирования генотип представляет собой компактное и обобщенное описание архитектуры, позволяющее формировать и масштабировать сложные нейросетевые структуры с минимальными затратами. Такой подход обеспечивает высокую гибкость при генерации архитектур, и особенно хорошо подходит для задач, где важны регулярность и повторяемость элементов. Однако сложность отображения между генотипом и фенотипом может затруднять применение стандартных операторов ЭА. Так, в [106] предложен метод

оптимизации структуры ИНС с помощью ГП, обеспечивающий автоматизированный поиск архитектур с учетом эффективности их работы на обучающих данных. Архитектуры ИНС кодируются в виде бинарных деревьев. Узлы дерева делятся на функциональные (+,>), определяющие операции объединения нейронов и их последовательного соединения в слои и терминальные элементы, которые являются скрытыми и входными нейронами или группами нейронов [107]. Позднее данный подход был расширен для формирования рекуррентных нейронных сетей. Для того чтобы обеспечить возможность автоматического проектирования таких сетей, функциональное множество ГП было дополнено новыми операциями, формирующими обратные связи между элементами структуры [108–110].

Наибольшее развитие в последние годы получили подходы, сочетающие ЭА с различными техниками повышения эффективности нейросетевых архитектур. Так, в работе [111] предложен метод AE-NAS — внимание-ориентированный эволюционный поиск архитектур, использующий механизм self-attention. Данный способ описания позволяет учитывать вклад отдельных направлений внутри сетей на итоговую эффективность. В процессе эволюции приоритет отдается тем архитектурам, в которых задействованы наиболее значимые пути. Метод ENAS (Ecological Neural Architecture Search) [112] относится к гибридизации методов самоадаптации ЭА и формирования ИНС. В ENAS вместе с описанием структуры ИНС в генотип также включаются параметры, управляющие самой эволюцией. Архитектура сети представляется в виде набора гиперпараметров, таких как количество слоев, число нейронов, ФА и другие структурные параметры, а также настройки ЭА. Это позволяет оценивать и отбирать не только сами архитектуры, но и условия, при которых они были получены, что способствует формированию более эффективных решений. В работе [113] представлен ЭА для решения многокритериальных задач MO-EvoPruneDeepTL, ориентированный на сокращение избыточности в полносвязных слоях глубокой ИНС с использованием переноса обучения (transfer learning). Алгоритм оптимизирует три критерия: точность модели, ее сложность (число активных нейронов) и устойчивость к

выбросам. Таким образом, современное развитие методов автоматизированного формирования ИНС демонстрирует устойчивую тенденцию к интеграции ЭА с различными стратегиями повышения эффективности поиска.

Формирование НЛС в значительной степени сводится к построению эффективной базы правил, описывающей поведение системы в условиях неопределенности. Эта задача представляет собой сложную комбинаторную задачу разношкальной оптимизации, особенно при большом количестве входных переменных и термов, поскольку возможное число комбинаций правил стремительно возрастает.

Методы применения ЭА для построения НЛС делятся на два основных класса: Питтсбургский и Мичиганский подходы. В рамках Питтсбургского подхода каждая особь в популяции представляет собой полную базу нечетких правил, и эффективность оценивается на уровне всей системы в целом [114–116]. В отличие от него, Мичиганский подход предполагает, что каждая особь кодирует одно нечеткое правило, а оценка пригодности осуществляется на уровне отдельных правил, работающих совместно в общей базе [117,118]. Оба подхода широко применяются в современных исследованиях, а также лежат в основе множества гибридных схем и усовершенствованных архитектур, направленных на объединение их преимуществ. Дополнительно были разработаны гибридные методы, сочетающие элементы Питтсбургского и Мичиганского подходов в рамках одного алгоритма. В таких схемах может использоваться, например, Мичиганский этап для локальной оптимизации отдельных правил, а затем Питтсбургская стратегия — для глобального отбора или сокращения базы правил [119,120].

Рассмотрим подробнее один из вариантов формирования НЛС с помощью питтсбургского подхода. Каждое правило представляет собой набор предпосылок, связанных логическим оператором "И", и заключения. Структура индивидуального кодирования определяется следующей длиной бинарной строки (1.21):

$$1 + \lceil log_2(n_{sets} + 1) \rceil \cdot \{n_{var}\} + \lceil log_2(n_{outsets}) \rceil \cdot \{n_{rules}\}, \qquad (1.21)$$

где n_{sets} — это количество нечетких термов на входных переменных, n_{var} — количество входных переменных, $n_{outsets}$ — количество термов на каждой выходной переменной, n_{rules} — максимальное количество правил в базе.

Дополнительно для каждого правила вводится управляющий бит, определяющий его использование в базе, а также специальный терм игнорирования, позволяющий адаптивно управлять длиной правил и объемом базы. При таком способе кодирования особенно эффективно применение ГА, так как он позволяет проводить эволюционный поиск в пространстве решений, представленном в виде бинарных строк. Это упрощает реализацию операторов мутации и скрещивания, а также способствует эффективному исследованию сложных дискретных пространств признаков.

ФП отражает качество предсказаний, обеспечиваемое НЛС при использовании данной базы правил, с учетом штрафов за увеличение количества и длины правил. В зависимости от типа выходных данных: при дискретных выходах используется схема классификации с присвоением меток классов; при непрерывных выходах применяются методы нечеткого логического вывода, такие как Мамдани, для аппроксимации выходных значений.

Другой эффективный метод формирования НЛС описан в [106] и базируется на применении ГП для автоматического синтеза правил. В качестве базовой структуры для представления НЛС используется дерево, отображающее поэтапное разбиение входного вектора. ГП формирует саму структуру логических правил, включая их количество, вложенность и последовательность условий. Исходные входные данные подвергаются делению на подмножества с помощью операторов вида %X, %Y и, что позволяет эффективно кодировать базы правил в бинарных деревьях.

Современные подходы формирования НЛС с помощью ЭА активно развиваются в направлении многокритериальной и коэволюционной оптимизации. В [121] описан метод, сочетающий конкурентные и кооперативные стратегии для одновременной оптимизации точности и интерпретируемости правил. Метод Fuzzy MoCoCo [122] применяет кооперативную коэволюцию в рамках обучения с

подкреплением (reinforcement learning), позволяя синтезировать компактные и понятные НЛС. В [123] представлен FRULER — метод формирования TSK-правил с использованием ГА и регуляризации Elastic Net, ориентированный на регрессионные задачи. Эти решения подчеркивают стремление к созданию интерпретируемых и адаптивных моделей, способных эффективно работать в условиях ограничений и неопределенности.

Кроме перечисленных подходов, в последние годы развиваются методы автоматизированного формирования коллективов с использованием ЭА. Такие методы охватывают задачи подбора состава ансамбля, структуры объединения и параметров отдельных моделей МО. Одним из направлений является эволюционный бэггинг, при котором ГА оптимизирует состав обучающих подвыборок, увеличивая разнообразие и точность ансамбля [124]. Другой подход — эволюционный стекинг, где одновременно подбираются базовые модели и мета-классификатор [125]. Также разрабатываются методы, в которых с помощью ГП строятся математические выражения для объединения выходов моделей или древовидные структуры, кодирующие архитектуры нейросетей [126–129]. Эти методы позволяют не только улучшать точность, но и оптимизировать сам процесс построения ансамбля.

Таким образом, ЭА зарекомендовали себя как эффективный инструмент автоматизированного построения ИИТ. Они успешно применяются для оптимизации структуры ИНС, синтеза НЛС и формирования коллективов моделей, обеспечивая адаптивность и высокую производительность в условиях сложных и многомерных задач.

1.4 Библиотеки и фреймворки для применения эволюционных алгоритмов в задачах формирования ИИТ

С развитием ЭА возникает потребность в эффективных инструментах, упрощающих реализацию, настройку и применение ЭА в задачах формирования ИИТ. Для этих целей разработан ряд специализированных библиотек, фреймворков и платформ, позволяющих исследователям и практикам применять ЭА. Фреймворк DEAP (Distributed Evolutionary Algorithms in Python) предоставляет инструментарий для быстрого прототипирования ЭА [130]. DEAP позволяет пользователю самостоятельно проектировать алгоритмы с нуля, собирая их из минимальных элементов, однако применение ее на практике требует значительного времени на освоение. Более прямой подход реализован в PyGAD — библиотеке с удобным пользовательским интерфейсом для реализации ГА для решения различных задач оптимизации, включая обучение ИНС прямого распространения и рекуррентных нейронных сетей с использованием фреймворка PyTorch [131,132]. Pymoo — библиотека для многокритериальной оптимизации, предлагающая широкий набор ЭА и инструментов для визуализации [133]. EvoX ориентирован на исследование стратегий адаптации в задачах эволюционной оптимизации. Он поддерживает распределенные вычисления и расчеты на графических процессорах. В его экосистему также входят несколько дочерних проектов для применения ГП, решения задач многокритериальной оптимизации и обучения с подкреплением [134]. BaumEvA — это современная реализация ГА на Python, ориентированная на решение задач оптимизации с бинарными и комбинаторными представлениями. Библиотека предоставляет широкий выбор операторов отбора, скрещивания и мутации, а также механизмов выбора родителей [135]. GPlearn — это библиотека, реализующая алгоритм ГП. Она предназначена для решения задач регрессии и классификации данным методом [136]. PyGMO — библиотека, разработанная в рамках Европейского космического агентства (ESA) для решения глобальных и многокритериальных задач оптимизации. Поддерживает широкий выбор ЭА и

распределенные вычисления [137]. FEDOT (Framework for Evolutionary Design of Optimal Trees) — это open-source фреймворк, разработанный для автоматического построения моделей МО с использованием ЭА. Основное назначение FEDOT — автоматизированный подбор структуры и параметров конвейеров (pipelines), состоящих из различных моделей МО и методов преобразований данных [138].

Помимо Python-библиотек, активно используются и решения на других языках программирования. Например, HeuristicLab — это мощная среда разработки на C#, предназначенная для визуального конструирования и анализа ЭА. Она предоставляет готовые модули для ГА, ГП, ДЭ, а также компоненты для визуализации, настройки параметров и логирования процесса оптимизации [139]. Другой заметной платформой является ECJ (Evolutionary Computation in Java) — расширяемая Java-библиотека, широко применяемая в академических исследованиях. Она поддерживает реализацию ГП, ГА, EDA (Estimation of Distribution Algorithms) и других методов МО [140]. Библиотека jMetal, также написанная на Java, ориентирована на многокритериальную оптимизацию и предлагает множество известных ЭА, включая NSGA-II, SPEA2, MOEA/D и их модификации [141]. Для задач высокопроизводительных вычислений на C++ применяются фреймворки Evolving Objects (EO) и Paradiseo, поддерживающие гибридные алгоритмы и параллелизм [142].

Таким образом, с учетом постоянно расширяющейся сферы применения ЭА и роста сложности решаемых задач, наличие специализированных библиотек и фреймворков становится все более актуальным и необходимым условием для эффективного проектирования ИИТ.

Выводы к Главе 1

В первой главе диссертации рассмотрены ЭА как эффективные методы глобальной оптимизации, применимые к задачам высокой размерности, в том

числе при наличии множества локальных экстремумов и отсутствии аналитических выражений целевой функции. Были описаны базовые принципы работы и особенности таких алгоритмов, как ГА, ГП и ДЭ, а также описаны генетические операторы селекции, скрещивания и мутации, механизмы формирования и эволюции популяции. Особое внимание уделено методам самоадаптации ЭА, обеспечивающим автоматическую настройку числовых параметров и конфигурации алгоритмов в процессе поиска решения. Рассмотрены современные подходы к самонастройке и самоконфигурации, а также гибридные и коэволюционные алгоритмы, демонстрирующие высокую эффективность в задачах оптимизации. Также в главе представлен систематический обзор ИИТ и моделей МО, включая ИНС, НЛС и ансамблевые методы. Проанализированы их архитектуры, принципы обучения, особенности применения, а также подходы к автоматическому проектированию структур ИИТ с использованием ЭА. На основе проведенного анализа обоснована целесообразность использования ЭА как универсального инструмента автоматизации проектирования и настройки интеллектуальных систем в условиях ограниченности априорной информации, высокой размерности и требований к интерпретируемости моделей. Во второй главе будет рассмотрена задача повышения эффективности ЭА за счет использования механизмов адаптации параметров на основе истории успеха.

2 Разработка и исследование самоконфигурируемых эволюционных алгоритмов оптимизации с адаптацией на основе истории успеха

2.1 Разработка и реализация самоконфигурируемых генетических алгоритмов оптимизации с адаптацией на основе истории успеха

Ранее в первой главе ГА рассматривался как один из базовых методов эволюционной оптимизации, эффективный при решении задач с вещественными, целочисленными, бинарными и разношкальными переменными, высокой размерностью и отсутствием аналитического выражения целевой функции. Особенностью ГА является представление решений в виде бинарной строки и последующий поиск оптимальной точки, соответствующей экстремуму целевой функции. Цикл работы ГА включает три этапа: селекция, скрещивание и мутация.

На первом шаге с использованием операторов селекции отбираются решения-родители, которые будут участвовать в следующих этапах эволюции. Существует три общепринятых оператора селекции:

- *пропорциональная* — вероятность выбора индивида пропорциональна его значению ФП;
- *ранговая* — вероятность выбора индивида пропорциональна его рангу по значению ФП;
- *турнирная* — выбирается индивид с лучшим значением ФП из случайно сформированной подгруппы.

На следующем шаге, скрещивании, отобранные решения комбинируются для передачи полезной информации потомкам. Различают следующие виды скрещивания в ГА:

- *одноточечное* — хромосомы родителей разделяются в одной случайной точке, и части хромосом меняются местами между родителями;
- *двухточечное* — хромосомы родителей разделяются в двух случайных

точках, и части хромосом меняются местами между родителями;

- *равномерное* — каждый ген потомка выбирается случайно (равновероятно) из генов одного из родителей.

На рисунке 2.1 представлены примеры применения генетических операторов скрещивания для ГА.

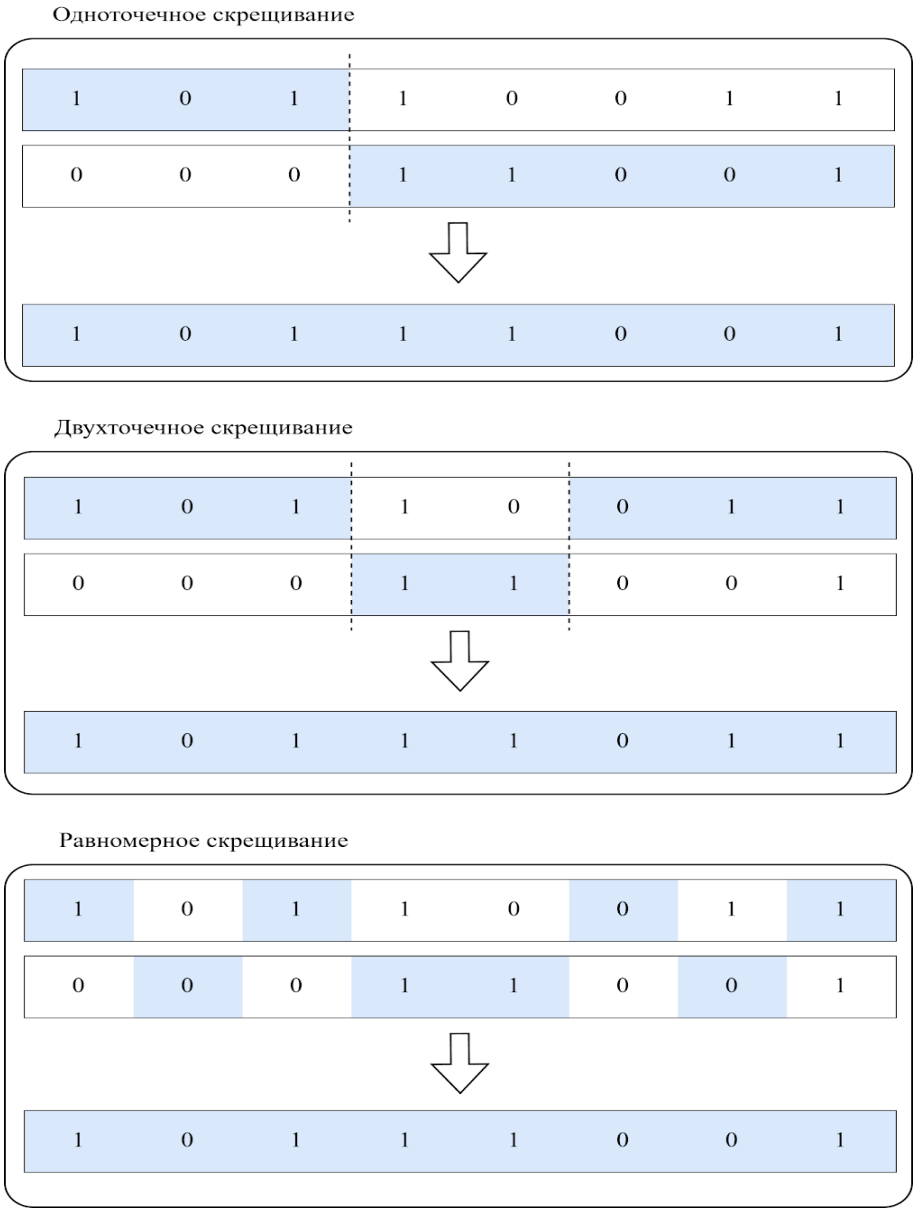

Рисунок 2.1 – Примеры применения вариантов генетического оператора скрещивания для ГА

Завершающим шагом является мутация, которая обычно вносит небольшие случайные изменения в решения, поддерживая таким образом разнообразие в

популяции. В ГА, где решения закодированы в виде бинарной строки, оператор мутации инвертирует биты с определенной вероятностью. На рисунке 2.2 представлен пример применения оператора мутации. Принято различать три вида мутации в зависимости от ее интенсивности:

- *средняя* — вероятность мутации равна обратной длине хромосомы;
- *слабая* — вероятность мутации в три раза меньше, чем средняя;
- *сильная* — вероятность мутации в три раза выше средней.

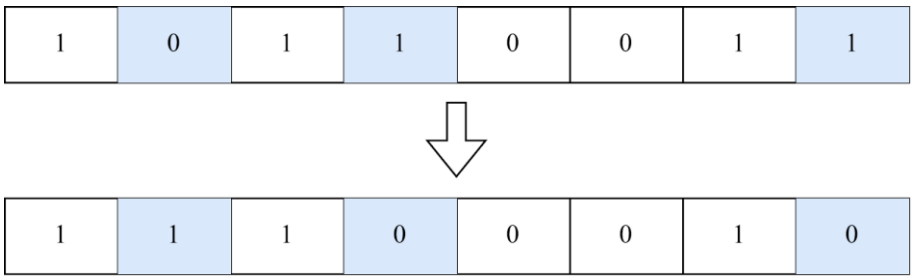

Рисунок 2.2 – Пример применения оператора мутации в ГА

В случае применения ГА для работы с вещественными или целочисленными переменными каждую переменную необходимо закодировать в бинарный вид. Прямой способ – разбить число на интервалы и составить соотношение между каждым интервалом и бинарной комбинацией [143]. Пусть x_{min} и x_{max} — это заданные исследователем минимальная и максимальная границы значений переменной, а ε — шаг разбиения, тогда количество возможных значений переменной равно (2.1):

$$n_{parts} = 1 + \frac{(x_{max} - x_{min})}{\varepsilon} \tag{2.1}$$

Необходимое для кодировки количество бит вычисляется как округленный в большую сторону логарифм n_{parts} по основанию 2 (2.2):

$$n_{bit} = Int_{+}\left(log_2 n_{parts}\right) \tag{2.2}$$

Из-за округления фактическая точность разбиения будет равна (2.3):

$$\tilde{\varepsilon} = \frac{(x_{max} - x_{min})}{2^n - 1}, \tag{2.3}$$

Для перевода бинарной строки в вещественное число можно воспользоваться следующими правилами: стандартным преобразованием из двоичного кода в десятичный получают значение x_{10}; по формуле (2.4) вычисляют вещественное значение переменной.

$$x = x_{min} + \tilde{\varepsilon} x_{10} \quad (2.4)$$

Значение ФП индивида рассчитывается по преобразованным переменным. Для повышения эффективности поиска и снижения вероятности больших скачков в фенотипе при малых изменениях генотипа часто используется код Грея. Кроме бинарного кодирования и кодирования с использованием кода Грея, в ЭА могут применяться и альтернативные схемы представления.

Ранее в главе 1 также рассматривался алгоритм SHAGA (Success History-based Adaptive Genetic Algorithm), особенностью которого является переработанный цикл работы, схожий с циклом алгоритма ДЭ. Однако в оригинальной версии, предложенной авторами, используются только равномерное скрещивание и турнирная селекция с размером турнира, равным 2. Эффективность применения других генетических операторов в SHAGA остается недостаточно исследованной. Кроме того, SHAGA выполняет только самонастройку числовых параметров, без адаптивного выбора вариантов генетических операторов. В связи с этим представляется перспективным доработка алгоритма SHAGA, направленная на построение варианта алгоритма, способного использовать схему адаптации числовых параметров SHA (Success History-based Adaptation) в сочетании с механизмами самоконфигурирования и с различными вариантами генетических операторов.

Цикл работы ГА изменен следующим образом: каждый индивид популяции рассматривается в качестве текущего (current) решения. На его основе формируется потомок, который заменяет текущее решение, если его значение ФП лучше.

Селекция является первым шагом цикла и сохраняется в классическом виде с одним уточнением: текущий индивид включается в процесс создания потомка в качестве одного из родителей, тогда как остальные родители выбираются из

популяции. При этом могут быть использованы различные операторы селекции, включая пропорциональную, ранговую и турнирную селекцию. Следует учитывать, что в данном случае необходимо выбирать на одного родителя меньше.

В SHAGA вероятность скрещивания определяет интенсивность изменений: высокие значения увеличивают вариативность и способствуют глобальному поиску, в то время как низкие значения делают поиск более локальным. В операторах равномерного скрещивания с селективным давлением, предложенных в [144], вероятность скрещивания применяется иначе: каждый ген наследуется случайно от одного из родителей, при этом вероятность выбора зависит от значений ФП родителей. Родители с более высокими значениями ФП имеют больше шансов передать свои гены потомку.

Оператор скрещивания в SHAGA можно модифицировать таким образом, чтобы учитывать как параметр *CR*, так и значения ФП родителей, одновременно регулируя интенсивность изменений и применяя селективное давление. Основная идея модификации заключается в том, что наследование каждого гена потомком происходит в два этапа:

Этап 1 — выбор между текущим решением и другими родителями: на этом этапе с вероятностью *CR* определяется, будет ли ген унаследован от первого родителя (текущего решения) или от других родителей. Если ген наследуется от первого родителя, то переходим к следующему гену (аналогично оригинальному SHAGA). Иначе случае переход осуществляется на этап 2.

Этап 2 — выбор среди оставшихся родителей: после того как определено, что ген наследуется не от текущего решения, производится выбор среди оставшихся родителей. Вероятность выбора между этими родителями зависит от их значений ФП, аналогично операторам из [144].

Подробное описание модифицированного оператора скрещивания с селективным давлением для SHAGA представлено в псевдокоде 1:

Псевдокод 1. Модифицированный оператор скрещивания для SHAGA с селективным давлением

```
BEGIN CROSSOVER (parent_1, other_parents, fitness_values, CR)
  offspring_genotype = new vector(length_of(parent_1))

  FOR each gene i from 1 to length_of(parent_1)
    // Stage 1
    IF random_number < CR THEN
      offspring_genotype[i] = parent_1[i]
      CONTINUE // to the next gene
    ELSE
      // Stage 2
      offspring_genotype[i] = SELECT(other_parents, fitness_values)
    END IF

  END FOR

  RETURN offspring_genotype
END CROSSOVER
```

Предлагаемая модификация оператора скрещивания для SHAGA позволяет использовать многородительское скрещивание, настраивать его интенсивность через параметр CR и учитывать селективное давление на этапе скрещивания. Если в процедуре участвуют только два родителя, оператор работает также, как в оригинальном SHAGA. Кроме того, можно использовать и другие варианты оператора скрещивания: одноточечное и двухточечное. В этом случае процедура не отличается от стандартной, но инициируется с вероятностью CR. Если скрещивания не происходит, оператор возвращает текущее решение.

Авторы [144] определили, что оптимальное число родителей для большинства операторов равномерного скрещивания составляет 2 и 7, а для турнирного скрещивания — 3 и 7. Однако в этом алгоритме на этапе скрещивания используется дополнительное селективное давление с помощью оператора селекции на этапе 2, что увеличивает общее число родителей на одного по сравнению с оригинальной реализацией. В настоящем исследовании используются следующие операторы скрещивания:

– равномерное, число родителей 2 (оператор из оригинального алгоритма SHAGA);

– одноточечное скрещивание;

– двухточечное скрещивание;

– равномерное равновероятное, число родителей 3 или 8;

– равномерное пропорциональное, число родителей 3 или 8;

– равномерное ранговое, число родителей 3 или 8;

– равномерное турнирное, число родителей 4 или 8 (размер турниров 2).

Мутация выполняется на завершающем этапе цикла работы алгоритма и полностью аналогична оператору мутации, применяемому в оригинальной версии SHAGA. При этом вероятность ее применения задается не фиксированным значением, а определяется параметром, который динамически настраивается в процессе поиска на основе истории успеха.

ГА с адаптацией параметров на основе истории успеха, использующий разные реализации операторов скрещивания и селекции, будет далее называться конфигурируемым SHAGA (Configuring SHAGA, CSHAGA).

Предложенный алгоритм CHSHAGA обладает 60 различными конфигурациями, что создает сложность в выборе оптимальных настроек для каждой новой задачи оптимизации. Решение данной проблемы возможно через интеграции методов самоконфигурирования, таких как SelfCEA или PDPEA, описанных в главе 1. Объединив все описанные модификации — изменение цикла работы ГА, модифицированный оператор скрещивания, адаптацию на основе истории успеха и методы самоконфигурирования — получается следующий алгоритм: самоконфигурируемый ГА с адаптацией на основе истории успеха [145]. Подробное описание данного алгоритма представлено в псевдокоде 2.

Псевдокод 2. Самоконфигурируемый генетический алгоритм с адаптацией на основе истории успеха

1. Инициализация.
- *1.1. Сгенерировать начальную популяцию индивидов.*
- *1.2. Вычислить значение ФП для каждого индивида.*
- *1.3. Инициализировать историю параметров:*
  - *1.3.1. Вектор MMR длиной M_size (для вероятности мутации) заполнить значениями 0.1.*
  - *1.3.2. Вектор MCR длиной M_size (для вероятности скрещивания) заполнить значениями 0.9.*
  - *1.3.3. Установить индекс истории k = 0,*
- *1.4. Инициализировать вероятности применения операторов для каждого типа:*
  - *1.4.1. P_sel (операторы селекции) – равновероятно по всем вариантам.*
  - *1.4.2. P_cross (операторы скрещивания) – равновероятно по всем вариантам.*
  - *1.4.3. P_mut (операторы мутации) – равновероятно по всем вариантам.*

2. Основной цикл (для каждого поколения):
- *2.1. Для каждого индивида i:*
  - *2.1.1. Случайно выбрать индекс r из диапазона [0, M_size].*
  - *2.1.2. Задать MR_r, используя распределение Коши с центром MMR[r] и масштабом 0.1.*
  - *2.1.3. Задать CR_r, используя нормальное распределение с центром MCR[r] и со стандартным отклонением 0.1.*
  - *2.1.4. Выбрать вариант оператора селекции с помощью вероятностей P_sel.*
  - *2.1.5. Выбрать вариант оператора скрещивания с помощью вероятностей P_cross.*
  - *2.1.6. Выбрать вариант оператора мутации с помощью вероятностей P_mut.*
  - *2.1.7. Применить выбранный оператор селекции для отбора родителей.*
  - *2.1.8. Применить выбранный оператор скрещивания к i-го индивиду (первому родителю) и родителям, формируя потомка с вероятностью CR_r.*
  - *2.1.9. Применить выбранный оператор мутации к полученному потомку с вероятностью MR_r.*
  - *2.1.10. Вычислить значение ФП потомка.*
- *2.2. Замещение:*
  - *2.2.1. Для каждого индивида i: если значение ФП потомка лучше, чем i-го индивида, заменить i-го индивида потомком.*
- *2.3. Обновление истории параметров:*
  - *2.3.1. Для всех индивидов, у которых произошла замена, собрать использованные значения MR и CR, а также величины улучшения значений ФП.*
  - *2.3.2. Обновить MMR[k] и MCR[k] с использованием взвешенного среднего успешных значений.*
  - *2.3.3. Увеличить k = k+1 или 0, если k > M_size.*
- *2.4. Обновление вероятностей применения операторов:*
  - *2.4.1. Обновить значения вероятностей применения генетических операторов P_sel, P_cross и P_mut, используя метод самоконфигурирования.*
- *2.5. Обновить глобально лучшего индивида.*

3. Завершение:
- *3.1. Вернуть лучшего найденного индивида и статистику работы алгоритма.*

Таким образом, в результате проведенных модификаций был разработан самоадаптивный ГА, отличающийся измененным циклом работы и комплексной

модифицированной процедурой скрещивания, позволяющей адаптировать интенсивность скрещивания, применять селективное давление на этапе скрещивания и многородительское скрещивание, а также интеграцией механизма адаптации вероятностей операторов скрещивания и мутации на основе истории успеха. При этом, в зависимости от применяемого метода самоконфигурирования, такой ГА реализован в двух вариантах: SelfCSHAGA (Self-Configuring Success History-based Adaptation Genetic Algorithm) и PDPSHAGA (Population-Level Dynamic Probabilities Success History-based Adaptation Genetic Algorithm), которые различаются механизмом настройки вероятностей применения генетических операторов.

