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

На правах рукописи

Ахмедова Шахназ Агасувар кызы

КОЛЛЕКТИВНЫЙ САМОНАСТРАИВАЮЩИЙСЯ МЕТОД ОПТИМИЗАЦИИ НА ОСНОВЕ БИОНИЧЕСКИХ АЛГОРИТМОВ

05.13.01 – Системный анализ, управление и обработка информации (информатика, вычислительная техника, управление)

ДИССЕРТАЦИЯ

на соискание ученой степени кандидата технических наук

Научный руководитель
Семенкин Евгений Станиславович
доктор технических наук,
профессор

Красноярск – 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования. При решении многих практических задач часто возникает необходимость выбора наилучшего решения по некоторому критерию из множества возможных. Математически такой выбор формализуется в виде задачи оптимизации.

Широко известные методы математического программирования для решения задач оптимизации представляют собой детерминированную итерационную процедуру пошагового улучшения одного текущего решения. Эффективность таких алгоритмов основывается на полном использовании удобных с точки зрения оптимизации свойств (выпуклость, гладкость, и т.п.) целевой функции, которые, к тому же, полагаются известными заранее. Для многих практических задач такие свойства либо не выполняются, либо неизвестны заранее, поэтому применение данных методов нецелесообразно. Для решения таких задач в настоящее время используются недетерминированные (стохастические), работающие одновременно с большим количеством текущих решений (многоагентные) алгоритмы, являющиеся более эффективными и универсальными. Например, генетический алгоритм [46], метод роя частиц [101], муравьиный алгоритм [79] и т.д.

Многоагентные алгоритмы, основанные на использовании популяции, работают с набором потенциальных решений. Каждое решение постепенно улучшается и оценивается, таким образом, каждое потенциальное решение влияет на то, как будут улучшены другие решения. Большинство популяционных методов заимствовало эту концепцию из биологии: процесс поиска наилучшего решения «копирует» некоторый природный процесс либо поведение определенных видов животных, причем учитываются их видовые особенности. Класс сложных систем, именуемых как алгоритмы стайного типа, также часто употребляется термин «бионические алгоритмы», – богатый источник нестандартных численных методов, с помощью которых можно решать сложные задачи, когда известно недостаточно информации об оптимизируемой функции.

Один из особенно популярных и часто используемых бионических методов известен как стайный алгоритм оптимизации (Particle Swarm Optimization, PSO), также называемый методом роя частиц (МРЧ). Данный метод был предложен для решения задач безусловной оптимизации с вещественными переменными Кеннеди и Эберхартом в 1995 году [101]. Идея стайного алгоритма была почерпнута из социального поведения стада копытных, стаи птиц, косяка рыб и т.д., то есть процесс поиска оптимального решения данным методом имитирует поиск пищи группой животных, насекомых или птиц одного вида без уточнения самого вида.

Помимо метода роя частиц есть множество других алгоритмов, аналогичных ему, но имитирующих некоторые свойства или поведение уже конкретных видов животных. Например, алгоритм поиска кукушек (Cuckoo Search Algorithm, CSA), разработанный Янгом и Дэбом в 2009 году [4], имитирующий гнездовой паразитизм некоторых видов кукушек, подкладывающих свои яйца в гнезда других птиц. Тем же Янгом были разработаны и другие алгоритмы стайного типа: в 2009 году алгоритм светлячков (Firefly Algorithm, FFA) [140], использующий свойство свечения этих насекомых, в 2010 году алгоритм летучих мышей (Bat Algorithm, BA) [139], имитирующий процесс поиска пищи летучими мышами, учитывая их способность к эхолокации. Также как пример можно привести еще один бионический метод, но разработанный уже Карабогом в 2005 году [99] – искусственный алгоритм пчелиной семьи (ABC), который тоже часто используется для решения оптимизационных задач и имитирует поведение кормовых медоносных пчел. Кроме того, следует упомянуть алгоритм поиска стаей волков (Wolf Pack Search, WPS) [75], речь о котором пойдет далее. Примеров бионических методов можно приводить множество, ведь с каждым годом их разрабатывается все больше.

Следует также отметить, что большинство алгоритмов стайного типа изначально разработано для решения задач безусловной оптимизации с вещественными переменными, и все они имеют некоторые параметры, которые необходимо подбирать при решении той или иной задач (наиболее значимый в

этом понимании «параметр» – размер популяции потенциальных решений). Однако год за годом предлагаются различные модификации этих методов:

1. Модификации, связанные с гибридизацией алгоритмов (например, гибридный алгоритм на базе метода роя частиц и алгоритма поиска кукушек, представленный в статье [135]; или гибридный метод на базе алгоритма светлячков и муравьиного алгоритма [91]);
2. Модификации для расширения круга решаемых задач (например, решение оптимизационных задач с бинарными переменными алгоритмом летучих мышей [115] или решение задач многокритериальной оптимизации алгоритмом светлячков [68]);
3. Модификации для настройки параметров алгоритмов (например, уже в 1998 году Ши и Эберхарт опубликовали работу [127], а в 2000 году они же ввели еще один параметр для алгоритма PSO [81]).

Однако исследования показали, что невозможно заранее определить, какой именно из этих алгоритмов следует применить для решения той или иной оптимизационной задачи, кроме того, как уже было упомянуто, для каждого алгоритма нужно выбирать заранее размер популяции потенциальных решений, что в свою очередь также является сложной задачей.

Таким образом, *цель диссертационной работы* состоит в повышении эффективности решения задач оптимизации бионическими алгоритмами за счет автоматизации их выбора и настройки параметров.

Поставленная цель предопределила необходимость решения следующего комплекса взаимосвязанных *задач*:

1. Исследовать эффективность различных бионических методов для решения задач условной и безусловной оптимизации с вещественными и бинарными переменными, определить наиболее успешный из них;
2. Разработать коллективный самонастраивающийся бионический алгоритм для решения задач условной и безусловной оптимизации с вещественными переменными;

3. Разработать модификации нового алгоритма для решения задач условной и безусловной оптимизации с бинарными переменными;
4. Реализовать разработанные алгоритмы в виде программных систем и оценить их эффективность на репрезентативном множестве тестовых задач;
5. Провести апробацию разработанных алгоритмов при решении реальных практических задач.

Научная новизна диссертационной работы состоит в следующем:

1. На основе пяти известных бионических алгоритмов разработан, реализован и исследован новый коллективный метод решения задач безусловной оптимизации с вещественными переменными, отличающийся от известных способом организации взаимодействия популяций и настройки параметров;
2. Разработана, реализована и исследована модификация нового метода для решения задач безусловной оптимизации с бинарными переменными;
3. Разработана, реализована и исследована модификация нового метода для решения задач условной оптимизации с вещественными и бинарными переменными;
4. На основе разработанного метода оптимизации предложены новые алгоритмы автоматического проектирования нейронных сетей и машин опорных векторов.

Апробация работы. Процесс разработки алгоритмов и результаты проведенных исследований докладывались в период 2010-2015 гг. на 35 конференциях различного уровня, среди которых 11 зарубежных, 8 Международных, 1 Всероссийская с международным участием и 15 молодежных научных конференций, в том числе: International Conference on Swarm Intelligence (ICSI, Hefei, China, 2014; Beijing, China, 2015), Informatics in Control, Automation and Robotics (ICINCO, Vienna, Austria, 2014; Colmar, France, 2015), International Congress on Advanced Applied Informatics (AAI, Okayama, Japan, 2015), Congress on

Evolutionary Computations of the IEEE World Congress on Computational Intelligence (CEC WCCI, Beijing, China, 2014), International Conference on Computer Science and Artificial Intelligence (ICCSAI, Wuhan, China, 2014), Conference on Engineering and Applied Sciences Optimization (OPT-I, Kos Island, Greece, 2014), Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, (Baltimore, USA, 2014), Congress on Evolutionary Computations (CEC, Cancun, Mexico, 2013), Genetic and Evolutionary Computation Conference (GECCO, Amsterdam, Holland, 2013), International Workshop on Mathematical Models and its Applications (IWMMA, Baykal, Russia, 2012; Krasnoyarsk, Russia, 2013, 2014, 2015), XIV Национальная конференция по искусственному интеллекту с международным участием (КИИ, г.Казань, 2014), V Международная конференция «Системный анализ и информационные технологии» (САИТ, Красноярск, 2013), XVI, XVIII и XIX Международные научные конференции «Решетневские чтения» (г. Красноярск, 2012, 2014, 2015 гг.), и др. Отдельные результаты работы обсуждались на научном семинаре института информационных технологий университета г. Ульм (Германия, 2014). Диссертация в целом обсуждалась на научно-технических семинарах кафедры системного анализа и исследования операций СибГАУ и кафедры систем автоматизированного проектирования (РК6) НИУ МГТУ им. Н.Э.Баумана.

Разработанные алгоритмы использованы при выполнении исследований в рамках российско-германских проектов (совместно с университетом г. Ульм) «Распределенные интеллектуальные информационные системы обработки и анализа мультилингвистической информации в диалоговых информационно-коммуникационных системах» (ФЦП ИР, ГК №11.519.11.4002) и «Математическое и алгоритмическое обеспечение автоматизированного проектирования аппаратно-программных комплексов интеллектуальной обработки мультилингвистической информации в распределенных высокопроизводительных системах космического назначения» (ФЦП НПК, ГК № 16.740.11.0742), российско-словенского проекта (совместно с университетом г. Марибор) «Manpower control strategy determination with self-adapted evolutionary

and biologically inspired algorithms» (ARRS Project BI-RU/14-15-047), а также в рамках проекта №8.5541.2011 «Развитие теоретических основ автоматизации математического моделирования физических систем на основе экспериментальных данных» и проекта № 140/14 «Разработка теоретических основ эволюционного проектирования интеллектуальных информационных технологий анализа данных» тематического плана ЕЗН СибГАУ. Данная работа поддержана грантом Фонда содействия развитию малых форм предприятий в научно-технической сфере в рамках программы У.М.Н.И.К., стипендией имени Аниты Борг от компании Google (Google Anita Borg Scholarship) и стипендией Президента РФ молодым ученым и аспирантам, осуществляющим перспективные научные исследования и разработки по приоритетным направлениям модернизации российской экономики.

Зарегистрированы следующие программные системы: «Коллективный метод безусловной оптимизации на основе стайных бионических алгоритмов» (Свидетельство о государственной регистрации программ для ЭВМ №2014610132), «Коллективный метод условной оптимизации на основе стайных бионических алгоритмов» (Свидетельство о государственной регистрации программ для ЭВМ №2013661150), «Коллективный метод безусловной оптимизации на основе стайных бионических алгоритмов для решения задач с бинарными переменными» (Свидетельство о государственной регистрации программ для ЭВМ №2014611007), «Машины опорных векторов, автоматически сгенерированные коллективными методами условной и безусловной оптимизации на основе стайных бионических алгоритмов» (Свидетельство о государственной регистрации программ для ЭВМ №2014610968), «Автоматическое генерирование нейронных сетей алгоритмами стайного типа» (Свидетельство о государственной регистрации программ для ЭВМ №2014616237), «Проектирование SVM-классификаторов коллективными бионическими алгоритмами с выбором информативных входов» (Свидетельство о государственной регистрации программ для ЭВМ №2015611847).

Публикации. По материалам данной работы опубликовано 22 печатные работы, в том числе 6 статей в научных изданиях Перечня ВАК, 10 в изданиях, индексируемых в международной базе Scopus, из них 4 индексированы также в Web of Science (публикации в сборниках ведущих международных конференций, а также публикации в сборниках различных всероссийских, региональных конференций).

1 Алгоритмы стайного типа: реализация и исследование эффективности

В настоящее время существует множество различных многоагентных стохастических алгоритмов оптимизации. Одним из наиболее изученных среди них является стайный алгоритм или Particle Swarm Optimization (PSO), разработанный Кеннеди и Эберхартом в 1995 году [101]. Идея данного метода почерпнута из социального поведения некоторых видов животных, например, стай птиц, косяка рыб или стада копытных. Исследования показали эффективность алгоритма и целесообразность его применения при решении задач как безусловной, так и условной оптимизации функций вещественных переменных. Постоянно предлагаются новые варианты алгоритма для улучшения производительности метода либо для расширения круга решаемых задач (например, одна из первых модификаций была связана с идеей решения однокритериальных задач безусловной оптимизации с бинарными переменными с помощью PSO).

Помимо PSO существуют и другие алгоритмы, использующие социальные и биологические идеи, имитирующие поведение определенных видов животных. Наибольший интерес из последних разработок представляют следующие бионические алгоритмы: алгоритм поиска кукушек (Cuckoo Search Algorithm, CSA) [141], алгоритм летучих мышей (Bat Algorithm, BA) [139], алгоритм светлячков (Firefly Algorithm, FFA) [140] и алгоритм стай волков (Wolf Pack Search, WPS) [75]. Перечисленные метаэвристики, как и PSO, изначально были разработаны для решения задач однокритериальной безусловной оптимизации с вещественными переменными, и так же, как и PSO, могут быть модифицированы для решения задач безусловной и условной оптимизации с бинарными переменными. Как уже было сказано, каждый из упомянутых алгоритмов имитирует некоторую характеристику определенного вида животных: CSA – способ откладывания яиц кукушками, BA – эхолокацию летучих мышей, FFA – излучение, исходящее от светлячков, WPS – процесс охоты стаи волков.

Далее в диссертации описаны перечисленные алгоритмы, а также результаты проведенных исследований. Для описания работы методов предположим, что необходимо решить задачу безусловной оптимизации с вещественными переменными.

1.1. Метод роя частиц (Particle Swarm Optimization, PSO)

Стайный алгоритм – метод численной оптимизации, базирующийся на моделировании поведения популяции частиц в пространстве оптимизации, для использования которого не требуется знать точного градиента оптимизируемой функции [101]. Данный метод привлекателен простотой реализации, он может использоваться для решения многих задач, включая обучение нейронных сетей.

Первоначально PSO был создан для задач с вещественными переменными. Пусть $f(x)$ целевая функция, которую необходимо минимизировать, причем $x = (x_1,\ldots, x_d,\ldots, x_D)$. Работа алгоритма начинается с создания популяции частиц (то есть множества потенциальных решений x_i, $i=\overline{1,N}$, где N – размер популяции) случайным образом. Частицы представляют собой вектор координат точки в пространстве оптимизации (вещественных чисел) $x_i = (x_{i1},\ldots, x_{id},\ldots, x_{iD})$, $i=\overline{1,N}$. Каждая частица передвигается по поверхности графика функции с какой-то скоростью $v_i = (v_{i1},\ldots, v_{id},\ldots, v_{iD})$, $i=\overline{1,N}$. Частицы изменяют свою скорость и координаты, основываясь на собственном опыте и опыте других частиц. Таким образом, моделируется многоагентная система, где агенты-частицы двигаются к оптимальным решениям, обмениваясь при этом информацией с соседями. Скорость и координаты частиц обновляются по следующим формулам:

$$v_{id}^{t+1} = \omega \cdot v_{id}^{t} + c_1 \cdot rand() \cdot (p_{id} - x_{id}^{t}) + c_2 \cdot Rand() \cdot (p_{gd} - x_{id}^{t}),$$

$$x_{id}^{t+1} = v_{id}^{t+1} + x_{id}^{t}.$$

В этих формулах v_{id}^{t+1}, v_{id}^{t} – скорости частицы на (t+1)-ой и t-ой итерациях соответственно, x_{id}^{t+1}, x_{id}^{t} – координаты частицы на (t+1)-ой и t-ой итерациях соответственно, p_{id} – лучшая позиция, найденная i-ой частицей за t предыдущих

поколений, p_{gd} – лучшая позиция, найденная всей стаей за все время работы алгоритма, *rand*() и *Rand*() – случайные числа из отрезка [0;1], c_1, c_2 – коэффициенты обучения из отрезка [0;2], ω – инерционный вес.

Константа c_1 называется когнитивным (познавательным) параметром, она позволяет учитывать «собственный опыт» (историю) частицы. Константа c_2 называется социальным параметром и позволяет частице учитывать «опыт всей стаи». Таким образом, c_2 управляет воздействием глобального лучшего положения, а c_1 управляет воздействием личного лучшего положения на скорость каждой частицы.

Работу алгоритма также определяет топология соседства частиц, а именно, каким образом выбирается лучший индивид для каждой частицы. Наиболее известными являются следующие топологии: кольцо, клика, двумерный тор, кластер [112].

Итак, идея алгоритма заключается в том, что частицы, которые сначала равномерно распределены по поверхности отклика функции, с течением времени (от поколения к поколению) начинают группироваться («сбиваться в стаи») около локальных минимумов, причем наибольшая стая собирается около глобального оптимума. При этом почти всегда имеются частицы, находящиеся в стороне от таких стай, а также частицы, выскакивающие за границы допустимой области.

В настоящее время существуют различные модификации стайного алгоритма оптимизации, разработанные для решения различных задач оптимизации. Одна из первых модификаций была предложена Кеннеди и Эберхартом в 1997 году для решения задач однокритериальной безусловной оптимизации с бинарными переменными [102]. В 1998 году Ши и Эберхарт опубликовали работу, в которой описывалась методика автоматизированной настройки параметров c_1, c_2, ω алгоритма [127]. В 1999 году в статье Кеннеди [100] рассматривалось влияние выбранной топологии на результат работы алгоритма при решении той или иной задачи оптимизации. В том же году была впервые предложена модификация алгоритма для решения задач многокритериальной оптимизации [117]. В последующие годы были разработаны

различные методы на основе роевого алгоритма, например, [138], [136]. Также стайный алгоритм применялся для обучения нейронных сетей [82] и решения различных практических задач, например, в работах [80], [89].

Ниже приведена схема стайного алгоритма оптимизации.

Particle Swarm Optimization

инициализация популяции $\boldsymbol{P} = \{x_{i1}, \ldots, xid, \ldots, x_{iD}), i = 1, \ldots, N\}$ *случайным образом*
для *каждой частицы* x_i *из популяции* $\boldsymbol{P}$
$p_{id} = x_{id}, d = 1, \ldots, D$
конец цикла
для *каждой частицы* x_i *из популяции* $\boldsymbol{P}$
$v_{id} = 0, d = 1, \ldots, D$
конец цикла
определение $p_{gd}, d = 1, \ldots, D$
пока *не выполнится критерий остановки*
для *каждой частицы* x_i *из популяции* $\boldsymbol{P}$
если $f(x_i) < f(p_i)$
$p_{id} = x_{id}, d = 1, \ldots, D$
конец цикла
обновить значение $p_{gd}, d = 1, \ldots, D$
конец цикла

В ходе выполнения диссертационной работы было проведено исследование эффективности метода роя частиц, результаты опубликованы в статьях [6], [9] и [12]. В статье [15] представлены результаты исследования эффективности эвристики PSO для решения задач оптимизации с бинарными переменными, а также ее практическое приложение. Разработка, реализация и исследование эффективности метода роя частиц, модифицированного для решения задач условной оптимизации, были темой публикаций [7], [5] и [13]. Разработка, реализация, а также исследование эффективности распараллеленного стайного алгоритма продемонстрированы в работе [10]. Кроме того, в статьях [4], [11], [16] и [14] можно найти описание решения задачи формирования оптимального инвестиционного портфеля предприятия с помощью различных разработанных модификаций алгоритма PSO. В монографии [25] представлено подробное описание всех исследований, связанных со стайным алгоритмом, описание всех реализованных его модификаций, а также представлены все полученные

результаты (как при решении тестовых задач, так и при решении практических). Краткое описание проведенных исследований опубликовано в работе [21].

1.2. Алгоритм поиска стаей волков (Wolf Pack Search, WPS)

Алгоритм поиска стаей волков – метаэвристический алгоритм, разработанный в 2007 году, идея которого почерпнута из социального поведения стай волков [75].

Для волков типичен семейный образ жизни: они живут стаями – семейными группами, состоящими из пары вожаков, их родственников, а также пришлых одиноких волков. Внутри стаи наблюдается строго обозначенная иерархия, на вершине которой находится вожак стаи, направляющий остальных особей на поиск добычи. Волки «исследуют» местность на наличие добычи, когда кто-то из них учует запах жертвы, начинается ее поиск. Чем сильнее ощущается запах, тем ближе волки к жертве. Таким образом, они перемещаются в направлении усиления запаха добычи. Причем волки разделяются на небольшие группы, и каждая группа осуществляет поиск в каком-то определенном направлении, отличном от направлений других групп. В итоге, когда один из волков найдет жертву, он подает сигнал вожаку и остальным, чтобы поделиться добычей с волками из стаи.

Метод поиска стаей волков копирует процесс их охоты. Предположим местность, на которой волки охотятся – это поисковая область в смысле оптимизации, а частицы – это волки. Пусть изначально сгенерировано N «волков» в евклидовом пространстве размерности D, то есть каждый волк представлен в виде вектора $x_i = (x_1, \ldots, x_D)$, определяющего его координаты в пространстве. Таким образом, стая (популяция) представляет собой множество потенциальных решений, координаты которых так же, как и для стайного алгоритма оптимизации, обновляются на каждой итерации, пока не будет найдено оптимальное решение или не будет выполнено максимально заданное количество вычислений целевой функций.

Тогда функция $f(x)$, характеризующая, насколько сильно ощущается запах добычи волками, является целевой, а координаты самой добычи – оптимальной точкой. Расстояние между двумя волками p и q описывается метрикой $L(p,q)$. Выбор метрики зависит от решаемой оптимизационной задачи. Например, если решается задача безусловной оптимизации с бинарными переменными, то следует использовать расстояние Хэмминга, в случае если решается задача с вещественными переменными, применяется стандартная метрика для евклидового пространства. В силу того, что из задачи минимизации легко можно получить задачу максимизации, предположим, что необходимо найти максимум целевой функции в заданном поисковом пространстве.

«Волки» осуществляют поиск оптимальной точки – добычи – кооперируясь, то есть делятся на группы, перемещаются в различных направлениях и обмениваются информацией между собой. Сам алгоритм поиска – совместной охоты – можно охарактеризовать с помощью трех правил.

1. «Волк» с лучшим значением целевой функции на данной итерации является вожаком. Если на последующей итерации найдется другая «особь» с лучшим значением целевой функции, чем у вожака, то, соответственно, стая «обретет» нового лидера.

2. Остальные волки исследуют местность на наличие добычи, $f(x_i)$ характеризует, как сильно ощущается запах добычи i-ым «волком». Тогда величина *GBest* характеризует, как сильно ощущается запах добычи вожаком стаи.

Если $f(x_i) > GBest$, тогда i-ый «волк» находится ближе к добыче, чем вожак стаи, поэтому i-ый волк становится вожаком на данном этапе и $GBest = f(x_i)$.

Если же $f(x_i) < GBest$, тогда «волк» перемещается в пространстве с некоторым заранее заданным шагом *step*.

3. Вожак стаи «сообщит» остальным «волкам» в стае о своем местоположении, как о наименьшем на данный момент расстоянии до добычи, чтобы они переместились в его направлении. На данном этапе «вожак» рассматривается почти также, как и добыча – цель, к которой необходимо

приблизиться. Тогда «волки» стаи перемещаются в направлении вожака с заранее заданным шагом *step*, причем *d*-ая координата *i*-ого «волка» на (*k*+1)-ой итерации вычисляется по формуле:

$$x_{id}^{k+1} = x_{id}^{k} + step \cdot \frac{\left(GBest_d^k - x_{id}^k\right)}{\left\|GBest_d^k - x_{id}^k\right\|},$$

где $GBest_d^k$ – *d*-ая координата вожака, определенного за *k* предыдущих итераций, ||…|| – норма, заданная для пространства поиска и метрики $L(p,q)$.

Из формул и описания алгоритма видно, что в методе поиска стаей волков обновляются лишь координаты «волков» без учета скорости их перемещения в пространстве. И если для стайного алгоритма необходимо настраивать-выбирать четыре параметра (коэффициенты обучения, инерционный вес, размер популяции), то для метода WPS достаточно подобрать лишь два параметра – размер популяции и шаг *step*, с которым перемещаются «волки» в направлении вожака и добычи.

Схема алгоритма выглядит следующим образом:

Wolf Pack Search

*инициализация популяции **W** случайным образом и шага итерации **step***
*определение **GBest***
пока *не выполнится критерий остановки*
    для *каждого волка **wolf***
        обновить координаты волка по формуле для алгоритма
        осуществить поиск добычи
    конец цикла
    *обновить значение **GBest***
конец цикла

Алгоритм писка стаей волков сравнивался с другими эволюционными алгоритмами оптимизации в работе [128]. Кроме того, метод был использован для решения различных практических задач, описания которых можно найти, например, в работах [143] или [106].

1.3. Алгоритм светлячков (Firefly Algorithm, FFA)

В мире насчитывается около двух тысяч видов светлячков, большинство которых обладают способностью светиться, производя короткие и ритмические вспышки. Считается, что основной функцией таких вспышек является привлечение особей противоположного пола и потенциальных жертв. Кроме того, сигнальные вспышки могут служить защитным механизмом предупреждения потенциальных хищников о том, что светлячок горек на вкус.

Известны два варианта популяционных алгоритмов оптимизации, разработанных на основе копирования поведения светлячков – алгоритм светлячков (Firefly Algorithm, FFA) [140] и алгоритм оптимизации роем светлячков (Glowworm Swarm Optimization, GSO) [104]. Основное различие между ними состоит в том, что вторые являются бескрылыми.

Алгоритм светлячков был разработан в 2007 году [140] Янгом. Алгоритм использует следующую модель поведения светлячков:

1. Все светлячки могут привлекать друг друга независимо от своего пола.

2. Привлекательность светлячка для других особей пропорциональна его яркости.

3. Менее привлекательные светлячки перемещаются в направлении более привлекательного светлячка.

4. Яркость излучения данного светлячка, видимая другим светлячком, уменьшается с увеличением расстояния между светлячками.

5. Если светлячок не видит возле себя светлячка более яркого, чем он сам, то он перемещается случайным образом.

Яркость излучения светлячка f_i из популяции F, где $i = 1, \ldots, N$, принимаем равной значению функции пригодности в его текущем положении.

Привлекательность светлячка f_i для светлячка f_j полагаем равной:

$$\beta_{i,j} = \beta_0 \cdot \exp\left(-\gamma \cdot r_{i,j}^2\right),$$

где $i, j = \overline{1, N}, i \neq j$, в то время как $r_{i,j}$ – расстояние между светлячками f_i и f_j, β_0 – взаимная привлекательность светлячков при нулевом расстоянии между ними; γ – вещественная величина, имеющая смысл коэффициента поглощения света средой.

Несложно показать, что случай, когда $\gamma \to 0$, соответствует стандартному стайному алгоритму (PSO).

Для ускорения и упрощения вычислений экспоненциальную функцию в выражении для привлекательности светлячка можно заменить функцией $\frac{1}{1+r_{i,j}^2}$.

При этом указанное выражение приобретает вид:

$$\beta_{i,j} = \frac{\beta_0}{1+\gamma \cdot r_{i,j}^2},$$

где $i, j = \overline{1, N}, i \neq j$.

Приведенные формулы определяют характерное расстояние $r_c = \frac{1}{\sqrt{\gamma}}$, на котором привлекательность изменяется от β_0 до $\beta_0 \cdot e^{-1}$ для первой формулы и от β_0 до $\beta_0 / 2$ для второй. В качестве функции $\beta(r)$ могут быть использованы и другие монотонно убывающие функции.

Движение светлячка f_j, который притягивается более привлекательным светлячком f_i, определяет формула:

$$x_j = x_i + \beta(r_{i,j}) \cdot (x_j - x_i) + \alpha \cdot U_{|X|}(-1;1),$$

где $i, j = \overline{1, N}, i \neq j$, x_j, x_i – координаты j-ого и i-ого светлячков соответственно, α – свободный параметр рандомизации, $U_{|X|}(-1;1)$ – случайное число из интервала (-1;1). Для обеспечения приемлемого баланса между диверсификацией и интенсификацией поиска значение параметра рандомизации рекомендуют уменьшать с ростом номера поколения, например, по формуле:

$$\alpha = \alpha_\infty + (\alpha_0 - \alpha_\infty) \cdot e^{-1},$$

где α_0 – начальное значение параметра рандомизации, α_∞ – его окончательное значение.

Для улучшения сходимости алгоритма в формуле, описывающей перемещение менее яркого светлячка к более привлекательному, может быть использован еще один член вида:

$$\lambda \cdot U(-1;1) \cdot \left(x_j - x^{best}\right),$$

где λ – параметр, подобный параметру α, x^{best} – координаты наиболее яркого светлячка (его координаты считаются лучшими).

Ниже приведена схема алгоритма светлячков для задачи глобальной безусловной оптимизации.

Firefly Algorithm

инициализация популяции светлячков ***F*** *случайным образом;*
определение ***GBest***
пока *не выполнится критерий остановки*
  для *каждого светлячка* $\boldsymbol{F^i}$
    для *каждого светлячка* $\boldsymbol{F^j}$
      если $\boldsymbol{F^i}$ *привлекательнее* $\boldsymbol{F^j}$
        перемещаем $\boldsymbol{F^j}$ *в направлении* $\boldsymbol{F^i}$
    конец цикла
  конец цикла
  обновить значение ***GBest***
конец цикла

В данной схеме ***GBest*** – это лучшее найденное решение всеми светлячками в популяции.

В настоящее время разработано множество модификаций алгоритма светлячков для решения различных задач оптимизации, например, [85], [86], кроме того, он также применялся для решения множества практических задач, например, [74] или [93].

1.4. Алгоритм поиска кукушек (Cuckoo Search Algorithm, CSA)

Алгоритм поиска кукушек (Cuckoo Search Algorithm, CSA) – метаэвристический алгоритм, разработанный в 2009 году Янгом, имитирующий

поведение кукушек во время откладывания яиц, а именно в процессе вынужденного гнездового паразитизма [141].

Существуют такие виды кукушек, которые откладывают яйца в коллективные гнезда вместе с другими кукушками, хотя могут выбрасывать яйца конкурентов, чтобы увеличить вероятность вылупления их собственных птенцов [119]. Целый ряд видов кукушек занимается гнездовым паразитизмом, подкладывая свои яйца в гнезда других птиц как своего вида, так и, часто, других видов.

Некоторые птицы могут конфликтовать с вторжением кукушек, то есть порой такое «вторжение» встречает отпор у некоторых птиц. Например, если хозяин гнезда обнаружит в нем яйца иного вида, то он либо выбросит эти яйца, либо просто покинет данное гнездо и соорудит другое на новом месте.

В алгоритме поиска кукушек каждое яйцо в гнезде представляет собой решение, в то время как яйцо, принадлежащее кукушке, представляет собой новое решение. Цель заключается в использовании новых и потенциально лучших решений (кукушкиных), чтобы заменить менее хорошие решения в гнездах. В простейшем варианте алгоритма в каждом гнезде по одному яйцу. Именно этот вариант и рассматривается в дальнейшем.

Предположим, что дана некоторая целевая функция $f(x)$, которую необходимо максимизировать. В основу эвристики заложены следующие правила.

1. Каждая кукушка откладывает одно яйцо за один раз в случайно выбранное гнездо.

2. Лучшие гнезда с яйцами «высокого качества» (то есть с лучшими значениями целевой функции) переходят в следующее поколение.

3. Яйцо кукушки, отложенное в чужое гнездо, может быть обнаружено хозяином гнезда с некоторой вероятностью $p_a \in (0;1)$ и удалено из гнезда.

Далее представлена схема алгоритма поиска кукушек.

Cuckoo Search Algorithm

инициализация популяции хозяйских гнезд ***H*** *случайным образом*
инициализация кукушки ***cuckoo*** *случайным образом*
определение ***GBest***

пока *не выполнится критерий остановки*
выполнить перемещение кукушки с помощью полетов Леви
определить новые координаты кукушки
случайным образом выбрать некоторое гнездо $\boldsymbol{H_i}$
если $f(cuckoo) > f(H_i)$
заменяем яйцо в гнезде на яйцо кукушки
с вероятностью p_a *удаляем некоторое число худших гнезд*
строим вместо удаленных гнезд новые (столько же, сколько удалили)
обновить значение ***GBest***
конец цикла

В данной схеме использованы следующие обозначения: *GBest* – лучшее найденное решение, $f(cuckoo)$, $f(H_i)$ – значения функции пригодности для кукушки *cuckoo* и для некоторого гнезда H_i. Стоит отметить, что для решения той или иной задачи данным алгоритмом необходимо подобрать следующие параметры: размер популяции, и вероятность обнаружения яйца кукушки хозяином гнезда p_a.

Полеты Леви кукушки реализуются по формуле:

$$x_c^{'} = x_c + v \cdot L_{|X|}(\lambda),$$

где $x_c^{'}$ и x_c – новые и старые координаты кукушки соответственно, $L_{|X|}(\lambda)$ – случайная величина распределения Леви, и обычно полагают все компоненты вектора v одинаковыми и равными, то есть $v = \left(v_j = v, j \in \overline{1,|X|}\right)$, причем $|X|$ – размерность пространства. Зачастую для генерирования величин распределения Леви применяют алгоритм, описанный в работе [110]. Также величина компоненты этого вектора v должна быть связана с масштабами поисковой области.

