Л. А. Гладков
Ю. А. Кравченко
В. В. Курейчик
С. И. Родзин

Институт
Компьютерных
Технологий и
Информационной
Безопасности

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ: МОДЕЛИ И МЕТОДЫ МЕТАЭВРИСТИЧЕСКОЙ ОПТИМИЗАЦИИ

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ: МОДЕЛИ И МЕТОДЫ МЕТАЭВРИСТИЧЕСКОЙ ОПТИМИЗАЦИИ

Монография

Чебоксары
Издательский дом «Среда»
2024

УДК 62(082)
ББК 30я43
И73

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 23-21-00089

Рецензенты:
заслуженный деятель науки РФ, д-р техн. наук, профессор, зав. отделом искусственного интеллекта в энергетике ФГБУН «Институт систем энергетики им. Л.А. Мелентьева Сибирского отделения РАН
Л. В. Массель;
д-р техн. наук, профессор, зав. кафедрой САПР и поискового конструирования ФГБУН «Волгоградский государственный технический университет»
М. В. Щербаков

И73 **Интеллектуальные системы: модели и методы метаэвристической оптимизации** : монография / Л. А. Гладков, Ю. А. Кравченко, В. В. Курейчик, С. И. Родзин. – Чебоксары: Среда, 2024. – 228 с.

ISBN 978-5-907830-56-1

В монографии рассмотрены основные принципы эволюции в живых и искусственных системах, описание эволюционного и генетического поиска. Представлена классификация методов биоинспирированного поиска и области их применения. Анализируются модели и методы роевого интеллекта, а также алгоритмы, инспирированные фауной.

Монография адресована специалистам в области разработки интеллектуальных систем и технологий для решения сложных задач принятия решений и оптимизации. Она может быть полезна студентам, магистрантам и аспирантам соответствующих специальностей.

ISBN 978-5-907830-56-1
DOI 10.31483/a-10639

ОБ АВТОРАХ

Гладков Леонид Анатольевич – к.т.н., доцент, профессор ФГАОУ ВО «Южный федеральный университет» (ЮФУ), г. Таганрог.

Кравченко Юрий Алексеевич – д.т.н., доцент, профессор ФГАОУ ВО «Южный федеральный университет» (ЮФУ), г. Таганрог.

Курейчик Владимир Викторович – д.т.н., профессор, зав. кафедрой Систем автоматизированного проектирования (САПР) ФГАОУ ВО «Южный федеральный университет» (ЮФУ), г. Таганрог.

Родзин Сергей Иванович – к.т.н., доцент, профессор ФГАОУ ВО «Южный федеральный университет» (ЮФУ), г. Таганрог.

ОГЛАВЛЕНИЕ

Предисловие

Из древних трактатов известно, что ученые при создании искусственных систем пытались копировать идеи, заимствованные из живой природы. При этом у них все время возникали вопросы, связанные с ее развитием. Как природа выбирает оптимальные пути своего развития? Почему она экономна? Какие механизмы использует при построении порядка из хаоса? Какие пути эволюции для этого она выбирает? Известный философ-просветитель Дени Дидро в своей работе «Энциклопедия, или Толковый словарь наук, искусств и ремёсел» привел следующее: «Похоже природе в процессе эволюции доставляет удовольствие варьировать один и тот же механизм бесконечным числом различных способов».

Моделирование процессов, протекающих в живой природе, по мнению авторов, позволит применить данный аппарат при проектировании новых интеллектуальных систем. При этом основными подходами при построении искусственных интеллектуальных систем (ИИС) являются модели и методы метаэвристической оптимизации, основанные на природных вычислениях. Данный подход является фундаментом, который совершенствуется, достраивается, создавая сложные иерархические структуры.

Природа имеет предпочтение к определенным устойчивым структурам – формам за счет самоорганизации. Данные процессы являются ключевыми понятиями, развивающейся области исследований метаэвристической оптимизации. Суть данной стратегии заключается в реализации целенаправленного процесса воспроизведения, т. е. появления новых элементов и устранения «плохих» в соответствии с критериями отбора, т. е. реализации принципа – «выживают сильнейшие».

Это медленный путь эволюции природы, реализующий переход от простых самоорганизующихся систем к более сложным. Таким образом она должна представляться не одной, а комплексом моделей, отражающий различные стороны ее поведения. Построение сложных эволюционирующих систем приводит к рассмотрению и использованию комбинированных, многоуровневых, интегрированных и гибридных подходов.

Таким образом, использование моделей, методов и алгоритмов, инспирированных природными вычислениями, является эффективным, перспективным подходом при создании и применению сложных интеллектуальных искусственных систем.

Данные исследования основаны на работах Н. Винера, И. Пригожина, Г. Хакена, Д. Холланда, Д. Гольдберга, Л. Дэвиса, Н. Моисеева, И. Шмальгаузена, А. Северцева, А. Шалыто, Е. Семенкина, И.Ходашинского, М. Дориго, А.Карпенко, Д. Карабога, Синь-Шэ Янга и других российских и зарубежных ученых.

Хочется выразить искреннюю благодарность членам коллектива, аспирантам и студентам кафедры САПР Л. Балюк, А.Кажарову, Д. Запорожцу, Д. Зарубе, Э. Кулиеву, В. Данильченко, В. Курейчику (младшему) за помощь в апробации алгоритмов и программ, директору ИКТИБ Г.Е. Веселову за поддержку исследований.

Мы понимаем, что монография не свободна от недостатков. Все замечания и предложения будут с благодарностью приняты.

Авторы

ВВЕДЕНИЕ

Одной из основных проблем в науке и технике сегодня является разработка теоретических основ, концепций, принципов, математических методов и моделей для эффективного принятия решений в сложных системах. В связи с отсутствием формального математического аппарата для построения адекватных моделей возникает необходимость в поиске, разработке и применении новых перспективных интеллектуальных технологий моделирования и синтеза. При этом важнейшими задачами здесь становятся моделирование процессов, протекающих в живой природе, оптимизация, принятия решений и построение искусственных интеллектуальных систем.

При построении ИИС в настоящее время активно применяются математический аппарат системного анализа, теории сложных систем, графов и гиперграфов, множеств, математического моделирования, а также метаэвристической оптимизации.

Способность строить сложные иерархические системы и оперировать с математическими структурами есть краеугольный камень человеческого интеллекта, а знания и способы работы с ними описываются теорией искусственного интеллекта в виде математических моделей, методов и алгоритмов [1–3].

Интеллектуальные системы сегодня являются самым быстро развивающимся направлением. Его неотъемлемой частью являются компьютерные методы моделирования процессов, протекающих в живой природе. Здесь главной трудностью при построении ИС, является то, что природные системы весьма хаотичны. Однако они обладают другими важными свойствами, такими как: адаптация, самоорганизация, воспроизводство, устойчивостью и гибкостью. Заметим, что в искусственных системах эти свойства могут присутствовать лишь частично.

Новая интеллектуальная ИС проходит процесс самоорганизации на основе путей эволюции, т. е. ее построение сводится к воспроизводству себе подобных систем при благоприятных условиях внешней среды и обладающих нужными свойствами которых не было изначально. Она характеризуется многоуровневостью, единством и непротиворечивостью [3, 4].

Основным отличием эволюции искусственных и естественных систем заключается в том, что искусственная система имеет цель для оптимального или эффективного решения определенного класса за-

дач. Тогда для искусственной системы необходимо использования таких механизмов отбора и воспроизведения, при которых генерируются только эффективные решения поставленной задачи.

При моделировании процессов, протекающих в природных системах выполнение преобразований, происходит аналогично реализации решения оптимизационных задач. Отличие заключается в виде используемых агентов, механизмов их воспроизводства, выживания и отбора. При математической формализации основная идея при решении оптимизационных задач методами, инспирированными природными системами заключается в следующем. С использованием известных принципов генерируется начальная популяция агентов, таким образом, чтобы по возможности использовать всю область поиска. Затем строится целевая функция решаемой оптимизационной задачи. Далее выполняются специфические для каждого из применяемых методов, инспирированных природными системами операторы поиска. При этом поиск проводится не только в рассматриваемой области решений, но и в ее в окрестностях. Данный подход позволяет исследовать полный спектр альтернативных решений для получения эффективных результатов поиска. Затем на основе значений целевой функции производится оценка результатов проведенного поиска. Процесс повторяется итерационно до достижения критерия остановки поиска.

Заметим, что основным преимуществом данных методов является их модульная структура, что позволяет использовать их достоинства и недостатки для проектирования новых вариаций алгоритмов, а также проводить их комбинирование, интеграцию и гибридизацию. Также данные методы, на основе проведенных исследований, демонстрируют эффективную работу при обработке больших массивов данных.

За многие тысячелетия природа в своих структурах научилась сохранять информацию о предыдущих состояниях и процессах, а также выработала специальные механизмы воздействия на процессы экономии и ускорения эволюции. Ученым необходимо правильно моделировать и распределять эти воздействия для учета самоорганизации и смены одной устойчивой структуры другой на различных иерархических уровнях [3, 5].

При этом экспоненциально возрастает сложность оптимизационных задач за счет обработки больших объемов информации, что приводит необходимости модернизации структуры ИИС, а также подходов и методов при решении оптимизационных задач. В качестве одного из таких подходов может рассматриваться использование методов, инспирированных природными системами.

Данный подход, на сегодняшний день, служит известной и эффективной методологией для решения различных оптимизационных задач. Она основана на моделировании естественных процессов, протекающих в природе, которые могут осуществляться как последовательными, так параллельными преобразованиями.