2.2 Исследование эффективности самоконфигурируемых генетических алгоритмов оптимизации на основе истории успеха

Для исследования алгоритмов CSHAGA, PDPSHAGA и SelfCSHAGA и сравнения с другими ГА было выбрано два набора задач:

1) набор сложных тестовых функций из CEC2005, CEC2013 и CEC2014 [146–148], содержащий задачи вещественной оптимизации различной размерности от 2 до 30 переменных (в таблице 2.1 приведены функции, их характеристики: размерность пространства, границы переменных, и параметры алгоритма: количество итераций и размер популяции для оптимизации целевой функции);

2) набор сложных задач глобальной псевдобулевой оптимизации из [149], содержащий функции F11-F16, которые характеризуются множеством локальных оптимумов (в таблице 2.2 представлены характеристики функций и параметры ГА).

Для проведения экспериментов использовалась специализированная программная система, разработанная для применения самоконфигурируемого ГА с адаптацией на основе истории успеха [150]. Программная система прошла

регистрацию в Федеральной службе по интеллектуальной собственности (Роспатент).

Таблица 2.1 – Функции и параметры для набора задач вещественной оптимизации

Функция	**Разм.**	**Л. граница**	**Пр. граница**	**Кол. итераций/**	**Размер популяции**
ShiftedRosenbrock	2	-100	100	270	270
ShiftedRotatedGriewank	2	-1000	1000	650	650
ShiftedExpandedGriewankRosenbrock	2	-3	1	110	110
RotatedVersionHybridCompositionFunction1	2	-5	5	130	130
RotatedVersionHybridCompositionFunction1 Noise	2	-5	5	130	130
RotatedHybridCompositionFunction	2	-5	5	250	250
HybridCompositionFunction3	2	-5	5	477	477
HybridCompositionFunction3H	2	-5	5	477	477
NonContinuousHybridCompositionFunction3	2	-5	5	477	477
HybridCompositionFunction4	2	-5	5	799	799
HybridCompositionFunction4withoutbounds	2	-10	10	1000	1000
Rosenbrock	2	-2.048	2.048	100	100
ExpandedScaffers_F6	2	-100	100	100	100

Продолжение таблицы 2.1

Weierstrass	5	-1	1	1157	1157
ShiftedSphere	10	-100	100	125	125
ShiftedSchwefe1_2	10	-100	100	728	728
ShiftedSchwefe1_2With Noise	10	-100	100	757	757
ShiftedRastrigin	10	-5	5	470	470
ShiftedRotatedRastrigin	10	-5	5	463	463
HybridCompositionFunction1	10	-5	5	799	799
Sphere	30	-5.12	5.12	210	210
HighConditionedElliptic	30	-100	100	355	355
Griewank	30	-600	600	600	600
Ackley	30	-32.768	32.768	247	247
Rastrigin	30	-5.12	5.12	799	799

Таблица 2.2 – Функции и параметры для набора задач псевдобулевой оптимизации

Функция	**Размерность**	**Количество итераций**	**Размер популяции**
OneMax	100	200	100
F11	30	200	250
F12	30	200	250
F13	24	200	250
F14	30	200	250
F15	30	200	250
F16	40	200	250

Поскольку ЭА являются стохастическими и результаты варьируются при каждом запуске, для их оценки использовалось множества запусков. Каждая конфигурация тестировалась 100 раз для каждой функции. Считалось, что ГА нашел решение задачи вещественной оптимизации, если точность по положению точки составляла 0.01, а для псевдобулевых задач — если бинарная строка полностью совпадала с оптимальным решением. Метрикой для сравнения алгоритмов выступала надежность — доля найденных с заданной точностью решений от общего числа запусков. Объем ресурсов для работы ГА выбирался для каждой функции так, чтобы надежность алгоритма не достигала 0 или 1, иначе сравнение алгоритмов становилось невозможным. Для псевдобулевых задач из [149] применялся объем ресурсов, указанный авторами.

Различия в метриках проверялись на статистическую значимость с помощью U-критерия Манна–Уитни, который подходит для малых выборок и данных, не соответствующих нормальному распределению. Уровень значимости составлял 0.05. Для каждого ГА надежность рассчитывалась 10 раз, выборка состояла из 10 измерений, полученных из 100 независимых запусков. В исследовании использовались следующие ГА:

1. Genetic Algorithm — ГА с операторами скрещивания с селективным давлением. Всего 165 конфигураций;

2. SelfCGA — самоконфигурируемая версия ГА на основе SelfCEA с расширенным набором операторов, включая операторы равномерного скрещивания с селективным давлением [144];

3. PDPGA — PDP-версия ГА, включая операторы равномерного скрещивания с селективным давлением [86];

4. SHAGA — реализация алгоритма SHAGA из оригинальной статьи [88];

5. CSHAGA — предлагаемая расширенная версия SHAGA с различными конфигурациями генетических операторов. Всего 60 конфигураций;

6. SelfCSHAGA — предлагаемая самоконфигурируемая версия CSHAGA на основе SelfCEA;

7. PDPSHAGA — предлагаемая самоконфигурируемая версия CSHAGA на основе PDPEA.

На рисунке 2.3 представлены усредненные значения надежности для шести различных вариантов ГА на задачах оптимизации с вещественными переменными из таблице 2.1. Если алгоритм не является самоадаптивным (например, GA или CSHAGA), то для него усредняются значения надежности по всем конфигурациям, а также фиксируется максимальная надежность, достигнутая на одной (лучшей) конфигурации.

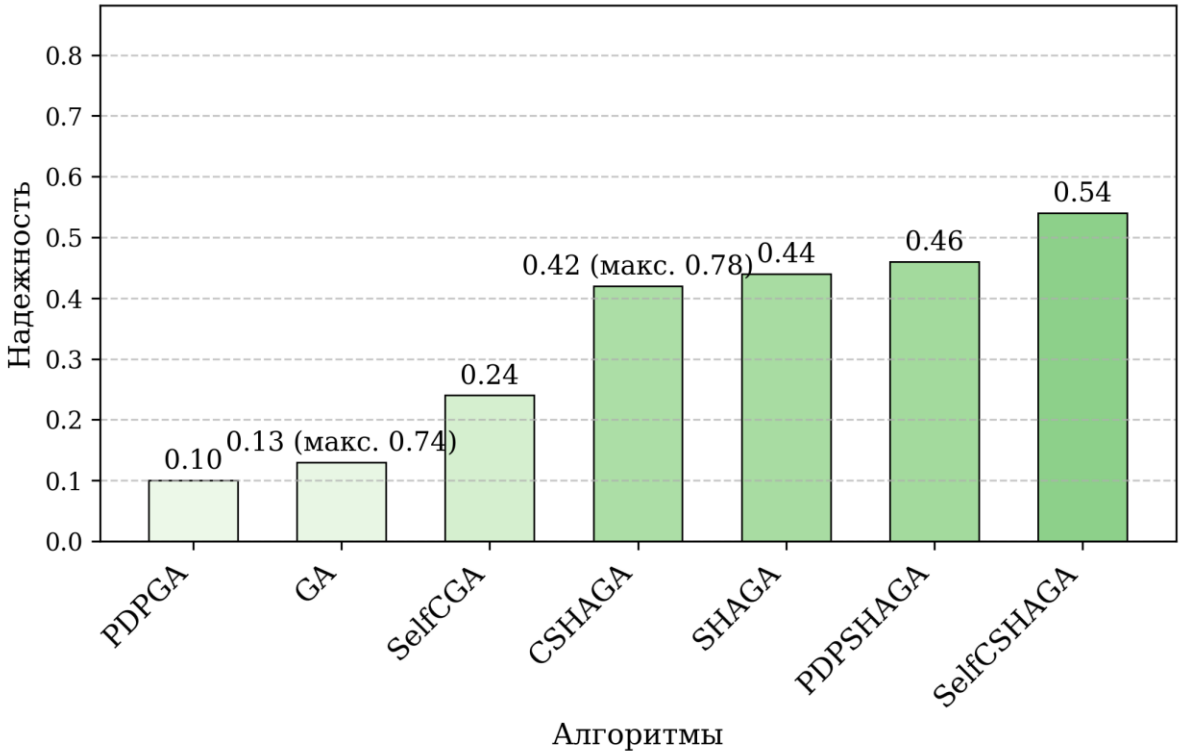

Рисунок 2.3 – Усредненные по 100 запускам значения надежности для предложенных и известных ГА на задачах вещественной оптимизации

Согласно рисунку 2.3, SelfCSHAGA достиг наивысшей средней надежности среди всех исследуемых алгоритмов. CSHAGA показал лучшие результаты, чем классический Genetic Algorithm, но его средняя надежность оказалась ниже, чем у SelfCSHAGA, PDPSHAGA и SHAGA. Таким образом, при отсутствии информации о свойствах оптимизируемой функции предпочтительнее использовать SelfCSHAGA, PDPSHAGA или SHAGA вместо CSHAGA с произвольной конфигурацией.

На рисунке 2.4 представлены результаты статистического теста для сравнения алгоритмов SelfCSHAGA и PDPSHAGA с другими ГП. Диаграммы показывают сравнение SelfCSHAGA и PDPSHAGA с SelfCGA, PDPGA и SHAGA на задачах вещественной оптимизации. Результаты разделены на три категории: «превосходит» (зеленый сектор) — количество функций, где SelfCSHAGA был лучше; «без различий» (серый сектор) — различия статистически незначимы; «уступает» (красный сектор) — другой алгоритм показал лучшие результаты.

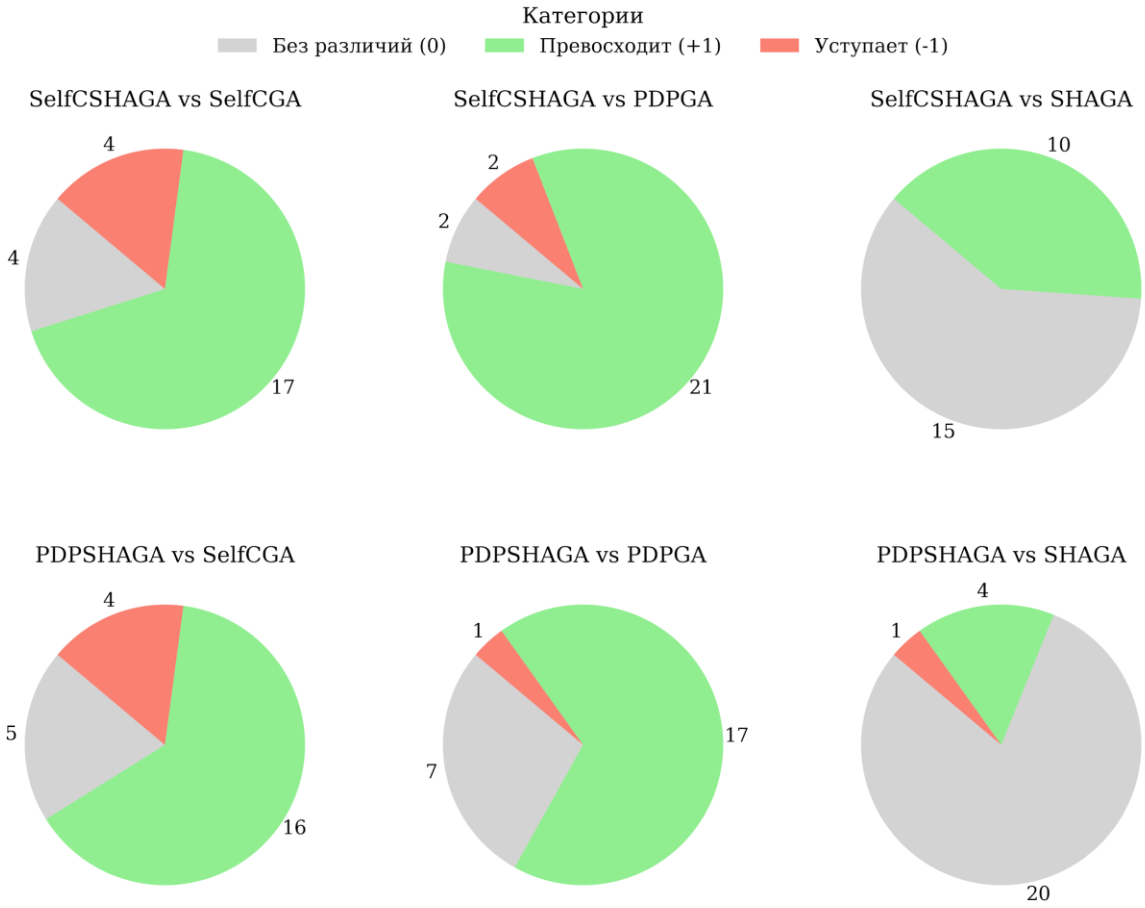

Рисунок 2.4 – Результаты сравнения предложенных алгоритмов с аналогичными на задачах вещественной оптимизации с использованием статистического критерия

Как видно на рисунке 2.4, предложенные алгоритмы SelfCSHAGA и PDPSHAGA либо превосходят, либо демонстрируют сопоставимую эффективность по сравнению с существующими самонастраиваемыми ГА на большинстве задач вещественной оптимизации.

Тем не менее, для задач с вещественным представлением решений более подходящими являются самонастраиваемые алгоритмы ДЭ — победители мировых соревнований по эволюционной оптимизации [98]. Основное преимущество ГА заключается в их универсальности и возможности решать задачи псевдобулевой, дискретной и разношкальной оптимизации.

В связи с этим проведено дополнительное исследование на задачах псевдобулевой оптимизации из таблице 2.2. Результаты представлены на рисунке 2.5.

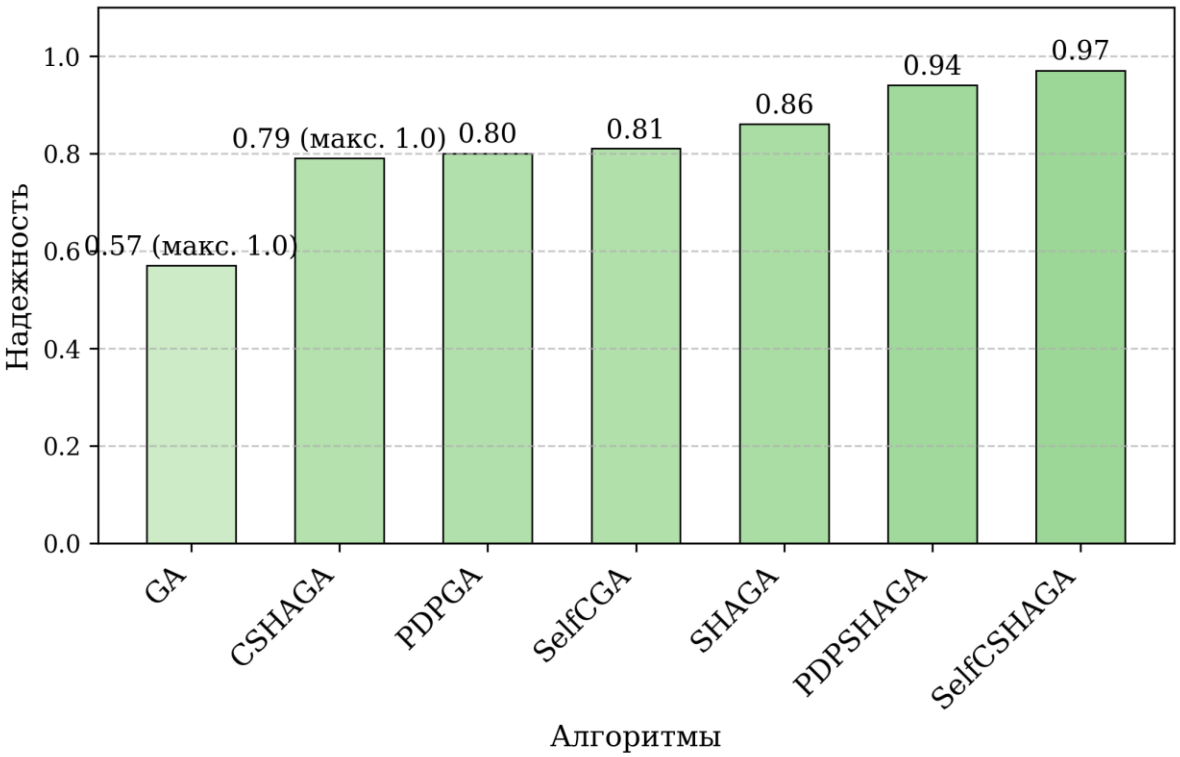

Рисунок 2.5 – Усредненные по 100 запускам значения надежности для предложенных и известных ГА на задачах псевдобулевой оптимизации

Результаты надежности и статистического сравнения показаны на рисунке 2.6.

Дополнительно предлагаемый алгоритм был протестирован на задаче формирования нечетко-логических классификаторов (НЛК), которая представляет собой задачу разношкальной оптимизации. В ней одновременно оптимизируются бинарные переменные, определяющие включение или исключение правил и

предпосылок, и дискретные значения — номера нечетких термов. Для оценки эффективности были выбраны задачи классификации из репозитория UCI Machine Learning Repository: Breast Cancer Wisconsin (BC), Credit Risk (CR), Banknote Authentication (BA), TwoNorm (TN) и RingNorm (RN) [151]. Для всех алгоритмов использовались одинаковые параметры: число итераций — 300, размер популяции — 300. Усредненные по 20 запускам значения F1-меры, достигнутые при формировании НЛК предложенными и известными самоадаптивными ГА представлены в таблице 2.3 (в данной и последующих таблицах наилучшие значения метрик по каждой задаче выделены полужирным шрифтом).

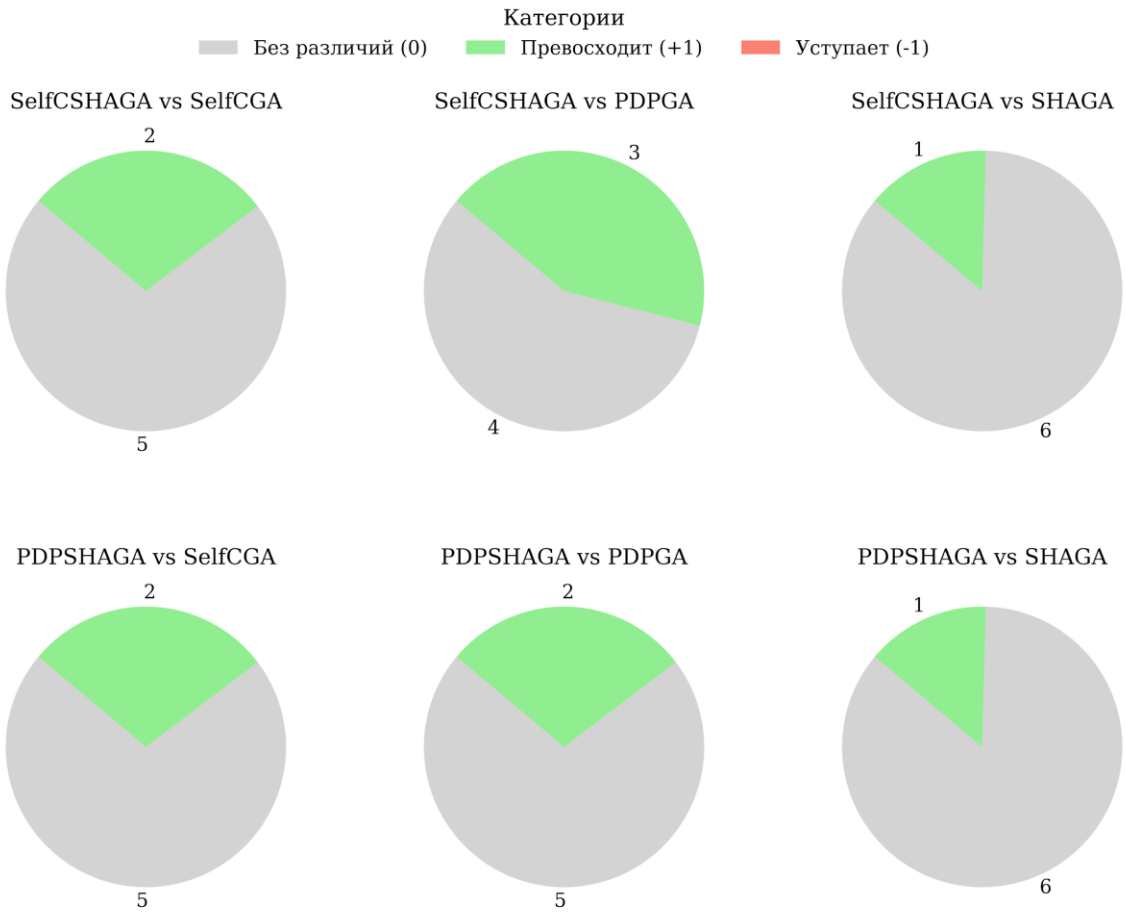

Рисунок 2.6 – Результаты сравнения предложенных алгоритмов с аналогичными на задачах псевдобулевой оптимизации с использованием статистического критерия

В результате SelfCSHAGA продемонстрировал наименьший средний ранг (2.0) среди всех алгоритмов и лучшие значения F1-меры на трех из пяти задач. При

этом, согласно результатам статистического тестирования (по U-критерию Манна–Уитни, уровень значимости 0.05), статистически значимые отличия результатов SelfCSHAGA по сравнению с аналогичными алгоритмами были зафиксированы только для одной из пяти задач (BA); на остальных задачах различия между результатами алгоритмов оказались статистически незначимыми.

Таблица 2.3 – Усредненные по 20 запускам значения F1-меры, достигнутые при формировании НЛК предложенными и известными самоадаптивными ГА

Алгоритм	BC	CR	BA	TN	RN	Ср. ранг
SelfCGA	0.9385	0.8827	0.9713	0.9545	0.9190	3.6
PDPGA	**0.9436**	**0.8967**	0.9716	0.9565	0.9225	2.2
SelfCSHAGA (предлагаемый)	0.9378	0.8863	**0.9809**	**0.9584**	**0.9267**	**2.0**
PDPSHAGA (предлагаемый)	0.9379	0.8879	0.9733	0.9566	0.9241	2.2

Таким образом, по результатам проведенных исследований на трех различных задачах — вещественной, псевдобулевой оптимизации и задаче формирования НЛК — предложенные алгоритмы SelfCSHAGA и PDPSHAGA стабильно демонстрируют либо лучшую, либо сопоставимую эффективность по сравнению с существующими аналогами.

2.3 Разработка и реализация самокофнигурируемых алгоритмов генетического программирования с адаптацией на основе истории успеха

Алгоритм ГП во многом структурно схож с ГА, однако между ними существует принципиальное отличие, связанное с формой представления решений.

В то время как в ГА индивиды, как правило, кодируются в виде бинарных строк фиксированной длины, в ГП используется более гибкий формат представления, допускающий различную структуру и размер генотипа. Наиболее широко распространенным является древовидное представление, в котором решения моделируются в виде синтаксических деревьев, составленных из функциональных и терминальных элементов. Тем не менее, в практике ГП применяются и альтернативные формы представления, включая линейные хромосомы (Linear Genetic Programming, LGP), представление в виде графов (например, Cartesian Genetic Programming, CGP). Выбор формы представления определяется спецификой задачи, требуемыми операциями над программами и желаемыми свойствами пространства решений. В рамках данной работы рассматривается классическая модель ГП с деревообразным представлением решений.

Деревом называется направленный граф, в котором каждая последующая вершина связана с одной и только одной предыдущей. Вершины дерева являются элементами одного из двух множеств:

- Множество всех возможных внутренних вершин дерева называется функциональным множеством F.
- Множество всех возможных внешних вершин дерева называется терминальным множеством T.

Элементы функционального множества обычно являются рабочими блоками программы (процедурами, функциями), в то время как в терминальном множестве находятся входные данные (переменные и константы). Пример бинарного дерева в методе ГП представлен на рисунке 2.7. Желтым цветом обозначены элементы из функционального множества, а зеленым из терминального.

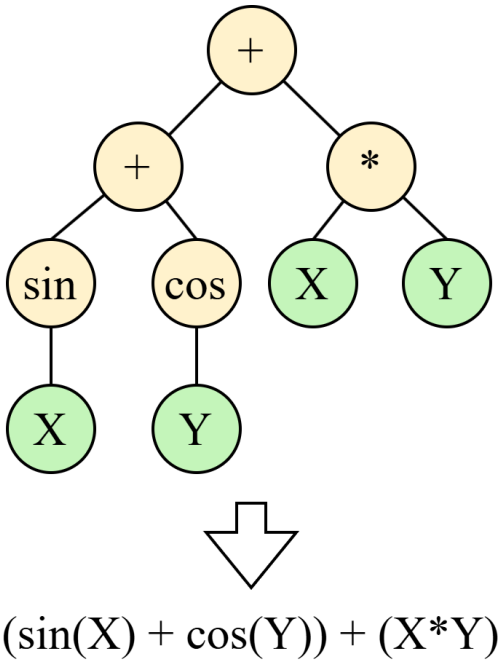

Рисунок 2.7 – Пример представления решения в генетическом программировании

Объединение функционального и терминального множеств называется универсальным множеством *C*. Для решения поставленной задачи методом ГП множества *F* и *T* должны удовлетворять следующим требованиям:

- *Замкнутость*. Для обеспечения замкнутости любой элемент из множества *F* должен принимать в качестве аргумента любой элемент множества *C*.
- *Достаточность*. Для обеспечения достаточности элементы множества *C* должны быть выбраны таким образом, чтобы можно было бы решить поставленную задачу.

Цикл работы ГП во многом повторяет схему работы ГА, однако отличаются генетические операции и форма решения:

1. На основе множеств *F* и *T*, формируется начальная популяция.
2. Вычисляются значения ФП решений в популяции.
3. С помощью операторов селекции, скрещивания и мутации формируется новое поколение.
4. Если выполнен один из критериев останова, выполнение алгоритма завершается, иначе переход к шагу 2.

Для инициализации популяции используются следующие методы выращивания деревьев:

- *Полный метод*. Задается глубина (максимальный уровень) дерева *d*. В качестве вершин на глубине меньше *d* выбираются элементы из функционального множества. на уровне глубины *d* элементы выбираются из терминального множества.
- *Метод выращивания*. Задается глубина дерева *d*. На глубине меньше *d* выбираются элементы как из функционального, так и из терминального множества. В случае выбора элемента из терминального множества рост текущей ветки заканчивается. На уровне *d* элементы выбираются из терминального множества.
- *Комбинированный метод*. Одна часть популяции выращивается полным методом, а другая методом роста.

Далее описываются генетические операторы алгоритма ГП. Операторы селекции, применяемые в методе ГП, аналогичны схемам селекции, применяемым в ГА. Скрещивание. Существует три варианта оператора скрещивания в ГП:

- *Стандартное скрещивание*. У каждого из родителей выбирается точка скрещивания. Родители обмениваются генами, находящимися ниже данных точек. Полученная пара является потомками.
- *Одноточечное скрещивание*. У родительской пары выбирается одна общая точка скрещивания, после чего скрещивание осуществляется по стандартной схеме.
- *Равномерное скрещивание*. Скрещивание осуществляется на уровне отдельных узлов: для каждого узла в общей области деревьев случайно выбирается, от какого родителя он будет унаследован. Такой подход обеспечивает более мелкомасштабную рекомбинацию и способствует увеличению разнообразия потомков.

Общая точка выбирается в общей области деревьев родителей, получаемой наложением одного дерева на другое. На рисунке 2.8 представлены примеры применения различных вариантов генетического оператора скрещивания.