Известно несколько модификаций алгоритма поиска кукушек. В канонической версии метода кукушка в своих полетах никак не учитывает информацию о лучших найденных решениях. В модифицированном алгоритме поиска кукушек (Modified Cuckoo Search, MCS) компоненты вектора v вычисляются по формуле вида:

$$v = U_1(0;1) \cdot \left(x^{best} - x_k\right),$$

где $x_k \neq x^{best}$ – случайно выбранное гнездо [132]. Так определенный вектор v обеспечивает большую вероятность полета кукушки к гнездам, имеющим высокую приспособленность.

Кроме того, в каноническом алгоритме вероятность обнаружения яиц кукушек p_a и параметры полета Леви являются фиксированными константами. В целях диверсификации поиска на ранних итерациях целесообразно использовать большие значения величин p_a и v. На завершающих итерациях для повышения точности локализации экстремума (интенсификации поиска) разумны меньшие значения указанных величин. Поэтому далее реализовывался улучшенный алгоритм поиска кукушек [133], использующий динамические значения этих параметров:

$$p_a(t) = p_a^{\max} - \frac{t}{T} \cdot \left(p_a^{\max} - p_a^{\min}\right),$$

$$v(t) = v^{\max} \cdot \exp\left(d^t\right),$$

$$d = \frac{1}{T} \cdot \ln\left(\frac{v^{\min}}{v^{\max}}\right).$$

Здесь $p_a^{\max}$, $p_a^{\min}$, $v^{\max}$, $v^{\min}$, T – заданные константы. Их значения были выбраны с учетом исследований, описанных в [133]. Таким образом, для реализованного алгоритма при решении той или иной задачи необходимо было подобрать размер популяции.

1.5. Алгоритм летучих мышей (Bat Algorithm, BA)

Алгоритм летучих мышей (Bat Algorithm, BA) – метаэвристический алгоритм, разработанный Янгом в 2010 году [139], и имитирующий свойство эхолокации летучих мышей [67].

Большинство видов летучих мышей обладает удивительно совершенными средствами эхолокации, которая используется ими для обнаружения добычи и препятствий, а также для обеспечения возможности разместиться в темноте на насесте. Параметры лоцирующего звукового импульса летучих мышей различных

видов меняются в широких пределах, отражая их различные охотничьи стратегии. Большинство мышей используют короткие частотно-модулированные в пределах примерно одной октавы сигналы. В то же время некоторые виды не используют частотную модуляцию своих звуковых импульсов.

В основу алгоритма летучих мышей положены следующие три правила.

1. С помощью эхолокации все мыши могут определять расстояние до добычи и препятствий, а также различать их.

2. Мыши движутся случайным образом. Текущее положение и скорость i-ой мыши b_i, где $i = 1, \ldots, N$, N – размерность популяции летучих мышей, равны $x_i = (x_1, \ldots, x_D)$ и $v_i = (v_1, \ldots, v_D)$ соответственно, D – размерность пространства поиска. Для поиска добычи мыши генерируют сигналы, имеющие частоту ω_i и громкость a_i. В процессе поиска мыши могут менять частоту этих сигналов, а также частоту повторения излучаемых импульсов $r \in [0;1]$;

3. Частота сигналов может изменяться в диапазоне $[\omega^{\min}, \omega^{\max}], \omega^{\max} > \omega^{\min} \geq 0$, а громкость сигналов в пределах [0;1].

Предположим, что необходимо решить задачу глобальной безусловной минимизации. Тогда схема алгоритма летучих мышей выглядит следующим образом:

Bat Algorithm

инициализация популяции ***B*** *и соответствующих параметров случайным образом*
определение ***GBest***
пока *не выполнится критерий остановки*
    обновить координаты летучих мышей (осуществить миграцию)
    для *каждой летучей мыши* $\mathbf{b_i}$ *сгенерировать случайное число* ***rand*** *в пределах* [*0;1*]
        если $\mathbf{rand} > \mathbf{r_i}$
            находим ее лучшее решение, и в его окрестности реализуем локальный поиск
            найденное решение принимаем за текущее положение мыши
            сгенерировать в окрестности $\mathbf{b_i}$ *случайным образом новое решение*
            сгенерировать случайное число ***Rand*** *в пределах* [*0;1*]
            если $\mathbf{Rand} < \mathbf{a_i}$ *и* $\mathbf{f(b_i)} < \mathbf{f(GBest)}$
                принимаем это решение в качестве нового положения мыши
                уменьшаем громкость сигнала
                увеличиваем частоту повторения сигналов
    конец цикла
    обновить значение ***GBest***
конец цикла

В данной схеме *GBest* – это лучшее найденное решение всеми летучими мышами в популяции.

На этапе инициализации алгоритма начальные значения частот ω_i^0, громкостей a_i^0 и частот повторения импульсов r_i^0, где i = 1, …, N, полагаются равномерно распределенными в соответствующих интервалах $[\omega^{\min}; \omega^{\max}]$, $[a^{\min}; a^{\max}]$, $[0;1]$.

Миграция летучей мыши b_i , i = 1, …, N, осуществляется по формулам:

$$x_i^{'} = x_i + v_i^{'},$$

$$v_i^{'} = v_i + \omega_i \cdot (x_i - x^{best}),$$

$$\omega_i = \omega^{\min} + (\omega^{\max} - \omega^{\min}) \cdot U_1(0;1),$$

где $x_i^{'}$, x_i – новые и старые координаты i-ой летучей мыши, $v_i^{'}$, v_i – старое и новое значение скорости i-ой летучей мыши, $U_1(0;1)$ – случайное число из интервала $(0;1)$.

Другими словами, на каждой итерации летучая мышь перемещается в направлении, определяемом суммой вектора перемещения на предыдущей итерации и случайным образом возмущенного вектора направления на лучшую мышь $(x_i - x^{best})$. Рассмотренная процедура миграции алгоритма летучих мышей имеет много общего с аналогичной процедурой стайного алгоритма оптимизации.

Случайный локальный поиск выполняется по следующей схеме:

1. Случайным образом варьируется текущее положение i-ой летучей мыши b_i в соответствии с формулой:

$$x_i^{'} = x_i + \overline{a} \cdot U_{|X|}(-1;1),$$

где i = 1, …, N, $x_i^{'}$, x_i – новые и старые координаты i-ой летучей мыши $\overline{a}$ – текущее среднее значение громкостей всех летучих мышей в популяции, такое что

$$\overline{a} = \frac{1}{N} \cdot \sum_{i=1}^{N} a_i ;$$

$|X|$ – размерность поисковой области, $U_{|X|}(-1;1)$ – величина, равномерно распределенная на интервале от -1 до 1;

2. Вычисляется значение целевой функции в новой точке: $f(x_i^{'}) = f_i^{'}$. Если оно лучше предыдущего значения, то есть $f_i^{'} < f_i$, то процедура локального поиска завершается, в противном случае фиксированное число раз осуществляется возврат к шагу 1.

Эволюция параметров a_i и r_i осуществляется по правилам:

$$a_i^{t+1} = b_a \cdot a_i^t,$$

$$r_i^{t+1} = r_i^0 \cdot (1 - \exp(-b_r \cdot t)),$$

где i = 1, …, N, a_i^{t+1}, a_i^t – громкости i-ой мыши на (t+1)-ой и t-ой итерациях соответственно, r_i^0 – частота повторения импульсов i-ой мышью при инициализации, r_i^{t+1} – частота повторения импульсов i-ой мышью на (t+1)-ой итерации, $b_a \in (0;1)$, b_r > 0 – заданные константы (свободные параметры алгоритма), рекомендуемые значении которых равны 0.9, t – номер поколения (итерации). Другими словами, с ростом числа итераций громкость импульсов, излучаемых каждой мышью, линейно уменьшается (добыча уже найдена), а частоту повторения импульсов r_i в тех же условиях уменьшают по экспоненциальному закону, так что имеют место предельные соотношения:

$$a_i^t \xrightarrow[t\to\infty]{} 0,\ r_i^t \xrightarrow[t\to\infty]{} r_i^0,\ i = 1, \ldots, N.$$

1.6. Исследование эффективности алгоритмов стайного типа

Изначально указанные алгоритмы были исследованы на шести тестовых задачах однокритериальной безусловной оптимизации с вещественными переменными (сфера, гипер-эллипсоид, функция Растригина, функция Розенброка, функция Акли и функция Гриванка), взятыми из [116]:

1. Сфера:

$$F(x,n)=\sum_{i=1}^{n}x_i^2$$

где n – размерность задачи. Минимум достигается в точке $\boldsymbol{x_i} = 0$, $i=\overline{1,n}$, $F(\boldsymbol{x},n) = 0$.

2. Гипер-эллипсоид:

$$F(x,n)=\sum_{i=1}^{n}\left(\sum_{j=1}^{n}x_j\right)^2,$$

где n – размерность задачи. Минимум достигается в точке $\boldsymbol{x_i} = 0$, $i=\overline{1,n}$, $F(\boldsymbol{x},n) = 0$.

3. Функция Растригина:

$$F(x,n)=200+\sum_{i=1}^{n}\left[x_i^2-10\cdot\cos(2\cdot\pi\cdot x_i)\right],$$

где n – размерность задачи. Минимум достигается в точке $\boldsymbol{x_i} = 0$, $i=\overline{1,n}$, $F(\boldsymbol{x},n) = 200 – 10\cdot n$.

4. Функция Розенброка:

$$F(x,n)=\sum_{i=1}^{n-1}\left[100\cdot\left[x_{i+1}-x_i^2\right]^2+(1-x_i)^2\right],$$

где n – размерность задачи. Минимум достигается в точке $\boldsymbol{x_i} = 1$, $i=\overline{1,n}$, $F(\boldsymbol{x},n) = 0$.

5. Функция Акли:

$$F(x,n)=20+\exp-20\cdot\exp\left(-0.2\cdot\sqrt{\frac{1}{n}\cdot\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\cdot\sum_{i=1}^{n}\cos(2\cdot\pi\cdot x_i)\right),$$

где n – размерность задачи. Минимум достигается в точке $\boldsymbol{x_i} = 0$, $i=\overline{1,n}$, $F(\boldsymbol{x},n) = 0$.

6. Функция Гриванка:

$$F(x,n)=\frac{-10}{0.005\cdot\sum_{i=1}^{n}x_i^2-\prod_{j=1}^{n}\cos\left(\frac{x_j}{\sqrt{j}}\right)+2}+10,$$

где n – размерность задачи. Минимум достигается в точке $\boldsymbol{x_i} = 0$, $i=\overline{1,n}$, $F(\boldsymbol{x},n) = 0$.

Результаты оценивались по следующим критериям: минимальное число вычислений целевой функции, требуемое для получения решения с заданной точностью, то есть произведение размера популяции и количества итераций-поколений, и минимальная ошибка, получаемая алгоритмом после выполнения заданного числа вычислений целевой функции.

Ниже в Таблице 1.1 и Таблице 1.2 приведены результаты сравнения описанных ранее бионических методов (средний размер популяции для алгоритма, среднее число итераций работы алгоритма, а также его надежность, то есть доля успешных прогонов, когда ошибка была меньше заданной заранее погрешности).

Таблица 1.1. Результаты сравнения алгоритмов

№ задачи	Кол-во пер-ых	PSO			WPS			FFA		
		Кол-во частиц	Кол-во поколений	Надежность	Кол-во частиц	Кол-во поколений	Надежность	Кол-во частиц	Кол-во поколений	Надежность
1	2	20	50	100	20	30	100	20	35	100
	3	30	60	100	**30**	**30**	**100**	**30**	**30**	**100**
	4	40	130	100	**30**	**40**	**100**	**30**	**40**	**100**
2	2	30	60	100	30	15	89	20	45	100
	3	70	70	100	**20**	**20**	**85**	30	60	100
	4	250	150	100	25	35	85	50	100	100
3	2	20	50	100	25	30	85	**20**	**30**	**100**
	3	50	75	96	**20**	**40**	**85**	30	70	91
	4	350	180	100	80	70	85	50	100	90
4	2	20	50	100	20	30	100	20	40	100
	3	50	70	95	20	30	85	**20**	**30**	**100**
	4	90	120	95	40	80	85	40	50	97
5	2	100	150	100	25	50	85	**20**	**35**	**98**
	3	500	380	95	100	165	86	**40**	**100**	**92**
	4	1600	220	95	200	510	86	200	250	84
6	2	60	100	90	30	45	90	40	50	96
	3	500	250	83	80	45	87	150	500	87
	4	3200	60	83	**90**	**50**	**87**	50	800	85

Таблица 1.2. Результаты сравнения алгоритмов

№ задачи	Кол-во пер-ых	BA			CSA		
		Кол-во частиц	Кол-во поколений	Надежность	Кол-во частиц	Кол-во поколений	Надежность
1	2	20	25	100	**25**	**15**	**100**
	3	30	35	100	20	50	100
	4	30	60	86	30	60	100
2	2	25	45	100	**20**	**20**	**100**
	3	40	70	95	25	20	100
	4	50	70	86	**10**	**50**	**90**

Продолжение Таблицы 1.2. Результаты сравнения алгоритмов

№ задачи	Кол-во пер-ых	BA			CSA		
		Кол-во частиц	Кол-во поколений	Надежность	Кол-во частиц	Кол-во поколений	Надежность
3	2	35	30	86	20	50	85
	3	75	50	86	40	55	85
	4	75	100	86	**40**	**40**	**85**
4	2	25	30	100	**20**	**20**	**100**
	3	25	40	100	30	60	100
	4	90	25	85	**20**	**45**	**100**
5	2	25	35	86	20	50	85
	3	500	40	90	40	100	85
	4	1700	20	85	**90**	**200**	**85**
6	2	**20**	**30**	**85**	50	15	95
	3	1600	20	88	**20**	**150**	**85**
	4	1600	20	85	60	100	86

В Таблицах 1.1 и 1.2 выделены лучшие комбинации размера популяции (числа индивидов) и количества итераций (поколений) для каждой задачи определенной размерности. Стоит отметить, что результат считался «лучшим», если надежность алгоритма была не меньше 85% (то есть 85 раз из 100 алгоритм находил оптимальное решение задачи с заданной погрешностью), и произведение числа итераций на число индивидов было минимальным.

В Таблице 1.3 и Таблице 1.4 представлены численные результаты для алгоритмов по критерию «минимальная ошибка». В таблицах использованы следующие обозначения: «Av_f» – значение ошибки, усредненное по числу успешных прогонов, «СКО» – среднеквадратическая ошибка. Прогон считался успешным, если оптимальное решение было найдено с заданной погрешностью.

Таблица 1.3. Результаты сравнения алгоритмов по критерию «минимальная ошибка»

№ задачи	Кол-во пер-ых	PSO		WPS		FFA	
		Av_f	СКО	Av_f	СКО	Av_f	СКО
1	2	0.000502201	0.000312496	0.000482788	0.000279011	0.000392907	0.000287461
	3	0.00062156	0.00025101053	0.000589053	0.000239496	0.000718224	0.000314287
	4	0.000663717	0.000246153	0.000690592	0.000220931	0.000637881	0.000263904

Продолжение Таблицы 1.3. Результаты сравнения алгоритмов по критерию «минимальная ошибка»

№ задачи	Кол-во пер-ых	PSO		WPS		FFA	
		Av_f	СКО	Av_f	СКО	Av_f	СКО
2	2	**0.000394192**	**0.00028577**	0.000526816	0.00030235	0.000584234	0.000295428
	3	0.000659686	0.000259448	0.000578738	0.000251964	0.000604299	0.000271862
	4	0.000655545	0.000233039	0.000624773	0.000232725	0.000634423	0.000243748
3	2	**0.000158651**	**0.000155061**	0.000391796	0.000230841	0.000773863	0.000146881
	3	0.000792495	0.000213476	0.000765909	0.000187156	0.00070231	0.000208729
	4	0.000822469	0.000152243	0.000808329	0.0001827	0.000574752	0.00023238
4	2	0.000468071	0.00028 4566	0.000501774	0.000286411	0.000605792	0.000294682
	3	0.000611749	0.000279977	**0.000529674**	**0.000246324**	0.000703628	0.000222479
	4	0.000703351	0.000241868	0.000739186	0.000216364	0.000639509	0.000220904
5	2	0.000505696	0.000296204	0.000580998	0.000295794	0.000625228	0.000256229
	3	0.000642948	0.000260757	0.000885613	0.000213842	**0.000471802**	**0.000229307**
	4	0.000969843	7.52751e-005	0.000977969	5.70546e-005	0.000666193	0.000296261
6	2	180.001	0.0002921	**180.001**	**0.0002907**	180.001	0.0003108
	3	170.001	0.0002687	170.001	0.0002706	**170.001**	**0.0001816**
	4	160.001	0.000213223	160.001	0.00024655	**160.001**	**0.000152797**

Таблица 1.4. Результаты сравнения алгоритмов по критерию «минимальная ошибка»

№ задачи	Кол-во пер-ых	BA		CSA	
		Av_f	СКО	Av_f	СКО
1	2	0.000776857	0.000478695	**0.000353949**	**0.000264723**
	3	**0.000137851**	**0.000115663**	0.000558841	0.000288605
	4	**0.000181049**	**0.000170724**	0.000652845	0.000270081
2	2	0.000805076	0.000436237	0.000491937	0.000310012
	3	**0.000192938**	**0.000203949**	0.000526477	0.000276469
	4	**0.000129756**	**0.000778811**	0.000638244	0.000265691
3	2	0.000163805	0.00145196	0.000162097	0.000165121
	3	0.000338486	0.000240149	**0.000274025**	**0.000191477**
	4	0.000396804	0.000222263	**0.000396317**	**0.000107665**
4	2	0.000714097	0.000355724	**0.000392817**	**0.000316075**
	3	0.000699483	0.000282309	0.000543911	0.000295872
	4	**0.000109123**	**0.000807283**	0.000650373	0.000248038
5	2	**0.000222734**	**0.000174269**	0.000536647	0.000302301
	3	0.000604966	0.000268792	0.000740171	0.000584139
	4	**0.00018217**	**0.00010808**	0.000553122	0.000321456
6	2	180.001	0.000350937	180.0001	0.000317671
	3	170.054	0.195605	170.001	0.000301185
	4	160.005	0.0165591	160.001	0.000283197

В Таблицах 1.3 и 1.4 выделены лучшие результаты по критерию «минимальная ошибка» для всех задач. Если несколькими алгоритмами было получено одинаковое значение ошибки, то лучшим считался тот результат, для которого было меньше значение «СКО».

Далее полученные результаты были использованы для проведения следующих численных экспериментов. Для каждой задачи с заданным количеством переменных определялся лучший алгоритм по критерию «минимальное число вычислений целевой функции» (комбинация размера популяции и числа итераций алгоритма). Затем это число вычислений целевой функции (размер популяции и количество итераций) использовалось для нахождения оптимального решения с заданной погрешностью для остальных алгоритмов при решении этой задачи.

Ниже в Таблице 1.5 и Таблице 1.6 приведены получившиеся результаты. В данных таблицах использованы следующие обозначения: «Av_ev» – число вычислений целевой функции, усредненное по числу успешных прогонов, «СКО» – среднеквадратичное отклонение, «Надежность» – число успешных прогонов. Прогон считался успешным, если оптимальное решение было найдено с заданной погрешностью, причем вычисления прекращались, если это решение было найдено.

Таблица 1.5. Результаты сравнения составляющих алгоритмов по критерию «число вычислений целевой функции» при минимальных установках

№ задачи	Кол-во пер-ых	PSO			FFA			WPS		
		Av_ev	СКО	Надежность	Av_ev	СКО	Надежность	Av_ev	СКО	Надежность
1	2	331	84	68	161	157	73	202	92	97
	3	793	146	71	**425**	**190**	**100**	**777**	**201**	**100**
	4	775	76	46	**613**	**199**	**100**	**685**	**263**	**100**
2	2	350	98	52	283	171	56	234	96	93
	3	385	43	31	384	96	23	**285**	**73**	**85**
	4	494	39	26	489	37	33	378	120	73
3	2	585	40	65	**560**	**93**	**100**	500	129	61
	3	795	28	53	702	183	35	**779**	**92**	**85**
	4	1600	0	32	1396	408	28	1585	44	29
4	2	302	96	73	269	137	63	198	108	93
	3	560	71	39	**318**	**114**	**100**	420	148	85
	4	878	61	21	543	282	59	702	173	85

Продолжение Таблицы 1.5. Результаты сравнения составляющих алгоритмов по критерию «число вычислений целевой функции» при минимальных установках

№ задачи	Кол-во пер-ых	PSO			FFA			WPS		
		Av_ev	СКО	Надежность	Av_ev	СКО	Надежность	Av_ev	СКО	Надежность
5	2	658	106	20	**569**	**154**	**98**	694	88	47
	3	3904	391	7	**3828**	**516**	**92**	3924	520	12
	4	17728	1525	5	17090	4995	12	17819	1896	5
6	2	598	17	6	474	190	26	553	119	57
	3	2857	313	31	2804	588	13	2297	788	59
	4	4490	74	8	4461	524	9	**4330**	**546**	**87**

Таблица 1.6. Результаты сравнения составляющих алгоритмов по критерию «число вычислений целевой функции»

№ задачи	Кол-во пер-ых	BA			CSA		
		Av_ev	СКО	Надежность	Av_ev	СКО	Надежность
1	2	281	91	74	**278**	**129**	**100**
	3	744	170	85	467	387	68
	4	739	53	48	734	509	68
2	2	344	106	69	**325**	**101**	**100**
	3	393	41	15	304	98	100
	4	498	19	5	**398**	**90**	**90**
3	2	544	118	45	462	212	53
	3	799	8	16	787	18	31
	4	1582	306	7	**1586**	**127**	**85**
4	2	255	132	95	**107**	**45**	**100**
	3	552	116	61	390	217	55
	4	839	57	10	**774**	**200**	**100**
5	2	600	184	54	377	247	66
	3	3841	279	7	3681	596	85
	4	17898	2053	3	**17691**	**1004**	**85**
6	2	**365**	**161**	**85**	573	95	28
	3	2998	250	2	**2169**	**763**	**85**
	4	4500	0	0	4265	624	60

Полученные результаты исследований были использованы для определения лучшего алгоритма для решения задач однокритериальной безусловной оптимизации с вещественными переменными. В Таблице 1.7 указаны те алгоритмы, которыми получены наилучшие значения по упомянутым ранее двум критериям для каждой функции.

Таблица 1.7. Результаты сравнения составляющих алгоритмов: обобщение

	Количество переменных	***Минимальное число вычислений***	***Мин. ошибка***
Сфера	2	CSA	CSA
	3	FFA	BA
	4	FFA	BA

Продолжение Таблицы 1.7. Результаты сравнения составляющих алгоритмов: обобщение

	Количество переменных	***Минимальное число вычислений***	***Мин. ошибка***
Функция Гриванка	2	CSA	PSO
	3	WPS	BA
	4	CSA	BA
Функция Акли	2	FFA	PSO
	3	WPS	CSA
	4	CSA	CSA
Гипер-эллипсоид	2	CSA	CSA
	3	FFA	WPS
	4	CSA	BA
Функция Розенброка	2	FFA	BA
	3	FFA	FFA
	4	CSA	BA
Функция Растригина	2	BA	WPS
	3	CSA	FFA
	4	WPS	FFA

В итоге было установлено, что нельзя с уверенностью утверждать, что тот или иной алгоритм предпочтительнее, так как лучшие результаты достигались для разных функций различными алгоритмами. Более того, исследования показали, что лучший результат даже для одной и той же задачи при смене количества переменных получается не одним и тем же алгоритмом. Кроме того, для всех алгоритмов при решении задач требовалось подобрать такое число вычислений целевой функции, чтобы было найдено оптимальное решение (с некоторой погрешностью) и чтобы оно было минимальным.

Далее определялся лучший алгоритм для задач однокритериальной условной оптимизации с вещественными переменными.

Указанные бионические алгоритмы (метод роя частиц, алгоритм поиска стаей волков и т.д.) были модифицированы для решения задач условной оптимизации. Учет ограничений для задач проводился с помощью метода динамических штрафов [83]. Исследования проводились на шести тестовых задачах однокритериальной условной оптимизации с вещественными переменными, взятых из [31]:

1. $z = (x-2)^2 + (y-3)^2 \to \max$

$x^2 + y^2 \le 52$

Точное решение: $x = -4, y = -6, z^* = 117$.

2. $z = 20 + e - 20\exp\left(-0.2\sqrt{\sum_{i=1}^{N}\frac{x_i^2}{N}}\right) - \exp\left(\sum_{i=1}^{N}\frac{\cos(2\pi \cdot x_i)}{N}\right) \to \min$

$N = 4$

$$\begin{cases} 2x_1 - 3x_2 + 4x_3 \le 10 \\ 4x_2 - 5x_3 + x_4 \le 1 \\ 10x_1 + 7.5x_3 - 8.4x_4 \le 3.5 \\ -3.1x_1 + 21.7x_2 - 36.4x_4 \le 16.2 \end{cases}$$

Точное решение: $x_i = 0, i = \overline{1,N}, z^* = 0$.

3. $z = -10 \cdot x - 5 \cdot y \to \min$

$$\begin{cases} x \ge 0 \\ y \ge -15 \\ y \le \frac{x^2}{2} \\ y \ge 2 \cdot x^2 - 20 \end{cases}$$

Точное решение: $x = 3.651483717$, $y = 6.66666667$, $z^* = -69.84817052$.

4. $u = x^3 + y^2 + z \to \max$

$$\begin{cases} x \ge 0 \\ y \ge 0 \\ z \ge 0 \\ x^2 + y^2 + z^2 \le 25 \end{cases}$$

Точное решение: $x = 5, y = 0, z = 0, u^* = 125$.

5. $z = 10x - 5y \to \max$

$$\begin{cases} y - 15 \le 0 \\ y + 2x^2 - 20 \le 0 \\ -\frac{x^2}{2} - y \le 0 \end{cases}$$

Точное решение: $x = 3.6514837, \mathrm{y} = -6.666667, \mathrm{z}^* = 69.8481705$.

6. $z = 5 \cdot x + 0.5 \cdot y \to \max$

$$\begin{cases} y \le -2 \cdot x + 5 \\ y \ge x - 1.5 \\ y \le 2 \cdot x + 1 \\ x \ge 0 \\ y \ge 0 \end{cases}$$

Точное решение: $x = \frac{13}{6} = 2.16666, y = \frac{2}{3} = 0.66666, z^* = \frac{67}{6} = 11.16667$.

Для упрощения процесса тестирования все задачи максимизации были преобразованы в задачи минимизации. Результаты оценивались так же, как и для задач безусловной оптимизации, по двум критериям: минимальное число вычислений целевой функции, требуемое для получения оптимального решения с заданной точностью, то есть размер популяции и количество итераций-поколений, и минимальная ошибка, получаемая алгоритмом после выполнения заданного числа вычислений целевой функции.

Ниже в Таблице 1.8 и Таблице 1.9 приведены результаты сравнения описанных ранее бионических методов: средний размер популяции (количество частиц) для алгоритма, среднее число итераций работы алгоритма (количество поколений), а также «надежность» алгоритма (доля успешных прогонов). Прогон считался успешным, если полученная в итоге ошибка была меньше заданной заранее погрешности.

Таблица 1.8. Результаты сравнения составляющих алгоритмов для задач условной оптимизации

№ задачи	PSO			WPS			FFA		
	Кол-во частиц	Кол-во поколений	Надежность	Кол-во частиц	Кол-во поколений	Надежность	Кол-во частиц	Кол-во поколений	Надежность
1	200	300	86	**100**	**40**	**85**	5	1800	95
2	100	100	96	50	50	88	10	1000	85
3	250	100	92	15	800	89	5	3000	85
4	200	200	94	**50**	**70**	**86**	5	1000	86
5	250	100	92	10	600	91	5	3000	85
6	150	100	98	50	45	95	10	1000	86

Таблица 1.9. Результаты сравнения составляющих алгоритмов для задач условной оптимизации

№ задачи	BA			CSA		
	Кол-во частиц	Кол-во поколений	Надежность	Кол-во частиц	Кол-во поколений	Надежность
1	**100**	**40**	**85**	100	250	86
2	**25**	**30**	**100**	50	40	100
3	200	65	86	**10**	**500**	**93**
4	**100**	**35**	**87**	100	200	85
5	100	100	86	**10**	**500**	**93**
6	**50**	**30**	**86**	40	100	86

В Таблицах 1.8 и 1.9 выделены лучшие комбинации размера популяции (числа индивидов) и количества итераций (поколений) для каждой задачи определенной размерности. Результат считался «лучшим», если надежность алгоритма была не меньше 85% (то есть 85 раз из 100 алгоритм находил оптимальное решение задачи с заданной погрешностью), и произведение числа итераций на число индивидов было минимальным.

В Таблице 1.10 и Таблице 1.11 представлены численные результаты для алгоритмов по критерию «минимальная ошибка». В данных Таблицах использованы те же обозначения, что и в Таблицах 1.3 и 1.4 (усредненное по количеству прогонов значение целевой функции и среднеквадратичное отклонение).

Таблица 1.10. Численные результаты сравнения составляющих алгоритмов для задач условной оптимизации

№ задачи	PSO		WPS		FFA	
	Av_f	СКО	Av_f	СКО	Av_f	СКО
1	**-116.999**	**0.000241136**	-116.999	0.000912559	-116.999	0.000317541
2	0.650223	1.04236	0.000768027	0.000192099	0.0752077	0.01818
3	-69.8474	0.000244777	-69.8473	0.000304947	-69.7817	0.0124845
4	**-125.004**	**0.0105656**	-124.979	0.00028859	-124.954	0.0483663
5	-69.8474	0.000208103	**-69.8477**	**0.000168608**	-69.7988	0.0140527
6	-11.1656	0.000347022	-11.1659	0.000239372	**-11.1659**	**0.000224535**

Таблица 1.11. Численные результаты сравнения составляющих алгоритмов для задач условной оптимизации

№ задачи	BA		CSA	
	Av_f	СКО	Av_f	СКО
1	-116.559	0.266042	-116.985	0.00977938
2	**0.000444089**	**0.000440098**	0.000592615	0.000232818
3	-69.7716	0.0266504	**-69.8476**	**0.000269929**
4	-121.752	17.0345	-125.0086	0.00935964
5	-69.7827	0.011322	-69.8477	0.000252173
6	-10.1573	0.901586	-11.1358	0.0253912

Далее опять для каждой задачи с заданным количеством переменных определялся лучший алгоритм по критерию «минимальное число вычислений целевой функции» (комбинация размера популяции и числа итераций алгоритма). Затем это число вычислений целевой функции (размер популяции и количество итераций) использовалось для нахождения оптимального решения с заданной погрешностью для остальных алгоритмов при решении этой задачи.

Ниже в Таблице 1.12 и Таблице 1.13 приведены получившиеся результаты. В данных Таблицах использованы те же обозначения, что и в Таблицах 1.5 и 1.6.

Таблица 1.12. Результаты сравнения составляющих алгоритмов для задач условной оптимизации при минимальных установках

№ задачи	PSO			WPS			FFA		
	Av_ev	СКО	Надежность	Av_ev	СКО	Надежность	Av_ev	СКО	Надежность
1	3875	472	40	**3502**	**990**	**85**	3236	1541	20
2	744	18	7	740	19	23	744	28	15
3	2798	13	6	4988	78	81	4877	519	38
4	3123	474	43	1999	814	86	3062	483	38
5	2798	24	8	4956	174	83	4938	381	38
6	1496	30	10	1474	100	66	3921	404	24

Таблица 1.13. Результаты сравнения составляющих алгоритмов для задач условной оптимизации при минимальных установках

№ задачи	BA			CSA		
	Av_ev	СКО	Надежность	Av_ev	СКО	Надежность
1	3765	338	85	3434	1161	23
2	**588**	**284**	**100**	612	105	90
3	2556	162	25	**4020**	**710**	**93**
4	**3498**	**15**	**87**	2084	156	25
5	2611	178	25	**4024**	**700**	**93**
6	**1358**	**365**	**86**	1387	246	41

Полученные результаты исследований были использованы для определения лучшего алгоритма для решения задач однокритериальной условной оптимизации с вещественными переменными. В Таблице 1.14 указаны те алгоритмы, которыми получены наилучшие значения по упомянутым ранее двум критериям для каждой задачи условной оптимизации.

Таблица 1.14. Результаты сравнения составляющих алгоритмов для задач условной оптимизации: обобщение

№ задачи	*Количество переменных*	*Минимальное число вычислений*	*Мин. ошибка*
1	2	WPS	PSO
2	4	BA	BA
3	2	CSA	CSA
4	3	BA	PSO
5	2	CSA	WPS
6	2	BA	FFA

По результатам, представленным в Таблице 1.14, можно сделать выводы, что для задач условной оптимизации алгоритм летучих мышей демонстрирует лучшие результаты, как по сравнению с другими условными модификациями, так и по сравнению с результатами, полученными для задач безусловной оптимизации. Однако, для задач условной оптимизации так же, как и для задач безусловной оптимизации, лучший алгоритм по разным критериям менялся, поэтому невозможно с уверенностью утверждать, что стоит применять для решения следующей задачи тот или иной бионический метод оптимизации.