Исследования в этой области интенсивно ведутся многими учеными из РФ и зарубежом. В данной монографии описываются результаты, полученные коллективом научной школы ЮФУ «Когнитивные биоинспирированные технологии в системах проектирования и поддержки принятия оптимальных решений» и посвящается памяти ее основателя заслуженного деятеля науки РФ Виктора Михайловича Курейчика.

Монография состоит из предисловия, введения, четырех глав, заключения и списка литературы.

В первой главе монографии авторы описывают различные подходы к эволюционному развитию природы, рассматривают модели эволюций как методы, применяемые в эволюционном моделировании для решения оптимизационных задач. Также рассмотрены основные положения эволюционного и генетического поиска приведено их формальное описание. Предложены различные типы генетических алгоритмов. Описаны основные операторы генетического поиска.

Во второй главе описываются общие положения теории биоинспирированного поиска, приводится классификация методов биоинспирированной оптимизации, а также рассматриваются области их применения.

В третьей главе монографии рассмотрено понятие роевого интеллекта. Описаны основные модели и методы, основанные на роевом интеллекте. Приведены и рассмотрены алгоритмы муравьиных колоний (ACO), пчелиного роя (ABC), светлячковой (FA), бактериальной (BFO) оптимизации, а также алгоритмы роя саранчи и колонии пауков (SSO).

В четвертой главе рассмотрены другие модели и методы метаэвристической оптимизации, инспирированные фауной. Приведены и описаны алгоритмы обезьяньего поиска (MSA) кукушки (CS), летучих мышей, бабочек монархов, червей, серых волков (GWO), белых кротов, косяка рыб (AFSA), прыжков лягушки (SFLA), горбатых китов (WOA), кузнечиков (GOA) и «мотылек – пламя» (MFOA).

Можно считать, что целью данной монографии является развитие научного направления ЮФУ, направленного на эффективное решение оптимизационных задач.

ГЛАВА 1. МЕТОДЫ ЭВОЛЮЦИОННОГО МОДЕЛИРОВАНИЯ

1.1. Эволюция естественных систем

Термин эволюция произошёл от латинского evolutio- развертывание, развитие (медленные, постепенные, количественные и качественные изменения, приводящие к более высокому уровню развития и организации) впервые был упомянут в биологии европейским ученым Ш. Бонне в 1782 году [6]. Эволюционный процесс приводит к развитию естественных систем за счет целеполагания преемственности поколений, усложнения и совершенствования своих структур.

В настоящее время ученые рассматривают достаточно большое число различных парадигм и моделей эволюции, отличающихся изменчивостью и механизмами выживания. Как известно эволюция естественных систем, согласно Ч. Дарвину, определяется наследственной изменчивостью, борьбой за существование, естественным и искусственным отбором, формируя адаптации, не являющейся внутренней сущностью, а возникающей и развивающейся под воздействием внешней среды [3, 7, 8]. Адаптация в современной науке – это процесс накопления и обработки информации, заключающийся в достижении системой оптимального состояния под воздействием изменяющихся факторов внешней среды [9].

В связи с развитием теории эволюции, данные парадигмы и модели в настоящее время широко используются при создании искусственных интеллектуальных систем. А это приводит в свою очередь к развитию новых междисциплинарных отраслей знаний и науки. Например, Биоинформатика – объединяет в себе биологию, кибернетику, генетику, химию, компьютерные науки, математику и статистику, направлена на изучение и разработку компьютерных методов получения, обработки и анализа больших объемов биологических данных. Бионика – объединяет в себе общую биологию, физику, химию, кибернетику и инженерные науки, направлена на изучение особенностей структуры и функционирования организмов для создания и совершенствования новых приборов, механизмов и систем.

Понятие естественного отбора, как общего принципа существования в живой природе, т. е. реализации направленного процесса «размножения-гибели» впервые было определено известным ученым-исследователем Ч. Дарвиным. Заметим, что именно отбор позволяет закрепляться в процессе эволюции объектам с «лучшими» признаками и удалять объекты с «худшими» в соответствии с определенным критерием, выработанным под воздействием внешней среды [7].

Известно, что одним из основных элементов эволюции являются популяции. В процессе их эволюционирования лежат изменения наследственных структур, которые известный голландский ботаник и генетик Хьюго де Фриз называл мутациями [10]. Они также являются элементарным материалом для эволюционного процесса и возникают под воздействием внешней среды в природных условиях. Изменения наследственных структур – изменчивость бывает направленная (определенная) и ненаправленная. Заметим, что направленная изменчивость несет в себе не изменение признака как такового, а реализует способность к его изменению, т. е. к модификациям [3, 7, 8, 10].

В общем же виде в результате смены поколений в эволюционном процессе всегда происходит усреднение генетического материала. В связи с этим как одним из основных движущих факторов эволюции для получения наилучших результатов развития в природе считается селекция – наука, созданная Ч. Дарвином, представляющая собой несколько форм отбора, включая естественный, бессознательный, методический и искусственный [7]. Также движущимися факторами эволюции, кроме отбора, Ч. Дарвин считал неопределенную изменчивость, и борьбу за существование. Ч. Дарвин в своих работах выделял следующие формы отбора, такие как: стабилизирующий, движущий и дизруптивный [7]. Первая форма направлена на закрепление среднего значения (нормы) признаков в популяции. Вторая – на закрепление одного из типов признаков, имеющих отклонения от нормы в сторону наилучшего значения. И третья – на закрепление множества признаков, имеющих отклонения от нормы.

В отличие от Ч. Дарвина известный советский биолог и теоретик эволюционного учения И.И. Шмальгаузен дополнил и выделил следующие четыре формы отбора: стабилизирующий, балансирующий, движущий и групповой [11–13]. Согласно И.И. Шмальгаузену, при реализации первой формы соотношение признаков организма и внешней среды в процессе эволюции остается неизменным рис. 1.1.

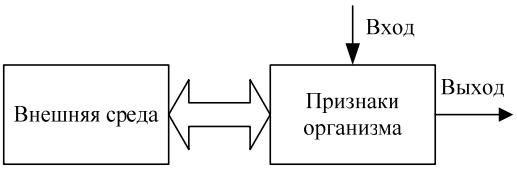


Рис. 1.1. Первая форма отбора

При реализации второй формы в процессе эволюции под воздействием внешней среды вырабатываются такие адаптационные признаки, которые нивелируют ее воздействия на популяцию рис. 1.2.

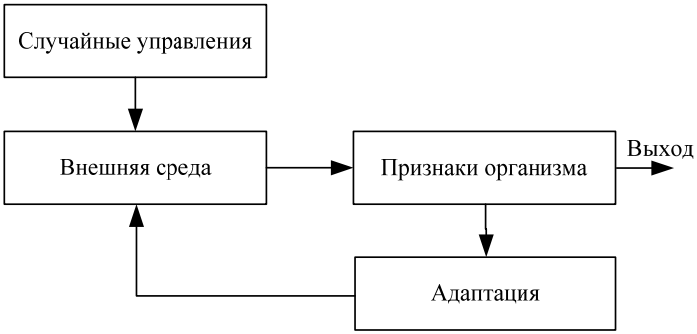


Рис. 1.2. Вторая форма отбора

При реализации третьей формы в процессе эволюции формируются новые адаптационные признаки за счет направленного воздействия внешней среды рис. 1.3.

При реализации четвертой формы в процессе эволюции происходит формирование групповой адаптации рис. 1.4.

Отметим, что основой эволюционных процессов в естественных системах служит популяция (совокупность организмов определенного вида, проживающая в определенном месте в течение длительного промежутка времени, в которой реализуется свободное скрещивание). Каждая особь в популяции уникальна, т. е. обладает своим генотипом и при жизни может передавать наследственную информацию [13, 14].

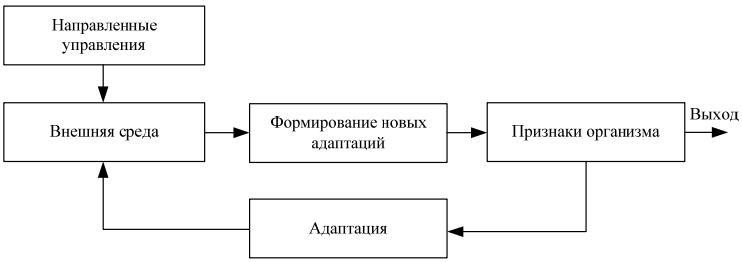


Рис. 1.3. Третья форма отбора

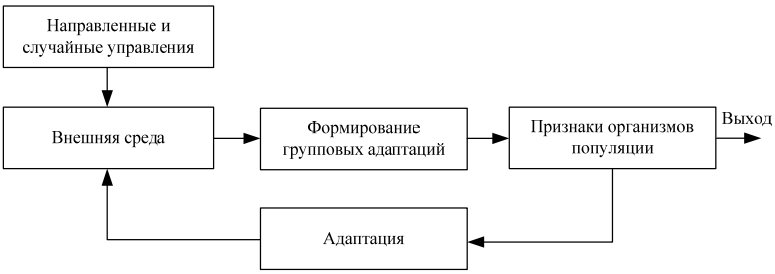


Рис. 1.4. Четвертая форма отбора

В настоящее время ученые рассматривают три механизма передачи наследственной информации при размножении это бесполое, половое и комбинированное [6–8, 13, 15]. Тогда эволюция происходит при взаимодействии следующих повторяющихся процессов воспроизводства, отбора, и мутации.