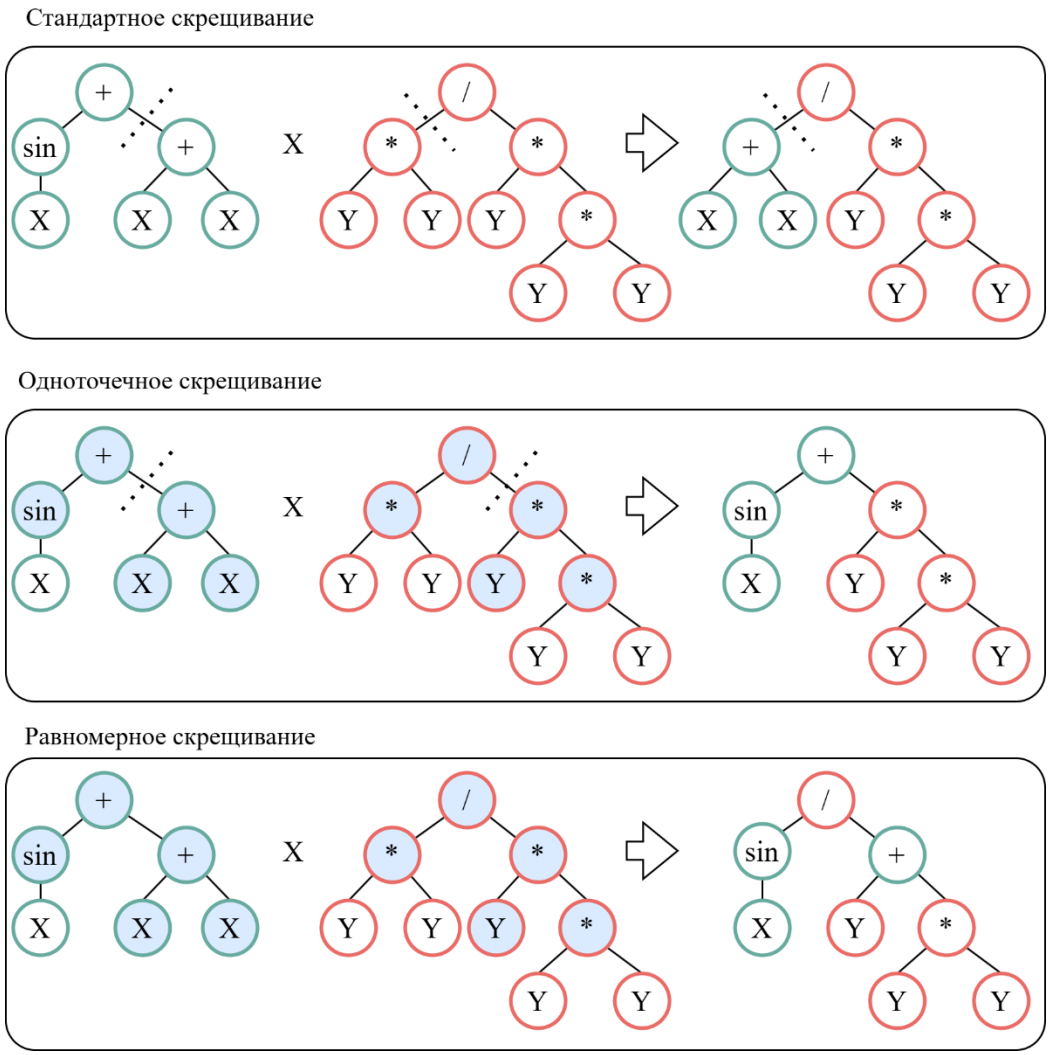

Рисунок 2.8 – Примеры применения вариантов скрещивания в генетическом программировании

Оператор мутации в ГП вносит случайные изменения в структуру дерева, моделирующего решение. Мутация обеспечивает сохранение генетического разнообразия в популяции и способствует исследованию новых областей пространства решений. В зависимости от стратегии изменения различают следующие основные схемы мутации:

- *одноточечная мутация (one-point mutation)*. Случайный узел дерева заменяется на элемент того же типа (функциональный или терминальный), выбранный случайным образом;
- *мутация поддерева (subtree mutation)*. Случайно выбранное поддерево в структуре решения полностью заменяется на новое поддерево, сгенерированное методами инициализации (например, методом роста);
- *обменная мутация (swap mutation)*. В дереве выбираются два совместимых по типу узла (например, два терминала или две функции), которые меняются местами;
- *сжатие (shrink mutation)*. Случайное поддерево в структуре решения заменяется на один терминальный узел.

Примеры применения различных операторов мутации представлены на рисунке 2.9.

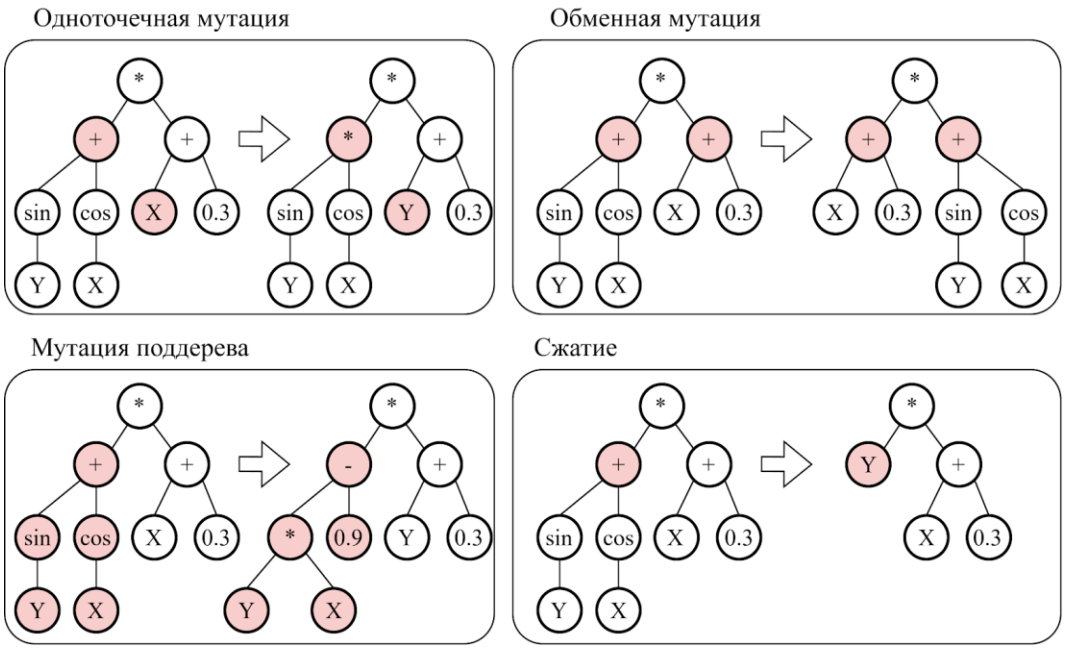

Рисунок 2.9 – Пример применения операторов мутации в ГП

Вероятность применения оператора мутации может вычисляться аналогично вероятности мутации в ГА, однако в числителе учитывается число элементов

дерева, а не длина бинарной строки.

Схема адаптации, основанная на истории успеха (SHA), зарекомендовала себя как высокоэффективный метод настройки вероятностей скрещивания и мутации, что подтверждено успешными экспериментальными результатами в [88,152] и ее регулярным применением в различных алгоритмах, включая ГА. Эффективность данного подхода была также подтверждена в предыдущем разделе, где были разработаны и исследованы самоконфигурируемые ГА с адаптацией на основе истории успеха — SelfCSHAGA и PDPSHAGA. Полученные результаты демонстрируют высокую универсальность метода SHA и позволяют предположить, что его использование в ГП может привести к аналогичным улучшениям. Для этого требуется соответствующим образом модифицировать цикл работы ГП: на каждом поколении для каждого *i*-го индивида из популяции последовательно применяются генетические операторы.

Сначала посредством селекции отбираются родители — поскольку *i*-й индивид уже служит первым родителем, выбирается на одного меньше, чем в стандартной схеме. Затем *i*-й индивид скрещивается с выбранными родителями, после чего к полученному потомку применяется оператор мутации. Это изменение схемы алгоритма введено для интеграции метода SHA, который адаптирует вероятности мутации и скрещивания на основе критерия: если значение ФП потомка выше, чем у *i*-го решения, текущая настройка параметров считается успешной.

Кроме того, требуется модифицировать оператор скрещивания. В методе SHA для каждого гена с вероятностью *CR* выбирается, передавать ли его от родителя или от мутанта, что во многом повторяет оператор равномерного скрещивания, но с динамически изменяемой вероятностью, отличной от фиксированной и равной 0.5. Дополнительно, при модификации оператора скрещивания в ГП необходимо обеспечить возможность селективного давления на данном этапе и выбора более чем двух родителей. Процесс скрещивания организован в два этапа: сначала для каждого гена с вероятностью *CR* определяется, будет ли он унаследован от первого родителя (текущего решения)

или от других родителей. Если ген выбирается от первого родителя, либо если родителей всего два, алгоритм переходит к следующему гену. В противном случае на втором этапе происходит выбор среди оставшихся родителей с учетом их значений ФП, что соответствует подходу, описанному в [73].

Предлагаемая модификация оператора скрещивания позволяет реализовать многородительское скрещивание с возможностью регулирования его интенсивности посредством параметра *CR* и учетом селективного давления на этапе скрещивания. Кроме того, допускается использование классических операторов (одноточечного и стандартного), где процедура выполняется без описанных ранее изменений, но инициируется с вероятностью *CR*; если скрещивание не выполняется, оператор возвращает первого родителя (текущее решение). Согласно [73], при использовании многородительсткого скрещивания, оптимальным числом родителей для большинства операторов равномерного скрещивания является 2 и 7, а для турнирного – 3 и 7. Однако в данном алгоритме на этапе скрещивания применяется дополнительное селективное давление посредством оператора селекции на втором этапе, поэтому общее число родителей увеличивается на 1 по сравнению с оригинальной реализацией.

Данная модификация позволяет использовать различные варианты операторов скрещивания: одноточечное; стандартное; равномерное равновероятное с двумя родителями; равномерное равновероятное с тремя родителями; равномерное равновероятное с восемью родителями; равномерное пропорциональное с тремя родителями; равномерное пропорциональное с восемью родителями; равномерное ранговое с тремя родителями; равномерное ранговое с восемью родителями; равномерное турнирное с тремя родителями; равномерное турнирное с восемью родителями. Оператор селекции при этом может быть любым. В данном исследовании используются: пропорциональная; ранговая; турнирная с размером турнира, равным 3, 5 и 7 индивидов. При этом применяются следующие операторы мутации: точечная, выращиванием, обмена, сжатия.

Алгоритм ГП с модифицированным циклом работы и адаптацией параметров на основе истории успеха далее будет называться конфигурируемым алгоритмом

генетического программирования на основе истории успеха SHAGP (Configuring SHAGP, CSHAGP).

Поскольку алгоритм CSHAGP обладает 220 возможными конфигурациями (учитывая варианты оператора селекции с различным количеством родителей), возникает проблема определения оптимальной настройки для каждой решаемой задачи. В этом случае целесообразно использовать методы самоконфигурирования ГП, которые динамически настраивают параметры в процессе работы, обеспечивая в среднем более высокую надежность по сравнению со случайным выбором конфигурации. Алгоритм CSHAGP с применением метода самоконфигурирования далее будет называться самоконфигурируемый алгоритм ГП с адаптацией на основе истории успеха [153]. Данный алгоритм реализован в двух вариантах: SelfCSHAGP (Self-Configuring Success History-based Adaptation Genetic Programming) и PDPSHAGP (Population-Level Dynamic Probabilities Success History-based Adaptation Genetic Programming), отличающихся методом настройки вероятностей применения генетических операторов.

Цикл работы такого ГП аналогичен ранее представленному циклу самоконфигурируемого ГА с адаптацией на основе истории успеха (псевдокод 2), с той существенной разницей, что индивиды представлены в виде бинарных деревьев, а применяемые операторы являются специализированными для ГП.

2.4 Исследование эффективности самоконфигурируемых алгоритмов генетического программирования с адаптацией на основе истории успеха

Для апробации предложенного метода использовался набор Feynman Symbolic Regression Database [154], содержащий 120 уравнений различной сложности с количеством неизвестных от 1 до 9. Эти уравнения охватывают широкий спектр физических явлений, включая механические, электромагнитные, квантовые и термодинамические процессы. Каждый из тестируемых

самоадаптивных алгоритмов обладал одинаковым набором типов генетических операторов и функциональным множеством. Всем алгоритмам было задано одинаковое количество поколений (1000), в течение которых они работали, и размер популяции (100).

В исследовании участвовали следующие алгоритмы:

- SelfCGP — версия ГП на основе SelfCEA с расширенным набором операторов, использующих селективное давление на этапе скрещивания;
- PDPGP — алгоритм, использующий механизм PDP для настройки вероятностей применения. Генетические операторы равномерного скрещивания с селективным давлением также используются;
- PDPSHAGP — PDP-модификация алгоритма CSHAGP, реализующая динамическую адаптацию вероятностей скрещивания и мутации;
- SelfCSHAGP — версия CSHAGP, основанная на SelfCEA.

Для проведения экспериментов использовалась специализированная программная система, разработанная для применения самоконфигурируемого алгоритма ГП с адаптацией на основе истории успеха [155]. Программная система прошла регистрацию в Федеральной службе по интеллектуальной собственности (Роспатент).

Для каждого из 120 уравнений использовалась выборка из 1000 точек, распределенных случайным образом в пространстве. Подробности формирования выборки описаны в [154]. После этого данные были разделены на обучающую (750 точек) и тестовую (250 точек) выборки. Чтобы учесть стохастическую природу ЭА, проводилось по 100 запусков для каждой задачи, при этом при каждом запуске сохранялось наилучшее значение метрики R^2. Для подтверждения статистической значимости различий результатов алгоритмов применялся статистический критерий Манна-Уитни с уровнем значимости 0.05.

При сравнении результатов решения задач регрессии с использованием большого количества задач возникает проблема интерпретации значения коэффициента детерминации R^2, которое может принимать отрицательные

значения и тем самым смещать средние показатели и искажать оценку методов. Часто для решения этой проблемы используют показатель надежности – долю успешно найденных решений, где успех определяется достижением заранее установленного порога ошибки. Однако такой подход может приводить к потере информации, поскольку результат сильно зависит от выбранного порога. Более информативным является вычисление надежности при различных пороговых значениях. На рисунке 2.10 представлен график значений усредненной по 120 уравнениям надежности при различных значениях порога (от 0.5 до 1 с шагом 0.01) для каждого из тестируемых методов.

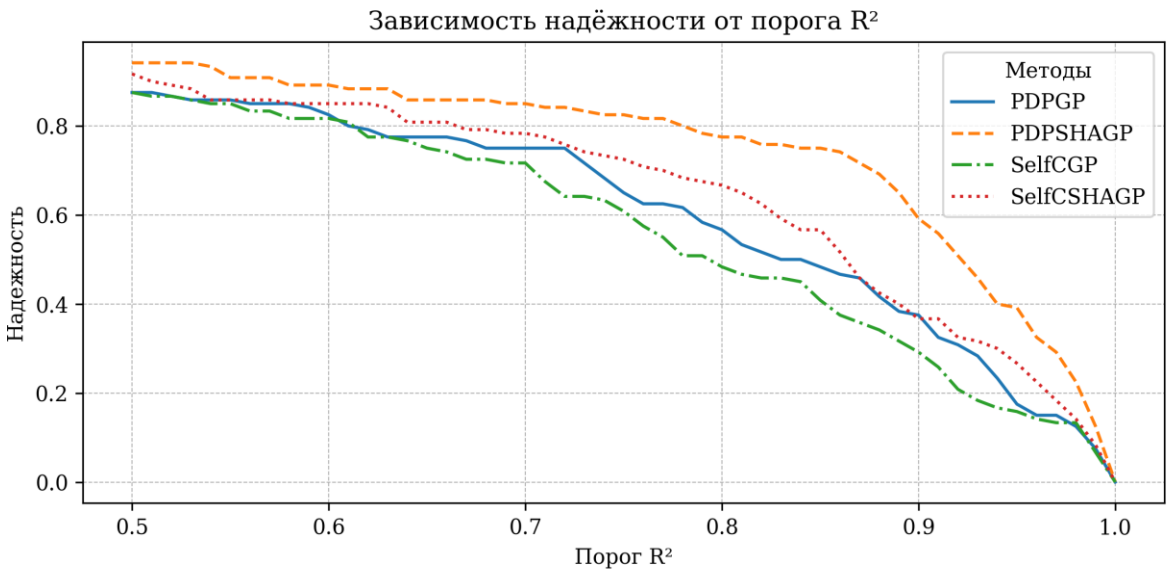

Рисунок 2.10 – Зависимость надежности от порогового значения коэффициента детерминации для различных самоадаптивных ГП

График на рисунке 2.10 показывает, как изменяется надежность различных методов при увеличении порога. PDPSHAGP (оранжевая пунктирная линия) показывает лучшие результаты, оставаясь выше остальных по всему диапазону. SelfCSHAGP (красная точечная линия) тоже выше остальных, но кривая спадает быстрее. PDPGP (синяя сплошная линия) и SelfCGP (зеленая штрихпунктирная линия) заметно уступают, особенно при высоких значениях порога. Усредненные значения надежности, рассчитанные по всем порогам и задачам, равны: SelfCGP:

0.742; PDPGP: 0.773; SelfCSHAGP: 0.797; PDPSHAGP: 0.848. Распределение рангов между алгоритмами, представленное на рисунке 2.11, отражает среднюю позицию каждого алгоритма ГП среди сравниваемых, определяемую на основе упорядочивания алгоритмов по уровню достигнутой надежности в каждой задаче: меньший ранг соответствует более высокому качеству решения.

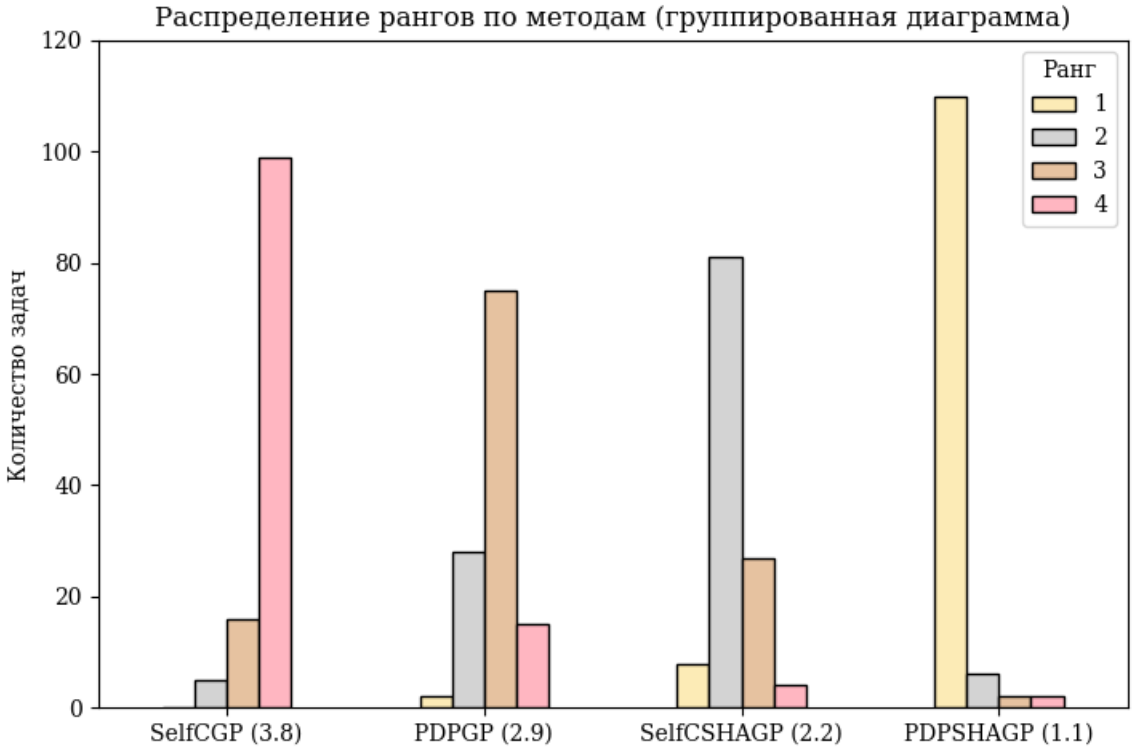

Рисунок 2.11 – Распределение рангов для различных самоадаптивных ГП

На рисунке 2.12 приведены круговые диаграммы, построенные на основе результатов статистического теста, выполненного для сравнения алгоритмов SelfCSHAGP и PDPSHAGP с другими самоадаптивными алгоритмами. Диаграммы разделены на три категории: «превосходит» (зеленый сектор) — количество функций, где первый алгоритм показал лучшие результаты; «без различий» (серый сектор) — статистически незначимые различия; «уступает» (красный сектор) — случаи, когда второй алгоритм продемонстрировал лучшие показатели.

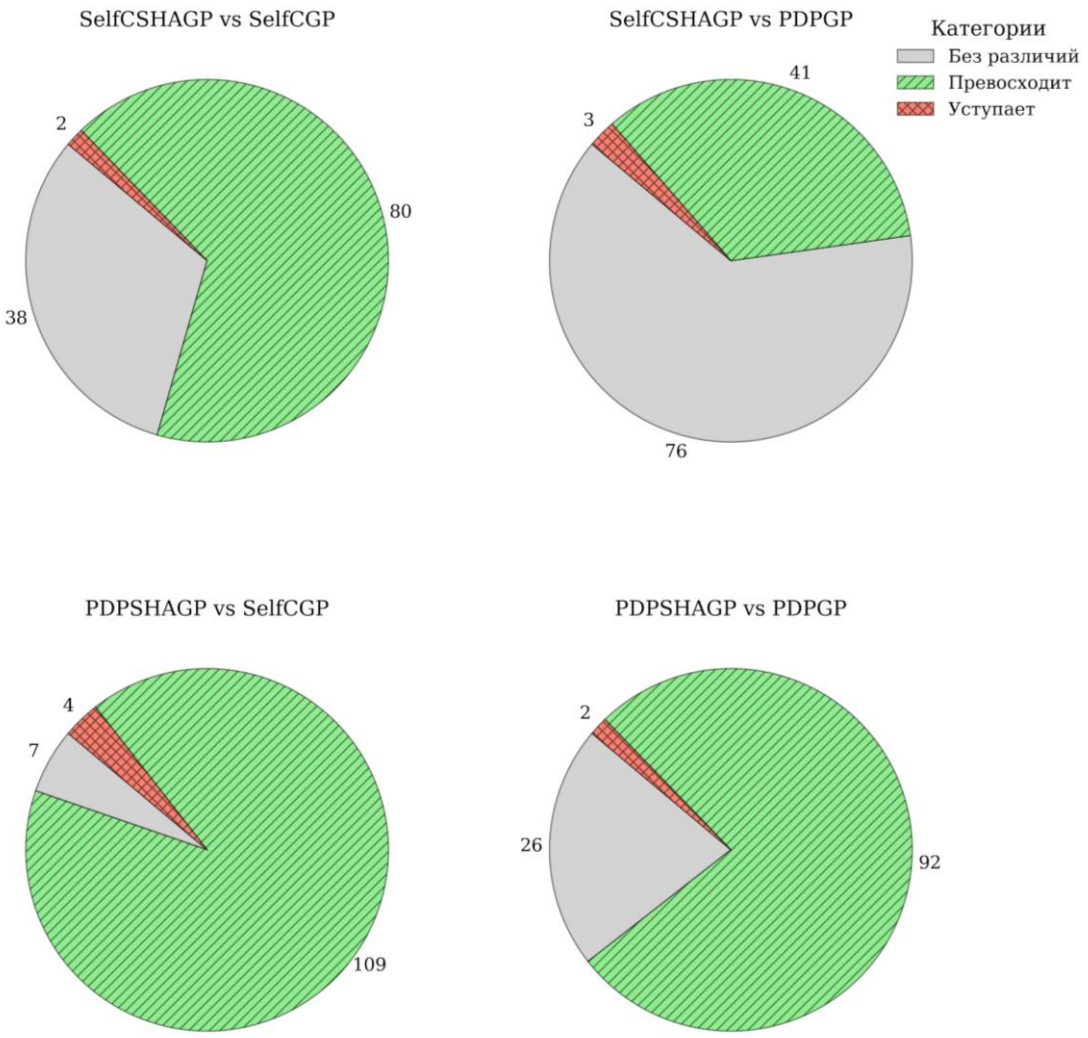

Рисунок 2.12 – Результаты сравнения самоадаптивных ГП с использованием статистического теста

Из представленных диаграмм видно, что оба алгоритма с использованием SHA (SelfCSHAGP и PDPSHAGP) превосходят конкурентов в большинстве тестовых функций. SelfCSHAGP превосходит SelfCGP (80 против 2) и PDPGP (41 против 3), хотя доля задач, в которых различия оказались статистически незначимыми достаточно велика (38 и 76 соответственно). PDPSHAGP же демонстрирует еще более высокие результаты: алгоритм превосходит SelfCGP (109 против 4) и PDPGP (92 против 2) с незначительным числом «ничейных» исходов, что указывает на его лидерство среди сравниваемых алгоритмов.

Дополнительно разработанные алгоритмы ГП были протестированы на задачах МО — как регрессии, так и классификации, где требуется построение

уравнения или разделяющей поверхности [156]. Для экспериментов были выбраны пять задач бинарной классификации из репозитория UCI Machine Learning Repository: (BC, CR, BA, TN, RN), а также две задачи регрессии — Diabetes Data Set (DD) и Combined Cycle Power Plant (CC). В каждом запуске сохранялось наилучшее значение метрики: коэффициент детерминации (R^2) для регрессии и F1-мера для классификации. Все алгоритмы выполнялись в течение 1000 поколений при размере популяции 300 особей. Результаты приведены в таблице 2.4.

Таблица 2.4 – Результаты решения задач классификации и регрессии предложенными и существующими самоадаптивными ГП

Алгоритм	BC	CR	BA	TN	RN	DD	CC	Ср. ранг
SelfCGP	0.954	0.912	0.990	0.958	0.779	0.309	0.901	3.28
PDPGP	0.951	0.942	0.980	0.961	0.782	**0.329**	0.907	2.57
SelfCSHAGP (предлагаемый)	0.950	0.934	0.985	0.972	0.821	0.318	0.905	2.71
PDPSHAGP (предлагаемый)	**0.954**	**0.960**	**0.994**	**0.975**	**0.838**	0.302	**0.909**	**1.43**

Из четырех протестированных реализаций на шести задачах наилучшую общую эффективность продемонстрировал PDPSHAGP, занявший первое место по среднему рангу (1.43) и показавший статистически значимо лучшие результаты в четырех из семи случаев. PDPGP и SelfCSHAGP продемонстрировали сопоставимые результаты, показав средние ранги 2.57 и 2.71 соответственно: PDPGP оказался сильнее в задачах классификации, тогда как SelfCSHAGP лучше сработал в части регрессионных задач. Алгоритм SelfCGP продемонстрировал наихудшую общую эффективность, особенно на задачах регрессии, со средним рангом 3.28.

Таким образом, по результатам тестирования на задачах символьной регрессии (Feynman Database), задачах бинарной классификации и регрессии из UCI репозитория, предложенные алгоритмы SelfCSHAGP и PDPSHAGP либо превосходят, либо демонстрируют сопоставимую эффективность по сравнению с аналогичными самоадаптивными алгоритмами ГП.

Выводы к Главе 2

Во второй главе диссертации были разработаны и исследованы усовершенствованные ЭА: самоконфигурируемый ГА (SelfCSHAGA, PDPSHAGA) и самоконфигурируемый алгоритм ГП (SelfCSHAGP, PDPSHAGP), использующие адаптацию параметров на основе истории успеха. В рамках работы был предложен и реализован модифицированный цикл работы для ГА и ГП, структурно приближенный к циклу работы ДЭ. Разработаны модифицированные операторы скрещивания с возможностью применения многородительского скрещивания, учетом селективного давления на этапе скрещивания и адаптацией вероятности скрещивания. Реализованные модификации обеспечили возможность эффективной интеграции механизма адаптации числовых параметров ЭА на основе истории успеха в алгоритмы ГА и ГП. Дополнительно в алгоритмы были интегрированы механизмы самоконфигурирования (SelfCEA и PDPEA), позволяющие автоматически настраивать вероятности применения различных генетических операторов в процессе поиска решения. Проведено исследование эффективности предложенных алгоритмов на задачах вещественной, псевдобулевой и разношкальной оптимизации (для ГА), а также на задачах формирования моделей МО и символьной регрессии (для ГП), при этом предложенные алгоритмы продемонстрировали стабильно более высокую или сопоставимую эффективность по сравнению с аналогичными алгоритмами.

Таким образом, в рамках проведенного исследования разработан метод самоадаптации ЭА, который интегрирует сильные стороны существующих методов и обеспечивает повышение эффективности ЭА. На основе данного метода разработаны и протестированы самоконфигурируемые ГА и ГП с адаптацией на основе истории успеха, сочетающие механизмы самонастройки и самоконфигурирования, что позволяет отнести их к классу самоадаптивных ЭА. При этом для задач, решаемых с помощью ГА, наилучшие результаты показал алгоритм SelfCSHAGA, тогда как для ГП лидирующие позиции занял алгоритм PDPSHAGP.

3 Разработка и исследование метода автоматизированного проектирования ансамблей нейронных сетей эволюционными алгоритмами

3.1 Метод кодирования нескольких структур в общем бинарном дереве

Основой кодирования нескольких структур в одной бинарном дереве является расширение функционального множества специальными операциями, позволяющих разделять одно дерево на несколько поддеревьев. Пример такой операции проиллюстрирован на рисунке 3.1, где показан специальный узел в структуре дерева. Этот узел является элементом функционального множества, но отличается тем, что сочетает характеристики как функционального, так и терминального узла. Такие узлы будут называться комбинированными, а дерево, содержащее комбинированные узлы — общим деревом.