На последнем этапе сравнения алгоритмов определялась лучшая эвристика стайного типа для решения задач однокритериальной оптимизации с бинарными переменными. Упомянутые ранее алгоритмы были модифицированы для решения задач с бинарными переменными по методике, описанной в работе Кеннеди и Эберхарта [102]. Для тестирования использовались следующие функции: сфера, гипер-эллипсоид, функция Растригина, функция Розенброка, функция Акли, функция Гриванка. Размерность задач менялась от 2 до 4. Работа алгоритма оценивалась по двум критериям: минимальное число вычислений целевой функции и минимальная ошибка. В Таблице 1.15 указаны те алгоритмы,

которыми получены наилучшие значения по упомянутым критериям для каждой функции.

Таблица 1.15. Результаты сравнения составляющих алгоритмов для задач безусловной оптимизации с бинарными переменными: обобщение

	Количество переменных	***Минимальное число вычислений***	***Мин. ошибка***
Сфера	2	BA	WPS
	3	WPS	FFA
	4	WPS	CSA
Функция Гриванка	2	CSA	WPS
	3	CSA	WPS
	4	CSA	PSO
Функция Акли	2	CSA	PSO
	3	WPS	PSO
	4	WPS	CSA
Гипер-эллипсоид	2	CSA	BA
	3	BA	FFA
	4	WPS	CSA
Функция Розенброка	2	BA	FFA
	3	FFA	FFA
	4	CSA	PSO
Функция Растригина	2	FFA	WPS
	3	BA	CSA
	4	WPS	CSA

Таким образом, было установлено, что для задач однокритериальной оптимизации с бинарными переменными невозможно заранее определить, какой алгоритм будет лучшим и следует применить для осуществления поиска оптимального решения некоторой заданной целевой функции.

Полученные выводы для задач однокритериальной условной и безусловной оптимизации, как с вещественными, так и с бинарными переменными послужили предпосылками для разработки нового оптимизационного метода и его модификаций для решения подобных задач, за основу которого были взяты пять описанных выше бионических алгоритмов.

Выводы

Стайный алгоритм оптимизации, разработка и исследование эффективности которого подробного описаны в монографии [25], а также еще четыре алгоритма роевого интеллекта, а именно алгоритм светлячков, метод поиска стаей волков, алгоритм поиска кукушек и алгоритм летучих мышей, наиболее похожих на эвристику PSO, были протестированы на множестве задач оптимизации с целью определения наиболее эффективного. Тестирование проводилось на различных задачах, а именно на задачах однокритериальной условной и безусловной оптимизации с вещественными или бинарными переменными.

Эффективность алгоритмов оценивалась по двум критериям: минимальная ошибка, а также минимальное число вычислений целевой функции для достижения оптимального решения с заранее заданной погрешностью. Размерность задач варьировалась. В итоге было установлено, что все перечисленные алгоритмы обладают как достоинствами, так и недостатками, хорошо решают задачи оптимизации, однако определить лучший алгоритм, и уж тем более, выбрать заранее, какую эвристику целесообразнее применить для той или иной задачи (независимо от того были ли ограничения на поисковую область и вида переменных), невозможно. Кроме того, необходимость довольно точной настройки параметров описанных в главе алгоритмов, от которых существенно зависит эффективность оптимизации, также является проблемой их применения.

В этой связи разработка универсального самонастраивающегося бионического метода оптимизации является актуальной научно-технической задачей.

2 Коллективный метод оптимизации на основе бионических алгоритмов

2.1. Коллективный метод оптимизации с вещественными переменными

Выводы, полученные в предыдущем разделе, послужили предпосылками для разработки нового оптимизационного метода, за основу которого были взяты пять описанных выше алгоритмов [62]. Главная идея заключается в генерировании пяти популяции (одной для каждой метаэвристики), которые бы в дальнейшем коллективно решали задачу оптимизации на основе конкуренции и кооперации.

Главный параметр, требующий настройки для всех алгоритмов, – это размер популяции или количество индивидов/частиц. Сама по себе задача выбора числа индивидов достаточно сложная, так как нужно определить такое их число, чтобы за определенное количество вычислений целевой функции было достигнуто оптимальное решение, с одной стороны, и чтобы этих вычислений было как можно меньше, с другой. Кроме того, от размера популяции зависит и количество итераций/поколений, необходимое алгоритму для достижения оптимального решения с заданной точностью. В совокупности указанные параметры определяют число вычислений целевой функции в ходе работы алгоритма, от которого зависит время выполнения одного программного запуска (прогона).

Поэтому было принято решение автоматизировать процесс настройки размера популяции таким образом, чтобы количество индивидов для каждой популяции определялось в процессе работы алгоритма. А именно размер каждой популяций может, как увеличиваться, так и уменьшаться в зависимости от того, как меняется значение функции пригодности. Другими словами, если на t-ой итерации средняя пригодность индивидов k-ой (k = 1, 2, 3, 4, 5) популяции лучше средней пригодности других популяций, то k-ая популяция считается «победителем», а все остальные «проигравшими». Из «проигравших» популяций удаляются индивиды – они добавляются к «победившей» популяции. Например,

на графике рисунка 2.1 ниже показано, как меняются размерности популяций при нахождении минимума для функции Швефеля с вращением.

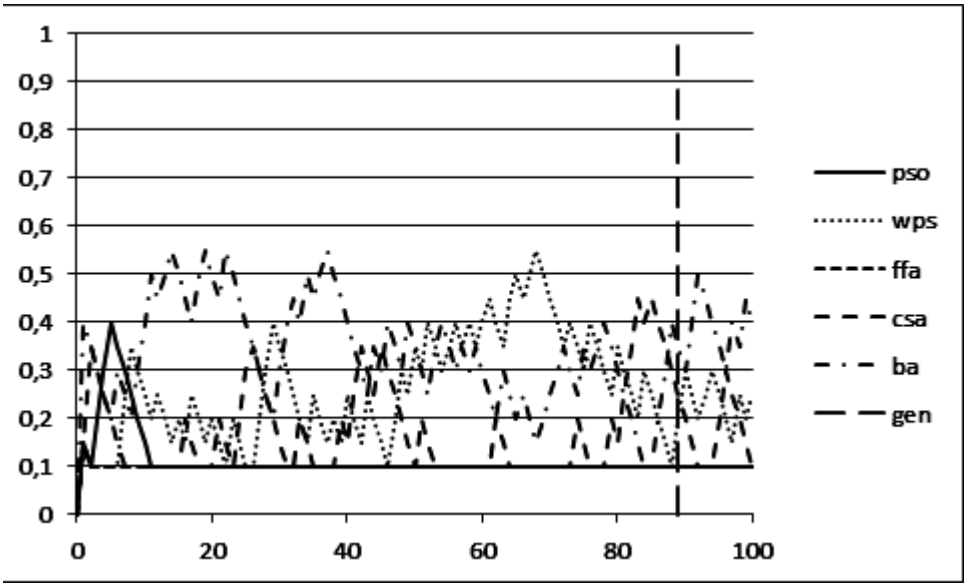

Рисунок 2.1. График изменения размера популяций для функции Швефеля с вращением

А на рисунке 2.2 показано, как менялись размерности популяций при нахождении минимума для функции Растригина. Вертикальной линией в обоих случаях отмечено поколение, на котором оптимальное решение было достигнуто с заданной точностью. Функции взяты из [107].

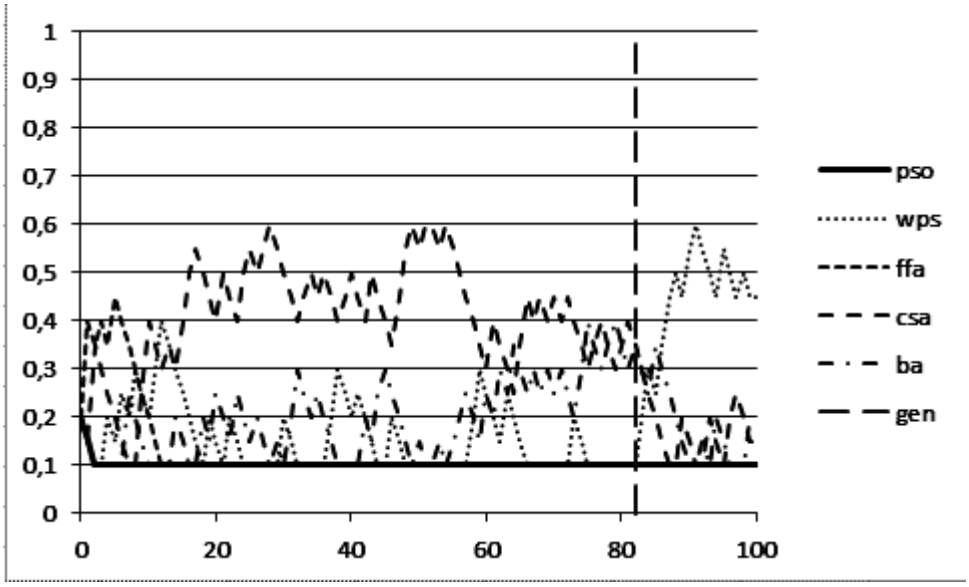

Рисунок 2.2. График изменения размера популяций для функции Растригина

Таким образом, определяется лучший алгоритм для задачи на каждой итерации.

С другой стороны, общее число всех индивидов может тоже либо увеличиваться, либо уменьшаться. Если на протяжении некоторого заданного числа поколений значение функции пригодности не улучшается, то увеличивается размер всех популяций. И наоборот, если на протяжении некоторого заданного числа поколений значение функции пригодности только улучшается, то размер всех популяций уменьшается.

Кроме того, все популяции сотрудничают друг с другом. Они обмениваются индивидами: худшие индивиды одной популяции заменяются лучшими индивидами других популяций, тем самым передается информация о наилучших решениях полученных всем коллективом алгоритмов в целом.

Разработанный коллективный метод оптимизации на основе стайных бионических алгоритмов был назван Co-Operation of Biology Related Algorithms (COBRA) [62], его схема представлена ниже.

Co-Operation of Biology Related Algorithms

установить минимальный размер будущих популяций
установить максимальный размер будущих популяций
*инициализация 5 популяции **P, W, F, C, B** минимального размера случайным образом*
инициализация соответствующих параметров для каждого алгоритма
*определение **GBest***
пока *не выполнится критерий остановки*
    *выполнить для популяции **P** стайный алгоритм*
    *выполнить для популяции **W** алгоритм стай волков*
    *выполнить для популяции **F** алгоритм светлячков*
    *выполнить для популяции **C** алгоритм поиска кукушек*
    *выполнить для популяции **B** алгоритм летучих мышей*
    вычисление средней пригодности популяции: популяция с максимальной пригодностью – «победитель», прочие популяции – «проигравшие»
    для *каждой «проигравшей» популяции*
        сократить размер «проигравшей» популяции на 10% (но так, чтобы ее размер был не меньше минимального)
    конец цикла
    пока *размер победившей популяции и всех популяции в целом не превысит максимальное допустимое значение*
        увеличить размер победившей популяции на число удаленных индивидов из проигравших популяций
    конец цикла
    если *совершенно* $100 \cdot k, k = 1,2,\ldots$ *вычислений целевой функции*

осуществить миграцию лучших индивидов
если *сменилось* $10 \cdot k, k = 1,2,...$ *поколений*
если ***GBest*** *не улучшалось*
пока *размер всех популяций в целом не превысит максимального*
увеличить размер каждой популяции на K индивидов
конец цикла
если ***GBest*** *все время улучшалось*
пока *размер всех популяций больше минимального*
уменьшить размер каждой популяции на K индивидов
конец цикла
обновить значение ***GBest***
конец цикла

В данной схеме использованы следующие обозначения: ***P*** – популяция для стайного алгоритма, ***W*** – популяция для алгоритма поиска стаей волков, ***F*** – популяция для алгоритма светлячков, ***C*** – популяция для алгоритма поиска кукушек, ***B*** – популяция для алгоритма летучих мышей, ***GBest*** – лучшее значение целевой функции, найденное данное число итераций.

Число вычислений целевой функции для осуществления миграции между популяциями, а также число итераций для увеличения или уменьшения размеров всех популяций были подобраны эмпирическим путем.

Разработанный алгоритм изначально был протестирован на тех же шести функциях, что и его компоненты: сфера, гипер-эллипсоид, функция Розенброка, функция Растригина, функция Акли и функция Гриванка. Полученные результаты представлены в Таблице 2.1 (в Таблице СКО – среднеквадратическое отклонение).

Таблица 2.1. Результаты тестирования на первых шести задачах предложенной метаэвристики

№ задачи	Кол-во пер-ых	Разработанная метаэвристика				
		Надежность	Средний размер популяции	Среднее число вычислений	Среднее значение функции	СКО (значений функции)
1	2	100	20	251	0.000351333	0.000346687
	3	100	24	298	0.000321549	0.000290652
	4	100	27	303	0.000303284	0.000297926
2	2	100	20	168	0.000316957	0.000153012
	3	100	22	236	0.000354898	0.000172061
	4	100	27	238	0.000428898	0.000190029
3	2	100	29	172	0.000419898	0.000207145
	3	100	33	224	0.000222704	0.000182613
	4	100	32	300	0.000371695	0.000172214

Продолжение Таблицы 2.1. Результаты тестирования на первых шести задачах предложенной метаэвристики

№ задачи	Кол-во пер-ых	Разработанная метаэвристика				
		Надежность	Средний размер популяции	Среднее число вычислений	Среднее значение функции	СКО (значений функции)
4	2	100	20	205	0.000188838	0.000220224
	3	100	25	274	0.000516846	0.000282332
	4	100	28	308	0.000300688	0.000186405
5	2	100	22	243	0.000156269	0.000100685
	3	100	22	1633	0.000380357	5.42609e-005
	4	100	28	698	0.000122768	0.000159342
6	2	100	27	324	180.00071	0.000273844
	3	100	40	436	170.00062	0.000147725
	4	100	56	407	160.00081	0.000146504

Новые результаты были сравнены с результатами, полученными ранее для алгоритмов-компонентов. В Таблице 2.2 указаны алгоритмы, продемонстрировавшие лучшие результаты по двум критериям: минимальная ошибка, усредненная по количеству прогонов, и минимальное число вычислений, также усредненное по числу прогонов.

Таблица 2.2. Результаты сравнения алгоритма COBRA и его компонент на первых шести задачах

	Количество переменных	***Минимальное число вычислений***	***Мин. ошибка***
Сфера	2	**COBRA**	**COBRA**
	3	**COBRA**	BA
	4	**COBRA**	BA
Функция Гриванка	2	**COBRA**	**COBRA**
	3	**COBRA**	BA
	4	**COBRA**	BA
Функция Акли	2	**COBRA**	PSO
	3	**COBRA**	**COBRA**
	4	**COBRA**	**COBRA**
Гипер-эллипсоид	2	CSA	**COBRA**
	3	**COBRA**	**COBRA**
	4	**COBRA**	BA
Функция Розенброка	2	**COBRA**	**COBRA**
	3	**COBRA**	**COBRA**
	4	**COBRA**	**COBRA**
Функция Растригина	2	**COBRA**	**COBRA**
	3	**COBRA**	**COBRA**
	4	**COBRA**	**COBRA**

В Таблице 2.2 выделены случаи, когда предложенным оптимизационным методом COBRA достигались лучшие результаты по одному из критериев.

В итоге по первому критерию (минимальное число вычислений целевой функции) алгоритм COBRA «выигрывал» 11 раз из 18, а по второму (минимальная ошибка) – 14 раз из 18. Таким образом, было установлено, что применение предложенного оптимизационного метода целесообразно: разработанная метаэвристика COBRA показывает для многих задач лучшие результаты, чем ее компоненты, а с увеличением размерности это происходит все чаще [60].

Затем все пять составляющих алгоритмов (PSO, WPS, FFA, CSA и BA) и COBRA были исследованы на 28 тестовых функциях, представленных на конкурсе CEC 2013 Special Session on Real-Parameter Optimization [107], среди которых 5 унимодальных функций, 15 базовых многоэкстремальных функций и 8 составных функций, являющихся специальными комбинациями базовых и унимодальных. Размерность задач варьировалась от 2-х до 50, а именно число переменных было равно 2, 3, 5, 10, 30 и 50. В соответствии с методикой проведения экспериментов [107] для каждой задачи программа запускалась 51 раз. Максимальное число вычислений целевой функции по правилам конкурса было равно произведению размерности на 10000. Таким образом, для задач размерности 2 максимальное число вычислений целевой функции было равно 20000, размерности 3 – 30000 вычислений, размерности 5 – 50000 вычислений, размерности 10 – 100000 вычислений, размерности 30 – 300000 вычислений, и размерности 50 – 500000 вычислений.

Для примера, в Таблице 2.3 и Таблице 2.4 представлены результаты, полученные разработанным алгоритмом COBRA для тестовых функций из [107] при размерностях пространства оптимизации 10 и 30. В Таблицах отмечены лучший результат, достигнутый за 51 программных запусков, худший результат, найденный за это же число прогонов, и средний результат, полученный как среднеарифметическое лучших значений целевой функции за 51 программных

запусков, а также среднеквадратическое отклонение полученных результатов для каждой тестовой задачи.

Тот факт, что с ростом размерности алгоритм получает худшие результаты, является очевидным следствием недостаточности выделяемых вычислительных ресурсов и предопределен условиями конкурса.

Таблица 2.3. Результаты, полученные разработанным методом для тестовых задач при размерности 10

Функция	***Лучший рез-тат***	***Худший рез-тат***	***Средний рез-тат***	***СКО***
1	7.22923 e-012	1.61329 e-008	2.97237 e-009	5.18367 e-009
2	0	9.27679 e-005	1.14343 e-005	2.08808 e-005
3	1.15928 e-008	0.782002	0.102093	0.170695
4	4.19687 e-008	0.243355	0.0128341	0.0395501
5	3.98746 e-008	0.0138995	0.000520312	0.00197243
6	3.23174 e-007	0.00111909	2.46492e-005	0.000155108
7	7.70566 e-006	4.79825	0.0978292	0.664745
8	3.55271 e-009	0.0125654	0.000667647	0.00241682
9	0.00205973	0.0105291	0.00362009	0.0023464
10	4.01914 e-005	0.0326493	0.00244537	0.00565084
11	4.23582 e-009	0.0290106	0.00135262	0.00556143
12	5.96411 e-005	0.013147	0.000783514	0.00247486
13	0.000255768	0.0153074	0.000647979	0.002103
14	0.00010575	6.15133	0.44075	1.1953
15	0.000126139	0.969424	0.207082	0.340472
16	0.148442	0.559909	0.428111	0.0976917
17	0.0102249	7.40559	0.775591	1.32106
18	0.485884	27.2888	17.0889	4.79148
19	2.54762 e-005	0.991762	0.0260478	0.140436
20	1.09368 e-005	1.76635	0.532022	0.331655
21	0.000165587	0.000862846	0.000409035	0.000189936
22	0.331956	0.56639	0.403035	0.0462226
23	0.00148287	0.20037	0.0170046	0.0281186
24	8.16801 e-005	0.0688156	0.00371946	0.0105508
25	0.0267766	0.0886245	0.0547158	0.0191086
26	0.000773574	0.128411	0.0161644	0.0241594
27	0.0731828	0.605071	0.189737	0.11922
28	0.00974123	0.72632	0.0709495	0.152096

Таблица 2.4. Результаты, полученные разработанным методом для тестовых задач при размерности 30

Функция	***Лучший рез-тат***	***Худший рез-тат***	***Средний рез-тат***	***СКО***
1	3.65981 e-011	0.735166	0.0172834	0.103529
2	0	5.69112 e-005	1.10173 e-005	1.47884 e-005
3	0.00130707	4.21033	0.288137	0.921748
4	6.79105 e-008	17.3102	0.386406	2.40148
5	1.08306 e-008	0.000928656	0.000265419	0.000283025
6	4.36387 e-006	6.29691 e-006	5.31432 e-006	4.8246 e-007
7	0.00105331	0.0313925	0.0212978	0.0134079
8	3.55271 e-005	0.00666942	0.00142249	0.00148681
9	0.220816	0.250201	0.221676	0.00405631
10	0.000118777	0.751481	0.240704	0.203541
11	1.30852 e-007	2.01199	0.242801	0.506378
12	0.00027047	0.9197	0.21758	0.235546
13	0.00275232	0.301866	0.0200902	0.0470579
14	0.0208192	372.306	12.9145	54.6424
15	0.018697	5.51728	2.64347	1.90269
16	0.210536	1.51656	0.798325	0.316804
17	0.477067	41.7021	2.26946	5.94534
18	15.4715	41.7186	23.4264	7.89417
19	0.00671998	0.0987405	0.0175377	0.0246683
20	2.74662	4.23006	3.07382	0.33092
21	0.000588721	0.033769	0.00676935	0.00623506
22	0.00460148	1.9737	0.973548	0.502061
23	0.0662653	35.3119	8.24008	9.60994
24	0.101707	8.60235	0.960898	1.6663
25	0.0228218	6.41579	0.398358	0.984635
26	0.0127528	15.8701	1.17299	2.7341
27	0.300251	3.13571	1.99258	1.22089
28	1.08122	3.80534	2.65293	1.08693

После того, как все алгоритмы были протестированы на 28 функциях, взятых из конкурса [107], было проведено сравнение полученных результатов по двум критериями: самый лучший результат, полученный алгоритмом для данной задачи, и лучший средний результат.

В результате исследований было установлено, что эвристика COBRA показывала все более и более лучшие результаты по обоим критериям с увеличением размерности задач. По первому критерию коллективный алгоритм «выигрывал» у остальных 2 раза при размерности 2, 11 раз при размерности 5, 17

раз при размерности 19 и 26 раз при размерности 30. По второму критерию были соответственно следующие результаты: 19 «побед», 24 «победы», 28 «побед» при размерностях 5, 10 и 30 соответственно. Как пример, в Таблице 2.5 представлены результаты сравнения алгоритмов-компонентов и разработанной метаэвристики COBRA при размерностях 10 и 30 (для каждой задачи указан алгоритм-«победитель»).

Таблица 2.5. Результаты сравнения эффективности алгоритмов-компонентов и метаэвристики COBRA на 28 тестовых функциях конкурса CEC'2013

Функция	***Лучший рез-тат D=10***	***Средний рез-тат D=10***	***Лучший рез-тат D=30***	***Средний рез-тат D=30***
1	PSO	PSO	PSO	COBRA
2	COBRA	COBRA	COBRA	COBRA
3	COBRA	COBRA	COBRA	COBRA
4	COBRA	COBRA	COBRA	COBRA
5	PSO	PSO	WPS	COBRA
6	COBRA	COBRA	COBRA	COBRA
7	COBRA	COBRA	COBRA	COBRA
8	WPS	COBRA	COBRA	COBRA
9	COBRA	COBRA	COBRA	COBRA
10	COBRA	COBRA	COBRA	COBRA
11	COBRA	COBRA	COBRA	COBRA
12	COBRA	COBRA	COBRA	COBRA
13	COBRA	COBRA	COBRA	COBRA
14	WPS	COBRA	COBRA	COBRA
15	COBRA	COBRA	COBRA	COBRA
16	COBRA	COBRA	COBRA	COBRA
17	PSO	COBRA	COBRA	COBRA
18	COBRA	COBRA	COBRA	COBRA
19	COBRA	COBRA	COBRA	COBRA
20	WPS	COBRA	COBRA	COBRA
21	PSO	COBRA	COBRA	COBRA
22	COBRA	COBRA	COBRA	COBRA
23	WPS	COBRA	COBRA	COBRA
24	COBRA	COBRA	COBRA	COBRA
25	FFA	COBRA	COBRA	COBRA
26	COBRA	COBRA	COBRA	COBRA
27	COBRA	COBRA	COBRA	COBRA
28	PSO	COBRA	COBRA	COBRA

Таким образом, описанный новый самонастраивающийся алгоритм оптимизации на основе коллективного поведения алгоритмов стайного

интеллекта показал свою эффективность на тестовых задачах оптимизации с вещественными переменными. Кроме того, исследование эффективности разработанного метода, проведенное в сравнении с составляющими его алгоритмами, показало целесообразность его применения для решения оптимизационных задач. С ростом размерности задачи превосходство коллективного алгоритма становится неоспоримым, а, следовательно, этот алгоритм можно применять вместо составляющих его алгоритмов [60].

Стоит отметить, что при реализации коллективного подхода использовались первоначальные версии составляющих его алгоритмов, для которых в настоящее время существует множество улучшающих их модификаций. Сделано это было сознательно, так как ставилась задача проверки полезности подхода самого по себе. Поэтому замена составляющих метод алгоритмов на их более эффективные модификации, как ожидается, приведет к улучшению работы алгоритма в целом. Кроме того, можно увеличивать число составляющих алгоритмов для улучшения работы новой метаэвристики. И, наконец, эффективность коллективного метода может быть повышена за счет совершенствования его самого – уточнение интервала адаптации, изменение способа миграции, правил изменения размеров популяций, и т. д.

2.2. Коллективный метод условной оптимизации

Известно, что большое количество задач на практике требует поиска экстремума некоторой целевой функции на строго заданной области определения, то есть на область определения налагаются некоторые ограничения. Подобного рода задачи также называют задачами условной оптимизации. Однако применение классических эволюционных алгоритмов для их решения невозможно, так как у них отсутствуют механизмы учета ограничений оптимизационной задачи. Можно указать несколько возможных методов непосредственного решения этой проблемы [96], [77].

Предположим, что необходимо решить задачу условной оптимизации с r ограничениями в виде неравенств, $(m–r)$ ограничениями в виде равенств и размерностью пространства равной D:

$$f(x) \to extr$$

$$\begin{cases} g_j(x) \le 0,\ j = \overline{1,r} \\ h_j(x) = 0,\ j = \overline{r+1,m} \end{cases}$$

Для решения подобных задач условной оптимизации с вещественными переменными была разработана модификация алгоритма COBRA, названная COBRA-c [20]. Модификация предложенной метаэвристики заключалась в учете ограничений тремя различными способами: методом динамических штрафов [83], с помощью правил Дэба [76], а так же методом, описанным в статье [108], использующим принципы селекции.

Для штрафных алгоритмов учета ограничений пригодность каждого индивида x вычисляется в общем случае по следующей формуле:

$$fitness(x) = f(x) + \delta \cdot \lambda(t) \cdot \sum_{j=1}^{m} f_j^{\beta}(x),$$

где t – номер текущего поколения, $\delta = 1$ для задач минимизации и $\delta = -1$ для задач максимизации, $f_j(x)$ – штраф за нарушение j-го ограничения, β – вещественное число. Штрафные функции $f_j(x)$ вычисляются по формуле:

$$f_j(x) = \begin{cases} \max\{0, g_j(x)\},\ j = \overline{1,r} \\ |h_j(x)|,\ j = \overline{r+1,m} \end{cases}.$$

Анализ различных штрафных методов учета ограничений показал, что целесообразнее использовать метод динамических или метод адаптивных штрафов, так как альтернативные методы обладают рядом недостатков. В частности, например, метод «смертельных» штрафов попросту отбрасывает недопустимые решения, тогда как они, находясь вблизи допустимой области, зачастую несут в себе полезную информацию для порождения новых допустимых решений. Данный метод хорошо работает на простых задачах, когда допустимая область обладает удобными для оптимизации свойствами (большой размер,

выпуклая и т.д.). Однако в реальных задачах, когда допустимая область является достаточно малой, по сравнению со всем поисковым пространством, не является выпуклой и даже связной, применение данного метода приведет к тому, что допустимые решения не будут найдены.

Еще один известный метод статических штрафов использует много настраиваемых параметров, которые пользователь должен самостоятельно подобрать перед решением каждой задачи условной оптимизации. Неверный выбор данных параметров в случае, когда значения целевой функции и ограничений отличаются на порядки, может привести к тому, что допустимые решения также не будут найдены.

Гибридные методы, использующие механизм «лечения», применяют на каждом шаге работы алгоритма методы локального поиска, для того, чтобы минимизировать значения штрафных функций $f_j(x)$. Это требует значительных дополнительных вычислительных ресурсов. В случае, когда вычисление целевой функции связано с какими-либо значительными затратами (материальные, время и др.), использование данных методов неприемлемо.

Метод адаптивных штрафов имеет большее число параметров, чем метод динамических штрафов, поэтому было принято решение использовать в дальнейшем последний.

Метод динамических штрафов определяет штрафную функцию $\lambda(t)$ следующим образом:

$$\lambda(t) = (C \cdot t)^{\alpha},$$

где C, α – некоторые константы. Таким образом, на *t*-ой итерации пригодность индивида *x* вычисляется для данного метода по следующей формуле:

$$fitness(x) = f(x) + \delta \cdot (C \cdot t)^{\alpha} \cdot \sum_{j=1}^{m} f_j^{\beta}(x).$$

Параметры C, α, β подбираются индивидуально для каждой решаемой задачи. Рекомендованные значения $C = 0.5, \alpha = \beta = 2$ [113].

Применение правил Дэба целесообразно, когда функции ограничения важнее по значимости для лица, принимающего решение, чем целевая функция.

Допустим, необходимо решить задачу минимизации с m ограничениями (в виде неравенств или равенств) с вещественными переменными и размерностью пространства поиска D. Далее предположим, что некоторым эволюционным алгоритмом найдены два решения x_k и x_l. Тогда лучшее из этих двух решений определяется следующим образом:

1. Если оба решения удовлетворяют всем m ограничениям задачи, то лучшее решение из двух определяется по значению целевой функции (чем меньше, тем лучше);
2. Если одно из решений, например, x_k, удовлетворяет всем m ограничениям задачи, а для другого, x_l, нарушается хотя бы одно ограничение, то лучшим решением будет x_k, и, наоборот;
3. Если для обоих решений нарушается хотя бы одно из m ограничений, то лучшим считается то, для которого модуль суммы значений функций ограничений меньше.

Метод, описанный в работе [108], заключается в том, что и оптимизируемая функция, и функции-ограничения изначально считаются целевыми функциями, однако, какую из них нужно минимизировать (максимизировать) на каждой итерации для каждого индивида в частности, определяется с помощью турнирной селекции. Пусть выполняется следующее правило:

$$a > b = \begin{cases} 1, a > b \\ 0, a \le b \end{cases}.$$

Кроме того, допустим, что все m ограничений представлены в виде неравенств $g_j(x) \le 0, j = \overline{1,m}$, а количество индивидов равно ps. Если для задачи определены некоторые ограничения в виде равенств, то их следует привести к виду неравенств.

Тогда далее вычисляются следующие значения:

$$p_i = \frac{\sum_{j=1}^{ps}\left(g_j(x_j) > 0\right)}{ps}, i = 1,...,m;$$

$$fp = 1 - \overline{p}, p = [p_1, p_2, ..., p_m],$$

$$\overline{p} = \frac{1}{m}\sum_{i=1}^{m} p_i ;$$

$$gp_i = \overline{p} \cdot \frac{p_i}{\sum_{i=1}^{m} p_i},$$

$$fp + \sum_{i=1}^{m} gp_i = 1.$$

В данных обозначениях gp_i и fp можно рассматривать как вероятности. Далее для каждого индивида с помощью турнирной селекции по значениям gp_i и fp выбирается оптимизируемая функция (либо функция-ограничение, либо целевая функция).

Итак, метод, описанный в работе [108], был использован для учета ограничений алгоритмом PSO, в то время как для остальных алгоритмов-компонент метаэвристики COBRA были сначала применены правила Дэба, а затем метод динамических штрафов.

Изначально алгоритм COBRA-c был протестирован на множестве относительно несложных задач условной оптимизации, которые ранее применялись в [13] для тестирования соответствующей модификации стайного алгоритма. Всего было 6 задач, количество переменных варьировалось от 2 до 4, подробное описание приведено в первой главе.

Задачи максимизации были сведены к задачам минимизации, а ограничения вида $g(x) \geq 0$ преобразованы к виду $g(x) \leq 0$. Для каждой задачи было выполнено 100 программных запусков, определялся лучший результат, найденный алгоритмом за все прогоны, худший результат, полученный алгоритмом за все прогоны, а также результат, усредненный по количеству прогонов и среднеквадратическое отклонение. Результаты тестирования алгоритма COBRA-c на упомянутых задачах приведены в Таблице 2.6.

Таблица 2.6. Результаты тестирования алгоритма COBRA-c для первых шести задач

1	-117	-116.996	-116.99964	0.000764437
2	-4.44089e-016	9.8671e-005	3.20875e-005	2.62605e-005
3	-69.8482	-69.8453	-69.8479	0.000584493

Продолжение Таблицы 2.6. Результаты тестирования алгоритма COBRA-c для первых шести задач

Функция	*Лучший рез-тат*	*Худший рез-тат*	*Средний рез-тат*	*СКО*
4	-125	-124.98	-124.992	0.000852199
5	-69.8482	-69.8467	-69.8479	0.000317708
6	-11.1667	-11.1664	-11.1666	7.87886e-005

Разработанный алгоритм COBRA-c успешно справился с решением первых шести тестовых задач условной оптимизации, более того, продемонстрировал результаты лучшие, чем его компоненты. Тестирование на этих задачах условной оптимизации показало, что новая метаэвристика COBRA-c работоспособна и ее применение целесообразно.

Затем модификация COBRA-c [20] и ее алгоритмы-компоненты были протестированы с помощью 18 задач условной оптимизации, представленных на конкурсе CEC 2010 Competition on Constrained Real-Parameter Optimization [109]. Размерность задач была равна сначала 10, потом 30. Программа запускалась для каждой задачи 25 раз. Максимальное число вычислений целевой функции было установлено равным 200000 для задач размерности 10 и 600000 для задач размерности 30. Результаты тестирования представлены в Таблице 2.7 и Таблице 2.8 (в таблицах «ППД» означает процент прогонов с допустимыми решениями).