Приведем простейшую математическую модель процесса эволюции, представленную Г. Дульневым [16, 17]. Рассмотрим систему, в которой изменение во времени некоторого параметра $\dot{q} = \frac{dq}{d\tau}$ будет пропорционально величине этого параметра. Тогда простейшее эволюционное уравнение примет вид:

$$\frac{dq}{d\tau} = \alpha q, \qquad (1.1)$$

где q – это особь в популяции, а α – коэффициент, отражающий характер и скорость изменения процесса эволюции. Решением будет следующее выражение:

$$q = q_0 e^{\alpha\tau}$$

где q_0 – постоянная интегрирования, в начальный момент времени $\tau = 0$, $\alpha \geq 0$. Так как для систем характерно свойство стохастичности, то уравнение (1) примет следующий вид:

$$\frac{dq}{d\tau} = \alpha q + f(\tau), \qquad (1.2)$$

где $f(\tau)$ – параметр, учитывающий флуктуации системы.

В заключении подраздела отметим, что эволюцию можно представить как многоступенчатый итерационный процесс, позволяющий отбирать и закреплять лучшие признаки. Заметим, что природный аналог в течение многих поколений только совершенствовал свою структуру. В связи с этим современная тенденция использо-

вания процессов, протекающих в природных системах при создании моделей, технологий, методик предоставит ученым эффективные алгоритмические средства для решения комплексных задач науки и техники в области построения искусственных систем.

1.2. Концепции и модели эволюции

В настоящее время известно достаточно много различных концепций и парадигм эволюционных учений. Рассмотрим основные из них, которые по мнению авторов, могут быть использованы при построении и развитии интеллектуальных искусственных систем для решения различных задач науки и техники.

Сегодня основой современного развития искусственных систем является результат синтеза различных концепций эволюционного учения, генетики и биологии, подтверждающим, что эволюция является созидательным процессом. Еще ученые древности вели наблюдения в области адаптации живых организмов и применяли свои знания о живой природе для построения искусственных систем.

Ч. Дарвин в 1859 году первым из ученых описал строго научную теорию эволюции, которую в последствии назвали «Дарвинизмом», состоящую из следующих положений [3].

1. В природных системах процесс воспроизводства потомков отличается по многим признакам за счет неопределенной наследственной изменчивости.

2. Процесс воспроизводства потомков в природных системах происходит в геометрической прогрессии, но численность всех организмов за счет естественного отбора в среднем остается постоянной.

3. Основным механизмом при отборе является принцип «выживают сильнейшие».

Модель эволюции Ч. Дарвина можно описать как постепенный многоступенчатый процесс воспроизводства потомков, реализующий механизм «выживания сильнейших» [3, 7, 8, 17]. На рис. 1.5. приведена условная упрощенная схема данной модели эволюции.

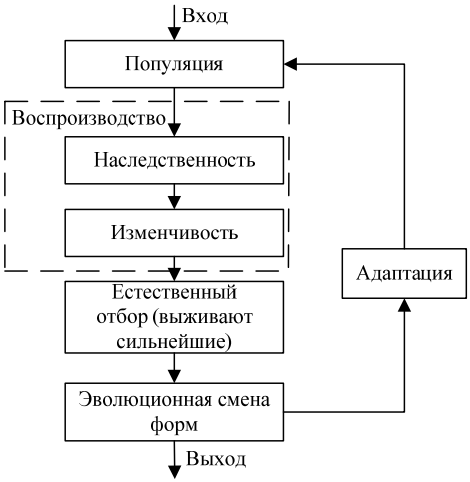


Рис. 1.5. Условная упрощенная схема модели эволюции Ч. Дарвина

Опишем работу блоков, представленной схемы более подробно. В первом блоке содержится популяция. Затем реализуется блок воспроизводства на основе неопределенной наследственной изменчивости с учетом воздействия внешней среды. Далее реализуется, согласно Ч. Дарвину, блок естественного отбора, основанный на механизме «выживают сильнейшие». И в последнем блоке происходит эволюционная смена форм, т. е. лучшие решения, выживают в результате реализации модели эволюции Ч. Дарвина становятся доминирующими в данной популяции.

На основе представленной и описанной схемы приведем словесный алгоритм решения оптимизационной задачи. На первом этапе создается популяция альтернативных решений исходной оптимизационной задачи и задается количество поколений эволюции. На втором этапе реализуется процесс воспроизводства потомков (новых альтернативных решений поставленной задачи. На третьем этапе отбираются лучшие альтернативные решения согласно принципу Ч. Дарвина «выживают сильнейшие». И на последнем этапе «лучшие» решения заменяют «худшие» на основе заданного критерия оптимизации для создания новой популяции и продолжения поиска лучших решений.

Еще до Ч. Дарвина выдающийся биолог, известный ученый-естествоиспытатель Жан Батист Ламарк в своей книге «Философия зоологии», опубликованной в 1809 году, описал стройную и целостную

теорию эволюции живого мира, получившей в дальнейшем название «ламаркизм». В ней он описывает внешние и внутренние факторы, оказавшие непосредственное влияние на весь ход эволюции, также Ламарк утверждал, что самым важным в процессе развития является природная целесообразность, что организмы эволюционируют, приспособляясь и усложняют свою структуру за счет своих внутренних свойств и прямого воздействия внешней среды [18].

Тогда Модель эволюции Ж. Ламарка можно описать как итерационный процесс появление форм, более сложных и совершенных за счет наследования благоприобретенных признаков и приспособляемости к воздействиям внешней среды [3, 8, 17, 18]. На рис. 1.6 приведена условная упрощенная схема такой модели эволюции.

Опишем работу блоков, представленной схемы более подробно. В первом блоке также, как и в модели Ч. Дарвина, содержится популяция. Затем в отличие от модели Ч. Дарвина здесь реализуется блок воспроизводства на основе наследования благоприобретенных признаков, а также выполнение блока приспособляемости организмов к прямым воздействиям внешней среды. Далее также, как и в модели Ч. Дарвина, реализуется отбор организмов, имеющих лучшие благоприобретенные признаки. И в последнем блоке происходит эволюционная смена форм, где лучшие решения закрепляются в новой популяции.

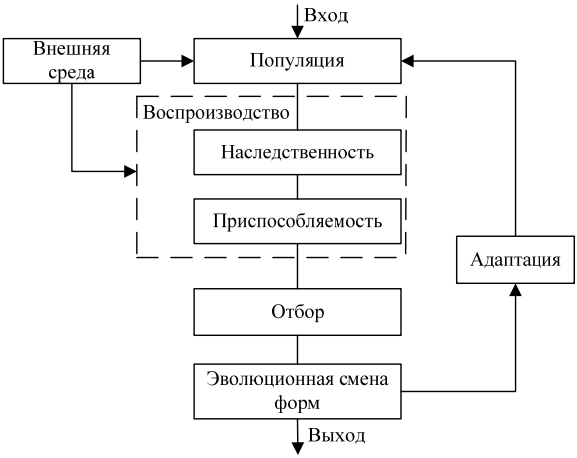


Рис. 1.6. Условная упрощенная модифицированная схема модели эволюции Ж. Ламарка

На основе представленной и описанной схемы приведем словесный алгоритм решения оптимизационной задачи. На первом этапе

также создается популяция альтернативных решений исходной оптимизационной задачи и задается количество поколений эволюции. На втором этапе реализуется процесс воспроизводства новых альтернативных решений поставленной задачи на основе учета предыдущих знаний. На третьем этапе производится изменение параметров поиска с учетом воздействия сигналов внешней среды. Затем отбираются лучшие альтернативные решения на основе заданного критерия оптимизации для создания новой популяции и продолжения поиска.

В 1886 году голландский ботаник и один из основателей генетики и мутационной теории Хуго де Фриз представлял эволюцию в виде скачкообразного процесса. Он утверждал, что новые виды в эволюции возникают не путём постепенного накопления непрерывных случайных изменений, как считал Ч. Дарвин, а путём внезапного появления резких скачков в развитии, названных им мутацией, что приводило к изменению признаков и превращало один вид в другой [10].

Тогда модель эволюции Хуго де Фриза можно описать как скачкообразный процесс появления новых форм за счет мутационной изменчивости организмов. Данная модель подтверждает палеонтологическую теорию социальных и географических катастроф, приводящих к резкому изменению видов и популяций и проявляющихся всего несколько раз за миллионы поколений [3, 8, 10, 17]. На рис. 1.7 приведена условная упрощенная схема такой модели эволюции.

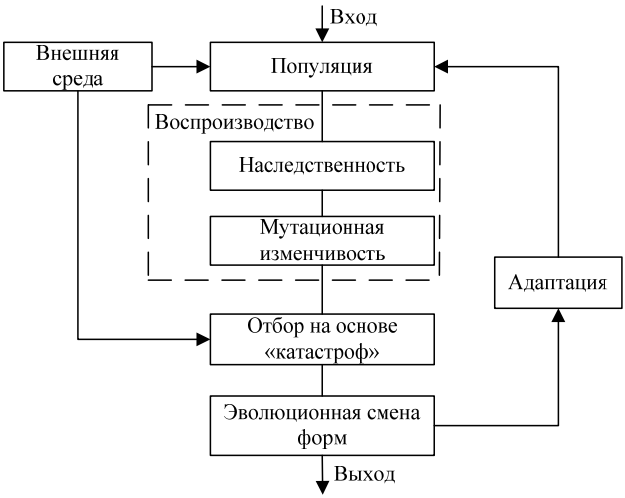


Рис. 1.7. Условная упрощенная модифицированная схема модели эволюции Хуго де Фриза

Опишем работу блоков, представленной схемы более подробно. В первом блоке также, как и в предыдущих моделях, содержится популяция. Затем в отличие от других моделей здесь реализуется блок воспроизводства с учетом мутационной изменчивости. Далее реализуется отбор организмов, после последствий проявления катастроф под воздействиями внешней среды, что приводит к сокращению популяции. И в последнем блоке происходит эволюционная смена форм, где оставшиеся решения составят новую популяцию с новыми качествами.