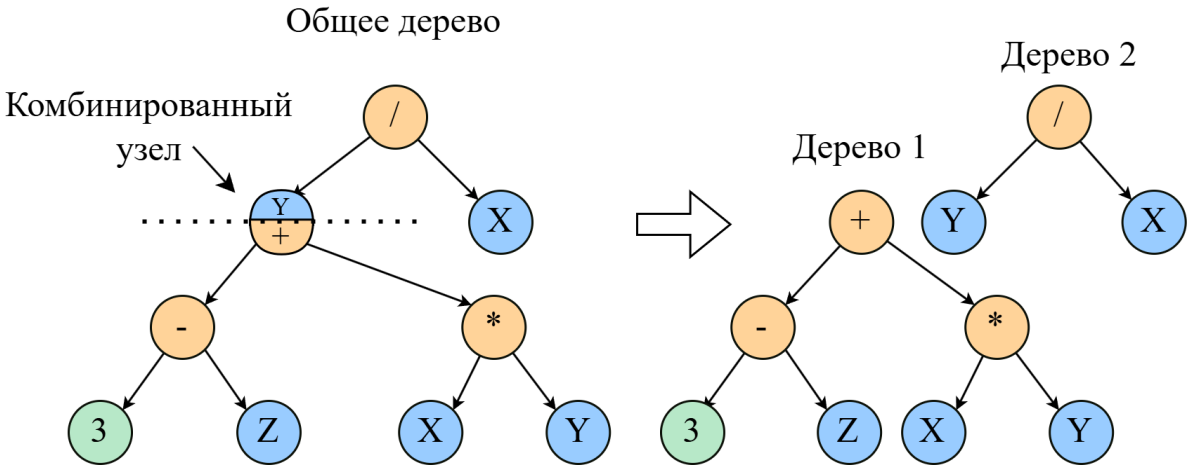

Рисунок 3.1 – Пример общего дерева

Функциональное множество для данного метода кодирования может быть модифицировано следующим образом: F = {+, −, *, /, (Y/+), X, Y, Z, Const}, где

(Y/+) — комбинированный узел, содержащий терминальный элемент «Y» и функциональный элемент «+». В отличие от обычного дерева, общее дерево нельзя напрямую выполнить. Сначала необходимо разделить общее дерево на отдельные деревья, число которых определяется количеством комбинированных узлов. Затем каждое из этих деревьев рекурсивно выполняется. Алгоритм разделения общего дерева на множество деревьев представлен в блок-схеме на рисунке 3.2.

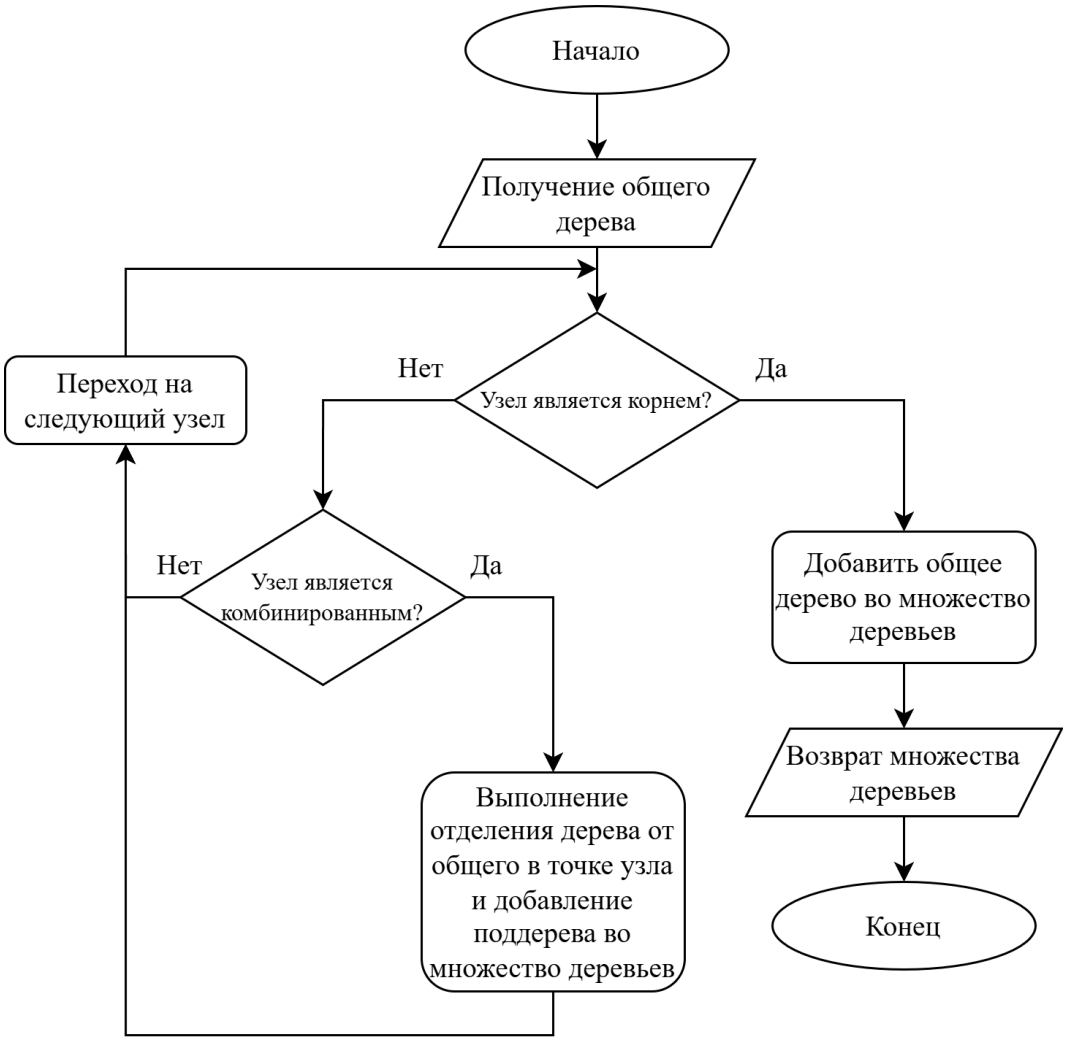

Рисунок 3.2 – блок схема алгоритма разделения общего дерева на множество деревьев

В процессе разбиения общего дерева на множество деревьев, выполняется отделение поддерева от общего дерева. При этом для оставшейся части общего дерева комбинированный узел превращается в терминальный, а для поддерева – в

функциональный. Процесс разделения продолжается до тех пор, пока не закончатся все комбинированные узлы.

Таким образом, данный способ позволяет кодировать в одном бинарном дереве сразу множество бинарных деревьев. При этом все генетические операции выполняются над общей структурой, что обеспечивает согласованную эволюцию всех составляющих и не требует переработки самих генетических операторов. Поскольку комбинированные узлы являются частью универсального множества, расположение и количество разбиений формируются динамически в процессе эволюции. Такой подход может быть использован, например, для одновременного поиска нескольких аналитических выражений либо для кодирования ансамблей ИИТ. В частности, далее в работе будет рассмотрено применение данного метода для представления ансамбля ИНС.

3.2 Формирование ансамблей нейронных сетей на основе общего бинарного дерева

Структура ИНС может быть закодирована в виде бинарного дерева. Для этого требуется определение функционального и терминального множества [106]. Терминальное множество состоит из входных (групп входных) и скрытых (групп скрытых) нейронов с различными ФА. Возможный вид терминального множества (3.1):

$$T = \{1sg, 2sg, 1th, 2th, 1gs, 2gs, 1rl, 2rl, in1, in2\}, \quad (3.1)$$

где число в начале означает количество скрытых нейронов в блоке, а два символа название ФА. Например, «$2sg$» является блоком с двумя скрытыми нейронами с логистической (сигма) ФА. Элементы $in1$, $in2$ – являются блоками входных

нейронов и их содержание определяется для каждой задачи индивидуально (3.2).

$$F = \{+, >\} \tag{3.2}$$

Операция «+» — это операция объединения нейронов в слои. Она добавляет в слои из левой структуры нейроны из соответствующих слоев правой структуры. Иллюстрация с примером применения данной операции представлена на рисунке 3.3.

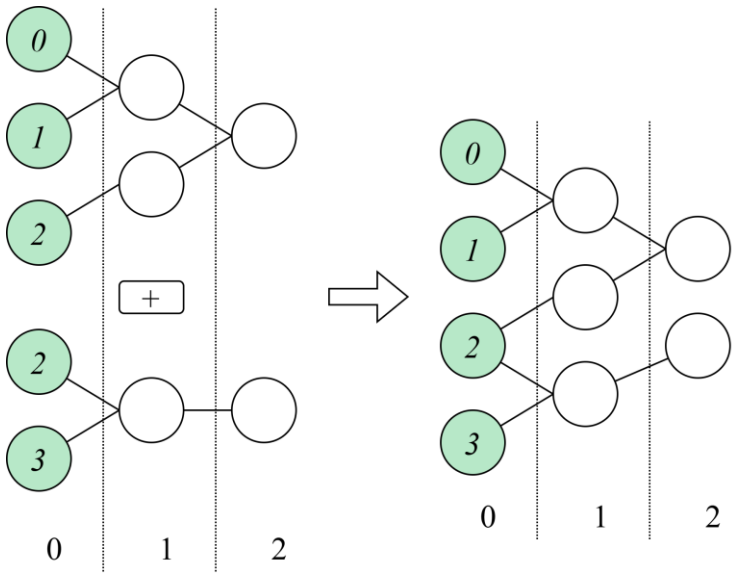

Рисунок 3.3 – Пример применения операции объединения

Структуры выравниваются по входному слою (зеленые нейроны), а в случае отсутствия его по первому скрытому (белые нейроны).

Операция «>» — это операция упорядочивания слоев друг за другом. Полносвязно соединяет крайние правые нейроны левой структуры с крайними левыми нейронными правой структуры. Иллюстрация с примером применения операции представлена на рисунке 3.4.

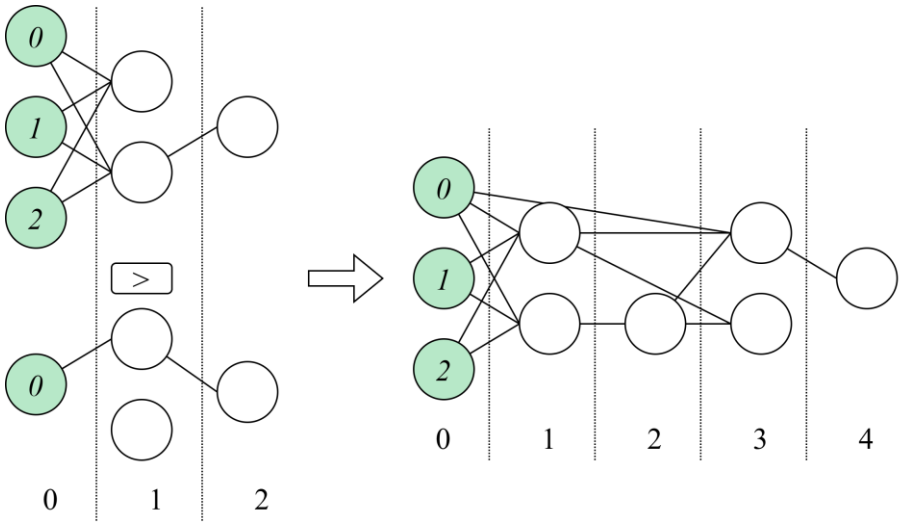

Рисунок 3.4 – Пример применения операции упорядочивания

Как можно видеть, с помощью такой операции в сети могут присутствовать связи не только от слоя к слою. На рисунке 3.5 ниже слева изображено бинарное дерево (генотип), а справа – полученная путем чтения данного дерева сеть (фенотип). Желтым цветом отмечены элементы функционального множества, зеленым входные нейроны, синим скрытые нейроны (блоки нейронов) и красным выходные нейроны.

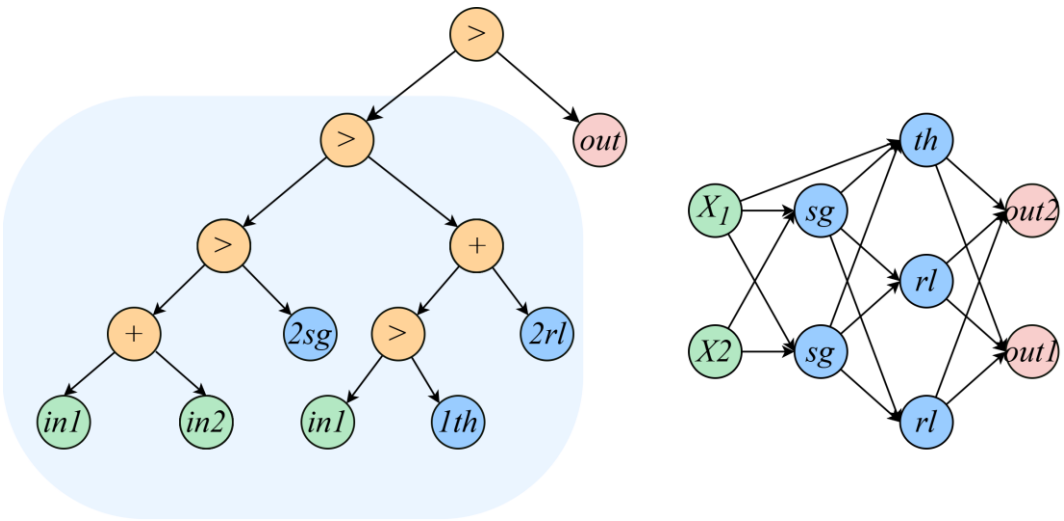

Рисунок 3.5 – Пример кодирования ИНС бинарным деревом

Синим выделена часть структуры, которая произвольно генерируется и участвует в генетических операторах ГП. Часть выше добавляется на этапе чтения

дерева и является обязательной для каждой структуры. Как можно отметить, такой подход позволяет компактно кодировать сети различной сложности с различными ФА на одном слое, а также создавать межслойные связи не только на следующий слой.

ФП может основываться на точности (величине ошибки) рассматриваемой сети, а также и на других параметрах сети: глубина дерева; количество скрытых нейронов; количество связей сети и т.д. Ниже представлен используемый в данной работе способ расчета ФП для индивида – сети (3.3):

$$fitness = \frac{1}{1+E+a_c complex+a_d depth}, \quad (3.3)$$

где $depth$ — это штраф за глубину дерева, a_d — коэффициент штрафа за глубину дерева, $complex$ — штраф за сложность (число связей сети), a_c — коэффициент штрафа за сложность, E — ошибка сети, рассчитываемая по формуле категориальной кросс – энтропии (3.4).

$$E(t,p) = \sum_{i=1}^{C} t(p), \quad (3.4)$$

где p — это выход нейросети (вероятности), t — целевые метки объектов, C — количество классов.

Ранее был рассмотрен метод кодирования структуры ИНС в бинарном дереве, позволяющий выполнять поиск оптимальной архитектуры с помощью алгоритма ГП. Однако, как известно, более высокие результаты при решении прикладных задач ИАД зачастую демонстрируют ансамблевые методы. Данный метод кодирования может быть естественным образом расширен, позволяя в одном бинарном дереве кодировать сразу несколько структур ИНС, которые впоследствии формируют ансамбль для решения конкретной задачи. Для этого предлагается использовать описанный ранее способ кодирования нескольких структур в одном общем бинарном дереве (рисунок 3.6).

Данный подход имеет несколько преимуществ: возможность кодирования сложных архитектур ИНС с разными ФА и связями, что повышает разнообразие ансамбля; возможность совместной оптимизации сетей с учетом их

взаимодействия; автоматизированное определение не только архитектуры сетей-участников ансамбля, но и их оптимального количества. Для получения итогового решения ансамбля используется агрегация прогнозов отдельных ИНС-участников. Простые методы агрегации включают голосование или усреднение прогнозов. Более сложный метод – стекинг – предполагает обучение дополнительной мета-модели на выходах участников ансамбля (мета-признаках).

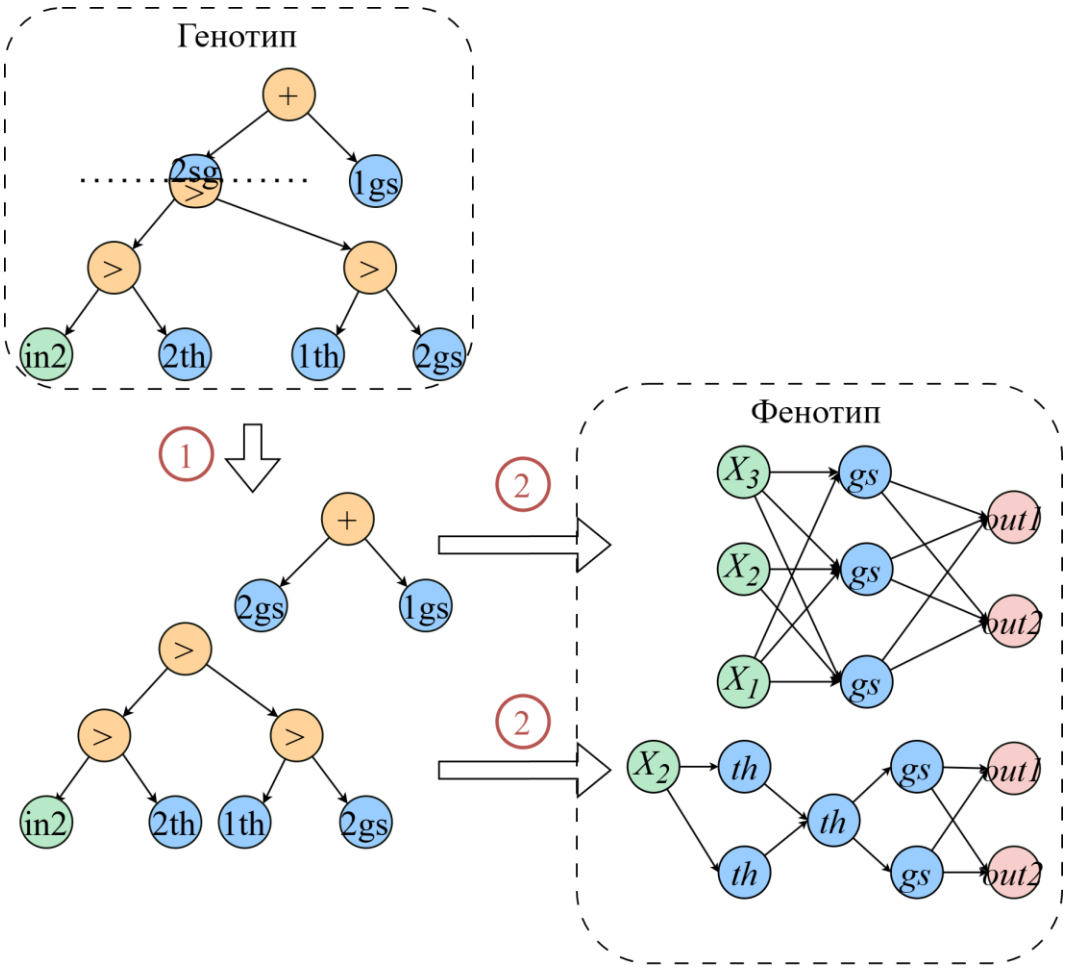

Рисунок 3.6 – Пример кодирования двух сетей бинарным деревом

Структура такого ансамбля может быть задана одним бинарным деревом, кодирующим только сети-участники, а структура мета-модели задается явно как настраиваемый параметр. В данном случае реализуется стекинг с использованием однослойной ИНС (перцептрона), соединяющей мета-признаки с итоговым выходом (рисунок 3.7).

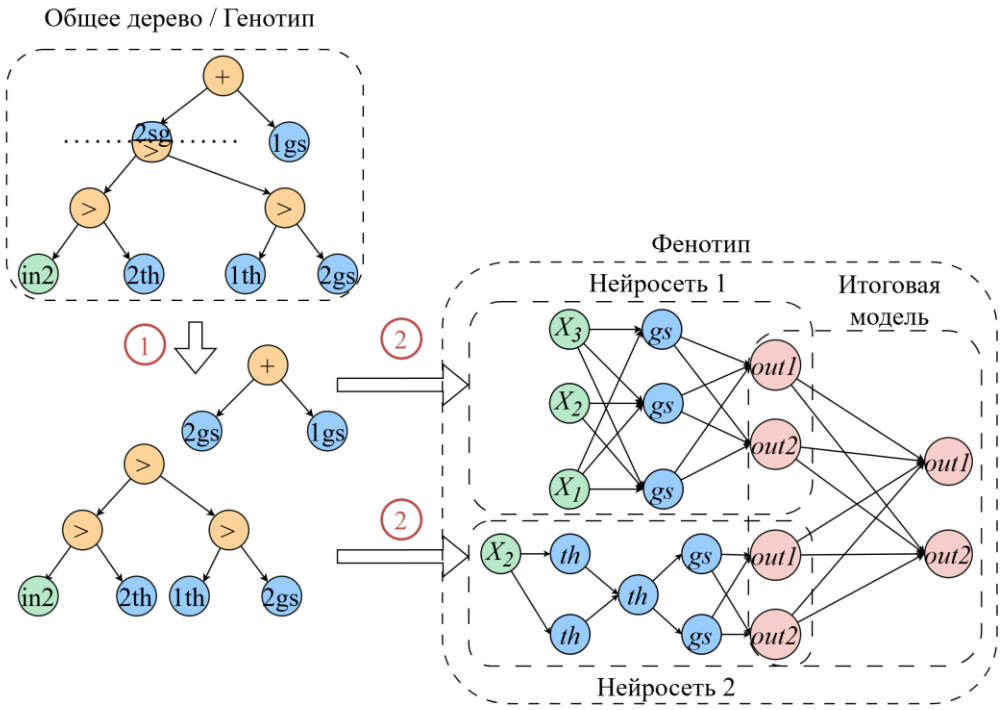

Рисунок 3.7 – Стекинг-ансамбль ИНС

Возможна дополнительная модификация метода, использующая два бинарных дерева, что дает возможность автоматически оптимизировать также и структуру мета-модели (рисунок 3.8). В этом случае одно дерево кодирует сети-участники, а другое – архитектуру итоговой мета-модели.

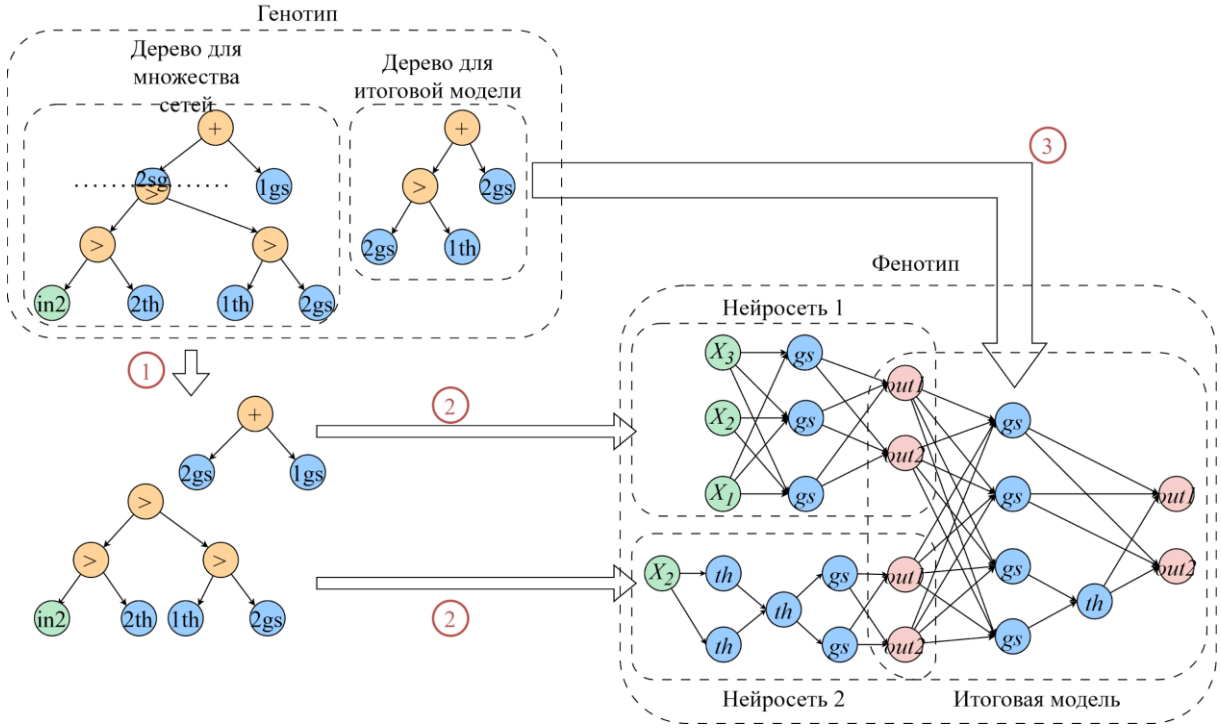

Рисунок 3.8 – Пример кодирования Стекинг-ансамбля и мета-модели

Для повышения разнообразия и снижения корреляции между ИНС-участниками ансамбля в рамках предлагаемого метода каждая модель обучается на собственной подвыборке обучающих данных, сформированной с использованием бутстреп-процедуры. Такая подвыборка формируется случайным образом с возвращением, то есть одни и те же объекты могут быть выбраны несколько раз, а другие — не попасть в подвыборку ни разу. При этом данный способ формирования подвыборок не является обязательным — в общем случае могут использоваться иные стратегии разбиения данных, адаптированные под специфику задачи или требований к ансамблю.

Таким образом, предложенный метод позволяет в одном бинарном дереве кодировать АНС с различными архитектурами и ФА. Алгоритм ГП может быть использован для поиска оптимального состава ансамбля и структуры ИНС-участников. Метод автоматизированного формирования АНС на основе общего бинарного дерева с помощью ГП будет называться GPENN (Genetic Programming Ensemble of Neural Networks) [157]. Далее эффективность разработанного метода будет исследована на тестовых задачах классификации и регрессии.

3.3 Исследование эффективности метода автоматизированного проектирования ансамблей нейронных сетей

Для исследования эффективности были выбраны следующие задачи: Breast Cancer Wisconsin (BC); Credit Risk (CR); User Knowledge Modelling (UK; Banknote Authentication (BA); Diabetes Data Set (DD); Combined Cycle Power Plant (CC). Для проведения экспериментов использовалась специализированная программная система, разработанная для применения метода GPENN [158]. Программная система прошла регистрацию в Федеральной службе по интеллектуальной собственности (Роспатент).

В настоящем исследовании рассматривается только первый из предложенных вариантов автоматизированного формирования ансамбля, основанный на использовании одного общего дерева, кодирующего архитектуры ИНС-участников. Итоговая мета-модель в данной конфигурации задается явно, без структурной оптимизации и задана однослойным персептроном. Оптимизация структуры и состава ансамбля проводилась с помощью самоконфигурируемого алгоритма ГП с адаптацией на основе истории успеха, при этом эволюция осуществлялась на протяжении 20 поколений с популяцией, состоящей из 50 деревьев. Настройка весовых коэффициентов ИНС производилась с помощью ДЭ, в частности, алгоритма SHADE [152], с числом поколений 100 000 и размером популяции 400. Алгоритм останавливался досрочно при отсутствии снижения ошибки в течение 100 последовательных поколений. Сравнение проводилось с рядом базовых и ансамблевых методов МО, реализованных в библиотеке scikit-learn [159]. Все алгоритмы использовались с параметрами по умолчанию, рекомендованными разработчиками библиотеки как общепринятые для типовых задач:

1. GPNN — метод построения одиночных ИНС с использованием ГП из [106];
2. Gradient Boosting — градиентный бустинг над решающими деревьями;
3. Logistic Regression / Linear Regression — логистическая и линейная регрессия;
4. MLP — многослойный перцептрон (Multilayer Perceptron);
5. Random Forest — случайный лес;
6. Voting — ансамбль, формируемый путем простого усреднения прогнозов нескольких моделей (в частности, MLP, линейной регрессии и случайного леса);
7. Stacking — стекинг, в котором базовыми моделями выступают MLP, случайный лес и метод опорных векторов, а роль мета-модели выполняет линейная регрессия;

8. XGBoost — градиентный бустинг с регуляризацией, реализованный в пакете XGBoost [46];

9. GPENN — предлагаемый в данной работе метод формирования АНС.

Чтобы обеспечить достоверность сравнения, все рассматриваемые методы, включая предлагаемый и альтернативные подходы, запускались по 30 раз с различными разбиениями данных на обучающую и тестовую выборки в пропорции 75/25. Значения метрик усреднялись, а статистическая значимость различий результатов оценивалась с помощью критерия Манна — Уитни при уровне значимости $\alpha = 0.05$. В таблице 3.1 представлены результаты исследования.