Таблица 2.7. Результаты тестирования алгоритма COBRA-c для задач размерности 10

Функция	Лучший рез-тат	Худший рез-тат	Средний рез-тат	СКО	ППД
1	-0.727373	-0.559235	-0.637199	0.0594299	100
2	1.82813	4.34865	2.75755	0.926259	100
3	8.87653	8.89164	8.87895	0.00318718	92
4	5.57793	5.58908	5.58215	0.00312711	28
5	189.952	516.713	338.063	82.963	32
6	103.515	563.247	369.431	72.4124	24
7	0.529035	0.888604	0.566389	0.0676125	100
8	21.8649	53.8881	23.4468	6.23478	100
9	1.9133e+012	2.4654e+012	2.04001e+012	2.12584e+012	68
10	2.30765e+011	2.84432e+012	6.58941e+011	8.31906e+011	84
11	0.0002073305	0.00843533	0.00190705	0.000479151	68
12	-169.36	671.962	-102.82	228.253	0
13	-57.1851	-54.9831	-57.0463	0.429771	100
14	7.9878	1.72346e+007	6.97436e+006	8.37255e+006	100
15	6.9986e+009	4.94244e+011	7.56845e+010	1.22078e+011	100

Продолжение Таблицы 2.7. Результаты тестирования алгоритма COBRA-c для задач размерности 10

Функция	Лучший рез-тат	Худший рез-тат	Средний рез-тат	СКО	ППД
16	0.724961	1.17519	0.826752	0.165962	56
17	103.275	321.092	111.988	42.6833	100
18	378.76	671.905	425.911	77.383	96

Таблица 2.8. Результаты тестирования алгоритма COBRA-c для задач размерности 30

Функция	Лучший рез-тат	Худший рез-тат	Средний рез-тат	СКО	ППД
1	-0.739337	-0.401494	-0.598349	0.0935372	100
2	3.89561	5.02194	4.126	0.400138	100
3	28.6756	28.6811	28.6762	0.00110221	76
4	9.1352	9.13546	9.13527	6.5384e-005	100
5	571..601	585.681	573.29	4.57558	88
6	216.562	508.433	333.311	142.987	56
7	4.47193	72.6267	20.8513	28.9008	100
8	0.173656	73.0897	5.76662	18.9706	100
9	9.19574e+011	1.49284e+013	5.96147e+012	4.43056e+012	64
10	1.13746e+010	1.57261e+011	3.99415e+010	2.95382e+010	60
11	-0.000785623	-4.81327e-005	-0.00048172	1.89917e-005	32
12	-45.2943	-45.2186	-45.2633	0.0216902	64
13	-60.9165	-59.9094	-60.8377	0.220922	100
14	2009.4	5.90293e+004	4343.39	1.11632e+004	100
15	6.26952e+010	1.50579e+012	3.32008e+011	3.95185e+011	100
16	1.18726	1.19005	1.18884	0.000796221	100
17	131.511	132.365	131.958	0.255058	100
18	2630.99	3036.88	2874.69	159.654	100

Таким образом, еще раз была доказана эффективность разработанного оптимизационного метода.

Далее, разработанная эвристика COBRA-c сравнивалась с алгоритмами, ставшими победителями в упомянутом соревновании.

Средние результаты, полученные алгоритмами-победителями конкурса (всего их было 12), а также средние результаты, полученные разработанным методом COBRA-c, были упорядочены по возрастанию. Алгоритму, которым было достигнуто минимальное значение, присваивался первый ранг; второй ранг присваивался алгоритму, которым было достигнуто минимальное из оставшихся значение целевой функции, и т.д.

Таким образом, «лучший» алгоритм для заданной функции имел первый ранг, а худший – тринадцатый ранг. В Таблицах 2.9, 2.10, 2.11 и 2.12 представлены ранги алгоритмов-победителей конкурса и предложенного метода COBRA-c для задач размерности 10 и 30.

Таблица 2.9. Ранжирование алгоритмов победителей и метода COBRA-c для задач размерности 10

Alg/Prob	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
jDEsoco	6	13	8	1	9	4	1	3	3
DE-VPS	11	6	11	9	5	10	9	11	5
RGA	8	8	13	7	10	11	10	5	6
E-ABC	9	7	12	11	7	8	12	12	8
εDEg	1	5	1	4	1	1	1	9	1
DCDE	12	4	1	10	1	1	7	10	4
Co-CLPSO	7	3	5	6	1	1	8	1	7
CDEb6e6rl	4	10	6	1	12	12	1	8	12
sp-MODE	1	12	10	13	13	13	1	7	13
MTS	13	11	7	12	6	6	13	13	9
IEMA	5	1	4	5	8	9	5	6	11
ECHT	1	2	1	1	4	5	6	4	2
COBRA	10	9	9	8	11	7	11	2	10

Таблица 2.10. Ранжирование алгоритмов победителей и метода COBRA-c для задач размерности 10

Alg/Prob	***10***	***11***	***12***	***13***	***14***	***15***	***16***	***17***	***18***
jDEsoco	4	3	6	3	4	7	8	9	9
DE-VPS	5	8	10	6	5	4	1	6	1
RGA	6	9	11	7	8	6	9	7	7
E-ABC	9	12	4	8	12	11	6	8	8
εDEg	1	1	1	1	1	2	7	4	1
DCDE	3	6	9	12	2	1	5	2	5
Co-CLPSO	7	11	3	9	3	3	2	5	6
CDEb6e6rl	12	7	2	1	11	13	12	12	12
sp-MODE	13	13	13	13	10	9	13	13	13
MTS	8	10	12	11	13	12	10	11	11
IEMA	11	2	5	4	6	5	3	1	1
ECHT	2	4	8	10	9	9	4	3	1
COBRA	10	5	7	5	7	8	11	10	10

Таблица 2.11. Ранжирование алгоритмов победителей и метода COBRA-c для задач размерности 30

Alg/Prob	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
jDEsoco	5	8	3	3	8	2	1	7	2
DE-VPS	12	6	8	8	4	7	6	10	8
RGA	7	7	12	7	5	8	12	13	7
E-ABC	8	9	11	10	6	6	13	9	9
εDEg	2	3	2	4	1	1	3	2	3
DCDE	10	2	1	9	10	10	5	4	13
Co-CLPSO	9	1	10	6	2	3	8	8	6
CDEb6e6rl	1	10	5	2	12	11	1	1	1
sp-MODE	3	12	9	13	13	13	10	11	12
MTS	13	13	7	11	7	5	7	12	10
IEMA	4	5	13	12	11	12	4	5	5
ECHT	6	4	4	1	3	4	9	6	4
COBRA	11	11	6	5	9	9	11	3	11

Таблица 2.12. Ранжирование алгоритмов победителей и метода COBRA-c для задач размерности 30

Alg/Prob	***10***	***11***	***12***	***13***	***14***	***15***	***16***	***17***	***18***
jDEsoco	2	3	1	1	4	6	7	10	9
DE-VPS	6	7	10	9	5	4	6	5	4
RGA	7	6	6	8	10	7	9	8	6
E-ABC	10	10	7	5	8	10	8	7	8
εDEg	3	2	11	4	1	2	1	6	7
DCDE	1	5	2	7	3	1	5	4	3
Co-CLPSO	8	11	3	11	5	3	1	3	5
CDEb6e6rl	13	1	9	3	11	12	10	13	12
sp-MODE	12	13	13	13	13	11	13	11	11
MTS	9	9	4	12	12	13	12	12	13
IEMA	5	12	12	2	2	5	4	1	1
ECHT	4	4	5	6	9	8	1	2	1
COBRA	11	8	8	10	7	9	11	9	10

После того, как для каждой задачи при заданной размерности результаты, полученные всеми методами, были упорядочены от лучшего к худшему, вычислялся средний ранг для алгоритма по задачам с данными количеством переменных. Алгоритм с наименьшим средним значением соответственно был лучшим, с наибольшим – худшим. Результаты сравнения разработанной эвристики COBRA-c с победителями конкурса представлены в Таблице 2.13.

Таблица 2.13. Результаты сравнения метода COBRA-c с алгоритмами-победителями конкурса

Алгоритм	*Ранжирование*			
	10D	***30D***	***Итоговое значение***	***Среднее значение***
jDEsoco	101	82	183	5.08
DE-VPS	123	125	248	6.88
RGA	148	145	293	8.14
E-ABC	164	154	318	8.83
εDEg	43	58	101	2.81
DCDE	95	95	190	5.27
Co-CLPSO	88	103	191	5.31
CDEb6e6rl	148	128	276	7.66
sp-MODE	193	206	399	11.08
MTS	188	181	369	10.25
IEMA	92	115	207	5.75
ECHT	76	81	157	4.36
COBRA	150	159	309	8.58

Таблица 2.14. Итоги сравнения метода COBRA-c с алгоритмами-победителями конкурса

Rank	***Algorithm***
1	εDEg
2	ECHT
3	jDEsoco
4	DCDE
5	Co-CLPSO
6	IEMA
7	DE-VPS
8	CDEb6e6rl
9	RGA
10	COBRA
11	E-ABC
12	MTS
13	sp-MODE

Как видно из Таблицы 2.14, алгоритм COBRA-c показывает лучшие результаты, чем некоторые из победителей соревнования, что еще раз доказывает целесообразность применения разработанного алгоритма. Кроме того, было установлено, что новый метод COBRA-c превосходит свои компоненты-алгоритмы, модифицированные для решения задач условной оптимизации.

Разработанная эвристика COBRA-с была применена для решения практической задачи в рамках российско-словенского проекта [129].

2.3. Коллективный метод оптимизации с бинарными переменными

Первоначально упомянутые ранее алгоритмы – PSO, WPS, FFA, CSA и BA – были разработаны для решения оптимизационных задач с вещественными переменными. В настоящий момент использование алгоритмов расширилось вплоть до дискретных задач и задач с бинарными переменными. Чтобы расширить перечисленные алгоритмы, работающие с вещественными переменными, в бинарное/дискретное пространство, наиболее важно понять смысл таких понятий, как траектория, скорость в бинарном/дискретном пространстве.

Первым бионическим методом, модифицированным для решения оптимизационных задач с бинарными переменными, был стайный алгоритм оптимизации PSO [102]. Модификация алгоритма была разработана Кеннеди и Эберхартом и заключалась в использовании скорости i-ой частицы v_i, а также вероятности для определения, в каком состоянии находится d-ая компонента i-ой частицы x_{id} – 1 или 0. Для этого они стягивали в точку d-ую компоненту i-ой частицы x_{id}, используя логистическую функцию:

$$s(v_{id}) = \frac{1}{1 + \exp(-v_{id})}.$$

Далее согласно методике Кеннеди и Эберхарта, генерировалось случайным образом некоторое число в пределах [0;1], и если оно было меньше значения логистической функции для соответствующей d-ой компоненты i-ой частицы x_{id} из популяции, то d-ая компонента i-ой частицы x_{id} была равна 1, в противном же случае, была равна 0.

Модификация алгоритма летучих мышей для решения задач безусловной оптимизации с бинарными переменными предложена в работе [118], [8]. Она идентична бинарному стайному алгоритму, поскольку в методе летучих мышей

так же, как и в методе роя частиц, каждый индивид описывается не только своими координатами в пространстве поиска, но и скоростью перемещения. Таким образом, состояние (1 или 0) *d*-ой компоненты *i*-ой мыши b_{id} определяется значением логистической функции, вычисленной от соответствующей компоненты скорости мыши и его сравнением с случайным числом, сгенерированным в пределах [0;1].

Алгоритм светлячков для решения задач оптимизации с бинарными переменными описан в статье [137], аналогичная модификация алгоритма поиска кукушек предложена в работе [122].

Известно, что в алгоритме поиска стаей волков, алгоритме светлячков и алгоритме поиска кукушек индивиды описываются лишь своими координатами. Поэтому для модификации перечисленных бионических методов *d*-ая компонента *i*-ого индивида x_{id} вычисляется следующим образом:

1) Определяется число $x_{id}^{'}$ с помощью логистической функции

$$x_{id}^{'} = s(x_{id}) = \frac{1}{1+\exp(-x_{id})};$$

2) Генерируется случайное число $rand \in [0;1]$;

3) Для *d*-ой компоненты *i*-ого индивида x_{id} определяется ее состояние (1 или 0) по правилу

$$x_{id} = \begin{cases} 1, x_{id}^{'} < rand \\ 0, x_{id}^{'} \geq rand \end{cases}.$$

Описанные модификации алгоритмов для решения задач оптимизации с бинарными переменными, на кооперативной работе которых основана эвристика COBRA, были использованы для разработки метода COBRA-b для решения задач безусловной оптимизации с бинарными переменными [59].

Упомянутые ранее шесть тестовых задач [116] (сфера, гипер-эллипсоид, функция Розенброка, функция Растригина, функция Акли и функция Гриванка) были использованы для исследования эффективности бинарной модификации алгоритма COBRA. Максимальное число вычислений целевой функции было установлено равным 100000, но работа алгоритма завершалась после нахождения

оптимального решения с заранее заданной погрешностью 0.001. Переменные для перечисленных задач являются вещественными, поэтому для тестирования алгоритма COBRA-b на них, требовалось данные переменные закодировать в бинарные строки. Проведенные ранее исследования показали, что для заданной погрешности достаточно каждую вещественную переменную представить в виде бинарной строки в 10 бит. Таким образом, размерность задач в бинарных переменных варьировалась от 20 до 40.

Результаты исследований приведены в Таблице 2.15. В Таблице указаны средний размер популяции в целом и число вычислений целевой функции, необходимые для достижения оптимального решения с заданной точностью (данные показатели оценивались только по успешным прогонам и усреднялись соответственно тоже по ним), а также итоговое значение целевой функции, усредненное по всем прогонам, и среднеквадратическое отклонение. В Таблице 2.15 количество переменных обозначено как «*D*», то есть, например, $D = 2$ означает 2 вещественные переменные или 20 бинарных переменных.

Таблица 2.15. Результаты тестирования алгоритма COBRA-b на первых шести функциях

Функция	D	Средний размер популяции	Среднее число вычислений целевой функции	Среднее значение целевой функции	СКО
1	2	31	740	0.000182069	0.282119
	3	68	3473	0.000188191	6.94652e-005
	4	80	6730	0.00579879	0.0129499
2	2	27	567	0.000236274	9.93911
	3	30	775	0.000150127	8.85592e-005
	4	32	916	0.000355086	0.0669021
3	2	32	1439	0.00019874	2.82798
	3	51	2046	0.00150713	0.222684
	4	62	3030	0.00126295	0.0515319
4	2	33	931	0.000209168	0.515766
	3	32	868	0.000191162	1.67698
	4	79	1710	0.000347666	0.306151
5	2	30	899	0.00032841	0.00046868
	3	65	1332	0.000506847	0.00140048
	4	160	2258	0.00411721	0.158903
6	2	28	1734	180.0002	0.00018536
	3	36	3294	169.801	0.169149
	4	41	5462	159.2	0.279294

В итоге работоспособность оптимизационного метода COBRA-b и целесообразность его применения были установлены и доказаны. Кроме того, было установлено, что разработанная метаэвристика COBRA-b превосходит по результатам свои алгоритмы-компоненты (бинарные версии алгоритмов PSO, WPS, FFA, CSA и BA).

Сравнение результатов, полученных «стандартной» версией алгоритма COBRA, и тех, что получены разработанным оптимизационным методом COBRA-b, показывает, что последний работает медленнее алгоритма COBRA, которым также достигаются лучшие значения целевой функции. Однако стоит отметить, что тестирование проводилось на задачах безусловной оптимизации с вещественными переменными, что могло отразиться на результатах предложенной метаэвристики COBRA-b.

Как уже упоминалось ранее, обычно на практике на поисковую область накладываются некоторые ограничения, ее сужающие, то есть на практике обычно необходимо решать задачи условной оптимизации. Этот факт послужил причиной разработки бинарной модификации предложенного ранее алгоритма решения задач условной оптимизации с вещественными переменными COBRA-c. Модификация проводилось аналогично той, что была применена для «стандартной» версии метода COBRA, то есть была использована методика Кеннеди и Эберхарта [102].

Бинарная модификация разработанного алгоритма COBRA-c сначала была протестирована на шести задачах условной оптимизации с небольшим числом переменных [31]. Данные задачи также являются вещественными, поэтому для тестирования алгоритма COBRA-b на них требовалось вещественные переменные закодировать в бинарные строки. Погрешность вычислений была задана равной 0.001, поэтому было достаточно каждую вещественную переменную представить в виде бинарной строки в 10 бит. Таким образом, размерность задач в бинарных переменных варьировалась от 20 до 40.

В Таблице 2.16 приведены результаты тестирования бинарной модификации предложенной метаэвристики COBRA-c на первых шести задачах

условной оптимизации. В данной Таблице «ППД» означает процент прогонов с допустимыми решениями.

Таблица 2.16. Результаты тестирования алгоритма COBRA-b на первых шести задачах условной оптимизации

№ задачи	Лучший рез-тат	Худший рез-тат	Средний рез-тат	СКО	ППД
1	-116.987	-116.954	-116.984	0.0063109	100
2	-4.44089e-016	0.0625593	0.00451869	0.0122674	100
3	-69.6921	-66.6813	-69.3636	0.699957	100
4	-124.828	-124.597	-124.726	0.0773779	100
5	-69.6921	-68.6657	-69.6154	0.26061	100
6	-11.1653	-11.1477	-11.1639	0.00477349	100

Тестирование алгоритмов, составляющих метод COBRA-b (бинарные версии их условных модификаций), на этих же задачах показало, что разработанная метаэвристика демонстрирует лучшие результаты.

Далее бинарная версия разработанного алгоритма COBRA-c была исследована на 18 вещественных задачах конкурса CEC’2010 [109] размерности 10. Максимальное число вычислений целевой функции было равно 200000, программа запускалась 25 раз (то есть было 25 прогонов). Каждая переменная была закодирована в виде бинарной строки в 10 бит, таким образом, решались 18 задач условной оптимизации с 100 бинарными переменными.

В Таблице 2.17 продемонстрированы результаты проведенных исследований: худший результат, найденный за 25 прогонов, лучший результат за 25 прогонов, итоговое значение целевой функции, усредненное по количеству прогонов, среднеквадратическое отклонение, а также процент прогонов с допустимыми решениями («ППД»).

Таблица 2.17. Результаты тестирования алгоритма COBRA-b на задачах условной оптимизации конкурса CEC’2010

№ задачи	Лучший рез-тат	Худший рез-тат	Средний рез-тат	СКО	ППД
1	-2.76873	-0.638384	-1.6363864	0.610586201	100
2	0.219038	1.49053	0.699351	0.39969	16
3	1.82901e+011	5.96292e+013	9.871e+012	1.22748e+013	0
4	-2.96071	4.74287	1.298189	2.440195863	0
5	-299.167	23.7935	-122.088188	130.7204842	0
6	-576.47	58.0159	-282.229	210.855	0

Продолжение Таблицы 2.17. Результаты тестирования алгоритма COBRA-b на задачах условной оптимизации конкурса CEC'2010

№ задачи	Лучший рез-тат	Худший рез-тат	Средний рез-тат	СКО	ППД
7	3.21247e+006	7.07832e+007	3.99E+07	2.73E+07	100
8	2.53262e+007	1.38524e+008	5.17861e+007	3.68168e+007	100
9	1.84722e+010	2.62639E+12	8.81456E+11	1.06383E+12	0
10	1.85464e+010	1.92614e+010	1.899993e+010	2.6286e+008	0
11	-25.1844	-1.49913	-12.5539	5.6182	0
12	-521.453	-164.062	-370.1806	108.9847041	0
13	-485.229	-480.557	-480.778	0.923323	8
14	1.85662e+009	2.00128e+009	1.889359e+009	4.413263e+007	100
15	1.85623e+009	2.01067e+009	1.902778e+009	7.034892e+007	100
16	0.305536	0.533195	0.4213278	0.062844125	70
17	1.85662e+009	1.89275e+009	1.876011e+009	1.580137e+007	100
18	27.6054	45.15	35.50327	4.636845695	100

Далее проводилось исследование эффективности составляющих алгоритмов (бинарных версий модификаций алгоритмов PSO, WPS, FFA, CSA и BA для решения задач условной оптимизации) с помощью этих же 18 задач размерности 10. Максимальное число вычислений и число прогонов не менялись (200000 и 25 соответственно). Полученные результаты сравнивались с теми, что были достигнуты бинарной версией разработанного алгоритма COBRA-c.

В Таблице 2.18 представлены результаты сравнения разработанной модификации алгоритма COBRA для решения однокритериальных задач условной оптимизации с бинарными переменными и его составляющих методов, то есть соответствующих модификаций алгоритмов PSO, WPS, FFA, CSA и BA. Наиболее важными показателями работы всех перечисленных методов оптимизации являются процент прогонов с допустимыми решениями и среднее значение целевой функции.

Таблица 2.18. Результаты тестирования алгоритма COBRA-b на задачах условной оптимизации конкурса CEC'2010

№ задачи	*COBRA*	*PSO*	*WPS*	*FFA*	*CSA*	*BA*
1	-1.6363864 (100)	-0.621562 (100)	-0.572244 (100)	-0.471491 (100)	-0.410512 (100)	-0.266626 (100)
2	0.699351 (16)	**243.218 (48)**	**10.541 (96)**	**43.0764 (76)**	**23.1881 (88)**	**192.864 (24)**
3	9.871e+012 (0)	3.82815e+015 (0)	2.77873e+015 (0)	2.86436e+015 (0)	3.64136e+015 (0)	2.9837e+015 (0)
4	1.298189 (0)	4.93681e+012 (0)	1.17385e+013 (0)	2.55671e+012 (0)	2.83572e+012 (0)	3.20304e+012 (0)
5	-122.088188 (0)	404716 (0)	192601 (0)	192513 (0)	178714 (0)	163246 (0)
6	-282.229 (0)	2.04208e+006 (0)	1.45486e+006 (0)	1.13862e+006 (0)	1.04578e+006 (0)	1.25087e+006 (0)
7	3.99E+007 (100)	**111914 (100)**	**167878 (100)**	4.09367e+007 (100)	3.49351e+007 (100)	3.6155e+007 (100)
8	5.17861e+007 (100)	**415087 (100)**	**165774 (100)**	3.51232e+007 (100)	3.01527e+007 (100)	2.88015e+007 (100)
9	8.81456E+011 (0)	**2.60347e+012 (4)**	**1.50473e+0012 (8)**	1.62709e+012 (0)	**1.3131e+012 (100)**	1.86997e+012 (0)
10	1.899993e+010 (0)	1.98028e+012 (0)	1.40288e+012 (0)	1.61206e+012 (0)	**1.72585e+012 (96)**	1.54865e+012 (0)
11	-12.5539 (0)	2.37415e+012 (0)	1.44017e+012 (0)	1.11742e+012 (0)	1.81747e+012 (0)	1.28953e+012 (0)
12	-370.1806 (0)	1.99001e+006 (0)	1.50186e+006 (0)	1.13485e+006 (0)	1.22991e+006 (0)	1.11382e+006 (0)
13	-480.778 (8)	**1.101012e+008(36)**	8.06574e+007 (4)	**-32.436 (100)**	**3.04547e+007 (72)**	**9.87519e+007 (20)**
14	1.889359e+009 (100)	**92.3234 (100)**	1.12288e+012 (100)	1.53511e+012 (100)	6.96763e+012 (100)	2.50752e+012 (40)
15	1.902778e+009 (100)	**9.46478 (100)**	**8.94005e+007 (100)**	**1.1124e+006 (100)**	**7.47061e+006 (100)**	**6.94987e+006 (100)**
16	0.4213278 (70)	7.85534e+006 (0)	322508 (24)	351549 (24)	**1.05269 (100)**	195496 (0)
17	1.876011e+009 (100)	2225.35 (48)	760.968 (84)	791.464 (48)	410.607 (96)	1390.21 (80)
18	35.50327 (100)	5918.58 (76)	5666.34 (84)	5383.14 (72)	5140 (100)	3122.51 (100)

Таким образом, в Таблице 2.18 указаны средние значения целевых функций для всех 18 задач, а в скобках процент прогонов с допустимыми решениями. Соответственно, лучше работал тот алгоритм, которым было достигнуто большее

значение показателя «процент прогонов с допустимым решением», если же некоторые алгоритмы имели одинаковое значение данного параметра, то лучшим считался тот, которым было получено меньшее значение целевой функции (все 18 задач были задачами минимизации). В Таблице 2.18 выделены случаи, когда алгоритмы-компоненты демонстрировали результаты лучшие, чем разработанная метаэвристика. Однако в целом можно сделать вывод, что предложенная модификация алгоритма COBRA для решения однокритериальных задач условной оптимизации с бинарными переменными превосходит свои алгоритмы-компоненты по обоим критериям и может быть применена вместо них.

2.4. Решение практических задач коллективными методами оптимизации

Описанная ранее бинарная модификация метаэвристики для решения задач условной оптимизации COBRA-с была применена для решения двух практических задач: формирования оптимального инвестиционного портфеля предприятия и формирования оптимального кредитного портфеля банка.

Рассмотрим задачу формирования оптимального инвестиционного портфеля предприятия. Она состоит в составлении такого портфеля инвестиционных проектов, который приносит инвестору наибольшую прибыль. При этом должны выполняться ограничения по выделяемым средствам, норме прибыли и общей рискованности портфеля.

Для формализованной записи критерия получения максимальной доходности от инвестиционных проектов, при соблюдении всех ограничений, введем следующие обозначения [44]:

1. m – количество центров финансовой ответственности (ЦФО) на предприятии;
2. N_i – количество инвестиционных проектов на i-м ЦФО;
3. $П_{ij}$ – плановый годовой объем прибыли, получаемый i-м ЦФО от внедрения j-го нововведения;

4. R_{ij} – экспертная оценка рискованности соответствующего инновационного проекта;
5. c_{ij} — плановые годовые затраты финансовых средств *i*-го ЦФО на *j*-е нововведение, способствующее увеличению мощности ЦФО;
6. C_i — плановые годовые объемы финансовых средств, выделяемые ЦФО в план нововведений;
7. $C = \sum_{i=1}^{m} C_i$ – сумма всех средств, выделяемых всеми ЦФО на реализацию их инвестиционных программ;
8. M — плановый годовой объем финансовых средств, выделяемый центральной компанией в планы нововведений ЦФО;
9. r — допустимая средняя прибыль на 1 руб. затрат (норма прибыли на капитал);
10. ρ – ограничение на суммарную рискованность инвестиционного портфеля;
11. x_{ij} – искомый параметр, показывающий, планируется ли к внедрению на *i*-м ЦФО *j*-е нововведение (если $x_{ij} = 1$, то планируется; если $x_{ij} = 0$, то не планируется).

Центры финансовой ответственности – это структурные подразделения предприятия, обладающие хозяйственной самостоятельностью и имеющие разрешение планировать инвестиционные проекты, финансируемые из собственных средств. Материнское предприятие может добавлять свои средства на инвестиционные программы ЦФО.

Таким образом, задача формирования оптимального инвестиционного портфеля предприятия может быть сформулирована как однокритериальная задача условной оптимизации с бинарными переменными.

$$f(x) = \sum_{i=1}^{m} \sum_{j=1}^{N_i} \Pi_{ij} x_{ij} \to \max$$

$$\frac{1}{\sum_{i=1}^{m}\sum_{j=1}^{N_i} x_{ij}} \sum_{i=1}^{m}\sum_{j=1}^{N_i} R_{ij} x_{ij} \le \rho$$

$$\sum_{i=1}^{m}\sum_{j=1}^{N_i} c_{ij} x_{ij} \le C + M$$

$$\frac{1}{\sum_{i=1}^{m}\sum_{j=1}^{N_i} c_{ij} x_{ij}} \sum_{i=1}^{m}\sum_{j=1}^{N_i} \Pi_{ij} x_{ij} \le r; x_{ij} \in \{1,0\}.$$

Исходные данные для задачи формирования инвестиционного портфеля предприятия приведены в Приложении 1.

Данные были взяты из практики работы предприятия Химзавод – филиала ФГУП «Красмаш» [56]. В Приложении 1 приведен список ЦФО, инновационные проекты, планируемые для внедрения, а также соответствующие им числовые данные: планируемые прибыльность, затраты и рискованность внедрения.

Ранее данная задача уже была решена соответствующей модификацией алгоритма PSO [15]. Известно, что оптимальное решение данной задачи достигается с бинарной строкой «101111111111111111110110001». Здесь единицы в бинарном представлении показывают, в проекты с какими номерами следует инвестировать деньги, а нули – в какие не следует инвестировать деньги, исходя из условия получения максимальной прибыли инвестором при выполнении ограничений задачи. Таким образом, во все проекты, кроме 2-ого (организация совместной деятельности по производству посуды из полимеров), 19-ого (производство сжиженного и газообразного кислорода), 22-ого (производство пропиленовых мешков), 23-ого (производство дифференциального редуктора) и 24-ого (производство электромагнитного инжектора), следует инвестировать деньги. В этом случае прибыль составит 117.7 млн. рублей.

Разработанной бинарной модификацией алгоритма COBRA-с указанный результат достигался при каждом запуске программы (всего их было совершено 100).

Задача формирования оптимального кредитного портфеля банка состоит в формировании оптимального кредитного портфеля с жесткими ограничениями по

суммам имеющихся в наличии свободных кредитных ресурсов, их стоимости, процентным ставкам на выдаваемые кредиты, срокам привлечения ресурсов, максимальному размеру кредита на одного заемщика.

Для формализованной записи критерия получения максимальной доходности от проводимых банком кредитных операций при соблюдении требования минимизации риска не возврата введем следующие обозначения:

1. F – сумма свободных пассивов, которыми располагает банк в данный момент времени;
2. N – количество заемщиков;
3. k_j – сумма кредита, запрашиваемая j-ым заемщиком, $j = \overline{1, N}$;
4. t_j – срок, на который j-ый заемщик берет кредит;
5. x_j – булева переменная, принимающая значения: 1, если кредит k_j выдается, и 0, если заявка на получение кредита отклоняется банком;
6. d_j – проценты за пользование j-ым кредитом (в данной постановке предполагается, что проценты выплачиваются единовременно с возвратом самого кредита);
7. P_j – вероятность невыполнения заемщиком обязательств по возврату кредита и процентов по нему $k_j \cdot (1 + d_j)$. В предлагаемой постановке задачи предполагается два варианта обслуживания долга заемщиком: 100% возврат суммы кредита и процентов по нему в установленный срок, либо полное отсутствие платежей в погашение кредита и процентов по нему;
8. ρ – ограничение на суммарную рискованность кредитного портфеля.

Ожидаемые проценты от комбинации кредитных заявок будут определяться по следующей формуле:

$$E(x) = \sum_{j=1}^{N} k_j \cdot (1 + d_j \cdot t_j) \cdot x_j .$$

Суммарная рискованность кредитного портфеля может быть определена по следующей формуле:

$$R(x) = \frac{1}{\sum_{j=1}^{N} x_j} \cdot \sum_{j=1}^{N} P_j x_j$$

Таким образом, постановка задачи формирования оптимального кредитного портфеля банка выглядит следующим образом:

$$E(x) \to \max$$

$$R(x) \le \rho$$

$$\sum_{j=1}^{N} k_j x_j \le F,\ x_j \in \{0,1\}.$$

Исходные данные для задачи формирования кредитного портфеля банка приведены в Приложении 2. Данные предоставлены В. А. Пуртиковым (Красноярский филиал банка Москвы) [50].

Точный ответ для данной задачи не был заранее известен в силу ее громоздкости. Бинарная модификация коллективного алгоритма условной оптимизации COBRA-с запускалась для ее решения также 100 раз. Результаты, полученные для обеих задач, представлены в Таблице 2.19.

Таблица 2.19. Результаты решения двух практических задач

Лучший рез-тат	*Худший рез-тат*	*Средний рез-тат*	*СКО*	*ППД*
177.7	177.7	177.7	0	100
4.35625e+009	4.34179e+009	4.35403e+009	5.08972e+006	100

Результаты для второй задачи (формирования оптимального кредитного портфеля банка) лучше полученных ее автором [50].

Таким образом, разработанный алгоритм условной оптимизации продемонстрировал свою работоспособность на двух практических задачах инвестиционного анализа.

Выводы

Выводы, полученные в первой главе, послужили предпосылками для разработки нового бионического метода безусловной оптимизации с вещественными переменными.

Пять алгоритмов роевого интеллекта, описанных ранее, были взяты за основу разработанной эвристики Co-Operation of Biology Related Algorithms (COBRA): они были использованы как компоненты нового самонастраивающегося метода [60]. Главное преимущество алгоритма COBRA заключается в том, что нет необходимости выбирать размер популяции для каждого бионического метода. Настройка данного параметра происходит автоматически в ходе работы разработанного алгоритма. Исследование эффективности эвристики COBRA на множестве различных задач безусловной оптимизации показало целесообразность его использования и работоспособность. Кроме того, было установлено, что в целом новый подход COBRA превосходит свои компоненты-алгоритмы при решении этих задач, что становится очевиднее с усложнением задач и увеличением размерности пространства поиска.