На основе представленной и описанной схемы приведем словесный алгоритм решения оптимизационной задачи. На первом этапе также создается популяция альтернативных решений исходной оптимизационной задачи и задается количество поколений эволюции. На втором этапе реализуется процесс воспроизводства новых альтернативных решений поставленной задачи с учетом применения мутационных преобразований. На третьем этапе с учетом изменения технического задания (катастрофа), т. е. под воздействием внешней среды производится отбор на основе на основе заданного критерия оптимизации для создания новой популяции, изменения параметров и режимов поиска.

И.И. Шмальгаузен – советский биолог, академик АН СССР, всемирно известный теоретик эволюционного учения XX столетия. В своих работах он исследовал механизмы эволюционного процесса и индивидуального развития организмов как саморегулирующихся систем, и изложил эволюционную теорию с позиций кибернетики и теории информации, ввел понятие обратной связи, а также микро-макро- и мета эволюции [11, 12]. Согласно И.И. Шмальгаузену большое количество информации через различные органы поступает в организмы из окружающей среды. При этом всякий организм приспосабливается в своем индивидуальном развитии не только к среде, но и к биологической работе всех своих частей в определенной среде. Все корреляции и координации в целостном организме происходят в результате совместной работы мобилизационных структур организма. При этом «внешние факторы не имеют специфического формообразовательного значения. Они служат для переключения определенного механизма, направляющего развитие по одному или другому из существующих уже, т. е. полностью детерминированных, путей («каналов») развития. В

связи с этим зависимость от факторов внешней среды полностью теряет своё формообразовательное значение и авторегуляция переходит в выраженное автономное развитие [11, 12].

Согласно, вышеописанному, модель эволюции И.И. Шмальгаузена можно представить как авторегулируемый процесс, основанный на обратных связях и определенных путях развития [8, 11, 12]. На рис. 1.8 приведена условная упрощенная схема такой модели эволюции.

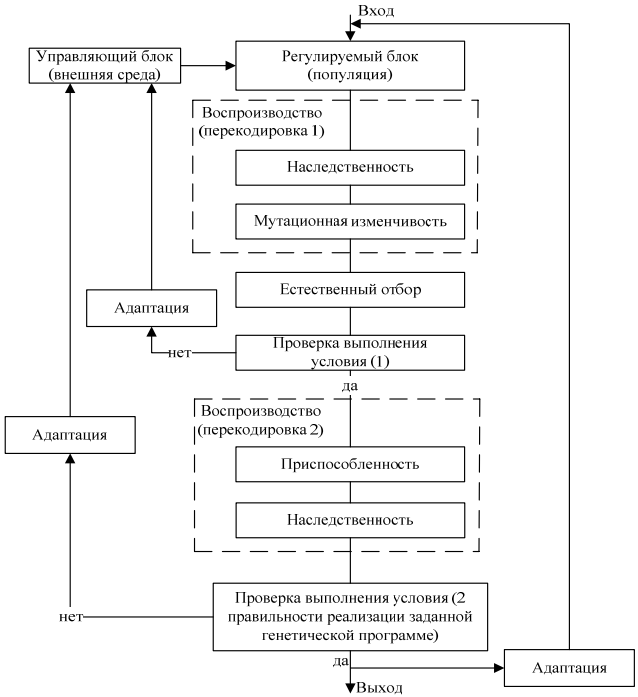


Рис. 1.8. Условная упрощенная модифицированная схема модели эволюции И. Шмальгаузена

Опишем работу блоков, представленной схемы более подробно. В первом блоке содержится популяция. При чем, согласно И.И. Шмальгаузену, она выступает в роли регулирующего блока, а управляющим блоком в этой схеме является внешняя среда. Далее

здесь реализуются два блока воспроизводства в одном с учетом мутационной изменчивости, где производится первая перекодировка информации, т. е. превращение генетической в фенотипическую и во втором – прошедших естественный отбор с учетом адаптационной приспособленности. При этом отбор здесь выступает как процесс авторегуляции. Также здесь вводятся два блока проверки выполнения логических условий. Первый соответствие выполнения условиям регулятора, второй – правильности реализации заданной генетической программе. Далее реализуется отбор и эволюционная смена форм для составления и реализации новой популяции.

На основе представленной и описанной схемы приведем словесный алгоритм решения оптимизационной задачи. На первом этапе также создается популяция альтернативных решений исходной оптимизационной задачи, а также ЛПР задается количество поколений эволюции и параметров поиска. На втором этапе реализуется процесс воспроизводства новых альтернативных решений поставленной задачи с учетом мутационной изменчивости, элитного отбора и адаптации. На третьем этапе также выполняется процесс воспроизводства новых альтернативных решений задачи согласно заданной ЛПР генетической программы. Далее выполняется отбор, проверка условий критериям оптимизации, адаптация и новый виток эволюции.

Сэр Карл Раймунд Поппер австрийский и британский философ и социолог. В 1972 году в своей работе «Объективное знание: эволюционный подход» сформулировал свои теории «передовой модели» и «активного дарвинизма». Согласно К.Р. Попперу живые организмы сами имеют цели и действуют в соответствии с этими целями, каждый из которых управляется центральным органом управления. При этом мутации в генах, определяющих структуру контроля, могут затем вызвать радикальные изменения в поведении, предпочтениях и целях, не оказывая влияния на признаки организма. Им интерпретирована эволюция Ч. Дарвина в виде триады: дедуктивизм – отбор – устранение ошибок и излагается следующим образом. Проблемы, возникающие в процессе эволюции всегда решаются методом проб и ошибок. При этом популяция является пробным решением, анализируемым, выбирающим под себя окружающую среду и преобразующим ее используя те механизмы управления для устранения ошибок, которые выработались в процессе эволюции. Таким образом эволюционный процесс согласно

К.Р. Попперу выражается следующей последовательностью событий: существование исходной проблемы, выработка пробных решений в процессе эволюции, устранение ошибок (неудачных форм) и появление новой проблемы [19].

Тогда модель К.Р. Поппера можно описать как процесс, реализующий иерархическую систему гибких механизмов управления, в которых мутация интерпретируется как метод случайных проб и ошибок, а отбор является одним из способов управления для устранения ошибок при взаимодействии с внешней средой [3, 8, 17, 19]. На рис. 1.9 приведена условная упрощенная схема такой модели эволюции.

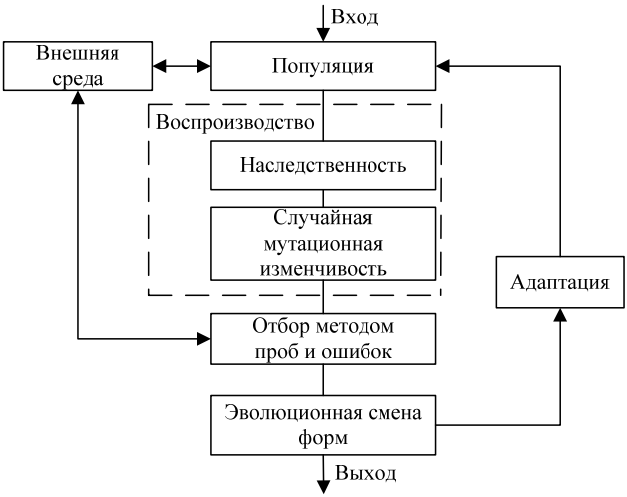


Рис. 1.9. Условная упрощенная модифицированная схема модели эволюции К. Поппера

Опишем работу блоков, представленной схемы более подробно. В первом блоке формируется популяция пробных решений. Затем реализуется блок воспроизводства на основе наследственности и мутационной изменчивости с учетом прямого и обратного воздействия внешней среды. Далее реализуется блок отбора под управлением внешней среды. И в последнем блоке происходит эволюционная смена форм, т. е. создается новая проблема (популяция), которая и участвует в дальнейшем эволюционном процессе.

На основе представленной и описанной схемы приведем словесный алгоритм решения оптимизационной задачи. На первом этапе создается пробная популяция альтернативных решений исходной оптимизационной задачи и ЛПР задается количество поколений эволюции и параметров поиска. На втором этапе реализуется процесс воспроизводства потомков (новых альтернативных решений поставленной задачи) на основе скрещивания и мутационных преобразований. На третьем этапе отбираются альтернативные решения согласно критерию оптимизации под управлением ЛПР. И на последнем этапе создается новая популяция для продолжения поиска.

Мотоо Кимура японский биолог. В 1968 году в своей книге «Молекулярная эволюция: теория нейтральности» описал нейтральную теорию молекулярной эволюции. Он утверждал, что подавляющее число мутаций на молекулярном уровне носит нейтральный по отношению к естественному отбору характер. Поэтому изменчивость организмов (особенно в малых популяциях) объясняется не действием отбора, а случайными мутациями, которые нейтральны или почти нейтральны [20].

Тогда модель М.Кимуры можно описать как процесс реализующий нейтральный отбор признаков [8, 17, 20]. На рис. 1.10 приведена условная упрощенная схема такой модели эволюции.