Таблица 3.1 – Усредненные по 30 запускам значения F1-меры и R^2, достигнутые методом GPENN и другими методами МО на задачах классификации и регрессии

Метод\Задача	**BCW**	**CR**	**UKM**	**BA**	**DDS**	**CCP**	**Ср. ранг**
GPENN (предлагаемый)	0.959	**0.993**	**0.955**	0.999	**0.567**	0.919	**2.83**
GPNN	0.952	0.978	0.939	0.990	0.455	0.921	6.75
Gradient boosting	0.958	0.968	0.928	**1.0**	0.559	0.927	4.91
Linear model	**0.977**	0.779	0.747	0.970	0.551	0.905	6.75
MLP	0.938	0.986	0.948	**1.0**	0.544	0.900	5.33
Random forest	0.958	0.978	0.941	0.993	0.524	0.924	5.50
Stacking	0.958	0.981	0.953	**1.0**	0.553	**0.929**	3.08
Voting	**0.977**	0.973	0.949	**1.0**	0.562	0.467	4.25
Xgboost	0.958	0.979	0.940	**1.0**	0.430	0.908	5.58

Для всех методов был рассчитан средний ранг по совокупности задач, отражающий их относительную эффективность (чем ниже значение, тем выше качество решений). Наилучший результат продемонстрировал метод GPENN, получивший минимальный средний ранг 2.83. Даже в тех случаях, когда он не занимал первое место, его показатели были близки к лучшим.

Для оценки статистической значимости различий результатов был проведен попарный анализ с использованием критерия Манна — Уитни. Результаты представлены на рисунке 3.9, где показано сравнение GPENN с восемью альтернативными методами МО по шести задачам. Каждая ячейка отражает результат сравнения GPENN с конкретным методом в рамках конкретной задачи. Результат классифицировался как «GPENN лучше», если различие по метрике качества оказалось статистически значимым ($p < 0.05$) и среднее значение у GPENN было выше; как «GPENN хуже», если различие также было значимым, но среднее — ниже; и как «нет различий», если различие не было статистически значимым.

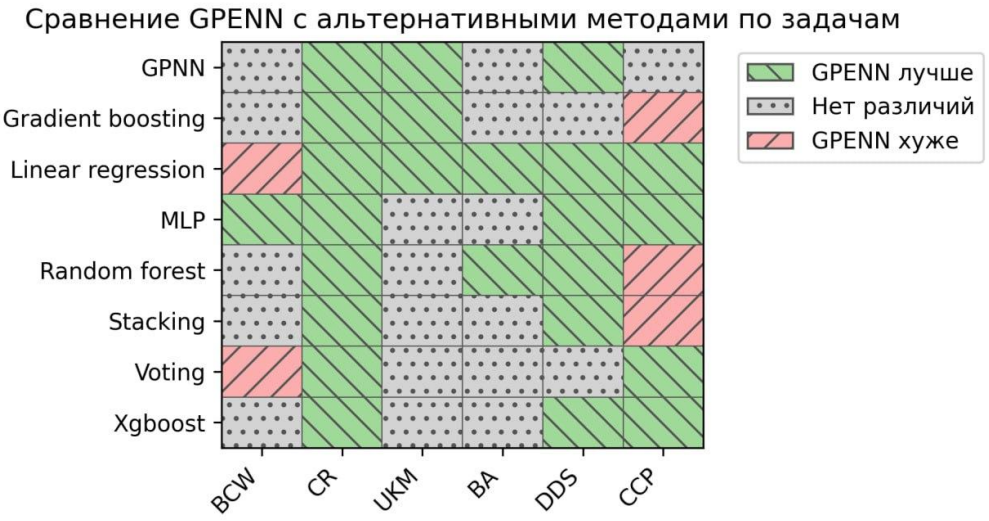

Рисунок 3.9 – Результаты статистического сравнения GPENN с альтернативными методами по задачам

Как видно из рисунка 3.9, метод GPENN продемонстрировал наибольшее число статистически значимо лучших результатов в задачах CR, BA и CC. В задачах BC, UK и BA доля сравнений без значимых различий была выше, в задачах

BC и CC зафиксированы случаи, в которых GPENN уступал другим методам. В частности, на задаче BC GPENN показал статистически худшие результаты по сравнению с более простыми моделями, такими как линейная регрессия и голосование (Voting). В задаче CC, напротив, худшие результаты наблюдались в сравнении с более сложными алгоритмами, включая XGBoost и градиентный бустинг.

На рисунке 3.10 представлена динамика усредненного количества участников ансамбля, формируемого предлагаемым методом GPENN, в процессе эволюции. Значения усреднены по результатам 30 независимых запусков для каждой из задач.

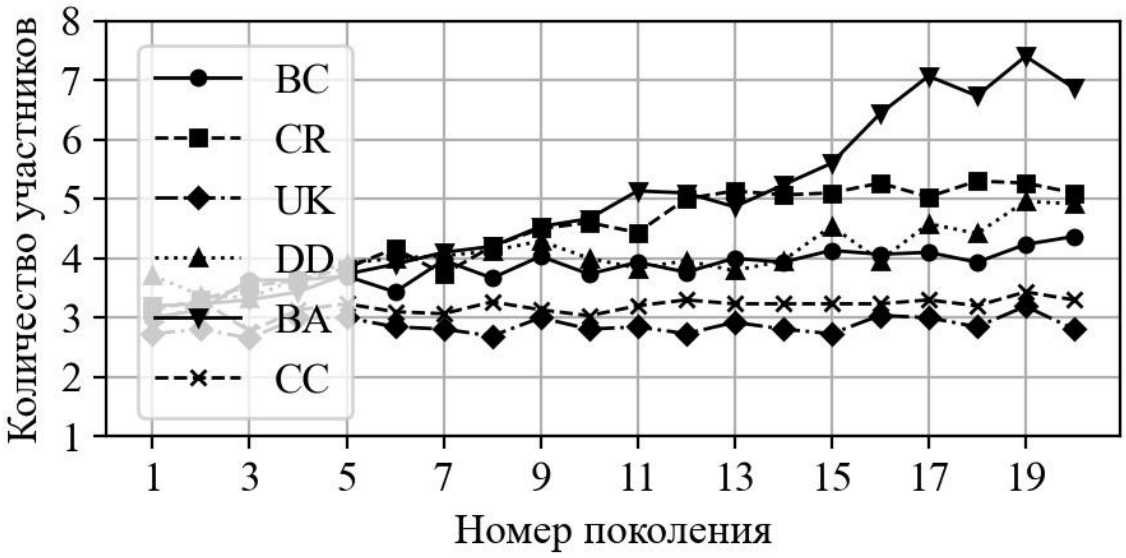

Рисунок 3.10 – Усредненное количество участников ансамбля по 30 запускам на различных итерациях эволюции

Как следует из рисунка 3.10, динамика изменения числа участников ансамбля существенно варьируется в зависимости от решаемой задачи. В задачах CR и BA наблюдается устойчивое увеличение количества моделей, входящих в ансамбль на протяжении итераций. Напротив, для задач UK и CC количество участников остается сравнительно стабильным и невысоким. Эти различия указывают на

способность предлагаемого алгоритма адаптивно определять необходимое число моделей в ансамбле, соответствующее сложности и характеру конкретной задачи.

На рисунке 3.11 представлены генотип (слева) и соответствующий ему фенотип (справа), полученные в одном из запусков GPENN при решении задачи классификации кредитных заявок (CR).

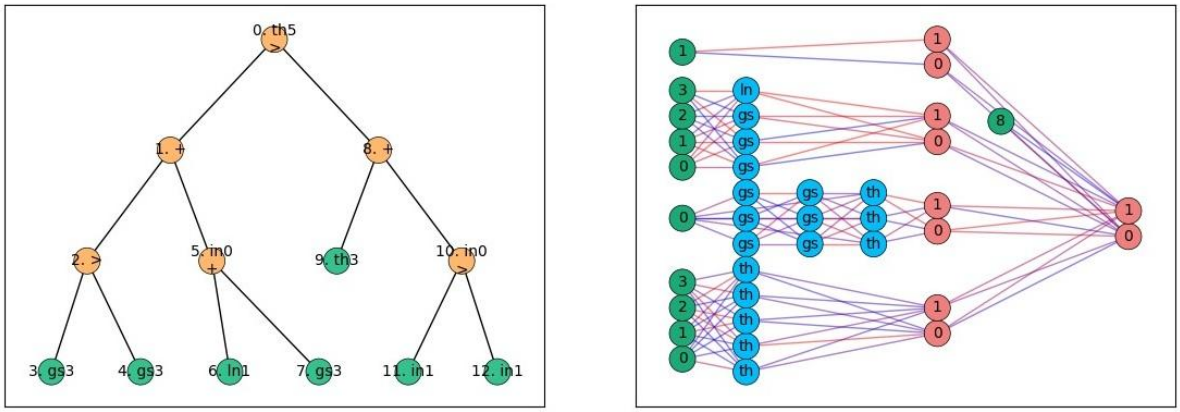

Рисунок 3.11 – АНС, полученный при решении задачи кредитного скоринга (CR)

Справа на рисунке 3.11 визуализирован фенотип — АНС, сформированный в одном из запусков метода при решении задачи классификации кредитных заявок. Ансамбль представляет собой совокупность моделей с различной архитектурой и входными признаками. Видно, что модели-участники обладают архитектурным разнообразием: одна из них реализует простую структуру, близкую к однослойному перцептрону, тогда как другая — многослойную сеть с несколькими скрытыми слоями. Также можно отметить, что входные данные для разных участников различаются, что способствует снижению корреляции между моделями. Данный пример демонстрирует адаптивность и гибкость предлагаемого метода в формировании ансамблей с учетом структуры данных и требований к обобщающей способности.

Выводы к Главе 3

В третьей главе диссертации разработан и исследован метод автоматизированного проектирования АНС на основе ГП и других ЭА, с применением предложенного метода кодирования нескольких деревьев в одном общем бинарном дереве. Исследование метода на тестовых задачах классификации и регрессии показало, что GPENN демонстрирует сопоставимые или более высокие результаты по сравнению с современными подходами, такими как бустинг, стекинг и случайный лес. При этом он не требует ручной настройки архитектур участников, выбора их количества, конкретных параметров объединения, состава ансамбля или числа входных переменных — все эти параметры автоматически оптимизируются в процессе эволюции. Такая автоматизация позволяет использовать метод при минимальном участии экспертов и делает его особенно полезным в задачах, где требуется генерация компактных и адаптивных ансамблей. Однако подход требует существенно больших вычислительных ресурсов по сравнению с традиционными методами, что важно учитывать при практическом применении. Предлагаемый способ кодирования нескольких структур в одном общем дереве может использоваться не только для АНС, но и для других моделей МО, например, при синтезе деревьев решений, НЛС или символьных выражений. Дальнейшее развитие метода включает расширение на случай оптимизации не только базовых моделей, но и мета-модели стекинг-ансамбля, а также адаптацию к стратегиям, таким как бустинг, где модели взаимодействуют последовательно.

4 Практическое применение разработанных методов для проектирования интерпретируемых моделей машинного обучения

4.1 Библиотека Thefittest для автоматизированного проектирования интерпретируемых моделей машинного обучения

С учетом разнообразия подходов к применению ЭА, а также высокой потребности их использования как в задачах оптимизации, так и при проектировании моделей МО, возникает необходимость в универсальном и удобном инструменте, упрощающем практическое применение данных методов. В качестве среды для реализации такого инструмента был выбран язык программирования Python, обладающий широкой распространенностью, высоким уровнем поддержки сообщества и развитой экосистемой библиотек для научных расчетов и ИАД.

В результате была сформулирована задача разработки специализированной программной библиотеки, интегрирующей реализованные в рамках диссертационного исследования алгоритмы, классические и модифицированные ЭА оптимизации, а также эффективные методы построения ИИТ на их основе, включая ИНС, НЛС, ансамблевые методы и другие методы ИАД.

В соответствии с данной целью была разработана библиотека с открытым исходным кодом на языке Python, предназначенная для использования как классических, так и модифицированных ЭА в задачах оптимизации, моделирования и автоматизированного проектирования моделей МО [160].

Разработанная библиотека Thefittest реализована как open-source проект и размещена в общедоступных репозиториях GitHub и PyPi с целью обеспечения доступности и возможности участия сообщества в ее развитии. Поддержка и развитие библиотеки осуществляются с соблюдением современных требований к

качеству программного обеспечения. В частности, используется статический анализатор типов MyPy для обеспечения корректной типизации кода, а также фреймворк PyTest для автоматизированного модульного тестирования. Стиль программного кода соответствует стандарту PEP8. Библиотека Thefittest характеризуется минимальными внешними зависимостями, что упрощает ее установку и использование. На момент написания диссертации для корректной работы пакета требуется наличие трех обязательных библиотек: NumPy — основная библиотека для численных вычислений; Numba — JIT-компилятор, применяемый для ускорения ресурсоемких операций; Scikit-learn для обеспечения полной совместимости с инструментами экосистемы МО. При необходимости обучения ИНС с аппаратным ускорением требуется установка библиотеки PyTorch.

Архитектура библиотеки основана на принципах функциональной декомпозиции и обеспечивает расширяемость за счет четкого разграничения модулей. Функциональная структура библиотеки включает следующие основные модули:

- base — модуль базовых структур. Содержит реализации ключевых объектов, включая n-арные деревья (Tree) и ИНС (Net) с произвольной архитектурой;
- optimizers — модуль реализаций ЭА. Включает как классические методы (например, ДЭ), так и модифицированные версии (SHADE, SelfCGA, PDPGA, SHAGA и др.), применимые как отдельно, так и в составе методов МО;
- classifiers и regressors — модули, содержащие средства построения моделей классификации и регрессии соответственно;
- benchmarks — модуль тестовых задач оптимизации и моделирования;
- utils — модуль вспомогательных инструментов.

Такое построение библиотеки обеспечивает модульность и повторное использование компонентов. На рисунке 4.1 представлен минимальный пример использования самоконфигурируемого ГА для решения классической задачи OneMax — максимизации количества единиц в бинарной строке.

```python
import numpy as np
from thefittest.optimizers import SelfCGA

def onemax(population):
    return np.sum(population, axis=1)

optimizer = SelfCGA(
    fitness_function=onemax,
    iters=200,
    pop_size=200,
    str_len=1000
)

optimizer.fit()
fittest = optimizer.get_fittest()
print("Best genotype:", fittest["genotype"])
```

Рисунок 4.1 – Пример кода для решения задачи OneMax с помощью SelfCGA и библиотеки Thefittest

Пример иллюстрирует типовой сценарий использования алгоритма оптимизации в Thefittest. Пользователь определяет ФП (стр. 4–5), инициализирует алгоритм с необходимыми параметрами (стр. 7–11), запускает оптимизацию (метод fit(), стр. 14) и получает результат через метод get_fittest() (стр. 14–15).

Сравнительный анализ возможностей Thefittest и аналогичных решений, рассмотренных в первой главе, представляет практическую ценность с точки зрения оценки полноты реализованного функционала и соответствия поставленным задачам. Учитывая, что упомянутые в первой главе библиотеки ориентированы на частично схожие цели, их сопоставление с Thefittest возможно лишь в контексте совпадения пользовательских задач с целевым назначением. Следовательно, выбор наиболее подходящего инструмента определяется спецификой исследовательских или инженерных потребностей. В целях

структурированного сопоставления ниже приведена таблица 4.1, отражающая перечень алгоритмов и методов, реализованных в Thefittest и в аналогичных библиотеках. Приведенный перечень составлен на основе документации и репозиториев соответствующих библиотек.

Таблица 4.1 – Сопоставление реализованных методов в Thefittest и аналогичных библиотеках

Algorithm\ package	**Thefittest**	**Deap**	**Scipy**	**PyGAD**	**Pymoo**	**gplearn**
Genetic algorithm	+	+	-	+	+	-
SelfCGA	+	-	-	-	-	-
SHAGA	+	-	-	-	-	-
Evolution Strategy	-	+	-	-	+	
Differential evolution	+	-	+	-	+	-
jDE	+	-	-	-	-	-
SHADE	+	-	-	-	-	-
Particle Swarm Optimization	-	+	-	-	+	-
Genetic programming	+	+	-	-	-	+
SelfCGP	+	-	-	-	-	-
Genetic programming of neural networks (GPNN)	+	-	-	-	-	-
Multilayer perceptron trained by evolutionary algorithms	+	-	-	+	-	-
Convolutional neural networks trained by evolutionary algorithms	-	-	-	+	-	-

Дополнительно в библиотеку Thefittest включены алгоритмы и методы, разработанные в рамках настоящего диссертационного исследования, а именно: самоконфигурируемые ГА оптимизации с адаптацией на основе истории успеха (SelfCSHAGA, PDPSHAGA); самоконфигурируемые алгоритмы ГП с адаптацией на основе истории успеха (SelfCSHAGP, PDPSHAGP); метод автоматизированного проектирования АНС с использованием самоконфигурируемых ЭА (GPENN).

С целью сравнения производительности реализаций ЭА, предоставленных в различных фреймворках и библиотеках, были проведены вычислительные эксперименты на идентичных задачах. Различия касались исключительно программной реализации алгоритма. В таблице 4.2 приведено среднее время выполнения (в секундах), усредненное по 30 запускам; прочерк означает отсутствие реализации в данной библиотеке.

Таблица 4.2 – Сопоставление реализованных методов в Thefittest и аналогичных библиотеках

Algorithm\ package	**Thefittest**	**Deap**	**Scipy**	**PyGAD**	**Pymoo**	**Gplearn**
Genetic algorithm	**8.20**	145.13	-	68.97	341.43	-
Differential evolution	**0.19**	-	91.86	-	1.39	-
Genetic programming	**30.71**	37.16	-	-	-	39.15

Библиотека Thefittest демонстрирует наименьшее среднее время выполнения по всем сравниваемым алгоритмам, что обусловлено использованием JIT-компиляции средствами библиотеки Numba. Наиболее заметное преимущество достигается в реализации алгоритма ДЭ, где время выполнения оказалось более чем на порядок ниже по сравнению с реализацией в SciPy. Несмотря на то, что вычислительные затраты в задачах с применением ЭА зачастую определяются ФП, эффективность самой реализации алгоритма приобретает особую значимость в

случаях вложенного использования ЭА или при параллельной обработке, где оптимизированный код позволяет дополнительно сократить общее время выполнения.

Таким образом, был разработан специализированная библиотека Thefittest, интегрирующая реализованные классические и модифицированные ЭА, а также эффективные подходы к построению ИИТ. На ее основе разработаны 6 программных систем, которые были использованны для решения практических задач, рассмотренных в данной главе. Данные программные были зарегистрированы в Федеральной службе по интеллектуальной собственности (Роспатент).

4.2 Гибридный подход к проектированию интерпретируемых моделей машинного обучения

Современные достижения в области ИИТ открывают новые возможности для решения сложных задач анализа данных. Однако по мере роста сложности применяемых моделей МО все более выраженным становится феномен «черного ящика», при котором внутренняя логика принятия решений недоступна для интерпретации. Особенно это характерно для многослойных ИНС, обладающих высокой точностью, но при этом лишенных свойства интерпретируемости, что существенно ограничивает их применение в ряде задач.

Альтернативой выступают интерпретируемые модели МО, такие как НЛС и деревья решений. Эти модели обеспечивают интерпретируемость за счет представления знаний в виде логических правил и позволяют явно описывать взаимосвязи между входными признаками и выходными результатами. Однако в большинстве случаев они уступают ИНС по точности. Дополнительной проблемой может быть рост сложности интерпретируемых моделей (например, увеличение

числа и длины правил в НЛС или глубины дерева в деревьях решений), что существенно затрудняет их восприятие и понимание человеком.

В связи с этим актуальной задачей становится объединение преимуществ сложных высокоточных моделей и интерпретируемых систем. В рамках такого подхода АНС используется для точного прогноза, а сопровождающая его НЛС — для интерпретации поведения ансамбля, объясняя процессы преобразования входных данных в выходные решения [161,162]. Как АНС, так и НЛС в данном подходе проектируются в автоматизированном режиме с использованием самоконфигурируемых ЭА, однако с разной целевой направленностью: АНС формируется с помощью метода GPENN для достижения высокой точности моделирования, тогда как НЛС строится с использованием алгоритма SelfCSHAGA на основе питтсбургского подхода кодирования для воспроизведения поведения АНС при сохранении интерпретируемости (рисунок 4.2). В контексте настоящей работы под интерпретируемостью понимается компактность и простота базы нечетких правил: чем меньше число правил и чем меньше предпосылок в каждом из них, тем более интерпретируемой считается модель. Полученная база правил НЛС должна объяснять, каким образом АНС преобразует входные данные в выходные результаты. Для обеспечения интерпретации необходимо, чтобы выходы НЛС максимально точно соответствовали выходам АНС при одинаковых входных данных.

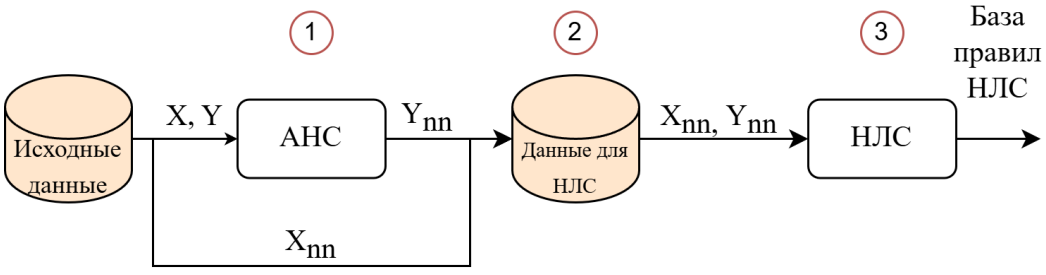

Рисунок 4.2 – Общая схема взаимодействия АНС и НЛС

Обобщенный алгоритм взаимодействия АНС и НЛС включает следующие этапы:

1. Проектирование АНС: автоматизированное построение эффективных структур и обучение АНС на исходной обучающей выборке (X,Y), где $X \subseteq R^n$ — пространство признаков, а $Y \subseteq R^m$ — множество целевых векторов выходов (в случае непрерывных задач).

2. Генерация обучающей выборки для НЛС: получение входов X_{nn} и соответствующих выходов Y_{nn} АНС на данных задачи.

3. Проектирование НЛС: автоматизированное формирование базы правил НЛС на основе данных (X_{nn}, Y_{nn}), с целью воспроизведения поведения АНС.

4. Анализ интерпретируемости: исследование сформированной базы правил для выявления закономерностей преобразования признаков во множество выходных параметров.

Таким образом, в результате процедуры получается интерпретируемая НЛС, способная объяснять работу АНС на множестве входных данных [163]. При этом используемый подход не ограничивается конкретными типами моделей или методами их построения: в качестве высокоточной модели может выступать не только ИНС, но и другие сложные алгоритмы, например ансамбли деревьев решений, градиентный бустинг или методы опорных векторов. Аналогично, роль интерпретируемой модели может выполнять не только НЛС, но и различные логико-ориентированные модели, деревья принятия решений, линейные аппроксимации и другие средства XAI (eXplainable Artificial Intelligence). Далее будут рассмотрены применения данного подхода к решению прикладных задач прогнозирования в различных предметных областях.

4.3 Задача краткосрочного прогнозирования силы ветра на морском побережье

Методы построения метеопрогнозов, использующие строго обоснованные математические модели, играют важную роль в предсказании различных аспектов погоды, включая условия на морских побережьях. Эти методы обычно описывают сразу несколько аспектов погоды (в зоне морского побережья обычно прогнозируют уровень волн, вызванных ветром определенной силы [164,165]), охватывают достаточно большие территории (например, 5x5 км или больше) и предоставляют прогнозы на несколько дней вперед с последующими уточнениями. Это делает такие прогнозы менее подходящими для задач оперативного прогнозирования, требующих высокой точности и быстроты реакции персонала.

Актуальность задачи оперативного прогнозирования особенно высока на Черноморском побережье Крыма. Центром ИИиАБД КФУ созданы базы данных метеорологических параметров, собранных на основе показаний автоматической метеостанции СОКОЛ-М, расположенной в семи точках побережья Черного моря. Эти данные используются как для научных исследований, так и для практических задач оперативного прогнозирования погодных условий на коротких временных интервалах [166,167]. Особую значимость такая система представляет для сельского хозяйства региона, зависящего от краткосрочных погодных условий при планировании работ, а также для организаторов массовых мероприятий на побережье, требующих точных и своевременных прогнозов.

Аналогичные задачи стоят и перед специалистами, работающими в районе Варны (Болгария), где краткосрочное прогнозирование метеоусловий имеет критическое значение для эффективного управления операциями морского порта. В рамках международного сотрудничества с Техническим университетом Варны (TUV) ведутся работы по интеграции оперативных метеопрогнозов в системы поддержки принятия решений при управлении портовой инфраструктурой [168]. Разработка таких моделей позволяет не только повысить безопасность и

эффективность портовой логистики, но и обеспечить необходимую информационную поддержку при проведении мероприятий на побережье.

В обоих регионах — на Черноморском побережье Крыма и в районе Варны — сбор и анализ локальных метеоданных является ключевым условием для построения точных и интерпретируемых моделей прогнозирования, способных работать в реальном времени и поддерживать принятие обоснованных решений в условиях ограниченного времени.

Проблема оперативного прогнозирования в условиях морского побережья, когда особенно актуально достаточно точное прогнозирование силы ветра на временном горизонте 1–3 часа, связана с необходимостью получения более точных данных для конкретных точек, а не общих прогнозов для больших площадей [169]. Существующие методы прогнозирования требуют определенных вычислительных ресурсов и времени для обработки данных, что может быть неприемлемо в ситуациях, когда информация нужна практически немедленно. Доступность достаточно точной и оперативной системы прогнозирования является критически важной для работников портовых служб и организаторов мероприятий на берегу моря [170]. Возможность получать актуальные данные здесь, сейчас и «на ладони» значительно повышает эффективность принятия решений и безопасность операций.

Для построения моделей прогнозирования использовался метеорологический датасет WindFc [171], содержащий данные о ветровых характеристиках и погодных условиях. Этот набор данных был сформирован лабораторией искусственного интеллекта Технического университета Варны (Болгария) в рамках исследовательского проекта по разработке систем оперативного метеопрогнозирования для прибрежных территорий. Набор данных включает измерения, зафиксированные в течение одного календарного года с частотой один раз в минуту. Впоследствии данные были агрегированы с интервалом в 15 минут для приведения к структуре прогнозируемых значений.

Целью прогнозирования является предсказание скорости ветра (WindSpeed) на горизонте 3 часа вперед. Прогноз осуществляется с временным разрешением в

15 минут, что соответствует необходимости формировать 12 предсказаний относительно каждого текущего момента времени. Основной фокус ставится на переменной WindSpeed (средняя скорость ветра).

На этапе подготовки данных был проведен анализ корреляций между признаками. Результаты показали, что значимую связь с целевой переменной имеют только скорость ветра и максимальная скорость ветра, тогда как температура воздуха, атмосферное давление, влажность и направление ветра продемонстрировали слабую корреляцию и были исключены. С учетом цели построения интерпретируемых моделей было принято решение оставить в качестве признаков только эти два параметра.

Для учета временных зависимостей был проведен анализ автокорреляционных и частных автокорреляционных функций (ACF и PACF). На основании полученных результатов в модель были включены три лага предыдущих значений признаков. Это позволило учитывать краткосрочные динамические эффекты без чрезмерного увеличения сложности модели. На этапе подготовки данных признаки и целевые переменные были сформированы по принципу скользящего окна: для каждой временной точки использовались значения скорости ветра и порывов за три предыдущих интервала и текущий момент времени. Целевыми переменными являются прогнозные значения скорости ветра на горизонте 3 часа вперед с шагом 15 минут.

Итоговый набор данных после обработки включает 31521 обучающих и 3490 тестовых наблюдений. Пример структуры сформированных входных признаков и выходных переменных приведен в таблице 4.3. Формирование АНС осуществлялось с использованием алгоритма PDPSHAGP с ограничением в 50 поколений при 20 индивидах в каждом поколении. Специфика данной задачи такова, что выделение большего вычислительного ресурса не имело смысла, т.к. это не приводило к увеличению точности, что было показано в ходе пробных запусков.

Таблица 4.3 – Описание входов и выходов для задачи краткосрочного прогноза ветра

Тип	Переменные	Описание
Вход	WindSpeed (T-45), WindSpeed (T-30), WindSpeed (T-15), WindSpeed (T)	Значения скорости ветра за 3 предыдущих интервала и в текущий момент времени
Вход	WindSpeedMax (T-45), WindSpeedMax (T-30), WindSpeedMax (T-15), WindSpeedMax (T)	Значения порывов ветра за 3 предыдущих интервала и в текущий момент времени
Выход	WindSpeed (T+15), (T+30), ..., (T+180)	Прогноз скорости ветра на горизонте от 15 до 180 минут вперед

Оптимизация весовых коэффициентов ансамбля проводилась с помощью алгоритма SHADE с максимальным числом 5000 итераций, размером популяции 200 особей и применением ранней остановки при отсутствии улучшения в течение 100 итераций. Формирование НЛС осуществлялось с помощью ГА с размером популяции 200 особей и максимальным числом поколений 400.

В эксперименте оценивались четыре показателя, выраженных через среднеквадратичную ошибку (RMSE, Root Mean Square Error) [172]:

1. RMSE АНС при решении исходной задачи прогнозирования.
2. RMSE НЛС, обученной на исходных данных (X, Y).
3. RMSE НЛС1, обученной на данных, сформированных по входам и выходам АНС (X_{NN}, Y_{NN}).
4. RMSE НЛС2 из пункта 3, рассчитанная на исходной задаче прогнозирования.

Проверка статистической значимости различий между сравниваемыми методами осуществлялась с использованием критерия Манна–Уитни при уровне значимости $\alpha = 0.05$.