Далее метод COBRA был модифицирован для решения задач условной оптимизации с вещественными переменными. Разработанная модификация была названа COBRA-c. Для ее реализации были использованы три технологии учета ограничений: динамические штрафы, правила Дэба, а также метод, использующий турнирную селекцию. Исследования показали, что эвристика COBRA-c работает эффективнее своих компонент-алгоритмов. Также тестирование алгоритма проводилось на задачах, взятых с конкурса CEC 2010, и было установлено, что разработанный алгоритм превосходит по результатам некоторых победителей данного конкурса. Таким образом, работоспособность алгоритма COBRA-c и целесообразность его применения вместо своих компонент, были доказаны.

Метод COBRA был модифицирован для решения задач безусловной и условной оптимизации с бинарными переменными, новый метод был назван

COBRA-b. Для бинаризации алгоритма применялась технология, описанная в работах Кеннеди и Эберхарта. Тестирование алгоритма также продемонстрировало его эффективность. Кроме того, им были решены две практические задачи: задача формирования оптимального инвестиционного портфеля предприятия и задача формирования оптимального кредитного портфеля банка.

В итоге, были разработаны и исследованы коллективные алгоритма решения задач условной и безусловной оптимизации, как с вещественными, так и с бинарными переменными, превосходящие по эффективности свои алгоритмы-компоненты, демонстрирующие высокую работоспособность на задачах различного уровня сложности.

Следующая глава посвящена разработке алгоритмического обеспечения для автоматизации проектирования информационных технологий интеллектуального анализа данных, а именно искусственных нейронных сетей и машин опорных векторов и их коллективов, при помощи предложенных коллективных методов оптимизации.

3 Алгоритмическое обеспечение автоматизации проектирования информационных технологий интеллектуального анализа данных

Информационные технологии интеллектуального анализа данных – методы, используемые для обозначения совокупности алгоритмов обнаружения в данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. Задачи интеллектуального анализа данных также называют задачами Data Mining [43].

Основу информационных технологий интеллектуального анализа данных составляют всевозможные методы классификации, моделирования и прогнозирования, основанные на применении деревьев решений [121], искусственных нейронных сетей (ИНС) [114, 120], генетического программирования [66], ассоциативной памяти [45], нечёткой логики [142] и т.д. К информационным технологиям интеллектуального анализа данных также нередко относят статистические методы, например дескриптивный анализ, корреляционный и регрессионный анализ, факторный анализ, дисперсионный анализ, компонентный анализ, анализ временных рядов, анализ выживаемости, анализ связей. Такие методы, однако, предполагают некоторые априорные представления об анализируемых данных, что несколько расходится с целью обнаружения ранее неизвестных нетривиальных и практически полезных знаний из набора данных.

Постановка задачи интеллектуального анализа данных выглядит следующим образом: имеется достаточно крупная база данных и предполагается, что в этой базе данных находятся некие «скрытые знания». Таким образом, необходимо разработать алгоритмы обнаружения знаний, скрытых в больших объёмах исходных данных. В текущих условиях глобальной конкуренции именно найденные закономерности (знания) могут быть источником дополнительного конкурентного преимущества.

Знания называются скрытыми, если они соответствуют определениям:

1. Ранее неизвестные, то есть такие знания, которые должны быть новыми (а не подтверждающими какие-то ранее полученные сведения);
2. Нетривиальные, то есть такие, которые нельзя просто так увидеть (при непосредственном визуальном анализе данных или при вычислении простых статистических характеристик);
3. Практически полезные, то есть такие знания, которые представляют ценность для исследователя или потребителя;
4. Доступные для интерпретации, то есть такие знания, которые легко представить в наглядной для пользователя форме и легко объяснить в терминах предметной области.

Эти требования во многом определяют суть информационных технологий интеллектуального анализа данных и то, в каком виде и в каком соотношении в этих технологиях используются системы управления базами данных, статистические методы анализа и методы искусственного интеллекта.

Задачи, решаемые информационными технологиями интеллектуального анализа данных, принято разделять на описательные и предсказательные. В данной работе рассматриваются только предсказательные задачи, на первом плане которых стоит вопрос о предсказании для тех случаев, для которых данных ещё нет. К предсказательным задачам относятся: классификация объектов (для заранее заданных классов) [78], регрессионный анализ, анализ временных рядов, прогнозирование [27].

Задача классификации – формализованная задача, в которой имеется множество объектов (ситуаций), разделённых некоторым образом на классы [78]. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется выборкой. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества. Классифицировать объект – значит, указать номер (или наименование) класса, к которому относится данный объект.

В математической статистике задачи классификации называются также задачами дискриминантного анализа. В машинном обучении задача классификации решается, как правило, с помощью методов искусственных нейронных сетей [114], метода опорных векторов и т.д. при постановке эксперимента в виде обучения с учителем [1]. Существует также множество других методов решения задач классификации, например, наивный байесовский классификатор, метод ближайших соседей, логистическая регрессия и т.д.

Прогнозирование – разработка прогноза, т. е. специальное научное исследование перспектив (прошлых тенденций) развития каких-либо явлений (технических, социально-экономических). Методология прогнозирования связана с планированием и моделированием. Можно считать, что прогнозирование является чуть ли не основной целью и задачей большого числа специалистов, занимающихся анализом данных.

Современные методы статистического прогнозирования позволяют с высокой точностью прогнозировать практически все возможные показатели. Однако надо помнить, что не существует универсальных методов прогнозирования на все случаи жизни. Выбор метода прогнозирования и его эффективность зависят от многих условий, и в частности от требуемой длины или времени прогнозирования.

По времени прогнозирования различают краткосрочный, среднесрочный и долгосрочный прогноз. Краткосрочный прогноз характеризует собой прогноз «на завтра», то есть прогноз на несколько шагов вперед. Для него применяют практически все известные методы, например, экспоненциальное сглаживание [39], нейронные сети и т.д. Среднесрочный прогноз – это обычно прогноз на один или на половину сезонного цикла. Для него используют АРПСС [28] и экспоненциальное сглаживание, которые позволяют отслеживать качество прогноза в зависимости от срока прогноза. А при построении долгосрочного прогноза стандартные статистические методы прогнозирования практически не используют, и требуется использование комплексных подходов. Например, использование нейронных сетей или регрессионных моделей.

После построения любой модели прогнозирования важно проверять, насколько адекватно она построена. Для этого можно, во-первых, провести визуальный анализ со сдвигом прогноза на несколько шагов назад. А во-вторых, воспользоваться анализом остатков – стандартным методом проверки адекватности любой построенной статистической модели.

В данной работе для решения различных задач классификации и прогнозирования использовались искусственные нейронные сети (Artificial Neural Networks, ANN) [114], а также машины опорных векторов (Support Vector Machines, SVM) [30]. Для их проектирования и настройки были применены разработанные и описанные ранее бионический метод оптимизации COBRA и две ее модификации COBRA-b и COBRA-c.

3.1. Нейросетевые классификаторы, генерируемые коллективными алгоритмами

Искусственная нейронная сеть (ИНС) – математическая модель, а также её программная реализация, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма [70]. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети У. Маккалока и У. Питтса [49]. Нейронные сети применяются для решения различных практических задач интеллектуального анализа данных: задач прогнозирования, задач распознавания образов, задач управления и др.

Искусственные нейронные сети представляют собой систему соединённых и взаимодействующих между собой простых процессоров – искусственных нейронов. Каждый нейрон подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим нейронам. И, тем не менее, будучи соединёнными в достаточно большую

сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

С точки зрения машинного обучения нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т.д. С математической точки зрения обучение нейронных сетей – это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники и программирования нейронная сеть – способ решения проблемы эффективного параллелизма [38].

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения – одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искаженных данных.

Чтобы отразить суть нейронных сетей, определение искусственного нейрона дается следующим образом:

1) Нейрон получает входные сигналы (исходные данные либо выходные сигналы других нейронов сети) через несколько входных каналов. Каждый входной сигнал проходит через соединение, имеющее определенный вес. Вычисляется взвешенная сумма входов и в результате получается величина активации нейрона.
2) Сигнал активации преобразуется с помощью функции активации и в результате получается выходной сигнал нейрона.

Нейроны сети соединены друг с другом. Если сеть предполагается для чего-то использовать, то у нее должны быть входы (нейроны, принимающие входные

данные, образуют, соответственно, входной слой) и выходы (прогнозы или управляющие сигналы). Кроме этого, в сети может быть еще много промежуточных (скрытых) слоев нейронов, выполняющих внутренние функции. Входные, скрытые и выходные нейроны должны быть связаны между собой.

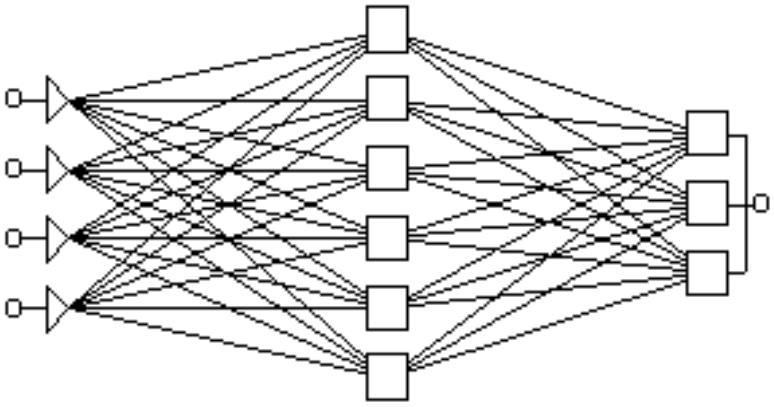

Рисунок 3.1. Пример сети с прямой передачей сигнала

Простейшая сеть имеет структуру прямой передачи сигнала, то есть сигналы проходят от входов через скрытые элементы и, в конце концов, приходят на выходные элементы. Такая структура имеет устойчивое поведение. Типичный пример сети с прямой передачей сигнала показан на рисунке 3.1.

Нейроны регулярным образом организованы в слои. Входной слой служит просто для ввода значений входных переменных. Каждый из скрытых и выходных нейронов соединен со всеми элементами предыдущего слоя. Сеть с подобной организацией может моделировать функцию практически любой степени сложности, причем число слоев и число элементов в каждом слое определяют сложность функции. Определение числа промежуточных слоев и числа элементов на них является важным вопросом при конструировании нейронных сетей.

Таким образом, для решения той или иной задачи при помощи нейронных сетей нужно выбрать ее структуру (количество скрытых слоев, количество нейронов на каждом слое, тип активационной функции на каждом нейроне), а затем для полученной сети настроить весовые коэффициенты связи нейронов.

Стоит отметить, что количество весовых коэффициентов зависит от числа входных параметров заданной задачи и числа нейронов сети.

Итак, для настройки нейронной сети необходимо решить две задачи безусловной оптимизации: одна задача связана с выбором структуры сети, а другая – подбором весовых коэффициентов [61].

В данной работе для нейронной сети генерировалось множество структур, причем каждая структура была представлена в виде бинарной строки. Длина бинарной строки определялась следующим образом:

1. Устанавливалось максимальное число скрытых слоев для сети (в данной работе – 5 слоев).
2. Устанавливалось максимальное число нейронов на каждом скрытом слое нейронной сети (в данной работе 5 нейронов на слое).
3. Каждый нейрон кодировался бинарной строкой заданной размерности. Длина бинарной строки для нейрона определялась количеством активационных функций (в данной работе рассматривались 15 функций, поэтому каждый нейрон кодировался бинарной строкой, длина которой была равна 4).

Как уже было сказано, в данной работе использовались 15 активационных функций, среди которых были тангенс гиперболический, сигмоидальная, линейная функция, пороговая функция, и т.д. Ниже представлен полный список использованных активационных функций:

1. $f(x)=\dfrac{1}{1+\exp(-x)}$;
2. $f(x)=1$;
3. $f(x)=th(x)$;
4. $f(x)=\exp\left(\dfrac{-x^2}{2}\right)$;
5. $f(x)=1-\exp\left(\dfrac{-x^2}{2}\right)$;
6. $f(x)=x^2$;

7. $f(x)=x^2$;
8. $f(x)=\sin(x)$;
9. $f(x)=\exp(x)$;
10. $f(x)=|x|$;
11. $f(x)=\begin{cases}-1, x<-1\\ x, -1\le x\le 1;\\ 1, x>1\end{cases}$
12. $f(x)=\begin{cases}0, x<-0.5\\ x+0.5, -0.5\le x\le 0.5;\\ 1, x>0.5\end{cases}$
13. $f(x)=\frac{2}{1+\exp(x)}-1$;
14. $f(x)=\frac{1}{x}$;
15. $f(x)=sign(x)$.

Если нейрон закодирован бинарной строкой вида «0000», значит, на заданном скрытом слое его не было. Для того чтобы определить, какая именно активационная функция закреплена за заданным нейроном, необходимо двоичное число, которым он кодировался, перевести в десятичное число. Последнее определяло номер активационной функции нейрона согласно приведенному выше списку.

Таким образом, в данной работе каждая структура нейронной сети представлялась бинарной строкой длиной $5\times 5\times 4=100$. То есть проблему выбора структуры нейронной сети можно рассматривать как задачу безусловной оптимизации с бинарными переменными, и размерность задачи в данной работе была равна 100. Для решения данной задачи (выбора структуры нейронной сети) был использован один из разработанных оптимизационных методов, а именно алгоритм COBRA-b [44].

В итоге при генерировании нейронной сети алгоритмом COBRA-b сначала инициализировалась популяция структур в виде заданного множества бинарных

строк. Для каждой структуры подбирались весовые коэффициенты, затем выполнялась «основная» работа эвристики, и в конечном итоге выбиралась та структура, которая решала бы задачу интеллектуального анализа данных, допустим задачу классификации, лучше всего.

Для каждой нейронной сети подбирались весовые коэффициенты в зависимости от полученной для нее структуры. Именно на этом этапе и происходит обучение сети. По своей сути настройка весовых коэффициентов нейронной сети – это задача безусловной оптимизации с вещественными переменными. Одним из наиболее известных и часто используемых методов «обучения» сети является алгоритм обратного распространения [36]. Однако в настоящее время разрабатывается все больше новых методов настройки весов ИНС, например, в статье [123] описан один из них. В данной работе для подбора весовых коэффициентов нейронной сети был применен один из разработанных методов, а именно алгоритм COBRA [23, 22].

Таким образом, после того как алгоритмом COBRA-b будет сгенерирована популяция структур (множество бинарных строк), для каждой структуры алгоритмом COBRA подбираются весовые коэффициенты. Затем выполняются остальные этапы метода COBRA-b. В итоге определяется наилучшая структура нейронной сети и для этой структуры вновь выполняется настройка весовых коэффициентов разработанной эвристикой COBRA.

3.2. Генерирование машин опорных векторов бионическими алгоритмами

Метод опорных векторов (Support Vector Machine, SVM) – набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа [30]. Принадлежит к семейству линейных классификаторов. Особым свойством метода опорных векторов является непрерывное уменьшение эмпирической ошибки.

Основная идея метода опорных векторов заключается в переводе исходных векторов в пространство более высокой размерности и поиске разделяющей

гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей данные из разных классов. Разделяющей гиперплоскостью будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей.

Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.

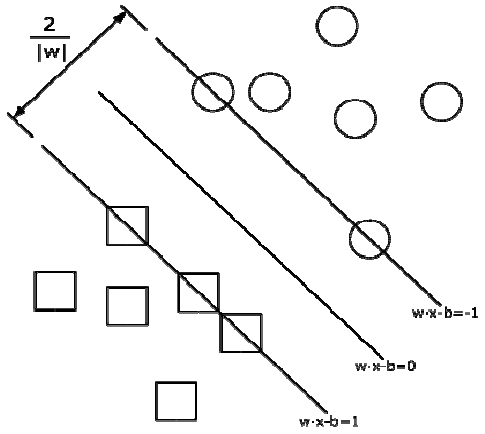

Рисунок 3.2. Оптимальная разделяющая гиперплоскость для метода опорных векторов, построенная на точках из двух классов

Часто в алгоритмах машинного обучения возникает необходимость классифицировать данные. Предположим, что данные поделены на два класса. Каждый объект данных представлен как вектор (точка) в p-мерном пространстве (последовательность p чисел). Каждая из этих точек принадлежит только одному из двух классов. Вопрос заключается в том, что неизвестно, возможно ли разделить точки гиперплоскостью размерностью ($p-1$). Это типичный случай линейной разделимости данных из разных классов (рисунок 3.2).

Таких гиперплоскостей может быть много. Поэтому вполне естественно полагать, что максимизация зазора между классами способствует более уверенной классификации. То есть главная задача – найти такую гиперплоскость, чтобы расстояние от неё до ближайшей точки было максимальным (то есть такую, чтобы

расстояние между двумя ближайшими точками, лежащими по разные стороны гиперплоскости, было максимально).

Такая гиперплоскость называется оптимальной разделяющей гиперплоскостью, а соответствующий ей линейный классификатор называется оптимально разделяющим классификатором.

Для большего понимания работы алгоритма опишем формально решаемую им задачу бинарной классификации. Пусть дан набор данных:

$$X^l = \{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\},\ x_i \in R^m, y_i \in \{-1;1\},$$

где $y_i \in \{-1;1\}$ означает, что точка x_i принадлежит либо классу -1, либо классу 1; l – это количество объектов в обучающей выборке, x_i – это m-мерный вещественный вектор, обычно нормализованный значениями [0;1] или [-1;1]. Если точки не будут нормализованы, то точка с большими отклонениями от средних значений координат точек слишком сильно повлияет на классификатор. Набор X^l рассматривается как обучающая выборка, в которой для каждого элемента уже задан класс, к которому он принадлежит. Необходимо, чтобы алгоритм метода опорных векторов классифицировал их таким же образом. Для этого строится разделяющую гиперплоскость, которая имеет вид [30]:

$$\omega \cdot x - b = 0,$$

где ω – вектор-перпендикуляр к разделяющей гиперплоскости. Параметр b равен по модулю расстоянию от гиперплоскости до начала координат. Если параметр b равен нулю, гиперплоскость проходит через начало координат, что ограничивает решение.

Так как нас интересует оптимальное разделение данных разных классов, то в дальнейшем будем рассматривать опорные вектора и гиперплоскости, параллельные оптимальной гиперплоскости и ближайшие к опорным векторам двух классов. Установлено, что эти параллельные гиперплоскости могут быть описаны следующими уравнениями (с точностью до нормировки): $\omega \cdot x - b = 1$ и $\omega \cdot x - b = -1$.

В самом общем виде метод опорных векторов [30] заключается в том, что необходимо построить гиперплоскость:

$$\omega \cdot x - b = 0,\ \omega \in R^m, b \in R.$$

Если обучающая выборка линейно разделима, то мы можем выбрать гиперплоскости таким образом, чтобы между ними не лежала ни одна точка обучающей выборки и затем максимизировать расстояние между гиперплоскостями. Ширину полосы между ними легко найти из соображений геометрии, она равна $\frac{2}{\|\omega\|}$ [32], таким образом, задача сводится к минимизации $\|\omega\|$.

Чтобы исключить все точки из полосы, мы должны убедиться, что для всех i выполняется следующее:

$$\begin{bmatrix} \omega \cdot x_i - b \geq 1, y_i = 1 \\ \omega \cdot x_i - b \leq -1, y_i = -1 \end{bmatrix}.$$

Последнее может также быть записано в виде:

$$y_i(\omega \cdot x_i - b) \geq 1, i = \overline{1,l}.$$

Таким образом, проблема построения оптимальной разделяющей гиперплоскости сводится к задаче условной оптимизации, а именно минимизации. Это задача квадратичной оптимизации, которая имеет вид:

$$\|\omega\|^2 \to \min$$

$$y_i(\omega \cdot x_i - b) \geq 1, i = \overline{1,l}.$$

Выше был описан случай линейной разделимости данных двух различных классов. Однако, в большинстве случае такое разделение невозможно, и применение метода опорных векторов для линейно неразделимых наборов данных приводит к неудовлетворительным результатам.

Одним из решений данной проблемы является перенос данных-векторов в пространство большей размерности, и уже для данных в новом пространстве применяется метод опорных векторов. То есть основная идея этого подхода заключается в том, что исходные данные в пространстве большей размерности

будут линейно разделимы, поэтому и применение SVM-классификаторов целесообразно. Проще всего реализовать данную идею, применив функцию-ядро [71].

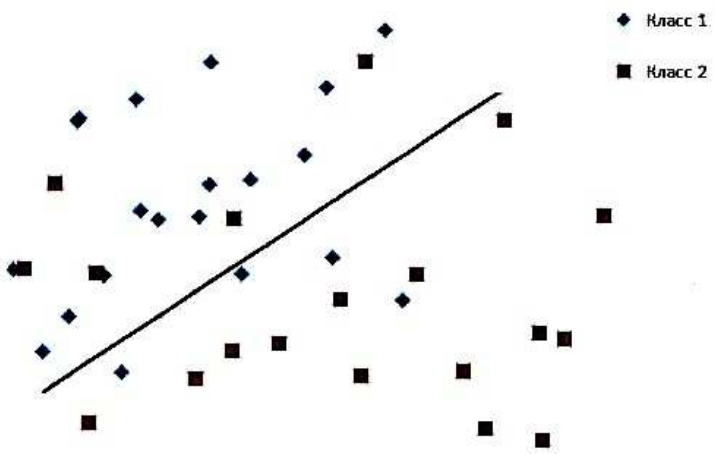

Рисунок 3.3. Случай линейной неразделимости классов

Результирующий алгоритм (после применения функции ядра) крайне похож на алгоритм линейной классификации, с той лишь разницей, что каждое скалярное произведение в приведённых выше формулах заменяется нелинейной функцией ядра (скалярным произведением в пространстве с большей размерностью). В этом пространстве уже может существовать оптимальная разделяющая гиперплоскость. Так как размерность получаемого пространства может быть больше размерности исходного пространства, то преобразование, сопоставляющее скалярные произведения, будет нелинейным, а значит функция, соответствующая в исходном пространстве оптимальной разделяющей гиперплоскости, будет также нелинейной.

Стоит отметить, что если исходное пространство имеет достаточно высокую размерность, то можно надеяться, что в нём выборка окажется линейно разделимой.

Вообще говоря, существует множество различных функций-ядер, например, полиномиальная функция, радиальная базисная, сигмоид, но для данной работы была использована полиномиальная функция

$$K(x,x')=(a(x,x')+\beta)^d,$$

где a,β,d – параметры функции-ядра $K(x,x')$. Далее было изменено правило классификации:

$$f(x)=\begin{cases}1,\left((K(\omega,x)+\beta)^d+b\right)\geq 1\\ -1,\left((K(\omega,x)+\beta)^d+b\right)\leq -1\end{cases}.$$

Тогда для построения гиперплоскости, наилучшим способом разделяющей объекты разных классов, решалась следующая задача условной оптимизации:

$$\|\omega\|^2\rightarrow \min$$

$$y_i\left((a(\omega,x_i)+\beta)^d+b\right)\geq 1, i=\overline{1,l}.$$

Таким образом, для решения задачи классификации методом опорных векторов, необходимо определить параметры функции-ядра a,β,d, вектор ω и параметр смещения b, иначе говоря, для каждого SVM-классификатора со своими параметрами функции-ядра a,β,d решить задачу условной оптимизации для построения разделяющей гиперплоскости, а затем уже из получившихся классификаторов выбрать лучший, то есть с минимальной ошибкой классификации.

Итак, для построения оптимальной гиперплоскости использовался разработанный ранее алгоритм COBRA-c [26], а для выбора параметров функции-ядра (решения задачи безусловной оптимизации) был применен метод COBRA [107].

3.3. Коллективы машин опорных векторов, сгенерированные бионическими алгоритмами

На практике решение задач интеллектуального анализа данных состоит в том, что производится выбор структуры и настройка параметров отдельной интеллектуальной информационной технологии (ИИТ), после чего лучшей найденной ИИТ уже осуществляется решение задачи. Однако часто оказывается, что желательно иметь некоторое множество информационных технологий

интеллектуального анализа данных и использовать их все одновременно согласно выбранной коллективной технологии. Такое множество ИИТ называют ансамблем [105]. Ансамбли информационных технологий могут состоять из нейронных сетей, машин опорных векторов, систем на нечеткой логике, деревьев принятия решений и т.д. [144].

Повышение качества решения задачи при объединении нескольких мнений в единое решение было замечено опытным путем достаточно давно. Общая теория композиций ИИТ была впервые предложена Ю. И. Журавлёвым в алгебраическом подходе к проблеме распознавания [41]. Например, отдельные информационные технологии интеллектуального анализа данных, а именно нейронные сети были объединены в коллектив уже в работе [92]. Было показано, что способность системы, использующей множество нейронных сетей, к обобщению может существенно повыситься за счет создания коллектива.

Однако на практике чаще всего применяются наиболее простые методы получения коллективного решения, такие как взвешенное голосование и комитетные системы [47]. Кроме того, в работах [52], [95], были предложены методы, основанные на идее областей компетентности.

Анализ существующих методов проектирования коллективов информационных технологий интеллектуального анализа данных [29] позволяет выделить следующие характеристики, индивидуализирующие конкретный метод:

- способ проектирования отдельных технологий;
- способ выбора технологий, формирующих коллектив;
- способ учета коллективом решений, получаемых от каждого его члена.

Выделяют два основных свойства коллективов информационных технологий интеллектуального анализа данных:

1. Коллективы объединяют различные информационные технологии, способные самостоятельно решать исходную задачу.

2. При создании коллектива отдельные информационные технологии воспринимаются как чёрные ящики, имеющие только две функции: обучение по заданной выборке и вычисление ответа для заданного примера.

При формировании ансамбля информационных технологий необходимо решать две задачи [55] – качественное обучение отдельной информационной технологии и выбор оптимального способа их объединения. Методы создания ансамблей делятся на два типа [69]:

1. Методы, обучающие членов коллектива независимо друг от друга и усредняющие решения (например, комитеты большинства, бустинг [88], бэггинг [72], монотонная коррекция [34]);

2. Методы, изменяющие распределение обучающих примеров в зависимости от результатов обучения других информационных технологий, тем самым упрощающие обучение каждой отдельной информационной технологии (например, комитеты старшинства [48] и смеси экспертов [98]).

Основные стратегии создания коллективов.

1. При последовательной оптимизации примеры обучающей выборки перераспределяются в зависимости от результатов обучения предыдущих информационных технологий интеллектуального анализа данных. Последующие ИИТ стремятся компенсировать ошибки предыдущих.

2. При параллельной оптимизации отдельные информационные технологии интеллектуального анализа данных обучаются независимо друг от друга.

3. При глобальной оптимизации одновременно настраиваются все информационные технологии интеллектуального анализа данных (при построении смесей экспертов отдельные ИИТ и оценки их эффективности пересчитываются с помощью итерационного процесса [98] или с помощью генетического алгоритма оптимизируются подмножества объектов и признаков, на которых настраиваются отдельные ИИТ [35]).

4. Применение алгебраического подхода, описанного в [40], [42], позволяет строить корректные алгоритмические композиции, не прибегая к оптимизации. Однако этот подход малоприменим на практике, так как склонен к переобучению.

На практике чаще используются алгебраические подходы, основанные на стратегии последовательной оптимизации [33].

Коллективный подход был успешно использован для решения широкого круга практических задач: задач распознавания образов, медицинской диагностики, классификации сейсмических сигналов, анализа частоты наводнений и многих других.

На практике метод голосования, а затем метод усреднения, а также его модификация, учитывающая ошибки каждого из членов коллектива [92], стали почти каноническими и используются практически во всех работах, посвященных данной тематике. В данной работе мы будем называть их схемами «усреднение» и «взвешенное усреднение» соответственно.

Схему «усреднение» можно описать следующим образом: выходные значения отдельных информационных технологий интеллектуального анализа данных складываются и делятся на число членов коллектива. Для схемы «взвешенное усреднение» учитывается «оценка компетентности» каждой информационной технологии. Всем членам коллектива назначаются веса на основании ошибки на обучающей выборке, вес обратно пропорционален ошибке [115]. Выходное значение ИИТ умножается на ее весовой коэффициент, после чего все полученные значения складываются и делятся на их число.

Помимо двух упомянутых схем в данной работе использовались еще две новые схемы коллективного принятия решений, которые будем называть «схема 1» и «схема 2» соответственно [64].

Для обеих схем данные делились случайным образом на три группы: обучающая выборка, тестовая выборка для отдельных информационных технологий интеллектуального анализа данных и тестовая выборка для коллектива. Сначала каждый член коллектива обучался, потом тестировался (использовались первые две выборки), оценивалась эффективность каждой информационной технологии коллектива по некоторому заданному критерию.

Все объекты (точки) из первых двух выборок назывались «старыми», а объекты (точки) из тестовой выборки для коллектива – «новыми»,

соответственно. Предположим, что решалась задача классификации. Для каждой «новой» точки находилась ближайшая к ней «старая», а затем определялся член коллектива, правильно классифицировавший найденную «старую» точку.

«Схемой 1» принятие решения коллективом осуществлялось следующим образом:

1. Если только один член коллектива правильно классифицировал «старую» точку, то он же использовался для классификации «новой» точки;

2. Во всех остальных случаях (ни один член коллектива не классифицировал «старую» точку правильно, несколько членов коллектива классифицировали правильно). Для классификации «новой» точки использовалась ИИТ, продемонстрировавшая наибольшую эффективность в целом.

«Схемой 2» принятие решения коллективом осуществлялось следующим образом:

1. Если только один член коллектива правильно классифицировал «старую» точку, то он же использовался для классификации «новой» точки;

2. Если ни один член коллектива не классифицировал «старую» точку правильно, то для классификации «новой» точки использовалась ИИТ, продемонстрировавшая наибольшую эффективность в целом;

3. Если несколько членов коллектива классифицировали «старую» точку правильно, то для классификации «новой» точки использовалась ИИТ, которая была наиболее близка к правильному ответу, что определялось по специально введенному критерию.

Итак, были реализованы коллективы информационных технологий интеллектуального анализа данных со схемами принятия решений «усреднение», «взвешенное усреднение» и двумя новыми схемами.

Выводы

В данной главе были разработаны и реализованы в виде программных систем следующие алгоритмы:

1. Алгоритм автоматического проектирования нейронных сетей;

2. Алгоритм автоматического генерирования машин опорных векторов;

3. Алгоритм автоматического генерирования коллективов информационных технологий интеллектуального анализа данных.

Для реализации перечисленных алгоритмов были применены разработанные ранее коллективные бионические методы оптимизации, а именно эвристика COBRA и две ее модификации COBRA-c и COBRA-b.

Практическую эффективность алгоритмов следует подтвердить в ходе решения реальных задач интеллектуального анализа данных.

4 Практическое применение информационных технологий интеллектуального анализа данных, автоматически генерируемых коллективными алгоритмами оптимизации

4.1. Решение задач распознавания машинами опорных векторов

В данном разделе мы рассмотрим две задачи распознавания, причем одна из них является задачей распознавания образов, а другая – задачей распознавания речи.

Распознавание речи является актуальной и сложной проблемой, состоящей из комплекса различных задач. Несмотря на значительные разработки в этой области, на настоящий момент не существует достаточно эффективной системы распознавания, независящей от языка речи и ее произношения неким диктором.

Большинство существующих систем распознавания речи, например скрытые Марковские модели [51] или нейронные сети с временной задержкой [37], разрабатывались с целью распознавания сигналов и на самом деле не используют информацию о специфике речи. С другой стороны аналитическими системами учитываются артикуляторный процесс, приводящий к тому, что фонемы языка воспринимаются различным образом. Для подобных систем предлагаются способы уменьшения влияния любых фонетических особенностей акустических наблюдений.

Главная проблема применения аналитических систем заключается в сложности получения достоверных акустических параметров. Акустические измерения должны удовлетворять следующим свойствам:

1. Содержание информации, связанной с рассматриваемой фонетической особенностью;

2. Независимость от диктора;

3. Независимость от содержания;

4. Устойчивость к шумам.

В основном акустические наблюдения сопровождаются учетом множества различных сведений, поэтому методы классификации не могут быть применены напрямую.

В ходе выполнения диссертационной работы была решена задача распознавания речи Phoneme, данные для которой представлены в международном репозитории KEEL [65].

Задача состояла в классификации носовых (первый класс) и ротовых (второй класс) гласных. Упомянутая база данных содержит гласные, полученные из 1809 слогов, например, pa, ta, pan и т.д. Пять различных вещественных признаков, выбранные для описания каждой гласной, представлены в Таблице 4.1.

Таблица 4.1. Описание признаков для задачи Phoneme

Признак	***Домен***
Aa	[-1.7, 4.107]
Ao	[-1.327, 4.378]
Dcl	[-1.823, 3.199]
Iy	[-1.581, 2.826]
Sh	[-1.284, 2.719]
Class	{0, 1}

Описание же самой задачи приведено в Таблице 4.2.

Таблица 4.2. Описание признаков для задачи Phoneme

Тип задачи	Классификация	***Тип данных***	Вещественные
Признаки	5	***Вещественные / Целые / Номинальные***	5 / 0 / 0
Всего данных	5404	***Классы***	2
Наличие неполных сведений			Нет

Вторая задача распознавания, Segment(0), решенная в ходе выполнения диссертационной работы, была связана с сегментацией изображений.