Опишем работу блоков, представленной схемы более подробно. В первом блоке формируется начальная популяция. Затем реализуется блок воспроизводства на основе наследственности и мутационной изменчивости, а далее реализуется блок нейтрального отбора, в котором отбирается ровно половина особей с лучшими и худшими либо средними признаками. И в последнем блоке, как всегда, происходит эволюционная смена форм, т. е. создается новая популяция, которая и участвует в дальнейшем эволюционном процессе.

На основе представленной и описанной схемы приведем словесный алгоритм решения оптимизационной задачи. На первом этапе создается популяция альтернативных решений исходной оптимизационной задачи и ЛПР задает количество поколений эволюции и параметров поиска.

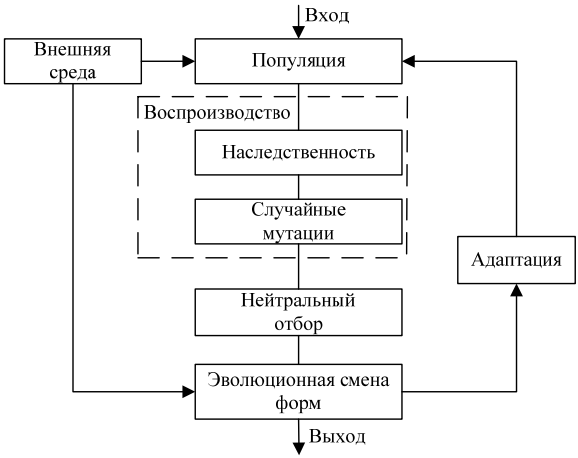


Рис. 1.10. Условная упрощенная модифицированная схема модели эволюции М.Кимуры

На втором этапе реализуется процесс воспроизводства потомков (новых альтернативных решений поставленной задачи) на основе скрещивания и мутационных преобразований. На третьем этапе отбираются лучшие и худшие, либо имеющие среднее значение относительно критерия оптимизации альтернативные решения. И на последнем этапе из отобранных создается новая популяция альтернативных решений для продолжения поиска.

В 1937 году русско-американский генетик и энтомолог-систематик Ф.Г. Добржанский опубликовал книгу Genetics and the Origin of Specie. Это время считается годом появления синтетической теории эволюции. А в 1949 году американский палеонтолог в своих работах использовал выражение «синтетическая теория эволюции». Создатели данной теории и ученые внесшие наибольший вклад в ее развитие – это Ф. Добржанский, Э. Баур, Дж. Хаксли, Э. Майр, И.И. Шмальгаузен, Г.Ф. Гаузе, Н. П. Дубинин, А.С. Северцев и другие. Они работали в разных областях науки и расходились во мнениях по ряду фундаментальных проблем, но вместе выработали ее основные положения, заключающиеся в следующем [11–13, 15, 21].

1. Основной элементарной единицей эволюции считается локальная популяция;

2. Основным материалом для эволюции служит мутационная и рекомбинационная изменчивость;

3. Естественный отбор является основной причиной возникновения и развития адаптаций, а также создает сложные генетические системы;

4. Случайные мутационные и генетические изменения, происходящие в небольшой популяции, являются причинами формирования нейтральных признаков;

5. Появление новых видов осуществляется преимущественно в условиях географической изоляции.

Тогда модель синтетической теории эволюции – это синтез различных концепций и парадигм эволюционных учений, представляющая собой процесс случайных, периодических и скачкообразных мутационных и генетических изменений, попадающих под действие естественного отбора с целью сохранения нейтральных и выгодных сочетаний признаков для дальнейшего размножения и рекомбинации [3, 8, 11–13, 15, 17, 21]. На рис. 1.11 приведена условная упрощенная модифицированная схема такой модели эволюции.

Опишем работу блоков, представленной схемы более подробно. В первом блоке формируется начальная популяция с учетом окружающей внешней среды. Затем реализуется блок воспроизводства на основе наследственности, мутационной и рекомбинационной изменчивости, а далее реализуется блок естественного отбора с целью сохранения нейтральных и выгодных сочетаний признаков. Далее происходит эволюционная смена форм ее адаптация и создание новой популяция, которая и участвует в дальнейшем эволюционном процессе.

На основе представленной и описанной схемы приведем словесный алгоритм решения оптимизационной задачи. На первом этапе создается популяция альтернативных решений исходной оптимизационной задачи и ЛПР задает количество поколений эволюции и параметров поиска. На втором этапе реализуется процесс воспроизводства потомков (новых альтернативных решений поставленной задачи) на основе скрещивания, мутационных и рекомбинационных преобразований.

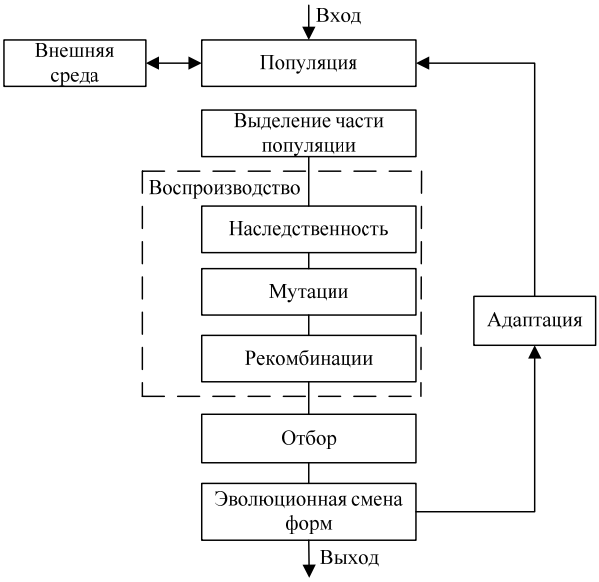


Рис. 1.11. Условная упрощенная модифицированная схема модели синтетической теории эволюции

На третьем этапе отбираются средние и лучшие альтернативные решения относительно критерия оптимизации. И на последнем этапе выполняется процедура адаптации поиска, создается новая популяция и процесс продолжается итерационно.

В заключении подраздела отметим, что авторы считают важным интеграцию различных моделей эволюций. Такая схема приведена на рис. 1.12 [8, 17]. Данный подход к построению эффективных интеллектуальных систем принятия оптимальных решений позволит учесть все достоинства рассмотренных моделей эволюции.

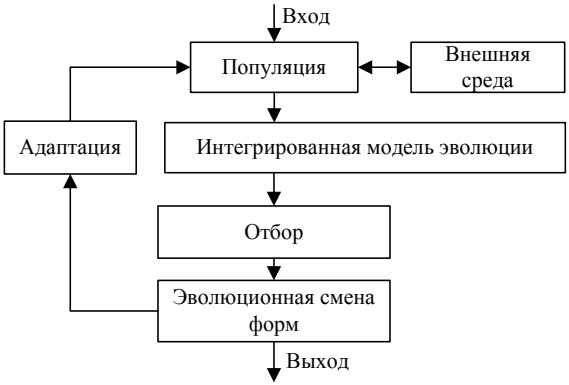


Рис. 1.12. Условная упрощенная модифицированная интегрированная схема эволюции

При чем возможно рассматривать интеграцию как двух моделей, так и всех.

1.3. Алгоритмы и методы эволюционного моделирования

При решении различных оптимизационных задач эффективно используют стратегии, концепции, методы, механизмы эволюционного моделирования.

Методы эволюционного моделирования сегодня – это направление исследований в теории и практике искусственного интеллекта. Ключевой идеей этих методов является использование в качестве основных операций некоторых формализованных принципов естественного эволюционного отбора [3, 8, 22–24]. Методы эволюционного моделирования являются приближенными (эвристическими) эффективными методами решения задач оптимизации и принятия решений. Они в основном основаны на статистическом подходе к исследованию ситуаций, а также при поиске решений используют итерационное приближение.

В настоящее время эволюционное моделирование – это эффективная оптимизационная методология, основанная на механизмах натуральной селекции, генетики и принципа «выживания сильнейших», опирающаяся на информацию, накопленную в процессе эволюции [3, 8, 21–23]. Основное отличие методов эволюционного моделирования от других поисковых процедур заключается в следующем: работают с закодированным множеством параметров рассматриваемой задачи, реализуют сразу несколько альтернатив на

заданном множестве решений, используют целевую функцию (ЦФ), а не ее различные приращения и наконец применяют вероятностные правила анализа поставленных задач.

Одним из основных методов эволюционного моделирования являются эволюционные алгоритмы.

Эволюционные алгоритмы (ЭА) как новое направление оптимизации начали развиваться в начале 80-х гг. [8, 23–27]. В настоящее время их начали широко использовать для решения различных практических задач науки и техники, в том числе для решения задач оптимизации и принятия решений. Данные методы позволяют обрабатывать большие области поиска с учётом множества критериев.

Другим основным представителем методов эволюционного моделирования являются генетические алгоритмы [3, 8, 26–31]. Впервые ГА были применены к таким научным проблемам, как распознавание образов и оптимизация. Основой для возникновения ГА послужили модель биологической эволюции и методы случайного поиска.

Л. Растригин отмечал, что случайный поиск возник как реализация простейшей модели эволюции, когда случайные мутации моделировались случайными шагами оптимального решения, а отбор – «устранением» неудачных вариантов [32].

Основателем генетических алгоритмов считается американский учёный, профессор психологии, профессор электротехники и информатики Джон Генри Холланд, который в 1975 году опубликовал свою знаменитую книгу «Adaptation in Natural and Artificial Systems». Он представил свой алгоритм, больше известный как «репродуктивный план Д. Холланда». Его основные шаги следующие [8, 28].

1. Формирование начальной популяция хромосом.

2. Задается число циклов (поколений) G. Устанавливается G = 0.

3. Рассчитывается приспособленность каждой хромосомы и среднее значение по всей популяции.