Усредненные по 20 независимым запускам результаты представлены в таблице 4.4. Строки соответствуют прогнозируемым значениям скорости ветра на

различных горизонтах от 15 до 180 минут вперед; столбцы содержат значения RMSE для соответствующих подходов.

Таблица 4.4 – Усредненные по 20 запускам значения RMSE на задаче прогнозирования силы ветра

Выход	АНС	НЛС	НЛС1	НЛС2
WindSpeed (T+15)	1.39	1.983	1.081	1.910
WindSpeed (T+30)	1.49	1.983	1.042	1.925
WindSpeed (T+45)	1.53	1.984	0.995	1.929
WindSpeed (T+60)	1.57	1.985	0.954	1.927
WindSpeed (T+75)	1.60	1.985	0.912	1.939
WindSpeed (T+90)	1.63	1.985	0.857	1.933
WindSpeed (T+105)	1.64	1.986	0.857	1.934
WindSpeed (T+120)	1.66	1.987	0.792	1.931
WindSpeed (T+135)	1.69	1.987	0.803	1.939
WindSpeed (T+150)	1.72	1.987	0.777	1.950
WindSpeed (T+165)	1.73	1.988	0.741	1.954
WindSpeed (T+180)	1.76	1.988	0.747	1.960
Среднее	1.62	1.975	0.880	1.936

НЛС1, обученная на входах и выходах АНС, показала среднюю ошибку 0.880 при воспроизведении выходов АНС. Это означает, что она с умеренной ошибкой

аппроксимирует поведение ансамбля и может быть использована для интерпретации его решений.

НЛС, построенная SelfCSHAGA и обученная напрямую на исходных данных, дала ошибку — 1.975, что является удовлетворительным результатом. Однако эта НЛС велика и ее правила содержат 8 предпосылок, что делает такую модель также практически не интерпретируемой. АНС показал хорошее качество прогнозирования скорости ветра, обеспечив наименьшую среднюю ошибку (RMSE) 1.62, однако это тоже не интерпретируемая модель.

Единственной интерпретируемой моделью оперативного прогноза ветра морского побережья оказывается НЛС2, дающая при ее применении к исходной задаче прогнозирования оценку ошибки 1.936, что лучше, чем у НЛС, построенной на исходных данных. Таким образом, передача поведения сложной модели через НЛС позволяет улучшить прогнозные свойства интерпретируемой модели. Дополнительно, при обучении НЛС на выходах АНС (НЛС2) удалось сократить среднее число предпосылок в базе правил с 8 до 5. В результате, построенная таким образом НЛС не только близка к поведению АНС, но и представляет собой более простую и удобную для анализа и принятия решений систему.

Точность, достигнутая всеми моделями является достаточной с точки зрения практики и соответствует точности прогнозов, получаемой в аналогичных целях другими, в том числе и намного более сложными, моделями МО [173,174].

Ниже приведен пример одного из правил, сформированных НЛС. Данное правило содержит восемь предпосылок, что делает модель излишне объемной и неудобной для интерпретации. Правило описывает связь между предшествующими значениями метеопараметров (предпосылками) и будущими состояниями прогнозируемой переменной — скоростью ветра (заключениями) на горизонте до 180 минут вперед с шагом 15 минут.

ПРАВИЛО 1: **ЕСЛИ** (WindSpeedMax[t−45мин] крайне низкое) **И** (WindSpeedMax[t−30мин] низкое) **И** (WindSpeedMax[t−15мин] очень низкое) **И** (WindSpeedMax[t] высокое) **И** (WindSpeed[t−45мин] очень низкое) **И** (WindSpeed[t−30мин] низкое) **И** (WindSpeed[t−15мин] очень низкое) **И** (WindSpeed[t] высокое)

ТОГДА WindSpeed[t+15мин] очень высокое, WindSpeed[t+30мин] высокое, WindSpeed[t+45мин] крайне низкое, WindSpeed[t+60мин] крайне высокое, WindSpeed[t+75мин] очень высокое, WindSpeed[t+90мин] низкое, WindSpeed[t+105мин] крайне низкое, WindSpeed[t+120мин] очень высокое, WindSpeed[t+135мин] крайне низкое, WindSpeed[t+150мин] крайне высокое, WindSpeed[t+165мин] очень низкое, WindSpeed[t+180мин] крайне низкое.

Далее представлена база из четырех нечетких правил НЛС1, обученной на входах и выходах АНС.

ПРАВИЛО 1: **ЕСЛИ** (WindSpeedMax[t−45мин] среднее) **И** (WindSpeedMax[t−30мин] среднее) **И** (WindSpeedMax[t−15мин] среднее) **И** (WindSpeed[t−15мин] высокое) **И** (WindSpeed[t] среднее)

ТОГДА WindSpeed[t+15мин] высокое, WindSpeed[t+30мин] высокое, WindSpeed[t+45мин] среднее, WindSpeed[t+60мин] среднее, WindSpeed[t+75мин] высокое, WindSpeed[t+90мин] высокое, WindSpeed[t+105мин] низкое, WindSpeed[t+120мин] среднее, WindSpeed[t+135мин] среднее, WindSpeed[t+150мин] высокое, WindSpeed[t+165мин] низкое, WindSpeed[t+180мин] среднее;

ПРАВИЛО 2: **ЕСЛИ** (WindSpeedMax[t−45мин] высокое) **И** (WindSpeedMax[t−30мин] высокое) **И** (WindSpeedMax[t−15мин] высокое) **И** (WindSpeed[t−15мин] высокое) **И** (WindSpeed[t] высокое)

ТОГДА WindSpeed[t+15мин] высокое, WindSpeed[t+30мин] очень высокое, WindSpeed[t+45мин] высокое, WindSpeed[t+60мин] высокое, WindSpeed[t+75мин] очень высокое, WindSpeed[t+90мин] среднее, WindSpeed[t+105мин] высокое, WindSpeed[t+120мин] высокое, WindSpeed[t+135мин] среднее, WindSpeed[t+150мин] высокое, WindSpeed[t+165мин] высокое, WindSpeed[t+180мин] высокое;

ПРАВИЛО 3: **ЕСЛИ** (WindSpeedMax[t−45мин] высокое) **И** (WindSpeedMax[t−30мин] высокое) **И** (WindSpeedMax[t−15мин] высокое) **И** (WindSpeed[t−15мин] крайне высокое) **И** (WindSpeed[t] очень высокое)

ТОГДА WindSpeed[t+15мин] среднее, WindSpeed[t+30мин] очень высокое, WindSpeed[t+45мин] высокое, WindSpeed[t+60мин] среднее, WindSpeed[t+75мин] очень высокое, WindSpeed[t+90мин] высокое, WindSpeed[t+105мин] высокое, WindSpeed[t+120мин] высокое, WindSpeed[t+135мин] высокое, WindSpeed[t+150мин] очень высокое, WindSpeed[t+165мин] высокое, WindSpeed[t+180мин] среднее;

ПРАВИЛО 4: **ЕСЛИ** (WindSpeedMax[t−45мин] очень высокое) **И** (WindSpeedMax[t−30мин] очень высокое) **И** (WindSpeedMax[t−15мин] очень высокое) **И** (WindSpeed[t−15мин] очень высокое) **И** (WindSpeed[t] очень высокое)

ТОГДА WindSpeed[t+15мин] крайне высокое, WindSpeed[t+30мин] очень высокое, WindSpeed[t+45мин] очень высокое, WindSpeed[t+60мин] очень высокое, WindSpeed[t+75мин] крайне высокое, WindSpeed[t+90мин] крайне высокое, WindSpeed[t+105мин] очень высокое, WindSpeed[t+120мин] очень высокое, WindSpeed[t+135мин] очень высокое, WindSpeed[t+150мин] очень высокое, WindSpeed[t+165мин] очень высокое, WindSpeed[t+180мин] крайне высокое.

Каждое правило направлено на предсказание значений скорости ветра на 12 шагов вперед (от t+15 мин до t+180 мин), что обусловливает наличие развернутой части заключений. Тем не менее, число предпосылок остается ограниченным — всего 4–5 условий на основе недавней истории параметров. Это свидетельствует о том, что модель может формировать комплексный прогноз на достаточно длительный интервал времени (3 часа), опираясь на компактное множество входных признаков. Подобное свойство делает модель интерпретируемой и пригодной для анализа и принятия решений, в отличие от «черного ящика» оригинального ансамбля.

Таким образом, представленная база правил одновременно отражает поведение исходной нейросетевой модели, упрощает структуру принятия решений за счет меньшего числа предпосылок, обеспечивает объяснимость и компактность описания прогнозируемого процесса. Если она будет реализована на переносимом устройстве, например – планшете, то может служить для поддержки принятия решений непосредственно персоналом морских портов и прибрежного обслуживания на месте выполнения им работы.

4.4 Задача прогнозирования деградации солнечных батарей космического аппарата

Солнечные батареи (СБ) являются основным источником энергии для большинства космических аппаратов (КА). Однако в условиях открытого космоса они подвергаются длительному и интенсивному воздействию различных негативных факторов, что приводит к их деградации. Это, в свою очередь, снижает эффективность работы космического аппарата и может ограничивать его функциональность и срок службы. Поскольку организация регулярных прямых измерений характеристик СБ в полете сопряжена с техническими и финансовыми трудностями, особенно в случае длительных миссий, возрастает потребность в создании математической модели, способной достаточно точно прогнозировать уровень деградации СБ по доступным параметрам окружающей среды [175].

Для построения прогностической модели деградации солнечных батарей космического аппарата используются реальные полетные данные, полученные с бортовых систем космического аппарата и спутников мониторинга солнечной активности. Эти данные структурированы в виде множества наблюдений, каждое из которых описывается фиксированным набором входных и выходных параметров. Модель получает на вход семь количественных характеристик, описывающих текущее воздействие внешней среды и состояние космического аппарата: Ресурс — число дней, прошедших с момента запуска аппарата; Интегральный флюенс протонов с энергией менее 1 МэВ; Интегральный флюенс протонов с энергией менее 10 МэВ; Интегральный флюенс протонов с энергией менее 100 МэВ; Интегральный флюенс электронов с энергией менее 0.6 МэВ; Интегральный флюенс электронов с энергией менее 2 МэВ; Коэффициент освещенности — параметр, отражающий интенсивность солнечного излучения, попадающего на поверхность СБ. На выходе модель должна предсказывать четыре электрических характеристики, по две для каждой из двух секций СБ КА: Uxx_1 —

напряжение холостого хода первой секции СБ; $I_{кз1}$ — ток короткого замыкания первой секции СБ; U_{xx2} — напряжение холостого хода второй секции СБ; $I_{кз2}$ — ток короткого замыкания второй секции СБ. Каждое наблюдение — это вектор из 7 входных и 4 выходных значений. Всего в выборке содержатся данные за 295 дней.

Для проведения эксперимента исходная выборка была разделена на две части: первые 169 наблюдений использовались для обучения моделей, а оставшиеся — для тестирования. Для количественной оценки точности прогноза модели использовалась относительная средняя ошибка, выраженная в процентах (4.1):

$$error = \frac{100}{s}\frac{1}{m}\sum_i^m \frac{1}{y_{max}^i - y_{min}^i}\sum_j^s \left|y_j^i - o_j^i\right|, \quad (4.1)$$

где: m — это количество выходных параметров; s — количество наблюдений в тестовой выборке; y — истинное значение; o — предсказанное моделью значение;

В таблице 4.5 представлены значения ошибок для четырех экспериментов [176]:

1. АНС — АНС, обученный методом GPENN с помощью PDPSHAGP на исходной задаче;
2. НЛС1 (исходные данные) — НЛС, обученная напрямую на входных и выходных данных исходной задачи с помощью SelfCSHAGA;
3. НЛС2 (по входам и выходам АНС) — НЛС, обученная на входах и выходах АНС с помощью SelfCSHAGA (аппроксимация его поведения);
4. НЛС2 (по входам и выходам АНС, тест на исходных данных) — та же НЛС, что в п.3, но протестированная на исходной задаче с помощью SelfCSHAGA (входы — как в исходной задаче, выходы — сравниваются с реальными значениями, а не с прогнозом АНС).

Таблица 4.5 – Усредненная по 20 запускам относительная ошибка (%) на задаче прогнозирования деградации солнечных батарей для рассматриваемых экспериментов

Экспертмент\модель	Относительная ошибка, %
АНС (обучена на исходных данных)	4.25
НЛС1 (обучена на исходных данных)	8.13
НЛС2 (обучена на входах/выходах АНС, тест на АНС)	6.26
НЛС2 (обучена на входах/выходах АНС, тест на исходных данных)	7.45

Для сравнения в таблице 4.6. также приведены результаты моделирования, полученные другими исследователями, с использованием различных методов интеллектуального анализа данных: SelfCGP-E — АНС, автоматически объединяемых с помощью ГП. Все параметры (число моделей, используемые признаки, структура ансамбля) подбираются автоматически [177]; SelfCGP+ANN — нейросеть с гибкой архитектурой, сформированная самоконфигурируемым методом ГП [178]; 3-layer MLP — трехслойная нейросеть (перцептрон) [179].

Таблица 4.6 – Результаты альтернативных подходов на задаче прогнозирования деградации солнечных батарей

Метод	Относительная ошибка, %
SelfCGP-E	4.18
SelfCGP+ANN	4.65
3-layer MLP	5.7

В результате экспериментов автоматически сформированный АНС продемонстрировал относительную ошибку 4.25 %, что сопоставимо с лучшим из известных результатов, полученным методом SelfCGP-E (4.18 %), и превышает точность как SelfCGP+ANN (4.65 %), так и классической трехслойной нейросети (5.7 %). При этом АНС не требует ручной настройки числа моделей в ансамбле и

параметров его структуры, так как формируется автоматически. НЛС2, обученная на входах и выходах АНС, с умеренной ошибкой воспроизводит прогнозы ансамбля для любых входных данных. При этом эту же НЛС2 можно использовать самостоятельно: ее база правил компактнее (НЛС2 — в среднем 14.4 правила с 3.6 предпосылками против НЛС1 — 19.7 правил с 6.9 предпосылками). Кроме того, НЛС2 демонстрирует более низкую ошибку прогнозирования (7.45 % против 8.13 %).

Для примера рассмотрим фрагмент базы правил, полученной в одном из запусков при обучении НЛС на входах и выходах АНС (здесь задействованы только признаки x1–x7 и четыре выходные переменные Uxx_1, Iкз_1, Uxx_2, Iкз_2):

1. Если (x1 Оч. мал.) и (x2 Оч. мал.) и (x3 Оч. мал.) и (x7 Мал.), то Uxx_1 Макс., Iкз_1 Оч. боль., Uxx_2 Макс., Iкз_2 Макс.

2. Если (x1 Оч. мал.) и (x2 Мал.) и (x3 Мал.) и (x7 Оч. боль.), то Uxx_1 Макс., Iкз_1 Оч. боль., Uxx_2 Оч. боль., Iкз_2 Оч. Боль.

3. Если (x1 Мал.) и (x2 Мал.) и (x3 Оч. мал.) и (x7 Оч. мал.), то Uxx_1 Мал., Iкз_1 Мин., Uxx_2 Оч. боль., Iкз_2 Оч. Боль.

4. Если (x1 Мал.) и (x2 Мал.) и (x3 Мал.) и (x7 Оч. боль.), то Uxx_1 Макс., Iкз_1 Боль., Uxx_2 Оч. боль., Iкз_2 Макс.

5. Если (x1 Мал.) и (x2 Средн.) и (x3 Оч. мал.) и (x7 Оч. мал.), то Uxx_1 Средн., Iкз_1 Боль., Uxx_2 Средн., Iкз_2 Мал.

6. Если (x1 Средн.) и (x2 Оч. мал.) и (x3 Оч. мал.) и (x7 Средн.), то Uxx_1 Мал., Iкз_1 Мал., Uxx_2 Мал., Iкз_2 Мал.

7. Если (x1 Средн.) и (x2 Оч. мал.) и (x3 Мал.) и (x7 Мал.), то Uxx_1 Мин., Iкз_1 Мин., Uxx_2 Мин., Iкз_2 Мин.

8. Если (x1 Боль.) и (x2 Оч. мал.) и (x3 Оч. мал.) и (x7 Средн.), то Uxx_1 Оч. мал., Iкз_1 Мал., Uxx_2 Мин., Iкз_2 Мин.

9. Если (x1 Боль.) и (x2 Оч. мал.) и (x3 Оч. мал.) и (x7 Оч. боль.), то Uxx_1 Макс., Iкз_1 Боль., Uxx_2 Оч. мал., Iкз_2 Оч. Мал.

Здесь: x1 — Ресурс; x2 — Интегральный флюенс протонов с энергией < 1 МэВ; x3 — Интегральный флюенс протонов с энергией < 10 МэВ; x7 — Коэффициент освещенности.

4.5 Задача прогнозирования уровня звукового давления деревянных панелей

Современные стандарты акустического комфорта и экологичности в строительстве предъявляют все более высокие требования к точности оценки звуковых характеристик конструкционных и отделочных материалов, особенно древесных панелей, широко применяемых в архитектуре, мебельной промышленности и интерьере. В связи с этим возрастает потребность в разработке надежных методов моделирования их акустического поведения, особенно в диапазоне частот 100–4000 Гц, значимом для слухового восприятия человека, где даже незначительные ошибки прогноза могут существенно снижать эффективность звукоизоляции. На этом фоне особую актуальность приобретает задача построения моделей зависимости уровня звукового давления от структурно-технологических параметров древесных панелей. К числу ключевых факторов, определяющих акустическую отдачу, относятся толщина и количество слоев, пористость, геометрия волокон, а также демпфирующие свойства исследуемого материала.

По результатам 130 лабораторных испытаний на специально разработанном стенде были получены зависимости среднего звукового давления (выходной параметр) от структурных характеристик панелей: радиуса, глубины и площади звуковых карманов, толщины образца древесной панели и клеевого слоя, влажности образцов и частоты сигнала (входные параметры) в диапазоне 100–4000

Гц. Требовалось создать модель, описывающую влияние этих факторов на акустическую отдачу с учетом нелинейных упругих и пористых эффектов.

Для решения задачи использовался описанный ранее подход, основанный на последовательном обучении АНС методом GPENN и ИНС с помощью самоконфигурируемых ЭА с адаптацией на основе истории успеха, предложенных в данной работе. Соответствующая программная система, реализующая данный подход, была разработана и зарегистрирована в Федеральной службе по интеллектуальной собственности (Роспатент) [180]. PDPSHAGP, используемый для оптимизации структуры АНС, искал решение в течение 100 поколений с размером популяции, равным 20. Для SelfCSHAGP было выделено 300 поколений с размером популяции, равным 300. В таблице 4.6 представлены значения RMSE для четырех экспериментов:

1. АНС — АНС, обученный на исходной задаче;
2. НЛС1 (исходные данные) — НЛС, обученная напрямую на входных и выходных данных;
3. НЛС2 (по входам и выходам АНС) — НЛС, обученная на входах и выходах АНС (аппроксимация его поведения);
4. НЛС2 (по входам и выходам АНС, тест на исходных данных) — та же НЛС, что в п.3, но протестированная на исходной задаче (входы — как в исходной задаче, выходы — сравниваются с реальными значениями, а не с прогнозом АНС).

Таблица 4.6 – Усредненные по 20 запускам значения RMSE на прогнозирования уровня звукового давления деревянных панелей для рассматриваемых экспериментов

Экспертмент\модель	**RMSE**
АНС (обучена на исходных данных)	3.3
НЛС1 (обучена на исходных данных)	4.9
НЛС2 (обучена на входах/выходах АНС, тест на АНС)	1.9
НЛС2 (обучена на входах/выходах АНС, тест на исходных данных)	5.1

В результате усредненная по 20 независимым запускам RMSE АНС составила 3.3 дБ (около 10 % отклонения от истинных значений) [181]. НЛС показали более высокую ошибку: 4.9 дБ у НЛС1, обученной на исходных данных, и 5.1 дБ у НЛС2, обученной на входах и выходах АНС. Статистически значимых различий в точности между НЛС1 и НЛС2 не выявлено. При этом структура НЛС2 оказалась существенно проще: при большем среднем количестве правил (9 против 5.15) число предпосылок в каждом правиле было более чем в два раза меньше (в среднем 4.04 против 10.9), что в итоге приводит к меньшему общему числу предпосылок. При воспроизведении выходов АНС на выборке (X_{nn}, Y_{nn}) НЛС2 показала ошибку 1.9 дБ.

Выводы к Главе 4

В четвертой главе описана разработанная библиотека с открытым исходным кодом Thefittest, предназначенная для автоматизированного проектирования моделей МО с использованием как существующих ранее, так и новых предложенных самоконфигурируемых ЭА. Архитектура библиотеки обеспечивает модульность, расширяемость и совместимость с существующими средствами анализа данных. Сравнение с аналогичными решениями продемонстрировало, что реализованная библиотека обеспечивает широкую функциональность и высокую эффективность реализации. На основе thefittest созданы программные системы, зарегистрированные в Федеральной службе по интеллектуальной собственности (Роспатент).

Был описан предлагаемый гибридный подход к построению интерпретируемых моделей, сочетающий автоматизированное проектирование АНС и НЛС, объясняющей поведение ансамбля. Такой подход повышает

интерпретируемость системы, не ухудшая при этом точность. Подход был апробирован на ряде прикладных задач в различных предметных областях, включая прогнозирование акустических характеристик древесных панелей, краткосрочный прогноз силы ветра на морском побережье и моделирование процесса деградации солнечных батарей космического аппарата.

Таким образом, разработанные алгоритмы и методы эффективно решают задачи моделирования и прогнозирования. АНС демонстрирует конкурентную точность, а НЛС, обученная на выходах ансамбля, формирует более компактную базу правил, чем НЛС, построенная на исходных данных, и обеспечивает меньшую либо сравнимую ошибку. В итоге формируется система из двух моделей: точного, но слабо интерпретируемого АНС и объясняющей его НЛС.

ЗАКЛЮЧЕНИЕ

В диссертационной работе получены следующие результаты:

1. Выполнен обзор современных методов проектирования ИИТ и подходов самоадаптации ЭА.
2. Разработан и исследован самоконфигурируемый ГА с измененным циклом работы, модифицированной процедурой скрещивания и адаптацией на основе истории успеха.
3. Разработан и исследован самоконфигурируемый алгоритм ГП с измененным циклом работы, модифицированной процедурой скрещивания и адаптацией на основе истории успеха.
4. Предложен, реализован и исследован метод формирования АНС с помощью бинарных деревьев, обеспечивающий одновременную оптимизацию структуры сетей-участников ансамбля и их количества.
5. Предложен и исследован подход гибридизации ИИТ на основе ЭА, в котором реализовано автоматизированное проектирование нейросетевых моделей, а также формирование базы нечетких правил, объясняющей процесс принятия решения нейросетевой моделью.
6. Разработана библиотека на языке Python, интегрирующая предложенные самоадаптивные ЭА, и реализовано 6 программных систем, использующих функционал библиотеки.
7. Проведено тестирование разработанных методов на ряде тестовых и прикладных задач, показавшее эффективность предлагаемых алгоритмов моделирования и оптимизации при формировании моделей МО.

Таким образом, в диссертационном исследовании поставлена и решена

актуальная научная задача повышения эффективности и интерпретируемости моделей МО за счет разработки самоадаптивных ЭА и их применения к автоматизированному проектированию ИИТ, что имеет существенное значение для теории и практики системного анализа, управления и обработки информации.

СПИСОК ЛИТЕРАТУРЫ

1. Боргоякова Т. Г., Лозицкая Е. В. Системный анализ и математическое моделирование // Инженерный вестник Дона. – 2018. – № 1(48).
2. Боргоякова Т. Г., Лозицкая Е. В. Математическое моделирование: определение, применяемость при построении моделей образовательного процесса [Электронный ресурс] // Науковедение. – 2017. – Т. 9, № 2. – Режим доступа: http://naukovedenie.ru/PDF/82TVN217.pdf (дата обращения: 21.06.2025).
3. Зёлко А. С. О математическом моделировании в психологических исследованиях [Электронный ресурс] // Концепт: Научно-методический электронный журнал. – 2013. – Т. 3. – С. 1741–1745. – Режим доступа: http://e-koncept.ru/2013/53351.htm (дата обращения: 21.06.2025).
4. Лысцов Н. А., Мартышкин А. И. Нейронные сети: применение и перспективы // Научное обозрение: Педагогические науки. – 2019. – № 3-2. – С. 35–38.
5. Abiodun O. I., Jantan A., Omolara A. E., Dada K. V., Umar A. M., Linus O. U., Arshad H. Comprehensive review of artificial neural network applications to pattern recognition // IEEE Access. – 2019. – Т. 7. – С. 158 820–158 846. – DOI: 10.1109/ACCESS.2019.2896880.
6. Pienaar S. W., Malekian R. Human activity recognition using LSTM-RNN deep neural network architecture // Proceedings of the 2019 IEEE 2nd Wireless Africa Conference (WAC). – Pretoria, South Africa: IEEE, 2019. – С. 1–5. – DOI: 10.1109/AFRICA.2019.8843403.
7. Иванько А. Ф., Иванько М. А., Сизова Ю. А. Нейронные сети: общие технологические характеристики // Научное обозрение: Технические науки. – 2019. – № 2. – С. 17–23.
8. Болдырев Д.В., Шерстнев П.А., Липинский Л.В. Исследование влияния параметров искусственной нейронной сети на её точность в задачах классификации // Решетнёвские чтения. Материалы XXIV Международной научно-практической конференции, посвящённой памяти генерального конструктора ракетно-космических систем академика М. Ф. Решетнёва. В 2 частях. 2020. С. 152–154.
9. Dubey S. R., Singh S. K., Chaudhuri B. B. Activation functions in deep learning: a comprehensive survey and benchmark // Neurocomputing. – 2022. – Т. 503. – С. 92–108. – DOI: 10.1016/j.neucom.2022.06.111.
10. Aggarwal C. C. Neural networks and deep learning: a textbook. – Cham: Springer, 2018. – 497 с. – ISBN 978-3-319-94463-0. – DOI: 10.1007/978-3-319-94463-0.
11. Hochreiter S., Schmidhuber J. Long short-term memory // Neural Computation. – 1997. – Т. 9, № 8. – С. 1735–1780. – DOI: 10.1162/neco.1997.9.8.1735.
12. Cho K., van Merriënboer B., Gulcehre C., Bahdanau D. et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation // Proceedings of the 2014 Conference on Empirical Methods in Natural Language

Processing (EMNLP 2014). – Doha, Qatar: Association for Computational Linguistics, 2014. – С. 1724–1734. – DOI: 10.3115/v1/D14-1179.
13. Schuster M., Paliwal K. K. Bidirectional recurrent neural networks // IEEE Transactions on Signal Processing. – 1997. – Т. 45, № 11. – С. 2673–2681.
14. Bahdanau D., Cho K., Bengio Y. Neural machine translation by jointly learning to align and translate [Электронный ресурс] // arXiv preprint. – 2016. – arXiv:1409.0473. – Режим доступа: https://arxiv.org/abs/1409.0473 (дата обращения: 21.06.2025).
15. Wang N., Su H., Aliverti A. Enhanced long short-term memory (LSTM)-based streamflow prediction // Journal of Hydrology. – 2022. – Т. 28. – С. 2107–2124.
16. Jameer S., Syed H. Deep SE-BiLSTM with IFPOA fine-tuning for human activity recognition using mobile and wearable sensors // Sensors. – 2023. – Т. 23, № 9. – С. 4319. – DOI: 10.3390/s23094319.
17. Xiong R., Shen W., Cao J., Lin C. Deep learning-based intelligent state of health estimation model for lithium-ion battery using long short-term memory network // Energy. – 2021. – Т. 231. – Статья 120882. – DOI: 10.1016/j.energy.2021.120882.
18. Hannun A. Y., Rajpurkar P., Haghpanahi M. et al. Cardiologist-level arrhythmia detection with convolutional neural networks // Nature Medicine. – 2019. – Т. 25. – С. 65–69. – DOI: 10.1038/s41591-018-0268-3.
19. Mohammed F. A., Tune K. K., Assefa B. G., Jett M., Muhie S. Medical image classifications using convolutional neural networks: a survey of current methods // Machine Learning and Knowledge Extraction. – 2024. – Т. 6, № 1. – С. 699–735. – DOI: 10.3390/make6010032.
20. Pradhan B., Rahman M. A., Tomar R., Chilamkurti N., Kim B.-G. Application of Internet of Things on the healthcare field using convolutional neural network processing: ECG classification [Электронный ресурс] // – 2023. – Режим доступа: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8808223/ (дата обращения: 29.05.2025).
21. Vie A., Kleinnijenhuis A. M., Farmer D. J. Qualities, challenges and future of genetic algorithms: a literature review [Электронный ресурс] // arXiv preprint. – 2020. – arXiv:2011.05277. – Режим доступа: https://arxiv.org/abs/2011.05277 (дата обращения: 29.05.2025).
22. Keshri M. K. BloombergGPT: Revolutionizing Finance with Large Language Models [Электронный ресурс] // SSRN Electronic Journal. – 2025. – DOI: 10.2139/ssrn.5215949.
23. Jaiswal J., Das R. Application of artificial neural networks with backpropagation technique in the financial data // IOP Conference Series: Materials Science and Engineering. – 2017. – Т. 263, № 4. – Статья 042139. – DOI: 10.1088/1757-899X/263/4/042139.
24. Rumelhart D. E., Hinton G. E., Williams R. J. Learning representations by back-propagating errors // Nature. – 1986. – Т. 323, № 6088. – С. 533–536. – DOI: 10.1038/323533a0.
25. Ioffe S., Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift // Proceedings of the 32nd International Conference

on Machine Learning (ICML). – 2015. – С. 448–456. – DOI: 10.5555/3045118.3045167.
26. Wu Y., He K. Group normalization // Proceedings of the 2018 European Conference on Computer Vision (ECCV). – Munich, 2018. – Т. 30. – С. 3–19. – DOI: 10.1007/978-3-030-01261-8_1.
27. Liu H. et al. An improvement of Adam based on a cyclic exponential decay and gradient norm constraint (CN-Adam) // Electronics. – 2024. – Т. 13, № 9. – Статья 1778. – DOI: 10.3390/electronics13091778.
28. Miikkulainen R. Neuroevolution // Scholarpedia. – 2009. – Т. 4, № 6. – С. 1448. – DOI: 10.4249/scholarpedia.1448.
29. Stanley K. O., Miikkulainen R. Evolving neural networks through augmenting topologies // Evolutionary Computation. – 2002. – Т. 10, № 2. – С. 99–127.
30. Stanley K. O., D'Ambrosio D. B., Gauci J. A hypercube-based encoding for evolving large-scale neural networks // Artificial Life. – 2009. – Т. 15, № 2. – С. 185–212. – DOI: 10.1162/artl.2009.15.2.15202.
31. Liu Y., Sun Y., Xue B., Zhang M., Yen G. G., Tan K. C. A survey on evolutionary neural architecture search // IEEE Transactions on Neural Networks and Learning Systems. – 2023. – Т. 34, № 2. – С. 550–570. – DOI: 10.1109/TNNLS.2021.3100554.
32. Zadeh L. A., Aliev R. A. Fuzzy logic theory and applications: Part I and Part II. – New Jersey: World Scientific, 2018. – 612 с. – ISBN 978-981-323-8176.
33. Zadeh L. A. Fuzzy sets // Information and Control. – 1965. – Т. 8, № 3. – С. 338–353. – DOI: 10.1016/S0019-9958(65)90241-X.
34. Sadat Asl A. A. S., Ershadi M. M., Sotudian S., Li X., Dick S. Fuzzy expert systems for prediction of ICU admission in patients with COVID-19 // Intelligent Decision Technologies. – 2022. – Т. 16, № 1. – С. 159–168. – DOI: 10.3233/IDT-200220.
35. Zeng Y., Hussein Z. A., Chyad M. H., Baloch H. Z., Li H. Integrating type-2 fuzzy logic controllers with digital twin and neural networks for advanced hydropower system management // Scientific Reports. – 2025. – Т. 15. – Статья 5140. – DOI: 10.1038/s41598-025-89866-5.
36. Şahin M., Sönmez M., Dede A. A type-2 fuzzy rule-based model for diagnosis of COVID-19 // Soft Computing. – 2022. – Т. 26, № 14. – С. 6789–6801. – DOI: 10.1007/s00500-021-06045-4.
37. Mamdani E. H., Assilian S. An experiment in linguistic synthesis with a fuzzy logic controller // International Journal of Man-Machine Studies. – 1975. – Т. 7, № 1. – С. 1–13. – DOI: 10.1016/S0020-7373(75)80002-2.
38. Takagi T., Sugeno M. Fuzzy identification of systems and its applications to modeling and control // IEEE Transactions on Systems, Man, and Cybernetics. – 1985. – Т. 15, № 1. – С. 116–132. – DOI: 10.1109/TSMC.1985.6313399.
39. Larsen P. M. Industrial applications of fuzzy logic control // International Journal of Man-Machine Studies. – 1980. – Т. 12, № 1. – С. 3–10.
40. Tsukamoto Y. An approach to fuzzy reasoning method // In: Gupta M. M., Ragade R. K., Yager R. R. (eds.) Advances in fuzzy set theory and applications. – Amsterdam: North-Holland, 1979. – С. 137–149.