В компьютерном зрении, сегментация – это процесс разделения цифрового изображения на несколько сегментов (множество пикселей, также называемых суперпикселями). Цель сегментации заключается в упрощении и/или изменении представления изображения, чтобы его было проще и легче анализировать [126]. Сегментация изображений обычно используется для того, чтобы выделить

объекты и границы (линии, кривые, и т. д.) на изображениях. Более точно, сегментация изображений – это процесс присвоения таких меток каждому пикселю изображения, что пиксели с одинаковыми метками имеют общие визуальные характеристики.

Результатом сегментации изображения является множество сегментов, которые вместе покрывают всё изображение, или множество контуров, выделенных из изображения. Все пиксели в сегменте похожи по некоторой характеристике или вычисленному свойству, например по цвету, яркости или текстуре. Соседние сегменты значительно отличаются по этой характеристике.

Данные для задачи Segment(0) были также взяты из репозитория KEEL [65]. В исходном виде для нее были определены 7 классов, однако мы будем делить объекты на два класса – первый и все остальные. Подробное описание задачи и признаков для данных представлены в Таблице 4.3 и Таблице 4.4.

Таблица 4.3. Описание признаков для задачи Segment(0)

Тип задачи	Классификация	***Тип данных***	Вещественные
Признаки	19	***Вещественные / Целые / Номинальные***	19 / 0 / 0
Всего данных	2308	***Классы (в итоге)***	2
1 класс (%)	14.25	***Остальные классы (%)***	85.75
Наличие неполных сведений			Нет

Таблица 4.4. Описание признаков для задачи Segment(0)

Признак	***Домен***	***Признак***	***Домен***
Region-centroid-col	[1.0, 254.0]	Rawred-mean	[0.0, 137.11111]
Region-centroid-row	[11.0, 251.0]	Rawblue-mean	[0.0, 150.88889]
Region-pixel-count	[9.0, 10.0]	Rawgreen-mean	[0.0, 142.55556]
Short-line-density-5	[0.0, 0.33333334]	Exred-mean	[-49.666668, 9.888889]
Short-line-density-2	[0.0, 0.22222222]	Exblue-mean	[-12.444445, 82.0]
Vedge-mean	[0.0, 29.222221]	Exgreen-mean	[-33.88889, 24.666666]
Vedge-sd	[0.0, 991.7184]	Value-mean	[0.0, 150.88889]
Hedge-mean	[0.0, 44.722225]	Saturatoin-mean	[0.0, 1.0]
Hedge-sd	[-1.5894573E-8, 1386.3292]	Hue-mean	[-3.0441751, 2.9124804]
Intensity-mean	[0.0, 143.44444]	Class	{positive, negative}

Описанные задачи были решены машинами опорных векторов, настройка которых осуществлялась разработанным ранее методом оптимизации COBRA. Максимальное число вычислений для настройки SVM-классификаторов было установлено равным 100000, что для данных задач считается небольшим.

Всего программа запускалась для каждой задачи 20 раз, далее для каждой задачи определялись машины опорных векторов, продемонстрировавшие лучший и худший результаты классификации соответственно. Критерием оценки результатов была ошибка классификации, чем меньше она была, тем лучше получался результат. В итоге результаты, собранные для каждой задачи, усреднялись по 20 прогонам. Также стоит отметить, что для решения задач распознавания применялась кросс-валидация.

Полученные результаты (доля правильно классифицированных данных), представлены в Таблице 4.5.

Таблица 4.5. Результаты, полученные для задач Phoneme и Segment(0)

Задача	*Лучший результат*	*Худший результат*	*Среднее значение*
Phoneme	1.000	0.991	0.995
Segment(0)	0.816	0.761	0.787

Помимо машин опорных векторов, генерируемых разработанной эвристикой COBRA, данные задачи решались системами на нечеткой логике [130] и нейронными сетями [103], настраиваемыми генетическими алгоритмами, причем число итераций и прогонов для них было таким же. Предложенные методики анализа данных демонстрировали лучшие результаты, чем упомянутые нейронные сети, и идентичные результатам систем на нечеткой логике. Кроме того, результаты альтернативных методов анализа данных, найденные в научной литературе, показали, что, несмотря на гораздо большее количество вычислительных ресурсов, ошибка классификации уменьшалась незначительно.

Таким образом, результаты, представленные в Таблице 4.5, доказывают работоспособность разработанных алгоритмов. Кроме того, исследования показали, что разработанная информационная технология интеллектуального анализа данных успешнее справляется с подобного рода задачами при равных

условиях, чем альтернативные методы классификации, что подтверждает целесообразность ее применения.

4.2. Решение задач банковского скоринга и медицинской диагностики

Разработанные алгоритмы проектирования информационных технологий интеллектуального анализа данных, а именно нейронных сетей и машин опорных векторов, были протестированы на задачах: двух задачах банковского скоринга и двух задачах медицинской диагностики [87]. Далее приведены решенные задачи:

1. Задача банковского скоринга в Австралии (Scoring in Australia или Australia);
2. Задача банковского скоринга в Германии (Scoring in Germany или Germany);
3. Задача определения типа (злокачественная, доброкачественная) раковой опухоли груди (Breast Cancer Wisconsin);
4. Задача определения наличия заболевания диабетом (Pima Indians Diabetes).

Выбор этих классификационных задач был обусловлен тем фактом, что они уже решались многими исследователями различными методами, поэтому с их помощью возможно не только протестировать классификаторы, но также и сравнить полученные ими результаты с уже известными результатами альтернативных алгоритмов [94].

Для начала были решены задачи банковского скоринга: австралийская и немецкая. Задача скоринга в Австралии имеет следующие характеристики: 14 атрибутов (входных параметров), из которых 6 численных и 8 категориальных, 2 класса («кредитоспособный клиент» и «некредитоспособный клиент»), причем выборка содержала сведения о 307 кредитоспособных клиентах и 383 о некредитоспособных клиентах. А задача скоринга в Германии имеет следующие характеристики: 20 атрибутов, из которых 13 качественных и 7 численных, 2 класса («кредитоспособный клиент» и «некредитоспособный клиент»), и выборка

содержала сведения о 700 кредитоспособных клиентах и 300 о некредитоспособных клиентах. Обе задачи с описанием и набором данных были взяты из [87].

Сначала эти задачи были решены нейронными сетями с заранее заданной структурой и настраиваемыми весовыми коэффициентами алгоритмом COBRA. Были выбраны следующие структуры [62, 20, 59, 3]:

1. Один скрытый слой с тремя нейронами, для всех нейронов использовалась одна и та же функция активации – сигмоид;
2. Один скрытый слой с пятью нейронами, для всех нейронов использовалась одна и та же функция активации – сигмоид;
3. Пять скрытых слоев, пять нейронов на каждом слое, для всех нейронов функция активации – сигмоид.

Задачи решались с помощью машин опорных векторов, полученного способом, описанным в предыдущей главе, а также нейронными сетями с настраиваемыми структурой и весовыми коэффициентами.

Таким образом, применение нейронных сетей с подбором структуры сводилось к решению двух оптимизационных задач: задача безусловной оптимизации с бинарными переменными размерности 100 и задача безусловной оптимизации с вещественными переменными, размерность которой менялась от 175 до 225. Для конечной настройки весовых коэффициентов (для лучшей найденной структуры) максимальное число вычислений целевой функции было установлено равным 10000. Полученные результаты (доля правильно классифицированных данных) представлены в Таблице 4.6. В Таблице использованы следующие обозначения:

1. ANN+COBRA (5x5) – искусственная нейронная сеть с 5 скрытыми слоями и 5 нейронами на каждом слое, на всех нейронах действует сигмоидальная функция активации, весовые коэффициенты настраиваются разработанным алгоритмом COBRA;

2. ANN+COBRA (5) – искусственная нейронная сеть с одним скрытым слоем и 5 нейронами на нем, на всех нейронах действует сигмоидальная функция

активации, весовые коэффициенты настраиваются разработанным алгоритмом COBRA;

3. ANN+COBRA (3) – искусственная нейронная сеть с одним скрытым слоем и 3 нейронами на нем, на всех нейронах действует сигмоидальная функция активации, весовые коэффициенты настраиваются разработанным алгоритмом COBRA;

4. SVM+COBRA – машины опорных векторов, генерируемые разработанным алгоритмом COBRA;

5. ANN+structure – искусственная нейронная сеть с настраиваемыми структурой и весовыми коэффициентами алгоритмами COBRA и COBRA-b.

Таблица 4.6. Результаты, полученные для задач банковского скоринга

Классификатор	*Скоринг в Австралии*	*Скоринг в Германии*
ANN + structure	**0.9075**	**0.7974**
2SGP	0.9027	0.8015
SVM+COBRA	**0.9022**	**0.7960**
ANN+COBRA (5x5)	**0.8985**	**0.7866**
C4.5	0.8986	0.7773
Fuzzy	0.8910	0.7940
ANN+COBRA (5)	**0.8907**	**0.7829**
GP	0.8889	0.7834
ANN+COBRA (3)	**0.8898**	**0.7809**
CART	0.8744	0.7565
LR	0.8696	0.7837
CCEL	0.8660	0.7460
RSM	0.8520	0.6770
Bagging	0.8470	0.6840
Bayesian	0.8470	0.6790
Boosting	0.7600	0.7000
k-NN	0.7150	0.7151

Таким образом, результаты, полученные нейронными сетями, настроенными разработанными алгоритмами, для задачи банковского скоринга в Австралии лучше тех, что получены другими классификационными методами. Результаты же, полученные для задачи банковского скоринга в Германии, хуже только результатов метода 2SGP. Как видно из таблицы, результаты, полученные

методом опорных векторов, также являются одними из лучших, но немного уступают результатам нейронных сетей с генерируемой структурой.

Также следует отметить, что все результаты нейронных сетей и метода опорных векторов являются усредненными значениями по 20 программным запускам. Среднеквадратичное отклонение (полученное нейронными сетями) итоговых результатов для австралийской задачи было равно 1.166%, а для немецкой задачи – 2.267%. Ниже приведен пример одной из структур, сгенерированной в конечном итоге для задачи скоринга в Австралии (5 скрытых слоев, 5 нейронов на каждом слое):

1. первый слой – 0011 0111 1101 1000 0011, то есть нейроны с 3-ей, 7-ой, 13-ой, 18-ой и опять 3-ей функциями активации;
2. второй слой –1111 1110 1101 1011 0111, то есть нейроны с 15-ой, 14-ой, 13-ой, 11-ой и 7-ой функциями активации;
3. третий слой – 0111 1110 1100 1101 1111, то есть нейроны с 7-ой, 14-ой, 12-ой, 13-ой и 15-ой функциями активации;
4. четвертый слой – 0001 0011 0001 0111 1100, то есть нейроны с 1-ой, 3-ей, опять 1-ой, 7-ой и 12-ой функциями активации;
5. пятый слой – 1011 0101 1101 0001 1111, то есть нейроны с 11-ой, 5-ой, 13-ой, 1-ой и 15-ой функциями активации.

Далее решались задачи медицинской диагностики: определения типа раковой опухоли груди и определения заболевания диабетом. Тестовые установки были такими же, что и для задач банковского скоринга.

Задача определения типа опухоли имеет следующие характеристики: 10 атрибутов (первый атрибут – идентификационный номер пациента, который не использовался в дальнейшем в расчетах, и 9 категориальных атрибутов), 2 класса («злокачественная опухоль» и «доброкачественная опухоль»). Выборка содержала данные о 458 пациентах с доброкачественной опухолью и 241 пациентах со злокачественной опухолью. Задача определения наличия диабета имеет следующие характеристики: 8 атрибутов (все численные), 2 класса («болен» и «здоров»), выборка же содержала сведения о 500 пациентах, не болеющих

диабетом, и о 268 пациентах, у которых он был обнаружен. Данные и описания задач были также взяты из [87].

Изначально эти задачи были так же, как и предыдущие, решены нейронными сетями с заранее заданной структурой и настраиваемыми весовыми коэффициентами алгоритмом COBRA. Были выбраны следующие структуры:

1. Один скрытый слой с тремя нейронами, для всех нейронов использовалась одна и та же функция активации – сигмоид;
2. Один скрытый слой с пятью нейронами, для всех нейронов использовалась одна и та же функция активации – сигмоид;
3. Три скрытых слоя с тремя нейронами на каждом слое, для всех нейронов использовалась одна и та же функция активации – сигмоид;
4. Пять скрытых слоев, пять нейронов на каждом слое, для всех нейронов функция активации – сигмоид.

И затем они так же, как и задачи банковского скоринга, решались с помощью метода опорных векторов, генерируемого способом, описанным в предыдущей главе, нейронными сетями с настраиваемыми структурой и весовыми коэффициентами [59], а также коллективами машин опорных векторов [64].

Таким образом, применение нейронных сетей с подбором структуры сводилось к решению двух оптимизационных задач: задача безусловной оптимизации с бинарными переменными размерности 100 и задача безусловной оптимизации с вещественными переменными, размерность которой менялась от 145 до 150. Для конечной настройки весовых коэффициентов (для лучшей найденной структуры) максимальное число вычислений целевой функции было установлено равным 10000. Полученные результаты (доля правильно классифицированных данных) представлены в Таблицах 4.7 и 4.8, кроме того, в таблицах также приведены результаты, полученные другими методами, найденные в научной литературе [111], [131]. В Таблицах 4.7 и 4.8 использовались те же обозначения для искусственных нейронных сетей,

настраиваемых разработанными алгоритмами COBRA и COBRA-b, что и в Таблице 4.6, а также:

1. SVME_01 – коллективы машин опорных векторов со «схемой 1» принятия решений;

2. SVME_02 – коллективы машин опорных векторов со «схемой 2» принятия решений;

3. SVM_av – коллективы машин опорных векторов со схемой принятия решений «усреднение»;

4. SVM_wav – коллективы машин опорных векторов со схемой принятия решений «взвешенное усреднение».

Таблица 4.7. Результаты, полученные для задачи определения типа раковой опухоли

Автор (год)	*Метод*	*Точность (%)*
Peng et al. (2009)	CFW	99.50
Akhmedova (2014)	**ANN+structure**	**98.95**
Albrecht et al. (2002)	LSA machine	98.80
Akhmedova (2014)	**SVME_01**	**98.57**
Polat, Günes (2007)	LS-SVM	98.53
Akhmedova (2014)	**SVME_02**	**98.32**
Akhmedova (2014)	**ANN+COBRA (5x5)**	**98.19**
Akhmedova (2014)	**SVM_wav**	**98.18**
Akhmedova (2013)	**ANN+COBRA (3x3)**	**98.16**
Akhmedova (2014)	**SVM_av**	**98.14**
Setiono (2000)	Neuro-rule 2a	98.10
Akhmedova (2013)	**ANN+COBRA (5)**	**97.67**
Akhmedova (2013)	**SVM+COBRA**	**97.64**
Akhmedova (2013)	**ANN+COBRA (3)**	**97.62**
Karabatak, Cevdet-Ince (2009)	AR + NN	97.40
Pena-Reyes, Sipper (1999)	Fuzzy-GA1	97.36
Ster, Dobnikar (1996)	LDA	96.80
Guijarro-Berdias et al. (2007)	LLS	96.00
Abonyi, Szeifert (2003)	SFC	95.57
Nauck and Kruse (1999)	NEFCLASS	95.06
Hamiton et al. (1996)	RAIC	95.00
Quinlan (1996)	C4.5	94.74

Таблица 4.8. Результаты, полученные для задачи определения диабета

Метод	*Точность (%)*
MLNN with LM	82.37
SVM_wav	**80.32**
SVME_01	**80.26**
GRNN	80.21
SVME_02	**80.10**
ANN + structure	**80.15**
SVM_av	**80.03**
SVM+COBRA	**79.98**
ANN+COBRA (5x5)	**79.89**
ANN+COBRA (3x3)	**79.83**
ANN+COBRA (5)	**79.71**
ANN+COBRA (3)	**79.65**
MLNN with LM (10xFC)	79.62
PNN	78.13
PNN (10xFC)	78.05
FCNN with PA	77.34
MLNN with LM	77.08
DTDN	76.00
LVQ	73.60
PNN	72.00
FFN	68.80
AIS	68.80
CFN	68.00
AIRS	67.40
TDN	66.80
Gini	65.97

Результаты для нейронных сетей и метода опорных векторов опять-таки усреднялись по 20 программным запускам. Среднеквадратичное отклонение результатов, полученных нейронными сетями для задачи определения типа раковой опухоли, было равно 0.3564%, а для задачи определения наличия заболевания диабетом – 1.2958%. Кроме того, полученные разработанными классификаторами результаты существенно не различались. Коллективы машин опорных векторов продемонстрировали наилучшие результаты среди прочих предложенных в ходе диссертационной работы алгоритмов.

Несмотря на то, что описанные задачи являются сложными практическими задачами анализа данных, их принято считать тестовыми. Разработанные алгоритмы успешно справились с их решением, тем самым продемонстрировав

свою работоспособность, и то, что они могут быть рекомендованы для решения других подобных задач, а именно реальных задач анализа данных, вместо прочих альтернативных методов классификации.

4.3. Решение задач категоризации текста

Категоризация текста – это процесс классификации текстовых документов при наличии фиксированного количества заранее определенных категорий или классов [97]. Существует множество различных приложений категоризации текста, например, идентификация жанра произведений, определение типа документов, фильтрация спама и так далее.

В настоящее время разработаны различные методы для классификации текстов (например, статистические методы). Одним из наиболее известных и часто используемых подходов, предназначенных для решения задач классификации, является метод опорных векторов (Support Vector Machine, SVM), описанный в предыдущей главе.

В данной работе задачи категоризации текста решены классификаторами, полученными обучением метода опорных векторов с полиномиальным ядром алгоритмами COBRA и COBRA-c, а также нейронными сетями с настраиваемыми структурой и весовыми коэффициентами.

Известно, что на работу классификаторов при решении задач категоризации текста существенно влияет методика обработки самого текста, а именно учета слов, встречающихся в тексте. Для каждого слова, встречающегося в тексте необходимо вычислить весовой коэффициент, который зависит от частоты его появления в самом тексте. В настоящее время предложено множество различных способов расчета весовых коэффициентов для слов, но в данной работе рассматривались лишь следующие методы:

1. TF-IDF 1 ($idf_i = \log\left(\frac{|D|}{n_i}\right)$, где $|D|$ – число текстовых документов в выборке, n_i – число текстовых документов, в которых встречается *i*-ое слово, idf_i – весовой коэффициент *i*-ого слова);
2. TF-IDF 2 ($idf_i = \log\left(\frac{|D|}{n_i}\right)$ где $|D|$ – число текстовых документов в выборке, n_i – число, с помощью которого обозначено сколько раз встречалось *i*-ое слово во всех текстовых документах выборки, idf_i – весовой коэффициент *i*-ого слова);
3. TF-IDF 3 ($idf_i = \left(\frac{|D|}{n_i}\right)^{\alpha}$, $\alpha \in (0,1)$ где $|D|$ – число текстовых документов в выборке, n_i – число текстовых документов, в которых встречается *i*-ое слово, idf_i – весовой коэффициент *i*-ого слова, α – параметр, в данной работе принимал значения 0.1, 0.5, 0.9);
4. TF-IDF 4 ($idf_i = \left(\frac{|D|}{n_i}\right)^{\alpha}$, $\alpha \in (0,1)$ где $|D|$ – число текстовых документов в выборке, n_i – число, с помощью которого обозначено сколько раз встречалось *i*-ое слово во всех текстовых документах выборки, idf_i – весовой коэффициент *i*-ого слова, α – параметр, в данной работе принимал значения 0.1, 0.5, 0.9);
5. ConfWeight ($p(x,n) = \frac{x + 0.5z^2_{\alpha/2}}{n + z^2_{\alpha/2}}$, $\Phi\left(z_{\alpha/2}\right) = \frac{\alpha}{2}$, $\alpha = 0.95$, $0.5z^2_{\alpha/2} = 1.96$, Φ – распределение Стьюдента).

Помимо перечисленных методик вычисления весовых коэффициентов для слов была использована эвристика, описанная в статье [90] (далее «C-values»). Главная идея метода заключается в том, что каждое слово, которое присутствует в тексте, должно как-то численно соотноситься с некоторым определенным классом. Таким образом, каждому слову ставится в соответствие вещественное

число, назовем его C_j для j-го слова, которое зависит от частоты появления этого слова в тексте. Данное число определяется с помощью модифицированной формулы, которая используется для нечетких классификаторов. Функция принадлежности заменяется частотой «встречаемости» слова в текстах определенного класса.

Пусть L – число классов для решаемой задачи; n_i – число объектов из набора данных i-го класса; N_{ji} – количество «появлений» j-го слова в текстах i-го класса; $T_{ij} = \frac{N_{ij}}{n_i}$ – частота «появления» j-го слова в текстах i-го класса. $R_j = \max_i T_{ij}$, $S_j = \arg\left(\max_i T_{ij}\right)$ – номер класса, который присваивается j-му слову. Тогда C_j вычисляется следующим образом:

$$C_j = \frac{1}{\sum_{i=1}^{L} T_{ij}} \left(R_j - \frac{1}{L-1} \sum_{i=1}^{L} T_{ij} \right).$$

Таким образом, каждый объект-текст из набора данных представляется в виде вектора из $L+1$ чисел, где первое число – идентификатор класса, а остальные являются суммами значений C_j всех слов, которые встречались в данном объекте-тексте.

Кроме того, далее в расчетах еще использовалась модификация этих методик (в дальнейшем будем называть их «предобработками данных» или просто «предобработками»), заключавшаяся в применении кластеризации. Модификация проводилась следующим образом:

1. Для каждого слова выбирался некий класс-победитель (то есть тот класс, в текстах которого оно встречалось чаще всего);
2. Все слова с одним и тем же классом собирались в отдельные группы;
3. Для каждого класса строилась дендрограмма с помощью иерархической агломерационной кластеризации;
4. Выбиралось число кластеров слов для каждого класса (в данной работе рассматривались случаи, когда кластеров было 50 и 100);

5. Для каждого кластера вычислялся весовой коэффициент, как среднее значение весовых коэффициентов всех слов в кластере;
6. Настройка весовых коэффициентов кластеров с помощью коэволюционного генетического алгоритма;

Таким образом, после выбора методов классификации и способов предобработки данных решались уже задачи текстовой категоризации. То есть для исследования эффективности предложенных методов и сравнения их с другими алгоритмами категоризации текста были решены три задачи с конкурса DEFT’07 («Defi Fouille de Texte») [57] и одна задача с конкурса DEFT’08 [84]. В конкурсе DEFT’07 были представлены следующие задачи: «A voir a lire» – категоризация отзывов на книги (далее просто «Книги»), «Video Games» – категоризация отзывов на видео игры (далее просто «Игры»), «Debates in Parliament» – категоризация отзывов на принятие или отклонение выдвигаемого закона (далее просто «Дебаты»). Задача, представленная на конкурсе DEFT’08, называлась «T1» и заключалась в классификации статей, напечатанных во французской газете «Le Monde».

Данные для всех задач были разбиты на две выборки: обучающую и тестовую. Причем сами выборки были заранее определены организаторами конкурса, и все участники конкурса должны были проверить свои алгоритмы именно на этих выборках. Для того, чтобы было возможным сравнить результаты классификационных методов, описанных в этой работе, с методами участников конкурса, в дальнейшем использовалось разбиение на обучающую и тестовую выборку, заданное организаторами конкурса. Более подробное описание задач (количество данных, размер выборок, количество классов) представлено в Таблицах 4.9-4.13.

Таблица 4.9. Данные для задачи «Книги»

Книги	***Обучающая выборка***	***Тестовая выборка***
Положительный отзыв	1150	768
Отрицательный отзыв	309	207
Нейтральный отзыв	615	411

Таблица 4.10. Данные для задачи «Игры»

Игры	***Обучающая выборка***	***Тестовая выборка***
Положительный отзыв	874	779
Отрицательный отзыв	497	332
Нейтральный отзыв	1166	583

Таблица 4.11. Данные для задачи «Книги»

Дебаты	***Обучающая выборка***	***Тестовая выборка***
За принятие закона	6899	4961
Против принятия закона	10400	6572

Таблица 4.12. Данные для задачи «T1»

T1	***Обучающая выборка***	***Тестовая выборка***
Спорт	5767	3844
Экономика	4630	3085
Искусство	3474	2315
Телевидение	1352	1352

Таблица 4.13. Полное описание задач с конкурса DEFT'07

Задача	***Размер выборок***	***Классы***
Книги	Обучающая выборка – 2074 Тестовая выборка – 1386 Словарь из 52507 слов	0: отрицательный отзыв 1: нейтральный отзыв 2: положительный отзыв
Игры	Обучающая выборка – 2537 Тестовая выборка – 1694 Словарь из 63144 слов	0: отрицательный отзыв 1: нейтральный отзыв 2: положительный отзыв
Дебаты	Обучающая выборка – 17299 Тестовая выборка – 11533 Словарь из 59615 слов	0: против принятия закона 1: за принятие закона
T1	Обучающая выборка – 15223 Тестовая выборка – 10596 Словарь из 202979 слов	0: Спорт 1: Экономика 2: Искусство 3: Телевидение

Эффективность классификации оценивалась с помощью вычисления параметра *F-score* при $\beta = 1$ [134]:

$$F-score=\frac{(\beta^2+1)\cdot precision\cdot recall}{\beta^2\cdot(precision+recall)},$$

$$precision=\frac{|\{relevant\_documents\}|\cap|\{retrieved\_documents\}|}{|\{retrieved\_documents\}|},$$

$$recall=\frac{|\{relevant\_documents\}|\cap|\{retrieved\_documents\}|}{|\{relevant\_documents\}|}.$$

Точность классификации «*precision*» для каждого класса вычислялась как число правильно классифицированных объектов из набора данных для данного класса, разделенное на число объектов, которые алгоритм отнес к данному классу. Критерий «полнота» – «*recall*» – определяется числом правильно классифицированных объектов из набора данных, поделенное на число объектов, которое должно было быть отнесено к данному классу.

В Таблицах 4.14-4.17 представлены результаты, полученные методом опорных векторов с полиномиальной функцией-ядром и нейронными сетями с настраиваемыми структурой и весовыми коэффициентами, для задач «Книги», «Игры», «Дебаты» и «T1» при различных предобработках данных. В таблицах жирным выделены лучшие результаты [58], [134], [63].

Таблица 4.14. Результаты, полученные для задачи «Книги» для различных предобработок данных (без модификации)

Книги	***SVM+COBRA***	***ANN+COBRA***
C-values	**0.619**	0.585137
CW	0.588023	**0.613048**
Binary_sum	0.558442	0.566378
TFIDF_1	0.580087	0.554113
TFIDF_2	0.563492	0.533189
TFIDF_3_0.1	0.577201	0.518038
TFIDF_3_0.5	0.576479	0.460317
TFIDF_3_0.9	0.55772	0.505051
TFIDF_4_0.1	0.549784	0.553719
TFIDF_4_0.5	0.559163	0.550767
TFIDF_4_0.9	0.561328	0.541913

Таблица 4.15. Результаты, полученные для задачи «Игры» для различных предобработок данных (без модификации)

Игры	***SVM+COBRA***	***ANN+COBRA***
C-values	**0.695772**	0.691919
CW	0.645218	**0.726551**
Binary_sum	0.681818	0.65368
TFIDF_1	0.668831	0.642136
TFIDF_2	0.661157	0.649351
TFIDF_3_0.1	0.686541	0.678932
TFIDF_3_0.5	0.65987	0.643579
TFIDF_3_0.9	0.603896	0.627706
TFIDF_4_0.1	0.691263	0.701299
TFIDF_4_0.5	0.645218	0.678932
TFIDF_4_0.9	0.657096	0.551948

Таблица 4.16. Результаты, полученные для задачи «Дебаты» для различных предобработок данных (без модификации)

Дебаты	***SVM+COBRA***	***ANN+COBRA***
C-values	0.699905	0.704587
CW	**0.714211**	**0.709269**
Binary_sum	0.641984	0.6471
TFIDF_1	0.638429	0.640336
TFIDF_2	0.64129	0.640683
TFIDF_3_0.1	0.661406	0.645712
TFIDF_3_0.5	0.668659	0.611983
TFIDF_3_0.9	0.669323	0.6436
TFIDF_4_0.1	0.663314	0.643284
TFIDF_4_0.5	0.665135	0.601231
TFIDF_4_0.9	0.660827	0.566375

Таблица 4.17. Результаты, полученные для задачи «T1» для различных предобработок данных (без модификации)

Дебаты	***SVM+COBRA***	***ANN+COBRA***
C-values	0.866648	0.624764
CW	**0.875519**	**0.689411**
Binary_sum	0.796338	0.609287
TFIDF_1	0.83758	0.644111
TFIDF_2	0.827576	0.513237
TFIDF_3_0.1	0.823424	0.581883
TFIDF_3_0.5	0.817761	0.586731
TFIDF_3_0.9	0.748112	0.569366
TFIDF_4_0.1	0.827388	0.617969
TFIDF_4_0.5	0.808985	0.562665
TFIDF_4_0.9	0.739713	0.598371

В итоге без кластеризации лучшие результаты достигаются при использовании двух предобработок: «C-values» и «ConfWeight». Причем для нейронных сетей лучшей предобработкой является метод «ConfWeight», а для метода опорных векторов – «C-values». Следует отметить, что лучшие результаты, полученные искусственными нейронными сетями и машинами опорных векторов, существенно отличаются для задачи «Игры»: нейронные сети заметно «выигрывают», однако для задачи «T1» заметно лучше работают машины опорных векторов независимо от способа предобработки данных.

В Таблице 4.18 представлены полученные различными методами, включая описанные в данной работе машины опорных векторов (SVM+COBRA) и нейронные сети (ANN+COBRA), генерируемые алгоритмом COBRA и его модификациями COBRA-b и COBRA-c, результаты для первых трех задач («Книги», «Игры» и Дебаты»). Причем словом «*rank*» обозначено, под каким номером в списке находится полученный результат, то есть (1) означает лучший результат для соответствующей задачи, (2) – второй, и т.д. до (12), что означает худший полученный результат для этой задачи. Таким образом, все результаты были ранжированы, а затем вычислялось значение «*Rank*» следующим образом:

$$Rank = \frac{rank1 + rank2 + rank3}{3}.$$

В конечном итоге «лучший» алгоритм определялся минимальным полученным значением «*Rank*», «худший» соответственно максимальным полученным значением параметра [24, 125].

Таблица 4.18. Результаты, полученные для задач с конкурса DEFT07 для различных предобработок данных (без модификации)

Исследовательская группа или метод	*Книги (rank1)*	*Игры (rank2)*	*Дебаты (rank3)*	*Rank*
J.-M. Torres-Moreno	0.603 (2)	0.784 (1)	0.720 (1)	1
G. Denhiere	0.599 (3)	0.699 (6)	0.681 (6)	4
S. Maurel	0.519 (7)	0.706 (5)	0.697 (5)	5
M. Vernier	0.577 (4)	0.761 (3)	0.673 (7)	6
E. Crestan	0.529 (6)	0.673 (8)	0.703 (4)	7
M. Plantie	0.472 (9)	0.783 (2)	0.671 (9)	8
A.-P. Trinh	0.542 (5)	0.659 (9)	0.676 (8)	9
M. Genereux	0.464 (10)	0.626 (10)	0.569 (12)	11
E. Charton	0.504 (8)	0.619 (11)	0.616 (10)	10
A. Acosta	0.392 (11)	0.536 (12)	0.582 (11)	12
Лучшая комбинация для SVM+COBRA	**0.619 (1)**	**0.696 (7)**	**0.714 (2)**	**3**
Лучшая комбинация для ANN + structure	**0.613 (2)**	**0.727 (4)**	**0.709 (3)**	**2**

Таким образом, разработанные методы классификации в итоге превзошли почти всех участников конкурса по результатам, причем нейронные сети показывают лучшие результаты, чем метод опорных векторов. С другой стороны, методом опорных векторов был достигнут лучший результат для задачи «Книги» с предобработкой «C-values», для реализации которой требуется значительно меньше вычислительных ресурсов, чем для той же предобработки «ConfWeight», а именно с этой методикой учета слов результат нейронных сетей оказался вторым. То есть по времени работы алгоритма выигрывает метод опорных векторов, однако результат нейронный сетей с настраиваемыми структурой и весовыми коэффициентами не намного, но все-таки лучше.

Для задачи «T1» результаты, полученные машинами опорных векторов оказались в числе лучших, а результаты, полученные нейронными сетями средними.

Далее первые три задачи решались разработанными методами, но с другими предобработками данных (предобработки были модифицированы с помощью кластеризации – 50 и 100 кластеров). В Таблицах 4.19-4.21 представлены полученные результаты для всех трех задач, жирным выделены лучшие результаты.