4. Наращивание циклов G = G + 1

5. Выбор из популяции, согласно их приспособленности двух родительских хромосом.

6. Реализация оператора кроссинговера для создания потомков.

7. Один из потомков сохраняется как член новой популяции для последовательной реализации мутации и инверсии. Полученные потомки также сохраняются в новой популяции.

8. Для сохранения размера популяции часть родительских хромосом заменяется полученными потомками.

9. Производится перерасчет приспособленностей хромосом и популяции.

10. Если число циклов соответствует заданному, то переход к п. 11, если нет, то переход к п. 4.

11. Конец работы алгоритма.

Первый простой генетический алгоритм был описан в книге американского специалиста по информатике Дэвида Эдварда Голдберга на основе работ Д Холланда, в Genetic Algorithms in Search, Optimization, and Machine Learning. Данный алгоритм состоял всего из нескольких простых операций [8, 29]. Сначала случайно генерировалась начальная популяция хромосом (альтернативных упорядоченных и неупорядоченных решений). Далее выполнялся оператор репродукции, т. е. копирование последовательности хромосом и перестановка их частей. Затем последовательная реализация операторов кроссинговера (скрещивания) и мутации. На последнем этапе производился элитный отбор, создание новой популяции и новый виток эволюции. Процесс поиска заканчивался по истечению заданного времени.

Еще один американский профессор информатики Лауренс Дэвид Дэвис в своей книге Handbook Of Genetic Algorithms представил следующий простой генетический алгоритм. Его словесное описание представлено следующим образом [8, 30].

1. Формирование начальной популяция хромосом.

2. Задается время решения Т.

3. Рассчитывается приспособленность каждой хромосомы в популяции.

4. Реализация оператора кроссинговера для создания потомков.

5. Далее реализация мутации и рекомбинации. Полученные потомки также сохраняются в новой популяции.

6. Оценка значений полученных потомков и если их приспособленность лучше родительских, то производится замена.

7. Если время соответствует заданному, то переход к п. 8, если нет, то переход к п. 4.

8. Конец работы алгоритма.

Сравнивая описание этих первых трех простых генетических алгоритмов, можно заметить, что в них реализована одна и таже основная идея моделирования эволюции с некоторыми модификаци-

ями, т. е. выполнением тех или иных процессов и механизмов, происходящих в живой природе, но эти изменения могут существенно повлиять на окончательное качество полученного решения [8].

Приведем некоторые понятия и определения. Все эти методы работают на основе начальной информации, в качестве которой выступает популяция альтернативных решений (хромосом), при чем каждая хромосома в популяции имеет определенный уровень качества, который характеризуется значением ЦФ (в литературе иногда называется функция полезности, приспособленности или пригодности (fitness)). Эта функция для сравнения хромосом друг с другом и выбора лучших. Хромосома – это любое альтернативное решение задачи, которое кодируется в виде последовательности конечной длины в некотором алфавите. В свою очередь хромосома состоит из дискретных элементов, называемых генами. Они размещаются по позициям – локусам и могут иметь различные функциональные значения (аллели). Часто начальные хромосомы называют родителями. Родители выбираются из популяции на основе заданных правил, для процесса воспроизводства потомков. Затем в результате одного цикла эволюции (поколения) из родителей и потомков, имеющих лучшие признаки создается новая популяция хромосом.

Приведем формальное определение ЭА и ГА [8].

ЭА и ГА=(Pio,N,PiT,k,T,Lj,A,(ЦФ,ОГР,ГУ),ГО, ОМГ,t), (3.1)

где Pio – начальная популяция хромосом; N – мощность (размер) популяции; T (G) = 0,1,2,… – номер генерации; Lj – длина i-ой хромосомы; (ЦФ,ОГР,ГУ) – целевая функция, ограничения и граничные условия; ГО – генетические операторы, ОМГ – оператор миграции, t – критерий окончания работы ЭА и ГА.

ЭА и ГА нацелены на оптимизацию целевой функции. Их эффективная реализация зависит от способа представления и отбора решений, выбора операторов поиска и принципов формирования начальной популяции альтернативных решений.

Процесс реализации данных методов заключается в последовательном выполнении следующих операций [8, 33]. Предварительно на основе известных принципов создается популяция решений рассматриваемой задачи. Далее процесс представляет собой последовательность циклов выполнения операторов преобразований с определенной вероятностью. При чем в каждом цикле происходит оценка всех

альтернативных решений решаемой задачи. Далее альтернативные решения, в зависимости от полученной оценки, либо участвуют в создании новой популяции, либо устраняются из дальнейшего поиска. Процесс поиска считается завершенным по выполнении критерия окончания работы алгоритма. Отметим, что критериев окончания работы алгоритма может быть три. Первый – это заданное количество поколений, второй – время решения и третий – нахождение оптимального решения, если оно известно заранее. На основе вышеописанного представим и опишем базовую структуру генетического поиска любой оптимизационной задачи рис 1.13 [8].

Согласно данной схеме, на первом этапе на основе известных принципов генерируется начальная популяция (множество решений) заданной мощности, поставленной оптимизационной задачи. Заметим, мощность популяции является величиной постоянной за все время поиска. Во втором блоке строится целевая функция на основе выбранного критерия или множества критериев, рассматриваемой оптимизационной задачи. Далее вычисляется значение ЦФ каждого решения и среднее значение ЦФ всей популяции. Заметим, что при построении ЦФ необходимо учитывать ограничения и граничные условия решаемой задачи. В следующем блоке производится сортировка решений от лучшего к худшему согласно значениям ЦФ. Затем на основе оператора селекции выбираются альтернативные решения для реализации различных операторов генетического поиска (перестановок). После реализации всех операторов производится объединение начального множества решений с полученным новым множеством потомков. Далее производится перерасчет значения ЦФ каждого решения и среднего значения ЦФ всей популяции. В следующем блоке, согласно модели эволюции Ч. Дарвина, отбираются лучшие решения на основе значения ЦФ. При этом необходимо учитывать, что мощность полученной популяции должна быть равной мощности начальной. Новая образованная популяция становится текущей и процесс продолжается итерационно до выполнения критерия остановки.

Рис. 1.13. Базовая структура генетического поиска

В отличие от базовой структуры генетического поиска, основанной только на одной модели эволюции Ч. Дарвина, авторы предлагают следующую модифицированную структуру и технологию генетического поиска, основанную на интеграции различных моделей эволюции, описанных в параграфе 1.2 и представленную на рис. 1.14.

В качестве модификации в известную архитектуру генетического поиска предлагается ввести несколько новых блоков. Это блок «внешней среды», согласно описанным моделям эволюции Ж.Б. Ламарка [18] и К. Поппера [19], блок эволюционной адаптации, согласно описанной модели И. Шмальгаузена [11, 12], блок «неперспективных решений», согласно описанной модели эволюции Ж.Б. Ламарка [18], блок «отбора» в котором будут использованы следующие механизмы: «выживание сильнейших» Ч. Дарвина [7], наследование «благоприобретенных признаков» Ж.Б. Ламарка [18], «нейтрального отбора» М. Кимуры [20] и «катастроф» Х. де Фриза [10].

Рассмотрим более детально модифицированную структуру генетического поиска и опишем работу ее блоков.

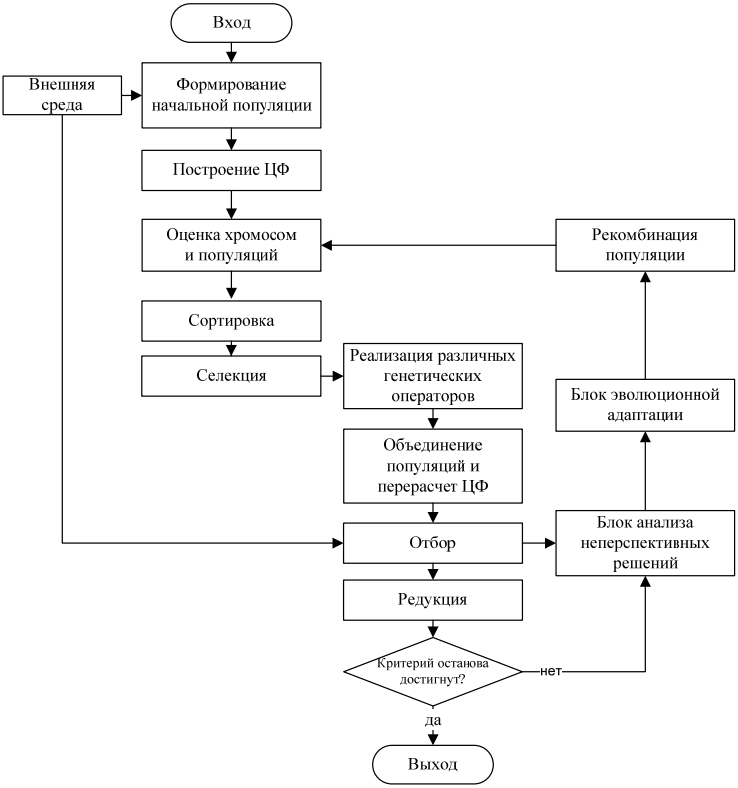


Рис. 1.14. Модифицированная структура генетического поиска

Предварительно задается область поиска допустимых решений с учетом ограничений и граничных условий решаемой оптимизационной задачи. Затем на основе известных принципов [32] создаётся начальная популяция альтернативных решений решаемой оптимизационной задачи с учетом ЛПР. Далее на основе выбранных критериев оптимизации строится целевая функция. Заметим, критериев у решаемой задачи может много, а целевая функция одна. Затем на основе ЦФ производится оценка всех альтернативных решений начальной популяции, также вычисляется среднее значение ЦФ всех решений в популяции. Далее производится сортировка этих решений и реализуется оператор репродукции (селекции). В