41. Quinlan J. R. Induction of decision trees // Machine Learning. – 1986. – Т. 1, № 1. – С. 81–106. – DOI: 10.1007/BF00116251.
42. Blockeel H., Devos L., Frénay B., Nanfack G., Nijssen S. Decision trees: from efficient prediction to responsible AI // Frontiers in Artificial Intelligence. – 2023. – Т. 6. – Статья 1124553. – DOI: 10.3389/frai.2023.1124553.
43. Rahman M. S. A., Jamaludin N. A. A., Zainol Z., Sembok T. M. T. The application of decision tree classification algorithm on decision-making for upstream business // International Journal of Advanced Computer Science and Applications (IJACSA). – 2023. – Т. 14, № 8. – DOI: 10.14569/IJACSA.2023.0140873.
44. Schapire R. E. The strength of weak learnability // Machine Learning. – 1990. – Т. 5, № 2. – С. 197–227. – DOI: 10.1007/BF00116037.
45. Freund Y., Schapire R. E. A decision-theoretic generalization of on-line learning and an application to boosting // Journal of Computer and System Sciences. – 1997. – Т. 55, № 1. – С. 119–139. – DOI: 10.1006/jcss.1997.1504.
46. Chen T., Guestrin C. XGBoost: a scalable tree boosting system // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016). – 2016. – С. 785–794. – DOI: 10.1145/2939672.2939785.
47. Ke G., Meng Q., Finley T., Wang T., Chen W., Ma W., Ye Q., Liu T. Y. LightGBM: a highly efficient gradient boosting decision tree // Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017). – 2017. – Т. 30. – С. 3146–3154.
48. Prokhorenkova L., Gusev G., Vorobev A., Dorogush A. V., Gulin A. CatBoost: unbiased boosting with categorical features // Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS). – 2018. – Т. 31.
49. Breiman L. Bagging predictors // Machine Learning. – 1996. – Т. 24, № 2. – С. 123–140. – DOI: 10.1007/BF00058655.
50. Breiman L. Random forests // Machine Learning. – 2001. – Т. 45, № 1. – С. 5–32.
51. Wolpert D. H. Stacked generalization // Neural Networks. – 1992. – Т. 5, № 2. – С. 241–259. – DOI: 10.1016/S0893-6080(05)80023-1.
52. Nanni L., Lumini A., Brahnam S. Ensemble of networks for multilabel classification // Information. – 2020. – Т. 11, № 4. – Статья 198. – DOI: 10.3390/info11040198.
53. Li Y., Ma R., Zhang Q., Wang Z., Zong L., Liu X. Neural architecture search using attention enhanced precise path evaluation and efficient forward evolution // Scientific Reports. – 2025. – Т. 15, № 1. – DOI: 10.1038/s41598-025-94187-8.
54. Begum A. M., Mondal M. R. H., Podder P., Kamruzzaman J. Weighted rank difference ensemble: a new form of ensemble feature selection method for medical datasets // BioMedInformatics. – 2024. – Т. 4, № 1. – С. 477–488. – DOI: 10.3390/biomedinformatics4010027.
55. Corne D., Shapiro J. (eds.) Evolutionary computing: AISB International Workshop, Manchester, UK, April 7–8, 1997. Selected papers // Berlin: Springer, 1997. – 307 с. – (Lecture Notes in Computer Science; Т. 1213). – DOI: 10.1007/BFb0032773.

56. Давронов Ш. Р. Обзор современных генетических алгоритмов и их применение на практике // Молодой ученый. – 2023. – № 36(483). – С. 15–18.
57. Alam T., Qamar S., Dixit A., Benaida M. Genetic algorithm: reviews, implementations, and applications // International Journal of Engineering Pedagogy (iJEP). – 2020. – Т. 10, № 6. – С. 57–77. – DOI: 10.3991/ijep.v10i6.14567.
58. Holland, J. H. Genetic algorithms // Scientific American. – 1992. – Vol. 267, № 1. – P. 66–72.
59. Bolotbekova A., Hakli H., Beskirli A. Trip route optimization based on bus transit using genetic algorithm with different crossover techniques: a case study in Konya/Türkiye // Scientific Reports. – 2025. – Т. 15. – Статья 86695. – DOI: 10.1038/s41598-025-86695-4.
60. Rivera G., Rodas-Osollo J., Bañuelos P., Quiroz M., Lopez M. Genetic algorithm for surgery scheduling optimization in a Mexican public hospital // Complexity. – 2019. – Article ID 269274. – DOI: 10.1155/2019/269274.
61. Mohammed M. A., Ghani M. K. A., Obaid O. I., Mostafa S. A. A review of genetic algorithm application in examination timetabling problem // Journal of Engineering and Applied Sciences. – 2017. – Т. 12, № 20. – С. 5166–5181.
62. Thierens D., Bosman P. A. N. Linkage neighbors, optimal mixing and forced improvements in genetic algorithms // Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO 2011). – 2011. – С. 687–694. – DOI: 10.1145/2001576.2001666.
63. Pelikan M., Goldberg D. E., Tsutsui S. Hierarchical BOA — toward a new generation of evolutionary algorithms // SICE Annual Conference. – 2003. – С. 2738–2743.
64. Thierens D. The Linkage Tree Genetic Algorithm // In: Schaefer R., Cotta C., Kołodziej J., Rudolph G. (eds.) Parallel Problem Solving from Nature – PPSN XI: 11th International Conference, Kraków, Poland, September 11–15, 2010, Proceedings. – Berlin: Springer, 2010. – (Lecture Notes in Computer Science; Т. 6238). – С. 264–273. – DOI: 10.1007/978-3-642-15844-5_27.
65. Chen Y.-P., Yu T.-L., Sastry K., Goldberg D. E. A survey of linkage learning techniques in genetic and evolutionary algorithms. – Urbana-Champaign: University of Illinois at Urbana-Champaign, IlliGAL Report No. 2007014, 2007. – 45 с.
66. Deb K., Pratap A., Agarwal S., Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II // IEEE Transactions on Evolutionary Computation. – 2002. – Т. 6, No. 2. – P. 182–197. – DOI: 10.1109/4235.996017.
67. Deb K., Jain H. NSGA-III: reference-point-based non-dominated sorting // IEEE Transactions on Evolutionary Computation. – 2014. – Т. 18, № 4. – С. 577–601.
68. Zhang Q., Li H. MOEA/D: multi-objective evolutionary algorithm based on decomposition // IEEE Transactions on Evolutionary Computation. – 2007. – Т. 11, № 6. – С. 712–731. – DOI: 10.1109/TEVC.2007.892759.
69. Koza J. R. Genetic programming: on the programming of computers by means of natural selection. – 6-е изд. – Cambridge, MA: MIT Press, 1998. – 609 с. – ISBN 978-0262111706.

70. Al-Helali B., Chen Q., Xue B., Zhang M. Genetic programming-based feature selection for symbolic regression on incomplete data // Evolutionary Computation. – 2024. – С. 1–27. – DOI: 10.1162/evco_a_00362.
71. Santoso L., Singh B., Rajest S., Rajan R., Kadhim K. A genetic programming approach to binary classification problem [Электронный ресурс] // EAI Endorsed Transactions on Energy Web. – 2020. – DOI: 10.4108/eai.13-7-2018.165523. – Режим доступа: https://doi.org/10.4108/eai.13-7-2018.165523 (дата обращения: 21.06.2025).
72. Wang C., Qu Y., Lu Z., An H., Xia H., Ma G. Trajectory optimization method for spacecraft orbit transfer with finite thrust // Journal of Southwest Jiaotong University. – 2013. – Т. 48. – С. 390–394. – DOI: 10.3969/j.issn.0258-2724.2013.02.030.
73. Semenkin E., Semenkina M. Self-configuring genetic programming algorithm with modified uniform crossover // 2012 IEEE Congress on Evolutionary Computation (CEC). – June 2012. – С. 1–6. – DOI: 10.1109/CEC.2012.6256587.
74. Kuranga C., Pillay N. A comparative study of genetic programming variants // In: Rutkowski L., Scherer R., Korytkowski M., Pedrycz W., Tadeusiewicz R., Zurada J. M. (eds.) Artificial Intelligence and Soft Computing. ICAISC 2022. – Cham: Springer, 2023. – (Lecture Notes in Computer Science; Т. 13588). – С. 377–386. – DOI: 10.1007/978-3-031-23492-7_32.
75. Al-Madi N., Ludwig S. A. Adaptive genetic programming applied to classification in data mining // Proceedings of the 2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC), Mexico City, Mexico. – 2012. – С. 79–85. – DOI: 10.1109/NaBIC.2012.6402243.
76. Harris S., Bueter T., Tauritz D. A comparison of genetic programming variants for hyper-heuristics // Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO 2015). – 2015. – С. 1043–1050. – DOI: 10.1145/2739482.2768456.
77. Miller J. F., Thomson P. Cartesian genetic programming // In: Proceedings of the European Conference on Genetic Programming. – Berlin: Springer, 2000. – (Lecture Notes in Computer Science; Т. 1802). – С. 121–132.
78. Brameier M., Banzhaf W. Linear genetic programming. – New York: Springer, 2007. – 259 с. – ISBN 978-0387310299.
79. Storn R., Price K. Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces // Journal of Global Optimization. – 1997. – Т. 11, № 4. – С. 341–359. – DOI: 10.1023/A:1008202821328.
80. Elsayed S., Sarker R., Essam D. Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems // Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC 2011). – 2011. – С. 1041–1048. – DOI: 10.1109/CEC.2011.5949724.
81. Tanabe R., Fukunaga A. S. Improving the search performance of SHADE using linear population size reduction // Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC 2014), Beijing, China. – 2014. – С. 1658–1665. – DOI: 10.1109/CEC.2014.6900380.

82. Lin A., Liu D., Li Z., Li D., Zeng J. A hybrid differential evolution and particle swarm optimization algorithm with local search strategy // Complex & Intelligent Systems. 2023. Т. 9. С. 6905–6925. DOI: 10.1007/s40747-023-01082-8.
83. Sopov A., Sherstnev P. An investigation of the hybridization of DE and BFGS algorithms // Hybrid Methods of Modeling and Optimization in Complex Systems (HMMOCS 2022): Proceedings of the I International Workshop. Krasnoyarsk, 2023. С. 336–342.
84. Tan X., Shin S.-Y., Shin K.-S., Wang G. A multipopulation differential evolution algorithm with uniform local search // Applied Sciences. 2022. Т. 12. № 16. С. 8087. DOI: 10.3390/app12168087.
85. Wolpert D. H., Macready W. G. No free lunch theorems for optimization // IEEE Transactions on Evolutionary Computation. – 1997. – Т. 1, № 1. – С. 67–82. – DOI: 10.1109/4235.585893.
86. Niehaus J., Banzhaf W. Adaption of operator probabilities in genetic programming // Proceedings of the 4th European Conference on Genetic Programming (EuroGP 2001). – Berlin: Springer, 2001. – (Lecture Notes in Computer Science; Т. 2038). – С. 325–336. – DOI: 10.1007/3-540-45355-5_25.
87. Семенкина М. Е. Самоадаптивные эволюционные алгоритмы проектирования информационных технологий интеллектуального анализа данных // Искусственный интеллект и принятие решений. – 2013. – № 1. – С. 13–23.
88. Stanovov V., Akhmedova S., Semenkin E. Genetic Algorithm with Success History based Parameter Adaptation // Proceedings of the 11th International Joint Conference on Computational Intelligence (IJCCI 2019). – SciTePress, 2019. – С. 180–187. – DOI: 10.5220/0008071201800187.
89. Potter M. A., De Jong K. A. A cooperative coevolutionary approach to function optimization // IEEE Transactions on Evolutionary Computation. – 2000. – Т. 4, № 1. – С. 75–87. – DOI: 10.1109/4235.996017.
90. Wang H., Sun Y., Liu X. An enhanced cooperative coevolutionary algorithm with adaptive grouping for large-scale optimization // Swarm and Evolutionary Computation. – 2022. – Т. 74. – Статья 102972. – DOI: 10.1016/j.swevo.2022.102972.
91. Zhang Q., Li S., Sun X. Adaptive decomposition-based cooperative coevolution for complex multi-modal optimization // Information Sciences. – 2023. – Т. 612. – С. 1–20. – DOI: 10.1016/j.ins.2022.11.001.
92. Mallipeddi R., Suganthan P. N., Pan Q.-K., Tasgetiren M. F. Ensemble differential evolution algorithm for CEC2011 problems // Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC). – IEEE, 2011. – С. 2157–2164. – DOI: 10.1109/CEC.2011.5949801.
93. Akhmedova Sh. A., Semenkin E. S. Collective bionic algorithm with biogeography based migration operator for binary optimization // Журнал СФУ. Математика и физика. – 2016. – № 1.
94. Basak A., Butz M. V., Kovacs T. A. Rank based adaptive mutation in genetic algorithm // International Journal of Computer Applications. – 2020. – Т. 175, № 10. – С. 49–55. – DOI: 10.5120/ijca2020920572.

95. Sherstnev P. Self-adaptation Method for Evolutionary Algorithms Based on the Selection Operator // High-Performance Computing Systems and Technologies in Scientific Research, Automation of Control and Production. HPCST 2023 / eds. Jordan V., Tarasov I., Shurina E., Filimonov N., Faerman V.A. Cham: Springer, 2024. (Communications in Computer and Information Science; vol. 1986).
96. IEEE Congress on Evolutionary Computation [Электронный ресурс]. – 2025. – Режим доступа: https://www.cec2025.org/ (дата обращения: 29.05.2025).
97. Stanovov V., Akhmedova S., Semenkin E. NL-SHADE-LBC algorithm with linear parameter adaptation bias change for CEC 2022 numerical optimization // Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC). – IEEE, 2022. – С. 1–8. – DOI: 10.1109/CEC55065.2022.9870295.
98. Stanovov V., Semenkin E. Success rate-based adaptive differential evolution L-SRTDE for CEC 2024 competition // Proceedings of the 2024 IEEE Congress on Evolutionary Computation (CEC). – IEEE, 2024. – С. 1–8. – DOI: 10.1109/CEC60901.2024.10611907.
99. Чахлев С. В., Рыбаков С. В. Автоматическая генерация мутационных операторов с использованием гиперэвристики // Вестник Тюменского государственного университета. Математика. Механика. Информатика. – 2018. – № 4. – С. 154–165.
100. Burke E. K., Gendreau M., Hyde M., Kendall G., Ochoa G., Özcan E., Qu R. Hyper-heuristics: a survey of the state of the art // Journal of the Operational Research Society. – 2013. – Т. 64, № 12. – С. 1695–1724. – DOI: 10.1057/jors.2013.71.
101. Sopov E., Videnin S. Online selective evolutionary hyperheuristic for large scale economic load dispatch problem // 2019 IEEE International Conference on Information Technologies (InfoTech-2019). – IEEE, 2019. – Статья 8860884. – 4 с. – DOI: 10.1109/InfoTech.2019.8860884.
102. Risi S., Stanley K. O. An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons // Artificial Life. – 2012. – Т. 18, № 4. – С. 331–363. – DOI: 10.1162/artl_a_00076.
103. Miikkulainen R. et al. Evolving deep neural networks [Электронный ресурс] // Artificial Intelligence in the Age of Neural Networks and Brain Computing / Kozma R., Alippi C., Choe Y., Morabito F. C. (eds.). – Amsterdam: Elsevier, 2019. – С. 293–312. – DOI: 10.1016/B978-0-12-816250-8.00017-6.
104. Real E., Aggarwal A., Huang Y., Le Q. V. Regularized evolution for image classifier architecture search // Proceedings of the AAAI Conference on Artificial Intelligence. – 2019. – С. 4780–4789.
105. Xu Y., Ma Y. Evolutionary neural architecture search combining multi-branch ConvNet and improved Transformer // Scientific Reports. – 2023. – Т. 13. – Статья 15791. – DOI: 10.1038/s41598-023-42931-3.
106. Липинский Л. В., Семенкин Е. С. Применение алгоритма генетического программирования в задачах автоматизации проектирования интеллектуальных информационных технологий // Вестник Сибирского государственного аэрокосмического университета им. акад. М. Ф. Решетнева. – 2006. – № 3(10). – С. 22–26.

107. Шерстнев П.А. Настройка структуры искусственной нейронной сети с помощью самоконфигурируемого метода генетического программирования // Актуальные проблемы авиации и космонавтики. Сборник материалов VIII Международной научно-практической конференции, посвященной Дню космонавтики. В 3 т. Красноярск, 2022. С. 101–103.
108. Sherstnev P.A., Polyakova A.S., Lipinskiy L.V., Semenkin E.S. Evolutionary algorithm for automated formation of recurrent neural networks // AIP Conference Proceedings. 2024. Vol. 3021, № 1. Article 060035.
109. Шерстнев П.А., Липинский Л.В. Эволюционный алгоритм проектирования искусственных нейронных сетей с перераспределением ресурсов // Российская наука, инновации, образование: РОСНИО-2022. Сборник научных статей по материалам Всероссийской научной конференции. Красноярск, 2022. С. 131–141.
110. Шерстнев П.А., Семенкин Е.С., Липинский Л.В., Полякова А.С. Программная система формирования рекуррентных нейронных сетей гибридным самоконфигурируемым эволюционным алгоритмом // Свидетельство о регистрации программы для ЭВМ № RU 2023686866. 11.12.2023.
111. Li Y., Wang Y., Wang Y., Wang Y. High-dimensional ensemble learning classification // Applied Sciences. – 2024. – Т. 14, № 5. – Статья 1956. – DOI: 10.3390/app14051956.
112. Winter B. D., Teahan W. J. Ecological neural architecture search [Электронный ресурс] // arXiv preprint. – 2025. – arXiv:2503.10908. – Режим доступа: https://arxiv.org/abs/2503.10908 (дата обращения: 29.05.2025).
113. Poyatos J., Molina D., Martínez A., Del Ser J., Herrera F. Multiobjective evolutionary pruning of deep neural networks with transfer learning for improving their performance and robustness // Applied Soft Computing. – 2023. – Т. 147. – Статья 110757. – DOI: 10.1016/j.asoc.2023.110757.
114. Carse B., Fogarty T. C. A fuzzy classifier system using the Pittsburgh approach // Parallel Problem Solving from Nature — PPSN III: Proceedings of the International Conference. – Berlin: Springer, 1994. – С. 259–269. – DOI: 10.1007/3-540-58484-6_270.
115. Ishibuchi H., Nakashima T., Murata T. Genetic fuzzy systems: recent advances and future challenges // Fuzzy Sets and Systems. – 2004. – Т. 141, № 1. – С. 3–31. – DOI: 10.1016/S0165-0114(03)00056-4.
116. Zhang M., Zhang M., Song Z. Fuzzy MoCoCo: interpretable policies via multi-objective cooperative coevolution for reinforcement learning [Электронный ресурс] // arXiv preprint. – 2023. – arXiv:2305.09922. – Режим доступа: https://arxiv.org/abs/2305.09922 (дата обращения: 29.05.2025).
117. Casillas J., Carse B., Bull L. Fuzzy-XCS: a Michigan genetic fuzzy system // IEEE Transactions on Fuzzy Systems. – 2007. – Т. 15, № 4. – С. 536–550. – DOI: 10.1109/TFUZZ.2007.900904.
118. Bernadó-Mansilla E., Garrell-Guiu J. M. Accuracy-based learning classifier systems: models, analysis and applications to classification tasks // Evolutionary Computation. – 2003. – Т. 11, № 3. – С. 209–238. – DOI: 10.1162/106365603322365289.

119. Становов В. В., Семенкина О. Э. Самоконфигурирующийся гибридный эволюционный алгоритм формирования нечетких классификаторов с активным обучением для несбалансированных данных // Вестник СибГАУ. – 2014. – № 5(57). – С. 128–135.
120. Nasiri A., Sadeghzadeh S. M., Nasiri A. A hybrid fuzzy genetic algorithm for sustainable agricultural optimization // Sustainability. – 2025. – Т. 17, № 7. – Статья 2829. – DOI: 10.3390/su17072829.
121. Wagner T., Hüllermeier E. Evolutionary multi-objective learning of fuzzy rules: competition-based versus cooperative strategies // Information Sciences. – 2021. – Т. 546. – С. 1030–1051. – DOI: 10.1016/j.ins.2020.09.042.
122. Bishop J. T., Gallagher M., Browne W. N. A genetic fuzzy system for interpretable and parsimonious reinforcement learning policies (Fuzzy MoCoCo) // Companion Proceedings of the 2021 Genetic and Evolutionary Computation Conference (GECCO ’21 Companion). – 2021. – DOI: 10.1145/3449726.3463198.
123. Rodríguez-Fdez I., Mucientes M., Bugarín A. FRULER: fuzzy rule learning through evolution for regression // Parallel Problem Solving from Nature – PPSN XIII: Proceedings of the 13th International Conference, Edinburgh, UK, September 2014. – Berlin: Springer, 2015. – (Lecture Notes in Computer Science; Т. 8672). – С. 173–183. – DOI: 10.1007/978-3-319-10762-2_17.
124. Ngo G., Beard R., Chandra R. Evolutionary bagging for ensemble learning // Engineering Applications of Artificial Intelligence. – 2022. – Т. 112. – Статья 104934. – DOI: 10.1016/j.engappai.2022.104934.
125. Oliveira J., Casanova H., Cardoso J. S. A genetic algorithm to optimize stacking ensembles in streaming scenarios // Applied Intelligence. – 2020. – Т. 53, № 5. – С. 2055–2070. – DOI: 10.1007/s10489-020-01862-x.
126. Хритоненко Д. И., Семенкин Е. С., Сугак Е. В., Потылицына Е. Н. Решение задачи прогнозирования экологического состояния города нейроэволюционными алгоритмами // Вестник Сибирского государственного аэрокосмического университета им. акад. М. Ф. Решетнева. – 2015. – Т. 16, № 1. – С. 137–142. – EDN TRIUYL.
127. Peng Y., Li E. A generative model-based coevolutionary training framework for noise-tolerant soft sensors in wastewater treatment processes // Complex & Intelligent Systems. – 2025. – Т. 11, № 2. – С. 123–140. – DOI: 10.1007/s40747-025-01845-5.
128. Dick G. An ensemble learning interpretation of geometric semantic genetic programming // Genetic Programming and Evolvable Machines. – 2024. – Т. 25, № 1. – С. 1–20. – DOI: 10.1007/s10710-024-09412-3.
129. Семенкин Е. С., Шабалов А. А. Система автоматизированного проектирования коллективов интеллектуальных информационных технологий для задач анализа данных // Программные продукты и системы. – 2012. – № 4. – С. 70–73. – EDN OXSJHP.
130. Fortin F. A., De Rainville F. M., Gardner M. A., Parizeau M. G. C. DEAP: evolutionary algorithms made easy // Journal of Machine Learning Research. – 2012. – Т. 13. – С. 2171–2175.

131. Paszke A., Gross S., Massa F., Lerer A. и др. PyTorch: An Imperative Style, High-Performance Deep Learning Library // Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). – Vancouver, Canada: NeurIPS, 2019. – С. 8024–8035.
132. Gad A. F. PyGAD: an intuitive genetic algorithm Python library // Multimedia Tools and Applications. – 2024. – Т. 83, № 20. – С. 58029–58042. – DOI: 10.1007/s11042-023-17167-y.
133. Blank J., Deb K. Pymoo: Multi-Objective Optimization in Python // IEEE Access. – 2020. – Т. 8. – С. 89497–89509.
134. Huang B., Cheng R., Li Z., Jin Y., Tan K. C. EvoX: Distributed GPU-accelerated Framework for Scalable Evolutionary Computation // IEEE Transactions on Evolutionary Computation. – 2024. – DOI: 10.1109/TEVC.2024.3388550.
135. BaumEvA: Bauman Evolution Algorithm [Электронный ресурс]. – 2025. – Режим доступа: https://github.com/DateOrMage/BaumEvolutionAlgorithms (дата обращения: 29.05.2025).
136. Gplearn's documentation [Электронный ресурс]. – 2023. – Режим доступа: https://gplearn.readthedocs.io/en/stable/ (дата обращения: 29.05.2025).
137. Biscani F., Izzo D. A parallel global multiobjective framework for optimization: pagmo // Journal of Open Source Software. – 2020. – Т. 5, № 53. – С. 2338. – DOI: 10.21105/joss.02338.
138. Nikitin N. O., Vychuzhanin P., Sarafanov M., Polonskaia I. S., Revin I., Barabanova I. V., Maximov G., Kalyuzhnaya A. V., Boukhanovsky A. Automated evolutionary approach for the design of composite machine learning pipelines // Future Generation Computer Systems. – 2021. – ISSN 0167-739X. – DOI: 10.1016/j.future.2021.08.022.
139. Wagner S., Kronberger G., Beham A., Kommenda M., Scheibenpflug A., Winkler S. HeuristicLab: a general purpose extensible open source software framework for heuristic optimization // Proceedings of the European Conference on the Applications of Evolutionary Computation (EvoApplications 2014). – Springer, 2014. – С. 538–549. – DOI: 10.1007/s10710-014-9214-4.
140. Luke S. ECJ — A Java-based Evolutionary Computation Research System [Электронный ресурс] // George Mason University. – 2025. – Режим доступа: https://cs.gmu.edu/~eclab/projects/ecj/ (дата обращения: 11.06.2025).
141. Durillo J. J., Nebro A. J. jMetal: a Java framework for multi-objective optimization // Advances in Engineering Software. – 2011. – Т. 42, № 10. – С. 760–771. – DOI: 10.1016/j.advengsoft.2011.05.014.
142. Keijzer M., Merelo J. J., Romero G., Schoenauer M., Alba E. Evolving Objects: a general purpose evolutionary computation library in C++ // Artificial Evolution. – Berlin: Springer, 2002. – С. 231–242. – DOI: 10.1007/3-540-46033-0_20.
143. Борисевич Д. И., Панфилов И. А. Генетический алгоритм с альтернативным представлением решений // Решетневские чтения. – 2015. – № 2(19). – С. 20–21.
144. Semenkin E. S., Semenkina M. E. Self-configuring genetic algorithm with modified uniform crossover operator // Lecture Notes in Computer Science. – 2012. – Т. 7331. – С. 414–421. – DOI: 10.1007/978-3-642-30976-2_52.