Таблица 4.19. Результаты, полученные для задачи «Книги» для различных предобработок данных модификацией

Книги	***SVM+COBRA***		***ANN + structure***	
	50	***100***	***50***	***100***
C-values	**0.602**	**0.594**	0.584	0.587
TFIDF_1	0.556	0.586	0.549	0.556
TFIDF_2	0.570	0.561	0.567	0.607
TFIDF_3_01	0.460	0.587	0.564	0.551
TFIDF_3_05	0.566	0.566	0.557	0.571
TFIDF_3_09	0.547	0.576	0.571	0.570
TFIDF_4_01	0.577	0.556	0.546	**0.613**
TFIDF_4_05	0.588	0.537	**0.612**	0.574
TFIDF_4_09	0.568	0.577	0.561	0.543

Таблица 4.20. Результаты, полученные для задачи «Игры» для различных предобработок данных модификацией

Игры	***SVM+COBRA***		***ANN + structure***	
	50	***100***	***50***	***100***
C-values	**0.681**	**0.668**	**0.712**	0.706
TFIDF_1	0.638	0.635	0.693	0.702
TFIDF_2	0.630	0.613	0.694	**0.725**
TFIDF_3_01	0.639	0.640	0.667	0.674
TFIDF_3_05	0.632	0.637	0.674	0.672
TFIDF_3_09	0.622	0.607	0.676	0.685
TFIDF_4_01	0.653	0.634	0.694	0.715
TFIDF_4_05	0.651	0.632	0.702	0.700
TFIDF_4_09	0.634	0.655	0.662	0.696

Таблица 4.21. Результаты, полученные для задачи «Дебаты» для различных предобработок данных модификацией

Дебаты	*SVM+COBRA*		*ANN + structure*	
	50	*100*	*50*	*100*
C-values	**0.668**	0.659	0.698	0.700
TFIDF_1	0.661	0.628	**0.700**	**0.704**
TFIDF_2	0.620	0.649	0.664	0.690
TFIDF_3_01	0.653	0.646	0.683	0.684
TFIDF_3_05	0.633	**0.663**	0.690	0.675
TFIDF_3_09	0.630	0.640	0.654	0.673
TFIDF_4_01	0.641	0.609	0.646	0.674
TFIDF_4_05	0.652	0.645	0.668	0.654
TFIDF_4_09	0.642	0.643	0.663	0.663

В итоге, для метода опорных векторов наиболее эффективной оказалась методика предобработки данных даже после кластеризации «C-values». В то же самое время для нейронных сетей «лучшая» методика предобработки данных менялась в зависимости от решаемой задачи и проведенной кластеризации.

Далее в Таблице 4.22 приведены полученные различными методами (включая описанные в данной работе методы COBRA+SVM и ANN+structure) результаты для всех трех задач после модификации способов предобработки данных. Результаты были ранжированы по тому же принципу, что и ранее в работе.

Таблица 4.22. Результаты, полученные для задач с конкурса DEFT07 для различных предобработок данных после модификации

Исследовательская группа или метод	*Книги (rank1)*	*Игры (rank2)*	*Дебаты (rank3)*	*Rank*
J.-M. Torres-Moreno	0.603 (3)	0.784 (1)	0.720 (1)	1
G. Denhiere	0.599 (5)	0.699 (7)	0.681 (6)	5
S. Maurel	0.519 (10)	0.706 (6)	0.697 (5)	6
M. Vernier	0.577 (7)	0.761 (3)	0.673 (7)	4
E. Crestan	0.529 (9)	0.673 (9)	0.703 (3)	7
M. Plantie	0.472 (12)	0.783 (2)	0.671 (9)	9
A.-P. Trinh	0.542 (8)	0.659 (11)	0.676 (8)	11
M. Genereux	0.464 (13)	0.626 (12)	0.569 (14)	13
E. Charton	0.504 (11)	0.619 (13)	0.616 (12)	12

Продолжение Таблицы 4.22. Результаты, полученные для задач с конкурса DEFT07 для различных предобработок данных после модификации

Исследовательская группа или метод	*Книги (rank1)*	*Игры (rank2)*	*Дебаты (rank3)*	*Rank*
A. Acosta	0.392 (14)	0.536 (14)	0.582 (13)	14
Лучшая комбинация для SVM+COBRA (50 кластеров)	**0.602 (4)**	**0.681 (8)**	**0.668 (10)**	**8**
Лучшая комбинация для SVM+COBRA (100 кластеров)	**0.594 (6)**	**0.668 (10)**	**0.663 (11)**	**10**
Лучшая комбинация для ANN + structure (50 кластеров)	**0.612 (2)**	**0.712 (5)**	**0.700 (4)**	**3**
Лучшая комбинация для ANN + structure (100 кластеров)	**0.613 (1)**	**0.725 (4)**	**0.704 (2)**	**2**

Таким образом, после кластеризации нейронные сети намного превзошли по результатам метод опорных векторов. Кроме того, с помощью искусственных нейронных сетей были получены первые два лучших результата для задачи «Книги» и второй лучший результат для задачи «Дебаты». В конечном итоге только один алгоритм «работал эффективнее» на этих задач, чем сгенерированные эвристикой COBRA нейронные сети.

4.4. Прогнозирование процесса деградации солнечных батарей космического аппарата (БС КА)

Солнечные батареи всегда были и остаются одним из основных источников электроэнергии космических аппаратов. Стремление довести время жизни КА до 10 и более лет значительно ужесточило требования к надежности и ресурсу БС. Очевидно, что для выполнения этих требований необходимо свести к минимуму деградацию характеристик БС.

Составить единую формальную модель влияния внешних факторов на характеристики БС не просто, так как большинство факторов действуют непостоянно и влияют не только непосредственно на БС, но и друг на друга. Проведение полномасштабных натурных экспериментов со всевозможными комбинациями этих факторов и их интенсивностью не представляется возможным в силу их сложности. В связи с этим возникает необходимость в построении дескриптивной модели, описывающей связь между внешними факторами и

характеристиками БС (без глубокого погружения в физику процесса), чтобы использовать ее при прогнозировании процесса деградации БС на основе имеющихся статистических данных об условиях полета.

Наиболее подходящим инструментом для этого являются искусственные нейронные сети (ИНС), однако возможно применение и других методов. В данной работе для прогнозирования процесса деградации БС были разработанные автоматически генерируемые нейросетевые модели прогнозирования, а так же машины опорных векторов.

Так как построение строгой формальной модели процесса деградации на настоящий момент невозможно, представляется вполне обоснованным использование модели типа «черного ящика», связывающей входы системы, то есть внешние факторы, воздействующие на солнечные батареи, и ее выходы, т.е. технические характеристики.

От специалистов ОАО ИСС (Ю.М. Князькин, Д.В. Вилков) была получена база данных, содержащая результаты наблюдений за внешними факторами и техническими характеристиками БС КА за трехлетний период функционирования на орбите, и предобработана специальным образом. В настоящее время эта база данных представляет собой полигон для испытания интеллектуальных технологий моделирования и прогнозирования.

Таким образом, были предоставлены результаты измерения параметров солнечных батарей в полёте (параметры секций БС3 и БС4 на КА ЭКСПРЕСС-А №2) и результаты измерения параметров СКЛ с КА ЭКСПРЕСС-А и GOES в годы активного Солнца (25 событий за период с 04.04.00 по 22.11.01, от экстремально мощных 14.07.00 и 22.11.01 до обычных - 16.10.00, примеры спектра). Необходимо было построить модель, прогнозирующую деградацию электрических характеристик солнечных батарей КА. Модель настраивалась на определение электрических характеристик солнечных батарей в зависимости от 7 факторов (входной вектор):

1. Интегральный флюенс протонов с различными значениями энергии (от 1 до 200 МеВ);

2. Интегральный флюенс электронов с различными значениями энергиями (от 0.6 до 2 МеВ);
3. Ресурс – это параметр, который задан как количество дней с момента контакта отделения КА, характеризует повреждения от метеоритных тел и от ультрафиолетового излучения;
4. Коэффициент освещенности КА – величина, характеризующая степень освещенности аппарата, зависит от взаимного положения КА, Земли и Луны, имеет принципиальное значение, т.к. некоторые выходные характеристики ФП, в частности сила тока короткого замыкания, зависят от освещенности солнечных батарей.

Данным факторам в соответствие ставятся 2 выхода для каждой секции, то есть всего 4 выхода: напряжение холостого хода Uxx для БС3 и для БС4 и сила тока Iкз БС3 и БС4 (вектор значений или выходной).

Нейросетевая модель строилась следующим образом [124], [2]: генерировались 4 нейронные сети (одна нейросеть на каждый выходной параметр), которые обучались на одной выборке, а затем тестировались на другой. Сначала использовались нейронные сети с заранее заданной структурой (1 скрытый слой с тремя нейронами, все функции активации – сигмоид). Далее для вычислений применялись нейронные сети с настраиваемой в процессе выполнения программы структурой. Для каждой нейросети сохранялось лучшее значение на выходе (то есть минимальная относительная ошибка). Ошибка прогнозирования вычислялась следующим образом:

$$e = \frac{1}{s}\frac{1}{m}\sum_{i}^{m}\frac{1}{y^{i}_{\max} - y^{i}_{\min}}\sum_{j}^{s}\left|y^{i}_{j} - o^{i}_{j}\right|,$$

где s – объем выборки, m – число выходов, y – истинный выход, o – выход модели.

Затем строилась модель машин опорных векторов следующим образом: генерировались 4 машины опорных векторов (одна машина на каждый выходной параметр), которые обучались на одной выборке, а затем тестировались на другой. Для каждой машины опорных векторов сохранялось лучшее значение на

выходе (то есть минимальная относительная ошибка), которая вычислялась так же, как и для нейронных сетей.

Результаты (относительная ошибка), полученные по указанному выше критерию, приведены в Таблице 4.23 (для сравнения также приведены результаты, опубликованные в статье [73]).

Таблица 4.23. Результаты, полученные для задачи прогнозирования процесса деградации БС КА

Метод	***Относительная ошибка (%)***
Neural Network Ensemble Model	4.3
Neural Network Ensemble Model (GASEN)	5.2
ANN + structure	5.32
ANN+COBRA (3)	5.35
SVM+COBRA	5.68
3-layer perceptron model	5.7
2-layer perceptron model	5.8
Symbolic model	6.3

Как видно из Таблицы 4.23 разработанные методы показывают неплохой результат. Нейронные сети уступают по критерию «относительная ошибка» ансамблям сетей, метод опорных векторов продемонстрировал не столь эффективную работу при решении задачи регрессии.

4.5. Прогнозирование успешности учебной деятельности студентов

Подготовка квалифицированных специалистов – одна из главных задач любого образовательного учреждения. В современных условиях одним из основных подходов, обеспечивающих высокое качество образования, является использование индивидуальных образовательных траекторий, когда усилия студента и коллектива преподавателей направлены на развитие именно этого студента, причем именно в тех областях, которые являются наиболее важными для него. Однако точный выбор дисциплин, разделов и тем, на которых необходимо дополнительно сосредоточить усилия, можно осуществить только на основе правильного понимания сильных и слабых сторон в уровне подготовке конкретного студента. Это возможно в случае студентов магистратуры, уже проучившихся четыре года в ВУЗе и хорошо знакомых преподавателям,

достаточно сложно для студентов 3-4 курса и практически невозможно для первокурсников, вчерашних абитуриентов, сильные и слабые стороны которых по понятным причинам не известны педагогическому коллективу.

Единственным подходом, позволяющим хотя бы приблизительно определить болевые точки (возможные пробелы в знаниях и места сосредоточения трудностей обучения), ликвидация которых является первоочередной задачей, и, наоборот, сильные стороны, требующие внимательного отношения и всемерного развития, является прогнозирование успешности учебной деятельности студента по информации, доступной на момент его поступления в ВУЗ.

При обучении на успеваемость студента могут влиять множество факторов. В ходе выполнения диссертационной работы была выделена следующая группа характеристик для оценивания «перспективности» абитуриента: пол абитуриента, его возраст, адрес проживания, номер школы, выпустившей абитуриента, результаты сдачи ЕГЭ, олимпиад, вступительных экзаменов в ВУЗе (если нет соответствующих результатов по ЕГЭ), выбранные приоритеты при поступлении в ВУЗ. Иными словами, учитывалась вся информация, предоставляемая студентом при поступлении. Кроме того, использовалась информация о результатах обучения студентов, сосредоточенная в документах деканатов и представляющая собой сведения в дискретных или порядковых шкалах (оценки, результаты аттестации), а иногда – в номинальных (благодарности, выговоры, и т.п.).

Цель заключалась в составлении единой формальной модели влияния перечисленных факторов на успеваемость будущих студентов, но так как рассматривались и количественные показатели, и качественные, то можно сказать, что исходные данные были сложно формализуемыми. Необходимо было собрать сведения для формирования базы данных, которая бы использовалась для генерирования интеллектуальной информационной технологии прогнозирования. Таким образом, необходимо было построить модель, описывающую связь между

группой характеристик абитуриента и его дальнейшей успеваемостью в ВУЗе в случае его поступления, основываясь на сформированной базе данных.

В качестве интеллектуальных информационных технологий прогнозирования были выбраны машины опорных векторов, генерируемые разработанным алгоритмом COBRA [18], [17], [19]. Помимо прочего также для данной задачи был автоматизирован процесс поиска наиболее информативных данных («входов»): определялись факторы из перечисленных выше, которые непосредственно влияли на успеваемость студента. Подобные алгоритмы генерирования ИИТ с выбором наиболее информативных входов, были рассмотрены ранее в работах [54], [53].

Для определения информативных входов для каждого вектора входных параметров генерировалась бинарная строка, количество бит в строке было равным количеству входов, так, что некоторый бит в строке был равен единице, если соответствующий ему параметр стоило учитывать (он был «информативен») и нулю в противном случае. Следовательно, помимо построения оптимальной гиперплоскости методом опорных векторов нужно было решить задачу безусловной оптимизации с бинарными переменными. В итоге генерировалась популяция бинарных строк (различные способы учета входных данных), для каждой строки решалась задача условной оптимизации и строилась гиперплоскость, далее выбиралась строка с лучшим показателем эффективности. После описанного этапа уже выполнялась работа бинарного алгоритма.

В данной работе для настройки поиска информативных входов была применена модификация коллективного бионического алгоритма COBRA для решения задач с бинарными переменными COBRA-b, а его модификация для решения задач условной оптимизации COBRA-c применялась для построения оптимальной разделяющей гиперплоскости.

Описанный алгоритм генерирования машин опорных векторов с выбором информативных входов был применен для решения двух задач классификации. Первая задача заключалась в определении поступит или нет абитуриент, соответственно, задача с двумя классами – поступившие и не поступившие

абитуриенты. Вторая задача состояла в прогнозировании возможности отчисления или прохождения первой экзаменационной сессии поступившими абитуриентами. Всего для второй задачи были определены три класса: поступившие выпускники школ, отчисленные после первой экзаменационной сессии; поступившие выпускники школ, сдавшие первую сессию; не поступившие абитуриенты.

Для обеих задач использовался один и тот же набор данных, состоявший из сведений об 742 абитуриентах. Каждый поступающий описывался следующими характеристиками: возраст, адрес, пол, баллы за ЕГЭ, сдавали ли экзамен в университете и т.д. Итак, входных параметров было 11, причем первый параметр был идентификатором класса. Все данные были изначально приведены к численному виду, а затем нормированы.

И для первой, и для второй задачи алгоритм запускался по 20 раз. Также задачи были решены нейронными сетями (GA-ANNinput) и системами на нечеткой логике (GA-FLinput), сгенерированными генетическими алгоритмами. В Таблице 4.24 приведены полученные результаты, усредненные по количеству запусков, причем в таблице для каждого алгоритма указана доля правильно классифицированных «объектов», то есть абитуриентов.

Таблица 4.24. Полученные результаты

Задача	***GA-ANNinput***	***GA-FLinput***	***SVM+COBRA***
2 класса	0.973	0.962	0.991
3 класса	0.931	0.915	0.969

По полученным результатам можно сделать вывод, что на данных задачах метод опорных векторов в среднем работает лучше нейронных сетей и систем на нечеткой логике, причем нейронные сети оказались эффективней метода GA-FLinput. Также стоит отметить, что все время работы всех алгоритмов (время необходимое для выполнения одного прогона) было примерно одинаковым (около 3-5 минут).

Одной из целей данной работы было определение наиболее информативных входов для этих задач, то есть определения самых значимых факторов для

поступления и сдачи первой экзаменационной сессии. Так, например, наилучший результат (98,7% правильно классифицированных данных) для задачи с тремя классами был достигнут методом опорных векторов SVM+COBRA, при учете следующих входных параметров: вид образовательного учреждения, оконченного абитуриентом, возраст абитуриента, баллы за ЕГЭ по информатике. Для первой же задачи наилучший результат достигался при учете тех же входных параметров, только вместо баллов за ЕГЭ по информатике включались баллы за ЕГЭ по математике. Обращает на себя внимание тот факт, что существенным фактором оказался возраст, хотя, казалось бы, у абитуриентов он практически одинаков. Объяснением этого может быть тот факт, что более старшие абитуриенты, окончившие школу не в текущем году, а ранее, менее успешны при поступлении и дальнейшем обучении. Очевидно, что среди таких поступающих значительна доля не поступивших в прошлые годы или поступивших и отчисленных после первых сессий, т.е. менее пригодных для обучения в ВУЗе.

Выводы

Разработанные алгоритмы автоматического проектирования искусственных нейронных сетей, машин опорных векторов, а также их коллективов были применены для решения различных задач анализа данных. Они были протестированы на двух задачах распознавания (сегментация изображений и распознавание речи), на двух задачах банковского скоринга (скоринг в Австралии и скоринг в Германии) и двух задачах медицинской диагностики (диагностика рака груди и диагностика диабета). Таким образом, было установлено, что сгенерированные классификаторы сравнимы по эффективности с другими методами категоризации, более того, они превосходили по результатам большинство из них. Далее предложенные алгоритмы были применены для решения практических задач: четыре задачи категоризации текста, взятых с конкурсов DEFT’07 и DEFT’08, задача прогнозирования процесса деградации солнечных батарей космического аппарата, а также задача прогнозирования

успешной учебной деятельности студентов. Были получены следующие результаты:

1. Для задач категоризации текста было установлено, что лишь один алгоритм работает лучше нейронных сетей и машин опорных векторов, сгенерированных с помощью разработанного алгоритма COBRA и его модификаций;
2. Для задачи прогнозирования процесса деградации БС КА нейронные сети по результатам превосходили разработанные модели регрессионного анализа данных;
3. В целом машины опорных векторов работали чуть хуже нейронных сетей, однако быстрее по времени.

Таким образом, в данной работе предложенные ранее алгоритмы прошли успешную апробацию на различных задачах классификации и прогнозирования. Показано, что разработанные алгоритмические схемы проектирования информационных технологий интеллектуального анализа данных демонстрируют лучшие результаты, чем большинство альтернативных методов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения диссертационной работы получены следующие результаты:

1) Разработан, реализован и апробирован новый коллективный самонастраивающийся метод решения задач безусловной оптимизации с вещественными переменными.

2) Разработан, реализован и апробирован новый коллективный самонастраивающийся метод решения задач безусловной оптимизации с бинарными переменными.

3) Выполнены модификации разработанных алгоритмов, позволяющие решать задачи условной оптимизации.

4) На основе разработанных алгоритмов и их модификаций построены алгоритмы автоматизированного генерирования информационных технологий интеллектуального анализа данных на основе искусственных нейронных сетей и машин опорных векторов.

5) Эффективность разработанных подходов продемонстрирована в ходе решения 12 прикладных задач оптимизации и анализа данных из различных областей науки и техники.

Таким образом, в данной работе разработаны, реализованы и проверены новые оптимизационные методы, а также эффективные алгоритмические схемы их применения при автоматизированном генерировании информационных технологий интеллектуального анализа данных, позволяющие успешно решать широкий круг практических задач и имеющие значение для развития теории и практики системного анализа и обработки информации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Айвазян, С.А., Енюков, И.С., Мешалкин, Л.Д. Прикладная статистика: основы моделирования и первичная обработка данных. – М.: Финансы и статистика, 1983. – 487 с.
2. Ахмедова, Ш.А. Автоматизированное проектирование нейросетевых предикторов деградации солнечных батарей космического аппарата кооперативными бионическими алгоритмами. Материалы XX научно-технической конференции молодых ученых и специалистов. – Королев: Ракетно-космическая корпорация «Энергия» имени С.П. Королева, С. 656-658, 2014.
3. Ахмедова, Ш.А. Генерирование нейросетевых классификаторов кооперативными бионическими алгоритмами оптимизации. Материалы XVIII Международной научной конференции «Решетневские чтения», посвященной 90-летию со дня рождения генерального конструктора ракетно-космических систем академика М. Ф. Решетнев, Том 2, С. 224-226, 2014.
4. Ахмедова, Ш.А. Исследование эффективности «стайного» алгоритма для задач многокритериальной оптимизации. Молодежь и наука: сборник материалов VIII Всероссийской научно-технической конференции студентов, аспирантов и молодых ученых, посвященной 155-летию со дня рождения К.Э. Циолковского, 2012.
5. Ахмедова, Ш.А. Исследование эффективности «стайного» алгоритма для задач условной оптимизации. Молодёжь и наука: сборник материалов VII Всероссийской научно-технической конференции студентов, аспирантов и молодых ученых, посвященной 50-летию первого полета человека в космос, 2011.
6. Ахмедова, Ш.А. Исследование эффективности «стайного» алгоритма оптимизации. Сборник статей студентов, аспирантов и молодых ученых по итогам Всероссийской научно-практической конференции, посвященной 80-

летию Сибирского государственного технологического университета, Том 2. Красноярск: СибГТУ, 2010. – С. 251-253.

7. Ахмедова, Ш.А. Исследование эффективности условного «стайного» алгоритма оптимизации. «Актуальные проблемы авиации и космонавтики». Труды Всероссийской научно-практической конференции творческой молодежи. Красноярск: СибГАУ, 2011.
8. Ахмедова, Ш.А. Коллективный бионический алгоритм для решения задач безусловной оптимизации с бинарными переменными. Материалы XVIII Международной научной конференции «Решетневские чтения», посвященной 90-летию со дня рождения генерального конструктора ракетно-космических систем академика М. Ф. Решетнева, Том 2, С. 17-18, 2014.
9. Ахмедова, Ш.А. Об эффективности «стайного» алгоритма оптимизации. Труды XLIII Краевой научной студенческой конференции по математике и компьютерным наукам. Красноярск: СФУ, 2010. – С. 9-12.
10. Ахмедова, Ш.А. Последовательный и параллельный «стайный» алгоритм для задач условной и безусловной оптимизации. Актуальные проблемы авиации и космонавтики: материалы VIII Всерос. науч.-практ. конф. творческой молодежи, посвященной Дню космонавтики и 55-летию запуска первого искусственного спутника Земли. Красноярск: СибГАУ, 2012.
11. Ахмедова, Ш.А. Разработка и исследование эффективности «стайного» алгоритма для задач однокритериальной и многокритериальной оптимизации. Наука и инновации в технических университетах: материалы 6-го Всероссийского форума студентов, аспирантов и молодых ученых. Санкт-Петербург: СПбПУ, 2012. – С. 68-70.
12. Ахмедова, Ш.А. Разработка и исследование эффективности «стайного» алгоритма оптимизации. Сборник материалов Всероссийского конкурса научно-исследовательских работ студентов и аспирантов в области технических наук. Санкт-Петербург: СПбГПУ, 2012.

13. Ахмедова, Ш.А. Разработка и исследование эффективности «стайного» алгоритма оптимизации. Сборник научных работ Всероссийского конкурса научно-исследовательских работ студентов и аспирантов в области информатики и информационных технологий, Том 3. Белгород: НИУ БелГУ, 2012. – С. 241-244.
14. Ахмедова, Ш.А. Формирование инвестиционного портфеля с помощью самонастраивающихся интеллектуальных информационных технологий эволюционного типа. Наукоёмкие инновационные проекты молодых ученых: материалы Всероссийского конкурса. Санкт-Петербург: СПбГПУ, 2012. – С. 96-114.
15. Ахмедова, Ш.А. Формирование оптимального инвестиционного портфеля предприятия бинарным «стайным» алгоритмом. Информационные технологии, системный анализ и управление: сборник трудов IX всероссийской научной конференции молодых ученых, аспирантов и студентов, Том 2. Таганрог: Изд-во ТТИ ЮФУ, 2011. – С. 218-220.
16. Ахмедова, Ш.А. Формирование оптимального инвестиционного портфеля предприятия многокритериальным «стайным» алгоритмом. «Решетневские чтения». Материалы XVI Международной научной конференции, посвященной памяти генерального конструктора ракетно-космических систем академика М. Ф. Решетнева, Том 2. - Красноярск: СибГАУ, 2012. – С. 471-472.
17. Ахмедова, Ш.А., Вишневская, С.Р. Метод опорных векторов для прогнозирования успешности обучения студентов. Материалы Всероссийской научно-практической конференции «Информационно-телекоммуникационные системы и технологии», Кемерово, С. 13-14, 2014.
18. Ахмедова, Ш.А., Вишневская, С.Р., Коромыслова, А.А. Интеллектуальные информационные технологии для прогнозирования успешности учебной деятельности абитуриентов. Вестник Сибирского государственного аэрокосмического университета имени академика М.Ф. Решетнева, № 3 (54), С. 16-20, 2014.

19. Ахмедова, Ш.А., Вишневская, С.Р., Коромыслова, А.А. Прогнозирование успеваемости абитуриентов методами вычислительного интеллекта. Евразийский Союз Ученых, № 6, часть 3, С. 8-10, 2014.
20. Ахмедова, Ш.А., Семенкин, Е.С. Development of a new optimization metaheuristic based on co-operation of biology related algorithms. Вестник Сибирского государственного аэрокосмического университета имени академика М.Ф. Решетнева, № 4 (50), С. 92-99, 2013.
21. Ахмедова, Ш.А., Семенкин, Е.С. Particle Swarm Optimization Algorithms Development and Effectiveness Investigation. Вестник Сибирского государственного аэрокосмического университета имени академика М.Ф. Решетнева, № 4 (44), 2012. – С.78-79.
22. Ахмедова, Ш.А., Семенкин, Е.С. Коллективный метод оптимизации на основе стайных бионических алгоритмов. Системный анализ и интеллектуальные технологии. Труды V Международной конференции (САИТ-2013), Том 2, С. 322-331, 2013.
23. Ахмедова, Ш.А., Семенкин, Е.С. Кооперативный бионический алгоритм безусловной оптимизации. Программные продукты и системы, № 4 (104), С. 133-136, 2013.
24. Ахмедова, Ш.А., Семенкин, Е.С. Проектирование систем автоматического определения субъективного мнения кооперативными бионическими алгоритмами. Системы управления и информационные технологии, №3.1 (57), С. 104-108, 2014.
25. Ахмедова, Ш.А., Семенкин, Е.С. Разработка и исследование эффективности стайного алгоритма оптимизации. LAP LAMBERT Academic Publishing. - Saarbrücken, Germany. – 67 с. – ISBN 978-3-659-37617-7.
26. Ахмедова, Ш.А., Семенкин, Е.С., Гасанова, Т.О., Минкер, В.М. Обучение машин опорных векторов для решения задач категоризации текста. Сборник трудов XIV национальной конференции по искусственному интеллекту с международным участием КИИ-2014, Том 2., С. 222-230.

27. Бокс, Дж. Анализ временных рядов, прогноз и управление. В 2-х т.: Пер. с английского / Дженкинс Г.М., Бокс Дж. – М.: Издательство «Мир», 1974. – 608 с.
28. Боровиков, В.П. Популярное введение в программу Statistica. – М.: Компьютер-Пресс, 1998. – 267 с.
29. Бухтояров, В.В. Эволюционные алгоритмы формирования коллективов нейронных сетей для решения задач моделирования и прогнозирования. Канд. техн. наук, 05.13.17. – СибГАУ, Красноярск, 2010.
30. Вапник, В.Н., Червоненкис, А.Я. Теория распознавания образов. – М.: Наука, 1974. – 416 с.
31. Ворожейкин, Ю.А., Гончар, Т.Н., Панфилов, И.А., Сопов, Е.А., Сопов, С.А. Об одной модификации вероятностного генетического алгоритма для решения сложных задач условной оптимизации. Вестник Сибирского государственного аэрокосмического университета имени академика М.Ф. Решетнева, № 4, 2009. – С.79-84.
32. Воронцов, К. В. Лекции по методу опорных векторов.
33. Воронцов, К.В. О проблемно-ориентированной оптимизации базисов задач распознавания. ЖВМ и МФ, Том 38, № 5, С. 870-880, 1998.
34. Воронцов, К.В. Оптимизационные методы линейной и монотонной коррекции в алгебраическом подходе к проблеме распознавания. ЖВМ и МФ, Том 40, № 1, С. 166-176, 2000.
35. Воронцов, К.В., Каневский, Д.Ю. Коэволюционный метод обучения алгоритмических композиций. Таврический вестник информатики и математики, № 2, С. 51-66, 2005.
36. Галушкин, А.И. Синтез многослойных систем распознавания образов. – М.: Энергия, 1974. – 367 с.
37. Гапочкин, А. В. Нейронные сети в системах распознавания речи. Science Time, №1 (1), С. 29-36, 2014.

38. Горбань, А.Н. Нейроинформатика: кто мы, куда мы идём, как путь наш измерить. Вычислительные технологии. М.: Машиностроение, № 4, с. 10-14, 2000.
39. Грешилов, А.А., Стакун, В.А., Стакун, А.А. Математические методы построения прогнозов. – М.: Радио и связь, 1997. – 112 с.
40. Журавлёв, Ю.И. Об алгебраическом подходе к решению задач распознавания или классификации. Проблемы кибернетики, Том 33, С. 5-68, 1978.
41. Журавлёв, Ю.И. Экстремальные алгоритмы в математических моделях для задач распознавания и классификации. Доклады АН СССР, математика, Т. 231, № 3, 1976.
42. Журавлёв, Ю.И., Рудаков, К.В. Об алгебраической коррекции процедур обработки (преобразования) информации. Проблемы прикладной математики и информатики, С. 187-198, 1987.
43. Журавлёв, Ю.И., Рязанов, В.В., Сенько, О.В. РАСПОЗНАВАНИЕ. Математические методы. Программная система. Практические применения. – М.: Изд. «Фазис», 2006. – 176 с.
44. Клешков, В.М., Семенкин, Е.С. Модели и алгоритмы распределения общих ресурсов при управлении инновациями реструктурированного машиностроительного предприятия. Проблемы машиностроения и автоматизации, № 3, С. 24-31, 2006.
45. Кохонен, Т. Ассоциативные запоминающие устройства. М.: Мир, 1982. – 384 с.
46. Курейчик, В.М. Генетические алгоритмы и их применение. Таганрог: Изд-во ТРТУ, 2002.
47. Мазуров, В.Д. Комитеты системы неравенств и задача распознавания. Кибернетика, № 3, 1971.
48. Мазуров, В.Д. Метод комитетов в задачах оптимизации и классификации. – М.: Наука, 1990.

49. Мак-Каллок, У.С., Питтс, В. Логическое исчисление идей, относящихся к нервной активности. Автоматы, под ред. К. Э. Шеннона и Дж. Маккарти. М.: Изд-во иностр. лит., 1956. – С. 363-384.
50. Пуртиков, В.А. Постановка задачи оптимизации выбора кредитного портфеля. Вестник НИИ СУВПТ, вып. № 2, С. 145-159, 1999.
51. Рабинер, Л.Р. Скрытые марковские модели и их применение в избранных приложениях при распознавании речи. ТИИЭР, №2, С. 86-120, 1989.
52. Растригин, Л.А., Эренштейн, Р.Х. Коллективные правила распознавания. М.: Энергия, 1981. – 244 с.
53. Семенкин, Е.С., Шабалов, А.А. Система автоматизированного проектирования коллективов интеллектуальных информационных технологий для задач анализа данных. Программные продукты и системы, № 4 (100), С. 70-73, 2012.
54. Семенкина, М.Е. Самоадаптивные эволюционные алгоритмы проектирования информационных технологий интеллектуального анализа данных. Искусственный интеллект и принятие решений, № 1, С. 13-23, 2013.
55. Терехов, С.А. Гениальные комитеты умных машин. Научная сессия МИФИ–2007. IX всероссийская научно-техническая конференция «Нейроинформатика–2007»: Лекции по нейроинформатике. Часть 2. – М.: МИФИ, 2007. – 148 с.
56. Хайниш, С.В., Клешков, В.М., Бородин, А.Н. Российское предприятие ВПК: выжить и развиваться. (На примере реформирования и развития Химзавода – филиала ФГУП «КРАСМАШ»). – М.: Рохос, 2003. – 240 с., цв. Вкл. (Из опыта управленческого консультирования.)
57. Actes de l'atelier DEFT'07, Plate-forme AFIA 2007, Grenoble, Juillet, 2007. URL: http://deft07.limsi.fr/actes.php.
58. Akhmedova Sh., Semenkin E., Gasanova T., Minker W. Co-Operation of Biology Related Algorithms for Support Vector Machine Automated Design. In

proceedings of the International Conference on Engineering and Applied Sciences Optimization (OPT-i’2014), pp. 1831-1837.