следующем блоке реализуются различные генетические преобразования, т. е. формируются новые решения – потомки на основе выполнения различных генетических операторов [32]. Затем происходит объединение родительской популяции решений с популяцией решений-потомков. В следующем блоке в зависимости от управляющих воздействий внешней среды (ЛПР) может реализовываться какой-либо из несколько механизмов отбора, таких как: «выживание сильнейших» Ч. Дарвина, наследование «благоприобретенных признаков» Ж.Б. Ламарка, «нейтрального отбора» М. Кимуры и «катастроф» Х. де Фриза. Далее часть отобранных решений передается в блок «анализа неперспективных решений», где им на основе анализа присваивается определённая метка (перспективное, неперспективное, тривиальное и др.) с учетом экспертных знаний ЛПР. Такое ранжирование решений за счет структуризации позволяет динамически управлять поиском, т. е. сужать его за счет элитных решений, либо расширять за счет неперспективных решений, либо усреднять за счет тривиальных решений. Это значительно позволит повысить эффективность работы предложенного генетического алгоритма. Таким образом, в данной структуре появляется дополнительный механизм для самоадаптации и настройки параметров поиска [88, 89]. Далее выполняется оператор редукции уменьшающий размер нашей популяции до заданной величины. В следующем блоке идет проверка критерия остановки. Заметим, что в предложенном алгоритме используются все три. Это либо число генераций, либо время, либо получение набора квазиоптимальных решений поставленной задачи. После этого результаты через блок «анализа неперспективных решений» передаются в блок эволюционной адаптации. Данный блок оказывает управляющее воздействие на процесс перестройки нашей популяции в блоке рекомбинации за счет анализа текущей ситуации, т. е. производится либо сужение, либо расширение области поиска при создании новой динамически изменяемой популяции альтернативных решений. Эволюционный процесс работы заканчивается при достижении того или иного определенного критерия окончания работы алгоритма.

В заключении подраздела отметим, что данные методы эффективно и одновременно способны анализировать различные области поискового пространства, а также позволяют быстро адаптироваться к нахождению новых, с точки зрения значения ЦФ, лучших областей, а за счет в ведения новых блоков и механизмов поиска выходить из локальных оптимумов.

1.4. Генетические операторы и их модификации

Основой работы любого генетического алгоритма, являются те или иные генетические преобразования, реализующие шаги из последовательности действий и названные генетическими операторами. В настоящее время генетических операторов очень много. Кратко рассмотрим основные из них [3, 8, 26–31, 33].

Оператор селекции в литературе часто встречается название оператор репродукции, – это инструкция при реализации которой более приспособленные хромосомы, т. е. обладающие более «лучшими» признаками, отбираются и закрепляются для воспроизводства потомков.

Из литературных источников известно большое число операторов селекции, основанных на моделировании «колеса рулетки», «заданной шкалы», «турнира», «элитизма». Кроме приведенных существуют и множество других, но все они подразделяются на три группы: случайные, направленные или комбинированные [8].

При случайной селекции частота R образования родительских пар полностью определяется численностью популяции N:

$$R = \frac{\beta}{N(N-1)}, \tag{1.3}$$

где β – коэффициент селекции, принимающий в зависимости от условий внешней среды значение 1 – 4.

При направленной селекции предпочтение отдается хромосомам с близкими и «лучшими» признаками (значение ЦФ), тогда вероятность $P_r(ОР)$ образования родительских пар рассчитывается на основе следующего выражения:

$$P_r(ОР) = \frac{\beta}{ЦФ(P_k^t)} \Big/ \sum_{i=1}^{N} ЦФ(P_k^t),\ k = \overline{1,N} \tag{1.4}$$

где P_k^t – это хромосома.

Также направленную селекцию можно проводить, используя следующие принципы «близкое» родство, «дальнее» родство, код Грея, Хеммингово расстояние и т.д.

При комбинированной селекции одна хромосома выбирается случайным образом, а вторая –на основе выражения (1.4).

Если $ЦФ(P_k^t) < ЦФ_{ср}$, где $ЦФ_{ср}$ – среднее значение ЦФ в популяции, то оператор репродукции моделирует процесс естественного отбора. При отборе различных случайных хромосом происходит расширение области поиска, повышается разнообразие, что в некоторых случаях позволяет выходить из локальных оптимумов.

Селекция считается эффективной если реализует переход из одной подобласти поиска в другую. Это повышает вероятность нахождения глобального оптимума целевой функции.

Следующим оператором генетического поиска является кроссинговер, в литературе часто встречается название кроссовер или скрещивание (ОК). Кроссинговер – это инструкция при реализации которой, процесс получение потомков производится на основе преобразования (скрещивания) родительских хромосом или их частей. Алгоритм выполнения этого процесса представим в виде словесного описания [8, 28, 31].

1. На основе реализации оператора селекции из текущей популяции выбираются две родительские хромосомы $P_1 = a_1, a_2,...,a_L$ и $P_2 = a'_1, a'_2,.., a'_L$.

2. В этих выбранных хромосомах случайным образом выбирается точка разреза r, ее еще называют точкой оператора кроссинговера. Она выбирается из множества {1, 2,…,L-1}, где L - длина хромосомы.

3. Потомки формируются путем преобразования (скрещивания), т.е. происходит частичный обмен признаками в родительских хромосомах согласно следующему правилу:

$$P_1{}' = a_1, a_2, \ldots, a_k, a'_{k+1}, \ldots, a'_L,$$
$$P_2{}' = a'_1, a'_2, \ldots, a'_k, a_{k+1}, \ldots, a_L.$$

Отметим, что после выполнения кроссинговера имеем две родительские хромосомы и дополнительно получаем две хромосомы-потомка. Например, имеем две родительские хромосомы P_1: (00000), P_2: (11111). Тогда получим хромосомы-потомки согласно представленному алгоритму.

P_1: 0 0 | 0 0 0
P_2: 1 1 | 1 1 1

P'_2: 0 0 | 1 1 1
P'_1: 1 1 | 0 0 0.

В настоящее время известно большое число различных операторов кроссинговера. Это двухточечный, многоточечный, частично-соответствующий, упорядоченный, циклический, универсальный и др., а также их различные модификации, так как их структура в основном и определяет эффективность всего генетического поиска.

Заметим, что вероятность реализации оператора кроссинговера лучших хромосом с худшими по значению ЦФ, должна уменьшаться на последних этапах поиска, а вероятность скрещивания лучших хромосом должна увеличиваться, так как позволяет закреплять желаемые признаки.

Следует отметить, что поиск универсальных и специализированных ОК, ориентированных на решение заданного класса задач продолжается.

Согласно эволюционной теории Х. де Фриза процессы преобразования происходят в природе толчками и основой таких процессов является мутационная изменчивость [10].

Мутация – это процесс, изменяющий только взаимное расположение генов, а не структуру и размер в родительской хромосоме при создании потомков. Алгоритм выполнения этого процесса представим в виде словесного описания [8, 28, 31].

1. Из текущей популяции выбирается родительская хромосома $P_1 = (a_1, a_2, a_3,...,a_{L-2}, a_{L-1}, a_L)$.

2. В ней случайным образом из множества {1, 2,...,L} определяются две позиции (два гена), например 1 и L.

3. Выбранные гены, соответствующие данным позициям, переставляются местами друг с другом формируя хромосому-потомок.

4. Гены, соответствующие выбранным позициям, переставляются, и формируется новая хромосома-потомок $P_1' = (a_L, a_2, a_3,...,a_{L-2}, a_{L-1}, a_1)$.

Например, имеем родительскую хромосому P_1 : 0 1 1 | 0 1 1, выбираем 3 и 4 позиции, тогда согласно алгоритму, имеем:

P_1 : 0 1 1 | 0 1 1

P'_1 : 0 1 0 | 1 1 1

В настоящее время известно также большое число различных операторов мутации. Это двухточечный, многоточечный, позиционный, кластерный, универсальный и др., а также их различные модификации, так как их структура предотвращают потерю генетического материала.

Оператор инверсии – это инструкция при реализации которой, производится инвертирование родительской хромосомы (или ее части) для формирования хромосомы-потомка. Алгоритм выполнения этого процесса представим в виде следующего словесного описания [8, 28, 31].

1. Из текущей популяции выбирается родительская хромосома $P_1 = (a_1, a_2, a_3,...,a_{L-2}, a_{L-1}, a_L)$.

2. В ней случайным образом из множества {1,2,...,L-2} определяется точка разреза между двумя генами (позициями), например 1 и L-2.

3. Новая хромосома-потомок формируется из P_1 путем инверсии сегмента, который лежит справа от выбранной позиции 1. В процессе выполнения получили: $P_1' = (a_1, a_L, a_{L-1}, a_{L-2,...,} a_3, a_2)$. Заметим, что также потомка можно получить путем инверсии сегмента, который лежит справа от выбранной позиции 1 и слева от позиции L-2.

Например, первый случай:

P_1 : A | B C D E F G H

P'_1 : A | H G F E D C B.

Второй случай:

P_1 : A B C | D E F | G H

P'_1 : A B C | F E D | G H

В настоящее время известно также большое число различных операторов инверсии, таких как многоточечный, позиционный, специальный, универсальный и др., а также их различные модификации, так как их структура, как и оператора мутации тоже предотвращают потерю генетического материала.