145. Шерстнев П.А., Семенкин Е.С. SelfCSHAGA: Самоконфигурируемый генетический алгоритм оптимизации с адаптацией на основе истории успеха // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2025. № 2 (151). С. 122–139.
146. IEEE Congress on Evolutionary Computation (CEC 2005) // Proceedings of the IEEE Congress on Evolutionary Computation. – Edinburgh, United Kingdom, 2–5 September 2005. – Piscataway, NJ: IEEE Press, 2005. – ISBN 0-7803-9363-5.
147. IEEE Congress on Evolutionary Computation (CEC 2013) // Proceedings of the IEEE Congress on Evolutionary Computation. – Cancún, Mexico, 20–23 June 2013. – Piscataway, NJ: IEEE Press, 2013. – ISBN 978-1-4799-0452-5.
148. IEEE Congress on Evolutionary Computation (CEC 2014) // Proceedings of the IEEE Congress on Evolutionary Computation. – Beijing, China, 6–11 July 2014. – Piscataway, NJ: IEEE Press, 2014. – ISBN 978-1-4799-1488-3.
149. Yu E. L., Suganthan P. N. Ensemble of niching algorithms // Information Sciences. – 2010. – Т. 180, № 15. – С. 2815–2833. – DOI: 10.1016/j.ins.2010.04.008.
150. Шерстнев П.А., Семенкин Е.С. Программная система решения задач оптимизации самоконфигурируемым генетическим алгоритмом с адаптацией на основе истории успеха // Свидетельство о регистрации программы для ЭВМ № RU 2025664786. 06.06.2025.
151. Kelly M., Longjohn R., Nottingham K. The UCI Machine Learning Repository [Электронный ресурс] // University of California, Irvine. – 2025. – Режим доступа: https://archive.ics.uci.edu (дата обращения: 21.06.2025).
152. Tanabe R., Fukunaga A. S. Success-History based parameter adaptation for differential evolution // Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC 2013), Cancun, Mexico. – 2013. – С. 71–78. – DOI: 10.1109/CEC.2013.6564644.
153. Шерстнев П. А., Семенкин Е. С. Самоконфигурируемые алгоритмы генетического программирования с адаптацией на основе истории успеха // Сибирский аэрокосмический журнал. 2025. Т. 26. № 1. С. 60–70.
154. Marian S., Tegmark M. E. AI Feynman: a physics-inspired method for symbolic regression // Science Advances. – 2020. – Т. 6, № 16. – Статья eaay2631. – DOI: 10.1126/sciadv.aay2631.
155. Шерстнев П.А., Семенкин Е.С. Программная система решения задач оптимизации самоконфигурируемым алгоритмом генетического программирования с адаптацией на основе истории успеха // Свидетельство о регистрации программы для ЭВМ № RU 2025664787. 06.06.2025.
156. Шерстнев П.А. Исследование эффективности самоконфигурируемых алгоритмов генетического программирования в задачах классификации и регрессии // Фундаментальные и прикладные исследования в науке и образовании: сборник статей Международной научно-практической конференции. Ижевск, 2025. С. 99–103.
157. Шерстнев П.А., Семенкин Е.С. Автоматизированное проектирование ансамблей нейронных сетей самоконфигурируемыми эволюционными

алгоритмами // Системы управления и информационные технологии. 2025. № 2 (100). С. 52–58.
158. Шерстнев П.А., Семенкин Е.С. Программная система автоматизированного проектирования ансамблей нейросетей с использованием самоконфигурируемых эволюционных алгоритмов с адаптацией на основе истории успеха // Свидетельство о регистрации программы для ЭВМ № RU 2025664073. 03.06.2025.
159. Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E. Scikit-learn: machine learning in Python // Journal of Machine Learning Research. – 2011. – Т. 12. – С. 2825–2830.
160. Sherstnev P. Thefittest: evolutionary machine learning in Python // Hybrid Methods of Modeling and Optimization in Complex Systems (HMMOCS-II 2023): Proceedings of the II International Workshop. Krasnoyarsk: ITM Web of Conferences, 2024. Vol. 59. Article 02020. 11 p.
161. Sherstnev P.A. Self-configuring evolutionary algorithms-based design of hybrid interpretable machine learning models // Hybrid Methods of Modeling and Optimization in Complex Systems (HMMOCS 2022): Proceedings of the I International Workshop. Krasnoyarsk, 2023. С. 313–320.
162. Шерстнев П.А., Семенкин Е.С. Применение эволюционных алгоритмов при проектировании интерпретируемых моделей машинного обучения в задачах классификации // Системы управления и информационные технологии. 2022. № 1 (87). С. 17–20.
163. Шерстнев П.А., Семенкин Е.С. Программная система проектирования интерпретируемых технологий искусственного интеллекта на основе нейронных сетей и нечеткой логики гибридными самоконфигурируемыми генетическими алгоритмами // Свидетельство о регистрации программы для ЭВМ № RU 2023684452. 15.11.2023.
164. Зеленько А. А., Струков Б. С., Реснянский Ю. Д. и др. Система прогнозирования ветрового волнения в мировом океане и морях России // Труды ГОИН. – М.: ГОИН, 2014. – Вып. 215. – С. 90–101.
165. Ратнер Ю. Б., Фомин В. В., Иванчик А. М., Иванчик М. В. Система оперативного прогноза ветрового волнения Черноморского центра морских прогнозов // Морской гидрофизический журнал. – 2017. – № 5. – С. 56–66.
166. Руденко М. А., Смирнов В. О., Снегур А. В., Скрипниченко Г. В. Агрометеорологическая база данных метеорологических параметров, собранная на основе показаний автоматической метеостанции «Сокол-М», расположенной в с. Маленькое // Свидетельство о регистрации базы данных № RU 2025621129. 13.03.2025.
167. Руденко М. А., Смирнов В. О., Снегур А. В., Скрипниченко Г. В. Агрометеорологическая база данных метеорологических параметров, собранная на основе показаний автоматической метеостанции «Сокол-М», расположенной в с. Светлое // Свидетельство о регистрации базы данных № RU 2025621114. 12.03.2025.

168. Parkinson S., Ceccaroni L., Edelist D., Robertson E., Horincar R., Laudy C., Ganchev T., Markova V., Pearlman J., Simpson P., Venus V., Muchada P., Kazanjian G., Bye B. L., Oliveira M., Paredes H., Sprinks J., Witter A., Cruz B., Das K., Woods S. M. The Iliad digital twins of the ocean: opportunities for citizen science // ARPHA Proceedings. – 2024. – Т. 6. – P. 61–65. – DOI: 10.3897/ap.e126643.
169. Solari G., Repetto M. P., Burlando M., De Gaetano P., Pizzo M., Tizzi M., Parodi M. The wind forecast for safety management of port areas // Journal of Wind Engineering and Industrial Aerodynamics. – 2012. – Т. 104–106. – С. 266–277. – DOI: 10.1016/j.jweia.2012.03.029.
170. Burlando M., De Gaetano P., Pizzo M., Repetto M. P., Solari G., Tizzi M. Wind short-term forecast in port areas // Proceedings of the Sixth European and African Conference on Wind Engineering, 7–11 July 2013, Cambridge, United Kingdom. – Nottingham: University of Nottingham, 2013.
171. Artificial Intelligence Laboratory [Электронный ресурс]. – 2025. – Режим доступа: http://ailab.tu-varna.bg/index.php/resources (дата обращения: 05.05.2025).
172. Шерстнев П. А., Семенкин Е. С., Митрофанов С. А., Ганчев Т. Д. Автоматизированное проектирование интерпретируемой модели машинного обучения для оперативного прогнозирования силы ветра на морском побережье // Моделирование, оптимизация и информационные технологии. 2025. Т. 13. № 2. DOI: 10.26102/2310-6018/2025.49.2.032. URL: https://moitvivt.ru/ru/journal/pdf?id=1945.
173. Zhang W., Tian M., Hai Sh., Wang F. Improving the forecasts of coastal wind speeds in Tianjin, China based on the WRF model with machine learning algorithms // Journal of Meteorological Research. – 2024. – Т. 38, № 3. – С. 570–585. – DOI: 10.1007/s13351-024-3096-z.
174. Chu X., Bai W., Sun Y., Li W., Liu C., Song H. A machine learning-based method for wind fields forecasting utilizing GNSS radio occultation data // IEEE Access. – 2022. – Т. 10. – С. 30258–30273. – DOI: 10.1109/ACCESS.2022.3159231.
175. Grigorieva G. M., Kagan M. B., Letin V. A., Nadorov V. P., Evenov G. D., Hartov V. V. Analysis of geostationary spacecraft solar arrays degradation from solar proton flares // Space Power: Proceedings of the Sixth European Conference. – Porto, Portugal: European Space Agency, 2002. – ESA SP 502. – С. 725–730. – ISBN 92-9092-812-3.
176. Шерстнев П.А. Применение самоконфигурируемых эволюционных алгоритмов при проектировании интерпретируемых моделей прогноза деградации солнечных батарей космического аппарата // Научные исследования: основа современной инновационной системы: в 2 ч. Ч. 1. Сборник статей Международной научно-практической конференции. Саратов, 2025. С. 171–176.
177. Семенкина М. Е., Семенкин Е. С., Рыжиков И. С. Прогнозирование динамики электрических характеристик солнечных батарей космических аппаратов методами вычислительного интеллекта // Вестник СибГАУ. – 2014. – № 3(55). – С. 139–145.
178. Semenkina M., Akhmedova S., Semenkin E., Ryzhikov I. Spacecraft solar arrays degradation forecasting with evolutionary designed ANN-based predictors //

Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2014). – SciTePress, 2014. – С. 421–428. – DOI: 10.5220/0005122004210428.

179. Bukhtoyarov V., Semenkin E. S., Shabalov A. A. Neural networks ensembles approach for simulation of solar arrays degradation process // Lecture Notes in Computer Science. – 2012. – Т. 7208, ч. 1. – С. 186–195. – DOI: 10.1007/978-3-642-28942-2_17. – EDN PDMJCX.
180. Шерстнев П.А., Храмов И.В., Храмова К.Р., Мохирев А.П., Мохирев И.А. Интеллектуальная система построения модели зависимости звукового давления усовершенствованной древесной плиты интерпретируемыми технологиями искусственного интеллекта на основе нечеткой логики // Свидетельство о регистрации программы для ЭВМ № RU 2024664367. 20.06.2024.
181. Шерстнев П.А., Храмов И.В., Гузоватова А.Д. Применение самонастраивающихся эволюционных алгоритмов для автоматизации проектирования моделей звукового давления древесных панелей // Прорывные научные исследования как двигатель науки: сборник статей Международной научно-практической конференции. Пермь, 2025. С. 42–45.

ПРИЛОЖЕНИЕ 1

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025664073

Программная система автоматизированного проектирования ансамблей нейросетей с использованием самоконфигурируемых эволюционных алгоритмов с адаптацией на основе истории успеха

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Сибирский федеральный университет» (RU)***

Авторы: ***Шерстнев Павел Александрович (RU), Семенкин Евгений Станиславович (RU)***

Заявка № **2025663251**
Дата поступления **26 мая 2025 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***03 июня 2025 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0692e7c1a6300bf54f2401670bcc2026
Владелец Зубов Юрий Сергеевич
Действителен с 10.07.2024 по 03.10.2025

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025664786

Программная система решения задач оптимизации самоконфигурируемым генетическим алгоритмом с адаптацией на основе истории успеха

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева» (RU)***

Авторы: ***Шерстнев Павел Александрович (RU), Семенкин Евгений Станиславович (RU)***

Заявка № **2025663908**
Дата поступления **06 июня 2025 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***06 июня 2025 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0692e7c1a6300bf54f240f670bcc2026
Владелец Зубов Юрий Сергеевич
Действителен с 10.07.2024 по 03.10.2025

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025664787

Программная система решения задач оптимизации самоконфигурируемым алгоритмом генетического программирования с адаптацией на основе истории успеха

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева» (RU)***

Авторы: ***Шерстнев Павел Александрович (RU), Семенкин Евгений Станиславович (RU)***

Заявка № **2025663909**

Дата поступления **06 июня 2025 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***06 июня 2025 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0692e7c1a6300bf54f240f670bcc2026
Владелец Зубов Юрий Сергеевич
Действителен с 10.07.2024 по 03.10.2025

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023684452

«Программная система проектирования интерпретируемых технологий искусственного интеллекта на основе нейронных сетей и нечеткой логики гибридными самоконфигурируемыми генетическими алгоритмами»

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Сибирский федеральный университет» (СФУ) (RU)***

Авторы: ***Шерстнев Павел Александрович (RU), Семенкин Евгений Станиславович (RU)***

Заявка № **2023669731**
Дата поступления **26 сентября 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***15 ноября 2023 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023686866

Программная система формирования рекуррентных нейронных сетей гибридным самоконфигурируемым эволюционным алгоритмом

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Сибирский федеральный университет» (СФУ) (RU)***

Авторы: ***Шерстнев Павел Александрович (RU), Семенкин Евгений Станиславович (RU), Липинский Леонид Витальевич (RU), Полякова Анастасия Сергеевна (RU)***

Заявка № **2023669811**
Дата поступления **26 сентября 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***11 декабря 2023 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2024664367

Интеллектуальная система построения модели зависимости звукового давления усовершенствованной древесной плиты интерпретируемыми технологиями искусственного интеллекта на основе нечеткой логики

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Сибирский федеральный университет» (СФУ (RU)***

Авторы: ***Шерстнев Павел Александрович (RU), Храмов Игорь Владимирович (RU), Храмова Ксения Романовна (RU), Мохирев Александр Петрович (RU), Мохирев Иван Александрович (RU)***

Заявка № **2024663786**
Дата поступления **20 июня 2024 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***20 июня 2024 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3853164baf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов

ПРИЛОЖЕНИЕ 2

МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Сибирский федеральный университет»
ИНСТИТУТ МАТЕМАТИКИ И ФУНДАМЕНТАЛЬНОЙ ИНФОРМАТИКИ
660041, Красноярский край,
г.Красноярск, проспект Свободный, д. 79
телефон: (391) 206-21-48
http://math.sfu-kras.ru, e-mail: math@sfu-kras.ru

№ 872 от «11» июня 2025 г.

Для предъявления по месту требования

Справка
об использовании результатов диссертационного исследования

Настоящим письмом институт математики и фундаментальной информатики Сибирского федерального университета подтверждает активное использование преподавателями института фреймворка ***thefittest*** (https://github.com/sherstpasha/thefittest), разработчик **Шерстнев Павел Александрович**.

Фреймворк использовался в 2024-2025 учебном году (и планируется к использованию в дальнейшем) в образовательном процессе при проведении практических работ по факультативному курсу «Оптимизация сложных систем» для студентов магистратуры, обучающихся по программе подготовки «Математическое моделирование» направления подготовки 01.04.02 «Прикладная математика и информатика». Курс читается базовой кафедрой математического моделирования и процессов управления ИМиФИ СФУ.

Необходимо подчеркнуть высокую полезность и эффективность моделей и алгоритмов, реализованных во фреймворке thefittest. Данный фреймворк впервые сделал доступным рядовому пользователю, не являющемуся экспертом в методах эволюционной оптимизации и вычислительного интеллекта, технологии высшего научного уровня.

Заместитель директора ИМиФИ
по науке, канд. физ.-мат. наук, доцент Сорокин Р.В.

ПРИЛОЖЕНИЕ 3

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное
учреждение высшего образования

**«Сибирский государственный
университет науки и технологий
имени академика М.Ф. Решетнева»
(СибГУ им. М.Ф. Решетнева)**

**Институт информатики
и телекоммуникаций**

проспект им. газеты Красноярский рабочий, 31
г. Красноярск, 660037,
тел.: +7 (391) 264-00-14 факс: +7 (391) 264-47-09
http://www.sibsau.ru e-mail: info@sibsau.ru
ОКПО 02069734, ОГРН 1022402056038
ИНН/КПП 2462003320/246201001

Для предъявления по месту требования

СПРАВКА

№ 142122 от 03.06.2025

СПРАВКА

об использовании результатов диссертационного исследования

Настоящим письмом институт Информатики и телекоммуникаций Сибирского государственного университета науки и технологий имени академика М.Ф. Решетнева подтверждает активное использование фреймворка ***thefittest*** (https://github.com/sherstpasha/thefittest), разработчик **Шерстнев Павел Александрович,** в образовательном процессе.

Фреймворк используется в образовательном процессе при проведении практических и лабораторных работ по курсам «Эволюционные алгоритмы моделирования и оптимизации сложных систем» магистерской программы «Интеллектуальные системы анализа данных и моделей принятия решений», «Системы поддержки принятия решений» магистерской программы «Системный анализ в управлении проектами и бизнес-процессами», «Человеко-машинные методы в управлении сложными системами» магистерской программы «Системный анализ в условиях неопределенности» кафедры системного анализа и исследования операций СибГУ им. М.Ф. Решетнева.

Подчеркиваем значительную полезность моделей и алгоритмов, реализованных во фреймворке thefittest, сделавшем доступным рядовому пользователю, не являющемуся экспертом в методах эволюционной оптимизации и вычислительного интеллекта, современных технологий моделирования и оптимизации сложных систем.

Директор ИИТК СибГУ,
д.ф.-м.н., профессор Сафонов К.В.

ПРИЛОЖЕНИЕ 4

МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение
высшего образования
«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
имени В.И. Вернадского»
(ФГАОУ ВО «КФУ им. В.И. Вернадского»)
проспект Академика Вернадского, 4
г. Симферополь, 295007
Тел.: +7(3652) 54-50-36
E-mail: cfuv@crimeaedu.ru
http://cfuv.ru

17 ИЮН 2025 № 10/3-12/634

На № _____ от _____

По месту требования

Справка о внедрении

Настоящим подтверждаем, что в Центре искусственного интеллекта и анализа больших данных Крымского федерального университета имени В. И. Вернадского применяются научные результаты, разработанные Шерстневым Павлом Александровичем в рамках его диссертационного исследования.

Центром ИИиАБД КФУ созданы базы данных метеорологических параметров, собранных на основе показаний автоматической метеостанции СОКОЛ-М, расположенной в семи точках побережья Черного моря, и проводятся прикладные научные исследования, направленные на построение точных и интерпретируемых моделей машинного обучения для оперативного прогнозирования локальных метеорологических характеристик.

В данных исследованиях используется в настоящее время и предполагается к использованию в дальнейшем библиотека открытого доступа Шерстнева П.А. *thefittest* (https://github.com/sherstpasha/thefittest) и входящие в нее программные системы эволюционной оптимизации и проектирования моделей машинного обучения.

В частности, используется авторский метод гибридизации точных моделей на основе ансамблей искусственных нейронных сетей и интерпретируемых моделей на основе нечеткой логики, проектирование и обучение которых выполняется с применением разработанной Шерстневым П.А. системы автоматизированного проектирования ансамблей нейронных сетей GPENN эволюционными самоконфигурируемыми алгоритмами моделирования и оптимизации с адаптацией на основе истории успеха PDPSHAGP и SelfCSHAGA. Применение данного метода повышает прикладную ценность создаваемых моделей и технологий оперативного прогнозирования метеорологической обстановки морского побережья.

Проректор по научной деятельности Н.В. Любомирский

ПРИЛОЖЕНИЕ 5

THE ARTIFICIAL INTELLIGENCE LABORATORY
Technical University of Varna,
Str. Studentska, №1, BG9010, Varna, Bulgaria.
Tel.: +359 878 011 083
Email: tganchev@tu-varna.bg
Web: http://ailab.tu-varna.bg/

Date: May 21th, 2025. TO WHOM IT MIGHT CONCERN

ACT OF IMPLEMENTATION AND USE

We hereby confirm that within the framework of the Cooperation Agreement of January 12, 2022, between the Reshetnev Siberian State University of Science and Technology (Krasnoyarsk, Russia) and the Technical University of Varna (Varna, Bulgaria) during the implementation of the initiative joint project "*Design of an intelligent information system for prompt forecasting of local wind characteristics on the sea coast*", the TUV's Artificial Intelligence Laboratory used the open-access framework ***thefittest*** (https://github.com/sherstpasha/thefittest), developed by **Pavel Sherstnev**, including the following author's algorithms:

1. Self-configuring genetic optimization algorithm with success history-based adaptation.
2. Self-configuring genetic programming algorithm with success history-based adaptation.
3. Automated design system for neural network ensembles using self-configuring evolutionary algorithms.

The result of this use was an interpretable machine learning model, namely a fuzzy logic-based prognostic system, suitable for implementation on a portable device (e.g. a smartphone) and capable of accurately forecasting the location-specific meteorological parameters at the specific area on the Black Sea coast for the upcoming three hours based on real-time sensor data buffered for last one and a half hours. The implementation of this result in practice allows the port authorities, marine traffic managers and organisers of events on the coast to make more informed decisions, especially in rapidly changing weather conditions, thus ensuring a higher level of safety.

The specified framework and algorithms are expected to be used in the future for further improvement of the developed intelligent information technology for prompt location-specific forecasting of meteorological parameters, as well as in other research and innovation projects of the TUV's Artificial Intelligence Laboratory.

Sincerely,
Prof. Todor Ganchev, Ph.D.
Head of the Artificial Intelligence Laboratory at TUV

Todor Dimitrov Ganchev

Digitally signed by Todor Dimitrov Ganchev
Date: 2025.05.21 10:18:02 +03'00'

Логотип лаборатории искусственного интеллекта

Лаборатория искусственного интеллекта

Технический университет г. Варны,
Болгария, г. Варна, BG9010,
ул. Студенстка, 1
тел.: +359 878 011 083
Email: tganchev@tu-varna.bg
Web: http://ailab.tu-varna.bg/

Логотип

Дата: 21 мая 2025 г.

По месту требования

Справка о внедрении

Настоящим подтверждаем, что в рамках договора о сотрудничестве от 12 января 2022 г. между Сибирским государственным университетом науки и технологий им. М.Ф. Решетнева (Красноярск, Россия) и Техническим Университетом Варны (Варна, Болгария) в ходе выполнения инициативного совместного проекта "«Разработка интеллектуальной информационной системы оперативного прогнозирования локальных характеристик ветра на морском побережье»" лаборатория искусственного интеллекта ВТУ использовала фреймворк открытого доступа ***thefittest*** (**https://github.com/sherstpasha/thefittest**), разработанный **Павлом Шерстневым**, в том числе следующие авторские алгоритмы:

- Самоконфигурируемый генетический алгоритмы оптимизации с адаптацией на основе истории успеха;
- Самоконфигурируемый алгоритм генетического программирования с адаптацией на основе истории успеха;
- Система автоматизированного проектирования ансамблей нейронных сетей с применением самоконфигурируемых эволюционных алгоритмов.

Результатом этого использования является интерпретируемая модель машинного обучения, а именно основанная на нечеткой логике прогнозирующая система, удобная для реализации на портативном устройстве (например, смартфоне) и способная достаточно точно прогнозировать локальные метеорологические параметры в конкретной зоне черноморского побережья на следующие три часа с использованием оперативных данных датчиков, собранных за последние полтора часа. Применение этого результата на практике позволяет руководству мороского порта, менеджерам морского трафика и организаторам мероприятий на морском побережье принимать более обоснованные решения, в особенности в условиях быстрых изменений погоды, обеспечивая тем самым более высокий уровень безопасности.

Указанные фреймворк и алгоритмы предполагается использовать в дальнейшем для совершенствования разрабатываемых интеллектуальных информационных технологий для оперативного локального прогнозирования метеорологических параметров, а также в других научно-исследовательских и инновационных проектах лаборатории искусственного интеллекта ВТУ.

С уважением,
Проф. Тодор Ганчев, Ph.D.
Заведующий лабораторией
искусственного интеллекта
ВТУ

Тодор Димитров Ганчев

Подписано электронно-цифровой подписью
Дата: 21.05.2025, 10:18:02 +03`00`

Перевод верен: переводчик управления международного сотрудничества Горбунова Светлана Владимировна
г. Красноярск, 23.05.2025

УПРАВЛЕНИЕ МЕЖДУНАРОДНОГО СОТРУДНИЧЕСТВА

ПРИЛОЖЕНИЕ 6

Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования

«Московский государственный
технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

105005, г. Москва, 2-я Бауманская ул., д. 5, стр. 1
тел. +7 (499) 263-63-91, факс +7 (499) 267-48-44
bmstu.ru bauman@bmstu.ru

ОГРН 1027739051779
ИНН 7701002520 КПП 770101001

_____ № _____

На № _____ от _____

По месту требования

Об использовании результатов
диссертационного исследования
Шерстнева П.А.

В Научно-образовательном центре «Технологии искусственного интеллекта» МГТУ им. Баумана ведется разработка программно-алгоритмического комплекса (фреймворка) BaumEvA, главной задачей которого является обеспечение автоматического проектирования математических моделей, моделей машинного обучения и технологий искусственного интеллекта на основе адаптивных стохастических алгоритмов моделирования и оптимизации сложных систем.

В ходе выполнения разработки установлено, что наиболее близким на настоящий момент к проекту аналогом является фреймворк ***thefittest***, разработанный и развиваемый Шерстневым Павлом Александровичем, инженером-исследователем Центра ИИ Сибирского федерального университета (г. Красноярск).

Идеология и наполнение данного фреймворка в целом имеет высокую степень схожести с нашим проектом, при этом имеет более широкий набор модельно-алгоритмических средств, а его автор – Шерстнев П.А – участвует в сравнительном анализе алгоритмического и программного обеспечения двух подходов к реализации фреймворка эволюционных алгоритмов решения сложных задач оптимизации и их применения в области автоматизированного проектирования технологий искусственного интеллекта, полностью соответствующих современным требованиям безопасности, интерпретируемости, доверительности и надежности.

Кроме того, фреймворк Шерстнева П.А. и входящие в него программные системы эволюционной оптимизации и проектирования моделей машинного обучения, реализующие метод гибридизации точных моделей на основе ансамблей искусственных нейронных сетей и интерпретируемых моделей на основе нечеткой логики, проектирование и обучение которых выполняется с применением разработанной Шерстневым П.А. системы автоматизированного проектирования ансамблей нейронных сетей GPENN на основе эволюционных самоконфигурируемых алгоритмов моделирования и оптимизации с адаптацией на основе истории успеха PDPSHAGP и SelfCSHAGA апробированы к настоящему моменту и предполагаются к использованию в дальнейшем при выполнении исследований по проекту «Инкассо» НОЦ «ФНС России и МГТУ им. Н.Э. Баумана» («Разработка модели, методов и алгоритмов машинного обучения в целях анализа операций по банковским выпискам, направленного на прогнозирование временного диапазона для выставления инкассового поручения на оптимальный счет налогоплательщика»).

Применение авторского метода гибридизации, использующего самоконфигурируемые генетические алгоритмы моделирования и оптимизации с адаптацией на основе истории успеха, позволяет получать более эффективные модели поддержки принятия решений.

Старший научный сотрудник
НОЦ «ФНС России и МГТУ им. Н.Э. Баумана»
НОЦ «Технологии искусственного интеллекта»
Московского государственного технического университета
им. Н.Э. Баумана, кандидат физико-математических наук — Виталий Александрович Красиков

Руководитель отдела разработки
НОЦ «ФНС России и МГТУ им. Н.Э. Баумана»
НОЦ «Технологии искусственного интеллекта»
Московского государственного технического университета
им. Н.Э. Баумана — Денис Александрович Суханов

Руководитель проектов
НОЦ «ФНС России и МГТУ им. Н.Э. Баумана»
НОЦ «Технологии искусственного интеллекта»
Московского государственного технического университета
им. Н.Э. Баумана — Дмитрий Юрьевич Евсюков

krasikov@bmstu.ru
sukhanov@bmstu.ru
evsyukov@bmstu.ru

ВЕРНО
НАЧАЛЬНИК ОТДЕЛА
КНЯПИНА А.Н.
ОТДЕЛ ПООРГАНИЗАЦИИ РАБОТЫ ЕДИНОЙ ПРИЁМНОЙ УКСИА МГТУИМЕН И.Н.Э.БАУМАНА

ПРИЛОЖЕНИЕ 7