59. Akhmedova, Sh., Semenkin, E. Co-Operation of Biology Related Algorithms Meta-Heuristic in ANN-Based Classifiers Design. In proceedings of the World Congress on Computational Intelligence (WCCI’14), pp. 867-873, 2014.
60. Akhmedova, Sh., Semenkin, E. Co-Operation of Biology Related Algorithms. In proceedings of 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 2207-2214, 2013.
61. Akhmedova, Sh., Semenkin, E. Data Mining Tools Design with Co-Operation of Biology Related Algorithms. Advances in Swarm Intelligence, Lecture Notes in Computer Science 8794, Y. Tan et al. (Eds.), Part 1, pp. 499-506, 2014.
62. Akhmedova, Sh., Semenkin, E. Development and Investigation of Biologically Inspired Algorithms Cooperation Metaheuristic. In proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO'13), pp. 1417-1418, 2013.
63. Akhmedova, Sh., Semenkin, E., Sergienko, R. Automatically Generated Classifiers for Opinion Mining with Different Term Weighting Schemes. In proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO’2014), Vol. 2, pp. 845-850, 2014.
64. Akhmedova Sh.A., Semenkin E.S. SVM-based classifier ensembles design with co-operative biology inspired algorithm. Вестник СибГАУ. – 2015. - № 1 (16). – С.22-27.
65. Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing, Vol. 17 (2-3), pp. 255-287, 2011. URL: http://sci2s.ugr.es/keel/index.php
66. Altenberg, L. The evolution of evolvability in genetic programming. Advances in Genetic Programming. Cambridge, MA: MIT Press, pp. 47–74, 1994.
67. Altringham, J.D. Bats: Biology and Behaviour. Oxford University Press, 1996.

68. Apostolopoulos, T., Vlachos, A. Application of the Firefly Algorithm for Solving the Economic Emissions Load Dispatch Problem. International Journal of Combinatorics 2011: Article ID 523806, 2011.
69. Bauer, E., Kohavi, R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, Vol. 36, pp. 105-142, 1999. URL: http://citeseer.ist.psu.edu/bauer99empirical.html
70. Bishop, C.M. Theoretical foundation of neural networks. Aston Univ., Neural computing research group, UK Tech. Rep. NCRG-96-024, 1996.
71. Boser, B., Guyon I., Vapnik, V. A training algorithm for optimal margin classifiers. In D. Haussler, editor, proceedings of the 5th Annual ACM Workshop on COLT, pp. 144-152, Pittsburgh, 1992.
72. Breiman, L. Bagging predictors. Machine Learning, Vol. 24 (2), pp. 123-140, 1996.
73. Bukhtoyarov, V., Semenkin, E., Shabalov, E. Neural Networks Ensembles Approach for Simulation of Solar Arrays Degradation Process. In proceedings of HAIS'2012, pp. 186-195, 2012.
74. Chatterjee, A., Mahanti, G.K. Design of a fully digital controlled reconfigurable switched beam conconcentric ring array antenna using firefly and particle swarm optimization algorithm. Progress in Elelectromagnetic Research B., Vol. 36, pp. 113-131, 2012.
75. Chenguang Yang, Xuyan Tu and Jie Chen. Algorithm of Marriage in Honey Bees Optimization Based on the Wolf Pack Search. In proceedings of the International Conference on Intelligent Pervasive Computing, pp. 462-467, 2007.
76. Deb, K. An efficient constraint handling method for genetic algorithms. Computer methods in applied mechanics and engineering, 186(2-4), pp. 311-338, 2000.
77. Deb, K. Optimization for engineering design Algorithms and examples. Prentice-Hall, New Delhi, India, 1995.
78. Dietterich, T.G. Machine learning research: Four current directions. AI Mag., Vol. 18, pp. 97-136, 1997.

79. Dorigo, M., Maria, G. Ant Colony System: a cooperative learning approach. IEEE Trans. Evol. Comput. 1, pp. 53-66, 1997.
80. Eberhart, R.C., Hu, X. Human tremor analysis using particle swarm optimization. In proceedings of the IEEE Congress on evolutionary computation (CEC 1999), pp. 1927-1930, 1999.
81. Eberhart, R.C., Shi, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In proceedings of the Congress on Evolutionary Computation 1, pp. 84-88, 2000.
82. Eberhart, R.C., Shi, Y. Evolving artificial neural networks. In proceedings of the International Conference on Neural Networks and Brain, pp. PL5-PL13, 1998.
83. Eiben, A.E., Smith, J.E. Introduction to evolutionary computation. Springer, Berlin, 2003.
84. European Language Recourses Association. DEFT'08 Evaluation Package. URL: http://catalog.elra.info/product_info.php?cPath=42_43&products_id=1165
85. Farahani, S.M., Abshouri, A.A., Nasiri, B., Meybodi, M.R. A Gaussian firefly algorithm. Machine Learning and Computing, Vol. 1 (5), pp. 448-453, 2011.
86. Fister, I.Jr., Fister, I., Brest, J., Yang, X.S. Memetic firefly algorithm for combinatorial optimization. In proceedings of the Bioinspired Optimization Methods and Their Applications (BIOMA'2012), pp. 75-86, 2012.
87. Frank, A., Asuncion, A. UCI Machine Learning Repository. Irvine, University of California, School of Information and Computer Science, 2010. URL: http://archive.ics.uci.edu/ml.
88. Freund, Y., Schapire, R.E. Experiments with a new boosting algorithm. In proceedings of the International Conference on Machine Learning, pp. 148-156, 1996.
89. Fukuyama, Y., Takayama, S., Nakanishi, Y., Yoshida, H. A particle swarm optimization for reactive power and voltage control in electric power systems. In proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO 1999), pp. 1523-1528, 1999.

90. Gasanova, T., Sergienko, R., Minker, W., Semenkin, E., Zhukov, E. A Semi-supervised Approach for Natural Language Call Routing. In proceedings of the SIGDIAL 2013 Conference, pp. 344-348, August, 2013.
91. Giannakouris, G., Vassiliadis V., Dounias, G. Experimental study on a hybrid nature-inspired algorithm for financial portfolio optimization. SETN 2010, LNAI 6040, pp. 101-111, 2010.
92. Hansen, L.K. Neural network ensembles. Pattern Analysis and Machine Intelligence, L.K. Hansen, P. Salamon IEEE Transaction, Vol. 12 (10), pp.993-1001, 1990.
93. Hassanzadeh, T., Vojodi H., Moghadam, A.M.E. An image segmentation approach based on maximum variance intra-cluster method and firefly algorithm. In proceedings of the 7th International Conference on Natural Computation (ICNC2011), pp. 1817-1821, 2011.
94. Huang, J.-J., Tzeng, G.-H., Ong, Ch.-Sh. Two-stage genetic programming (2SGP) for the credit scoring model. Applied Mathematics and Computation, 174, pp. 1039-105, 2006.
95. Jacobs, R.A. Adaptive mixtures of local experts. Neural Computation, R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, no. 3., pp. 79-87, 1991.
96. Jiménez, F., Verdegay, J. Evolutionary techniques for constrained optimization problems. In proceedings of the 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99), Springer-Verlag, 1999.
97. Joachims, T. Text categorization with Support Vector Machines: Learning with many relevant features. In Machine Learning: ECML-98, Tenth European Conference on Machine Learning, pp. 137-142, 1998.
98. Jordan, M.I., Jacobs, R.A. Hierarchical mixtures of experts and the EM algorithm. Neural Computation, no. 6., pp. 181-214, 1994.
99. Karaboga, D. An Idea Based On Honey Bee Swarm for Numerical Optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department 2005.

100. Kennedy, J. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In proceedings of IEEE Congress on Evolutionary Computation (CEC 1999), pp. 1931-1938, 1999.
101. Kennedy, J., Eberhart, R. Particle Swarm Optimization. In proceedings of IEEE International Conference on Neural Networks, IV, pp. 1942-1948, 1995.
102. Kennedy, J., Eberhart, R.C. A discrete binary version of the particle swarm algorithm. In proceedings of the International Conference on Computational Cybernetics and Simulation, pp. 4104-4108, 1997.
103. Khritonenko, D.I., Semenkin, E.S. Distributed self-configuring evolutionary algorithms for artificial neural networks design. Вестник Сибирского государственного аэрокосмического университета имени академика М.Ф. Решетнева, № 4 (50), С. 112-116, 2013.
104. Krishnanand, K.N., Ghose, D. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. Swarm Intelligence, Vol. 3, No. 2, pp. 87-124, 2009.
105. Levin, E., Tishby, N., Solla, S. A statistical approach to learning and generalization in layered neural networks. In proceedings of the IEEE Special Issue on Neural Networks, C. Lau, Guest Ed., 1990, to be published.
106. Lia, C.-M., Duc, Y.-C., Wua, J.-X., Lind, C.-H., Hoe, Y.-R., Lina, Y., Chen, T. Synchronizing chaotification with support vector machine and wolf pack search algorithm for estimation of peripheral vascular occlusion in diabetes mellitus. Biomedical Signal Processing and Control, Vol. 9, pp. 45-55, 2014.
107. Liang, J.J., Qu, B.Y., Suganthan, P.N., Hernandez-Diaz, A.G. Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization. Zhengzhou University, Computational Intelligence Laboratory, Zhengzhou China, and Nanyang Technological University, Singapore, 2012.
108. Liang, J.J., Shang Zhigang, Li Zhihui. Coevolutionary Comprehensive Learning Particle Swarm Optimizer. In proceedings of Congress on Evolutionary Computation (CEC), pp. 1505-1512, 2010.

109. Mallipeddi, R., Suganthan, P.N. Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real-Parameter Optimization. Nanyang Technological University, Singapore, 2009.
110. Mantegna, R.N. Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes. Physical Review E, Vol. 49, pp. 4677-4683, 1994.
111. Marcano-Cedeño, A., Quintanilla-Domínguez, J., Andina, D. WBCD breast cancer database classification applying artificial metaplasticity neural network. Expert Systems with Applications: An International Journal, vol. 38, issue 8, pp. 9573-9579, 2011.
112. Mendes, R., Kennedy, J., Neves, J. The Fully Informed Particle Swarm: Simpler, Maybe Better. IEEE Transactions of Evolutionary Computation, Vol. 1, No. 1, pp. 204-210, 2004.
113. Michalewicz, Z. Genetic algorithms, numerical optimization and constraints. In proceedings of the Sixth International Conference on Genetic Algorithms and their Applications, Pittsburgh, PA, 1995.
114. Minsky, M.L. Theory of neural-analog reinforcement systems and its application to the brain-model problem: Ph.D. Thesis, Princeton University, Princeton, NJ., 143 p., 1954.
115. Mishra, S., Shaw, K., Mishra, D. A new metaheuristic classification approach for micro array data. Procedia Technology, Vol. 4, pp. 802-806, 2012.
116. Molga, M., Smutnicki, Cz. Test functions for optimization need. 3 kwietnia, 2005.
117. Moore, J., Chapman, R. Application of particle swarm to multiobjective optimization. Department of Computer Science and Software Engineering, Auburn University. 1999.
118. Nakamura, R.Y.M., Pereira, L.A.M., Costa, K.A., Rodrigues, D., Papa, J.P., Yang, X.S. BBA: A binary bat algorithm for feature selection. In proceedings of the 25th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), IEEE Publication, pp. 291-297, 2012.

119. Payne, R.B., Sorenson, M.D., Klitz, K. The Cuckoos. Oxford University Press, 2005.
120. Perrone, M.P., Cooper, L.N. When networks disagree: ensemble method for neural networks. Artificial Neural Networks for Speech and Vision, in R.J. Mammone (Ed.), Chapman & Hall, New York, pp.126-142, 1993.
121. Quinlan, J. R., Induction of Decision Trees. Machine Learning, Kluwer Academic Publishers, Vol. 1, pp. 81-106, 1986.
122. Rodrigues, D., Pereira, L.A.M., Almeida, T.N.S., Papa, J.P., Souza, A.N., Ramos, C.C.O., Yang, X.S. BCS: A Binary Cuckoo Search algorithm for feature selection. In proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), pp. 465-468, 2013.
123. Sasaki, T., Tokoro, M. Evolving Learnable Neural Networks under Changing Environments with Various Rates of Inheritance of Acquired Characters: Comparison between Darwinian and Lamarckian Evolution. Artificial Life, 5 (3), pp. 203-223, 1999.
124. Semenkina, M., Akhmedova, Sh., Semenkin, E., Ryzhikov, I. Spacecraft Solar Arrays Degradation Forecasting with Evolutionary Designed ANN-based Predictors. In proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO'2014), Vol. 1, pp. 421-428, 2014.
125. Sergienko, R., Gasanova, T., Akhmedova, Sh., Semenkin, E., Minker, W. Opinion Mining and Topic Categorization with Novel Term Weighting. In proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis (WASSA2014), pp. 84-89, 2014.
126. Shapiro, L.G., Stockman, G.C. Computer Vision, New Jersey, Prentice-Hall, pp. 279-325, 2001.
127. Shi, Y., Eberhart, R.C. Parameter selection in particle swarm optimization. In proceedings of Evolutionary Programming VII (EP98), pp. 591-600, 1998.
128. Shoghian, Sh., Kouzehgar, M. A Comparison among Wolf Pack Search and Four other Optimization Algorithms. World Academy of Science, Engineering & Technology, Vol. 6, Issue 72, pp. 418-423, 2012.

129. Škraba, A., Semenkin, E., Semenkina, M., Kofjač, D., Žnidaršič, A., Rozman, Č., Maletič, M., Stanovov, V., Akhmedova, Sh. Development of discrete manpower model and determination of optimal control strategies. Материалы XVIII международной научной конференции «Решетневские чтения», посвященной 90-летию со дня рождения генерального конструктора ракетно-космических систем академика М.Ф. Решетнева, Том 2, С. 421-423, 2014 г.
130. Stanovov, V.V., Semenkin, E.S. Self-adjusted evolutionary algorithms based approach for automated design of fuzzy logic systems. Вестник Сибирского государственного аэрокосмического университета имени академика М.Ф. Решетнева, № 4 (50), С. 148-152, 2013.
131. Temurtas, H., Yumusak, N., Temurtas, F. A comparative study on diabetes disease diagnosis using neural networks. Expert Systems with Applications, vol. 36, no. 4, pp. 8610-8615, 2009.
132. Tuba, M., Subotic, M., Stanarevic, N. Modified cuckoo search algorithm for unconstrained optimization problems. In proceedings of the 5th European Computing Conference (ECC’2011), pp. 263-268, 2011.
133. Valian, E., Mohanna, S., Tavakoli S. Improved Cuckoo Search Algorithm for Global Optimization. International Journal of Communications and Information Technology, IJCIT, Vol. 1, Issue 1, pp. 31-44, 2011.
134. Van Rijsbergen, C.J. Information Retrieval (2nd ed.). Butterworth, 1979.
135. Wang, F., Lou, L., He, X., Wang, Y. Hybrid optimization algorithm of PSO and Cuckoo Search. In proceedings of the 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC'11), pp. 1172-1175, 2011.
136. Wei, J., Wang, Y., Wang, H. A hybrid particle swarm evolutionary algorithm for constrained multi-objective optimization. Computing and Informatics, Vol. 29, pp. 701-718, 2010.
137. Wong, L.A., Shareef, H., Mohamed, A., Ibrahim, A.A. Application of binary firefly algorithm for optimal power quality monitor positioning. In proceedings of

the 7th International Conference on Power Engineering and Optimization Conference (PEOCO), pp. 386-390, 2013.

138. Yang, S., Wang, M., Jiao, L. A Quantum Particle Swarm Optimization. In proceedings of the Congress on Evolutionary Computing, Vol. 1, pp. 320-324, 2004.
139. Yang, X.S. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization, Springer, SCI 284, pp. 65-74, 2010.
140. Yang, X.S. Firefly algorithms for multimodal optimization. In proceedings of the 5th Symposium on Stochastic Algorithms, Foundations and Applications, pp. 169-178, 2009.
141. Yang, X.S., Deb, S. Cuckoo search via Levy flights. In proceedings of World Congress on Nature & Biologically Inspired Computing, pp. 210-214, 2009.
142. Zadeh, L.A. Fuzzy Sets. Information and Control, Vol. 8, pp. 338-353, 1965.
143. Zahadat, P., Schmickl, T. Wolfpack-inspired evolutionary algorithm and a reaction-diffusion-based controller are used for pattern formation. In proceedings of the 2014 Conference on Genetic and Evolutionary Computation (GECCO'2014), pp. 241-248, 2014.
144. Zhou, Z.-H., Jiang, Y. NeC4.5: Neural Ensemble Based C4.5 IEEE Transactions Knowledge Data Engineering, Vol. 16 (6), pp. 770-773, 2004.

ПУБЛИКАЦИИ ПО ТЕМЕ РАБОТЫ

Статьи в ведущих рецензируемых научных журналах и изданиях

1. Akhmedova Sh.A., Semenkin E.S. SVM-based classifier ensembles design with co-operative biology inspired algorithm // Вестник СибГАУ. – 2015. - № 1 (16). – С.22-27.
2. Ахмедова Ш.А., Семенкин Е.С. Проектирование систем автоматического определения субъективного мнения кооперативными бионическими алгоритмами // Системы управления и информационные технологии. – 2014. – №3.1 (57). – С. 104-108.
3. Ахмедова Ш.А., Вишневская С.Р., Коромыслова А.А. Интеллектуальные информационные технологии для прогнозирования успешности учебной деятельности абитуриентов // Вестник СибГАУ. – № 3 (54). – 2014. – С. 16-20.
4. Ахмедова Ш.А., Семенкин Е.С. Кооперативный бионический алгоритм безусловной оптимизации // Программные продукты и системы. – № 4 (104). – 2013. – С. 133-136.
5. Akhmedova Sh.A., Semenkin E.S. New optimization metaheuristic based on co-operation of biology related algorithms // Вестник СибГАУ. – № 4 (50). – 2013. – С. 92-99.
6. Akhmedova Sh.A. Development and investigation of the effectiveness of the particle swarm optimization algorithm // Вестник СибГАУ. – № 4 (44). – 2012. – С. 78-79.

Публикации в изданиях, индексируемых в международных базах

7. Akhmedova Sh., Yakimov I., Zaloga A., Burakov S., Semenkin E., Dubinin P., Piksina O., Andryushchenko E. Genetic Algorithm Based X-Ray Diffraction Analysis for Chemical Control of Aluminium Smelters Baths // 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO’2015, France). (**Scopus**).

8. Skraba A., Semenkin E., Kofjac D., Semenkina M., Znidarsic A., Maletic M., Akhmedova Sh., Rozman C., Stanovov V. Modelling and optimization of strictly hierarchical manpower system // 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO'2015, France). (**Scopus**).
9. Zaloga A., Burakov S., Semenkin E., Yakimov I., Akhmedova Sh., Semenkina M., Sopov E. On the Application of Co-Operative Swarm Optimization in the Solution of Crystal Structures from X-Ray Diffraction Data // ICSI-CCI 2015, Part I, LNCS 9140, pp. 89–96. (**Web of Science**, **Scopus**).
10. Akhmedova Sh., Semenkin E. Data Mining Tools Design with Co-Operation of Biology Related Algorithms // ICSI 2014, Part 1, LNCS 8794, pp. 499-506. **Web of Science**, **Scopus**).
11. Akhmedova Sh., Semenkin E. Co-Operation of Biology Related Algorithms Meta-Heuristic in ANN-Based Classifiers Design // Congress on Evolutionary Computations of the IEEE World Congress on Computational Intelligence (CEC WCCI 2014, China), pp. 867-873. (**Web of Science**, **Scopus**).
12. Akhmedova Sh., Semenkin E., Gasanova T., Minker W. Co-Operation of Biology Related Algorithms for Support Vector Machine Automated Design // International Conference on Engineering and Applied Sciences Optimization (OPT-i'2014, Greece), pp. 1831-1837. (**Scopus**).
13. Akhmedova Sh., Semenkin E., Sergienko R. Automatically Generated Classifiers for Opinion Mining with Different Term Weighting Schemes // 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO'2014, Austria), Vol. 2, pp. 845-850. (**Scopus**).
14. Semenkina M., Akhmedova Sh., 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, Austria), Vol. 1, pp. 421-428. (**Scopus**).
15. Akhmedova Sh., Shabalov A. Development and Investigation of Bio-logically Inspired Algorithms Cooperation Metaheuristic // Genetic and Evolutionary

Computation Conference Companion (GECCO'13, Holland), pp. 1417-1418. (**Scopus**).

16. Akhmedova Sh., Semenkin E. Co-Operation of Biology Related Algorithms // 2013 IEEE Congress on Evolutionary Computation (CEC’2013, Mexico), pp. 2207-2214. (**Web of Science**, **Scopus**).
17. Sergienko, R., Gasanova, T., Akhmedova, Sh., Semenkin, E., Minker, W. Opinion Mining and Topic Categorization with Novel Term Weighting // 5th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis (WASSA2014), pp. 84-89, 2014.

Публикации в сборниках трудов конференций

18. Ахмедова, Ш.А. Автоматизированное проектирование нейросетевых предикторов деградации солнечных батарей космического аппарата кооперативными бионическими алгоритмами. Материалы XX научно-технической конференции молодых ученых и специалистов. – Королев: Ракетно-космическая корпорация «Энергия» имени С.П. Королева, С. 656-658, 2014.
19. Ахмедова, Ш.А. Генерирование нейросетевых классификаторов кооперативными бионическими алгоритмами оптимизации. Материалы XVIII Международной научной конференции «Решетневские чтения», посвященной 90-летию со дня рождения генерального конструктора ракетно-космических систем академика М. Ф. Решетнев, Том 2, С. 224-226, 2014.
20. Ахмедова, Ш.А. Исследование эффективности «стайного» алгоритма для задач многокритериальной оптимизации. Молодежь и наука: сборник материалов VIII Всероссийской научно-технической конференции студентов, аспирантов и молодых ученых, посвященной 155-летию со дня рождения К.Э. Циолковского, 2012.
21. Ахмедова, Ш.А. Исследование эффективности «стайного» алгоритма для задач условной оптимизации. Молодёжь и наука: сборник материалов VII Всероссийской научно-технической конференции студентов, аспирантов и

молодых ученых, посвященной 50-летию первого полета человека в космос, 2011.

22. Ахмедова, Ш.А. Исследование эффективности «стайного» алгоритма оптимизации. Сборник статей студентов, аспирантов и молодых ученых по итогам Всероссийской научно-практической конференции, посвященной 80-летию Сибирского государственного технологического университета, Том 2. Красноярск: СибГТУ, 2010. – С. 251-253.
23. Ахмедова, Ш.А. Исследование эффективности условного «стайного» алгоритма оптимизации. «Актуальные проблемы авиации и космонавтики». Труды Всероссийской научно-практической конференции творческой молодежи. Красноярск: СибГАУ, 2011.
24. Ахмедова, Ш.А. Коллективный бионический алгоритм для решения задач безусловной оптимизации с бинарными переменными. Материалы XVIII Международной научной конференции «Решетневские чтения», посвященной 90-летию со дня рождения генерального конструктора ракетно-космических систем академика М. Ф. Решетнева, Том 2, С. 17-18, 2014.
25. Ахмедова, Ш.А. Об эффективности «стайного» алгоритма оптимизации. Труды XLIII Краевой научной студенческой конференции по математике и компьютерным наукам. Красноярск: СФУ, 2010. – С. 9-12.
26. Ахмедова, Ш.А. Последовательный и параллельный «стайный» алгоритм для задач условной и безусловной оптимизации. Актуальные проблемы авиации и космонавтики: материалы VIII Всерос. науч.-практ. конф. творческой молодежи, посвященной Дню космонавтики и 55-летию запуска первого искусственного спутника Земли. Красноярск: СибГАУ, 2012.
27. Ахмедова, Ш.А. Разработка и исследование эффективности «стайного» алгоритма для задач однокритериальной и многокритериальной оптимизации. Наука и инновации в технических университетах: материалы 6-го Всероссийского форума студентов, аспирантов и молодых ученых. Санкт-Петербург: СПбПУ, 2012. – С. 68-70.

28. Ахмедова, Ш.А. Разработка и исследование эффективности «стайного» алгоритма оптимизации. Сборник материалов Всероссийского конкурса научно-исследовательских работ студентов и аспирантов в области технических наук. Санкт-Петербург: СПбГПУ, 2012.
29. Ахмедова, Ш.А. Разработка и исследование эффективности «стайного» алгоритма оптимизации. Сборник научных работ Всероссийского конкурса научно-исследовательских работ студентов и аспирантов в области информатики и информационных технологий, Том 3. Белгород: НИУ БелГУ, 2012. – С. 241-244.
30. Ахмедова, Ш.А. Формирование инвестиционного портфеля с помощью самонастраивающихся интеллектуальных информационных технологий эволюционного типа. Наукоёмкие инновационные проекты молодых ученых: материалы Всероссийского конкурса. Санкт-Петербург: СПбГПУ, 2012. – С. 96-114.
31. Ахмедова, Ш.А. Формирование оптимального инвестиционного портфеля предприятия бинарным «стайным» алгоритмом. Информационные технологии, системный анализ и управление: сборник трудов IX всероссийской научной конференции молодых ученых, аспирантов и студентов, Том 2. Таганрог: Изд-во ТТИ ЮФУ, 2011. – С. 218-220.
32. Ахмедова, Ш.А. Формирование оптимального инвестиционного портфеля предприятия многокритериальным «стайным» алгоритмом. «Решетневские чтения». Материалы XVI Международной научной конференции, посвященной памяти генерального конструктора ракетно-космических систем академика М. Ф. Решетнева, Том 2. - Красноярск: СибГАУ, 2012. – С. 471-472.
33. Ахмедова, Ш.А., Вишневская, С.Р. Метод опорных векторов для прогнозирования успешности обучения студентов. Материалы Всероссийской научно-практической конференции «Информационно-телекоммуникационные системы и технологии», Кемерово, С. 13-14, 2014.

34. Ахмедова, Ш.А., Вишневская, С.Р., Коромыслова, А.А. Прогнозирование успеваемости абитуриентов методами вычислительного интеллекта. Евразийский Союз Ученых, № 6, часть 3, С. 8-10, 2014.
35. Ахмедова, Ш.А., Семенкин, Е.С. Коллективный метод оптимизации на основе стайных бионических алгоритмов. Системный анализ и интеллектуальные технологии. Труды V Международной конференции (САИТ-2013), Том 2, С. 322-331, 2013.
36. Ахмедова, Ш.А., Семенкин, Е.С. Разработка и исследование эффективности стайного алгоритма оптимизации. LAP LAMBERT Academic Publishing. - Saarbrücken, Germany. – 67 с. – ISBN 978-3-659-37617-7.
37. Ахмедова, Ш.А., Семенкин, Е.С., Гасанова, Т.О., Минкер, В.М. Обучение машин опорных векторов для решения задач категоризации текста. Сборник трудов XIV национальной конференции по искусственному интеллекту с международным участием КИИ-2014, Том 2., С. 222-230.
38. Škraba, A., Semenkin, E., Semenkina, M., Kofjač, D., Žnidaršič, A., Rozman, Č., Maletič, M., Stanovov, V., Akhmedova, Sh. Development of discrete manpower model and determination of optimal control strategies. Материалы XVIII международной научной конференции «Решетневские чтения», посвященной 90-летию со дня рождения генерального конструктора ракетно-космических систем академика М.Ф. Решетнева, Том 2, С. 421-423, 2014 г.

Зарегистрированные программные системы

39. Ахмедова Ш.А., Семенкин Е.С. Коллективный метод безусловной оптимизации на основе стайных бионических алгоритмов. Свидетельство №2014610132 о гос. регистрации в Реестре программ для ЭВМ от 09.01.2014.
40. Ахмедова Ш.А., Семенкин Е.С. Коллективный метод безусловной оптимизации на основе стайных бионических алгоритмов для решения задач с бинарными переменными. Свидетельство №2014613715 о гос. регистрации в Реестре программ для ЭВМ от 03.04.2014.

41. Ахмедова Ш.А., Семенкин Е.С. Машины опорных векторов, автоматически сгенерированные коллективными методами условной и безусловной оптимизации на основе стайных бионических алгоритмов. Свидетельство №2014613716 о гос. регистрации в Реестре программ для ЭВМ от 03.04.2014.
42. Ахмедова Ш.А., Семенкин Е.С. Автоматическое генерирование нейронных сетей алгоритмами стайного типа. Свидетельство №2014616237 о гос. регистрации в Реестре программ для ЭВМ от 26.08.2014.
43. Ахмедова Ш.А., Семенкин Е.С. Коллективный метод условной оптимизации на основе стайных бионических алгоритмов. Свидетельство №2013661150 о гос. регистрации в Реестре программ для ЭВМ от 29.11.2013.
44. Ахмедова Ш.А., Семенкин Е.С. Проектирование SVM-классификаторов коллективными бионическими алгоритмами с выбором информативных входов. Свидетельство №2015611847 о гос. регистрации в Реестре программ для ЭВМ от 06.02.2015.

ПРИЛОЖЕНИЕ 1. ИСХОДНЫЕ ДАННЫЕ ДЛЯ ЗАДАЧИ ФОРМИРОВАНИЯ ОПТИМАЛЬНОГО ИНВЕСТИЦИОННОГО ПОРТФЕЛЯ ПРЕДПРИЯТИЯ

№	ЦФО	$П_{ij}$, млн. руб.	R_{ij}	C_i млн. руб.	c_{ij}, млн. руб.
	1. ПТНП				
1	Производство форточных вентиляторов из УВС	3.5	2.2		2.5
2	Организация совместной деятельности по производству посуды из полимеров	5.0	3.0		3.0
3	Литье под давлением изделий из полимеров	3.2	1.8		4.9
4	Изготовление блоков-приборов и других комплектующих для завода холодильников "Бирюса"	6.2	1.9		8.0
5	Продукция ветеринарного и зоотехнического назначения	4.0	1.2		11.3
6	Производство настенных воздухоочистителей-ионизаторов из УВС	11.5	3.0		25.3
7	Производство из УВС нагревателей воздуха для автокарбюраторов	6.7	2.4		15.1
8	Производство тепловентиляторов из УВС	7.5	2.3		15.5
	Всего по ЦФО 1:	***47.6***	***17.8***	***70***	***85.6***
	2. ПЭП				
9	Трубы полиэтиленовые с радиационной обработкой	9.4	3.1		22.0
10	Производство профилей из соэкструдированного жесткого и мягкого ПВХ	7.6	1.5		9.8
11	Экструзия полимеров	4.5	1.4		10.8
12	Производство комплектующих для уплотнителей дверей холодильников	6.0	1.2		11.0
13	Утилизация ПЭТ бутылок	4.0	1.5		17.0
14	Производство напольных офисных покрытий	9.5	2.3		12.5
	Всего по ЦФО 2:	***41***	***11***	***72***	***83.1***
	3. Цех 43				
15	Организация совметной деятельности по газификации жидкого кислорода	4.3	1.5		10.2
16	Утилизация супертоксикантов	4.5	1.2		17.0
17	Производство сжиженного и газообразного аргона	3.7	2.2		6.8
18	Производство сжиженного и газообразного азота	4.6	2.2		5.3
19	Производство сжиженного и газообразного кислорода	3.0	2.2		3.2
	Всего по ЦФО 3:	***20.1***	***9.3***	***35***	***42.5***
	4. ПТПП				

20	Производство геополотна	6.0	2.1		13.6
21	Производство тканой полипропиленовой продукции	10.0	3.1		20.0
22	Производство пропиленовых мешков	6.9	3.3		9.1
	Всего по ЦФО 4:	***22.9***	***8.5***	***32***	***42.7***
	5. Цех 40				
23	Производство дифференциального редуктора	3.0	3.5		3.0
24	Производство электромагнитного инжектора	2.5	4.0		3.5
25	Комплектация ГБО ГИГ	1.0	1.5		0.5
	Всего по ЦФО 5:	***6.5***	***9***	***12***	***7***
	Итого:	**138.1**	**55.6**	**221**	**260.9**

ПРИЛОЖЕНИЕ 2. ИСХОДНЫЕ ДАННЫЕ ДЛЯ ЗАДАЧИ ФОРМИРОВАНИЯ ОПТИМАЛЬНОГО КРЕДИТНОГО ПОРТФЕЛЯ БАНКА

№	Сумма	%	Период	Риск
1	10000000	25	75	0.042
2	5300000	28	80	0.039
3	2400000	25	91	0.029
4	50000000	23	84	0.033
5	1000000	28	64	0.026
6	500000	30	76	0.046
7	250000	37	91	0.044
8	100000	30	86	0.012
9	330000	26	90	0.026
10	5600000	28	88	0.039
11	7300000	25	76	0.02
12	1220000	27	80	0.037
13	2900000	31	84	0.03
14	950000	29	86	0.041
15	4360000	25	88	0.021
16	3700000	26	90	0.035
17	400000	26	79	0.029
18	280000	28	84	0.03
19	5200000	30	91	0.039
20	1280000	27	90	0.04
21	8400000	25	86	0.035
22	670000	29	80	0.015
23	790000	28	84	0.024
24	950000	26	83	0.034
25	580000	27	90	0.038
26	640000	24	91	0.042
27	440000	28	67	0.029
28	460000	28	91	0.018
29	6000000	27	62	0.021
30	7100000	26	78	0.036
31	3260000	27	87	0.027
32	2670000	25	75	0.014
33	620000	27	82	0.038
34	20000000	24	91	0.019
35	10000000	24	90	0.026
36	35000000	22	89	0.022
37	5100000	29	69	0.036
38	865000	30	74	0.021
39	675000	27	63	0.017
40	4650000	29	69	0.026
41	135000	28	70	0.03
42	400000	27	76	0.041
43	1640000	26	87	0.017
44	1380000	29	88	0.021
45	1950000	26	71	0.013
46	1000000	27	85	0.014
47	6900000	27	82	0.016
48	9000000	27	86	0.024
49	22000000	29	91	0.016
50	350000	27	69	0.026
Итого	256695000			
	188500000 - Свободные ресурсы			