Оператор транслокации – это инструкция, при реализации которой хромосомы-потомки формируются в результате скрещивания и инвертирования родительских хромосом (или их частей). Данный оператор представляет собой комбинацию из операторов кроссинговера и инверсии. Алгоритм выполнения этого процесса представим в виде следующего словесного описания.

1. Из текущей популяции выбираются две родительские хромосомы $P_1 = a_1, a_2,.., a_L$ и $P_2 = a'_1, a'_2,.., a'_L$.

2. В этих выбранных хромосомах случайным образом выбирается точка разреза r,. Она выбирается из множества {1,2,...,L-1}, где L – длина хромосомы.

3. Потомки формируются путем преобразования (скрещивания) и инвертирования т.е. происходит частичный обмен признаками в родительских хромосомах согласно следующему правилу:

$$P_1{}^{/} = a_1, a_2, \dots, a_k, a_L^{/}, \dots, a_{k+1}^{/},$$

$$P_2{}^{/} = a_1^{/}, a_2^{/}, \dots, a_k^{/}, a_L, \dots, a_{k+1}.$$

Другими словами, первый потомок формируется из части левого до точки разрыва первого родителя и инверсии правой части

после точки разрыва второго родителя. Второй потомок формируется левой части второго родителя и инверсии правой части после точки разрыва первого родителя. Например:

P_1 : 1 2 | 3 4 5 6
P_2 : 2 1 | 4 3 6 5

P'_1 : 1 2 5 6 3 4
P'_2 : 2 1 6 5 4 3

Заметим, что здесь, как и после реализации кроссинговера, имеем две родительские хромосомы и дополнительно получаем две хромосомы-потомка. Отметим, что до последнего времени оператор транслокации не применялся при генетическом поиске.

Часто при реализации генетического поиска выбираются части альтернативных решений (тесно связанные между собой гены), которые нежелательно разрывать. Они называются строительными блоками. Их часто используют для формирования наиболее эффективных решений. Оператор транспозиции является частным случаем оператора инверсии, но работающий непосредственно только с блоками.

Оператор сегрегации является частным случаем оператора кроссинговера, но в отличии от него реализуется на части или всей рассматриваемой популяции решений.

Заметим, что операторы (кроссинговера, транслокации и сегрегации), реализующие процесс полового размножения могу воспроизводить не реальные решения. Для устранения этого недостатка вводятся операторы удаления и вставки. Эти операторы применяются только совместно Первый удаляет дублирование генов в хромосомах, а второй на их место вставляет недостающие гены, чтобы соответствующее альтернативное решение оставалось реальным.

В процессе воспроизводства потомков при генетическом поиске возникает существенная проблема-экспоненциального роста размера популяции. Для устранения этой проблемы используется оператор редукции. Это инструкция, при реализации которой производится отбор решений с целью уменьшения размера популяции до заданной величины. Основной сложностью при реализации оператора редукции является установление компромисса между разнообразием генетического материала и качеством решений. Поэтому в процессе генетического поиска рассматриваются различные способы реализации этого оператора, такие как: элитная, случайная, выбор «лучших» и «худших», «близкое» родство, «дальнее» родство, на основе кода Грея, Хэммингова расстояния, «турнира» и др.

В качестве модификаций этого процесса также рассматривают следующие виды отбора[8]:

равновероятностный

$$P_k(s) = \frac{1}{N}, \quad (1.5)$$

где N – размер популяции; и пропорциональный

$$P_k(s) = \frac{ЦФ(P_k)}{\sum_{k=1}^{n} ЦФ(P_k)} \quad (1.6)$$

Отметим, что оператор редукции в процессе генетического поиска может выполняться после реализации каждого оператора или после реализации всех в пределах одного поколения эволюции.

В конце генетического поиска реализуется оператор рекомбинации[8]. Это технология анализа и преобразования популяции при переходе из одного поколения в другое. Он может выполняться разными способами после реализации каждого оператора или после реализации всех в пределах одного поколения эволюции. Основными механизмами при реализации этого процесса являются: установление меток для специальных хромосом, которые сразу переходят в новую популяцию; вытеснение «близких» и похожих хромосом и разделение-устранение копий хромосом из создаваемой популяции. При реализации этого оператора задается параметр W(P), находящегося в пределах от 0 до 1, с помощью которого происходит управление процессом преобразования текущей популяции в новую.

Важным понятием в процессе генетического поиска является вероятность, которая определяет, какой процент из общего числа генов в хромосомах подвергается изменениям при реализации того или иного генетического оператора. Другими словами, сколько раз (сколько точек разрыва необходимо взять), чтобы выполнить тот или иной генетический оператор за одно поколение эволюции. Из литературы и экспериментальных исследований известны следующие значения: вероятность кроссинговера 90%; мутации – 60%; инверсии – 60%; транслокации – 50%; транспозиции – 50%; сегрегации – 90%.

Согласно известным гипотезам, увеличение вероятности приводит к большому разнообразию генетического материала, но также увеличивает время поиска и может привести к полной потере лучших признаков. И наоборот уменьшение вероятности, приводит к уменьшению разнообразию генетического материала, времени поиска, но часто приводит к попаданию решения в локальный оптимум, далекий от глобального. Поэтому необходим поиск разумного компромисса в этом вопросе.

Чтобы уменьшить эти значения и ускорить процесс получения результата и при этом не потерять лучшие признаки многие исследователи используют разные методы и механизмы для модификации генетических операторов. Это фрактальные множества; метод дихотомии, золотое сечение, ряд Фибоначчи, телесные, простые, фигурные числа и др.

Рассмотрим пример построения и реализации модифицированного оператора мутации на основе фигурных многоугольных чисел. Пусть задана последовательность треугольных чисел на основе следующей формулы:

$$T_n = \frac{1}{2}n(n+1),\ n = 1,\ 2,\ 3,\ \ldots. \qquad (1.7)$$

Ее отображение представлено на рис. 1.15.

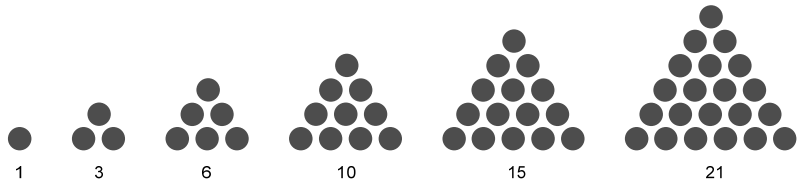


Рис. 1.15. Отображение последовательности треугольных чисел

Здесь число фигурных чисел из последовательности выбирается таким образом, чтобы это число было меньше или равно длине хромосомы L, отобранной для выполнения оператора мутации. Имеем родительскую хромосому P: 1234567 длины 7, тогда последнее фигурное число берется равное 6, т. е. имеем три точки разрыва после первой, третьей и шестой позиций в родительской хромосоме. После определения точек разрыва происходит перестановка генов. Отметим, что выполнять оператор мутации можно двумя способами. Первый заключается в расстановке сразу всех точек разрыва. Тогда хромосома потомок будет иметь следующий вид P_1:2143576. А второй в последовательном применении сначала одной, потом второй и в конце в третьей. Заметим, что при реализация второго способа мы получаем дополнительно еще два потомка итого имеем три потомка: P_1:2134567, P_2:2143567, P_3:2143576. При этом потомок, полученный первым способом, полностью соответствует потомку P_3 полученному при реализации второго способа.

Отметим, что остальные модифицированные генетические операторы, использующие различные механизмы при определении точек разрыва строятся и реализуются аналогично.

ГЛАВА 2. БИОИНСПИРИРОВАННЫЕ МОДЕЛИ И МЕТОДЫ

2.1. Общие положения теории биоинспирированного поиска

Еще в древнее века ученые столкнулись с проблемой, что делать если задачу нельзя решить с помощью математических и логических приемов. Впервые понятие эвристика ввел в своем трактате греческий математик Паппа «Искусство решать задачи» (300 год н. э.). Но еще до него свою систему словесного обучения, основанного на принципе «знающего незнания», ввел Сократ (469–399 гг. до н. э.). Его труды были уточнены Платоном. В (287–212 гг. до н. э.) Архимед произнес свое знаменитое: «Эврика!». Дальше на долгие годы в связи с упадком античных наук про это направление было забыто. И только в XVI–XVII вв. благодаря трудам Г. Галилея, Ф. Бэкона и других учёных эвристические подходы начали возрождаться. Затем (1781–1848 гг.) чешский математик Б. Больцано в своем труде «Наукоучение» изложил различные эвристические методы и приёмы. В России разработкой теории эвристики в начале XX столетия занимался инженер-патентовед П.К. Энгельмейер, который описал эвристику в изобретательстве. Долгое время в основе эвристик лежали методы проб и ошибок. На современном этапе развития первые эвристические методы оптимизации появились в 1950-х гг. Эти методы постоянно совершенствуются и развиваются одновременно с созданием новых электронных вычислительных устройств. В настоящее время известно около трехсот различных эвристик (подходов, методов, алгоритмов, процедур, приемов) их основной список опубликован в книге Дж. К. Джонс, «Методы проектирования», Москва: Мир, 1986 г. Среди них появились эвристики, основанные на моделирование процессов, происходящих в живой природе. Данные эвристики сегодня известны под термином биоинспирированные методы [23, 26, 34–40].

Данные эвристические методы, основаны на моделях, описывающих природную систему и в отличии от классических производят обработку не одного варианта решения рассматриваемой задачи, а позволяют одновременно обрабатывать несколько ее альтернатив. Хотелось бы заметить, что в этих методах сходимость в достижении глобального оптимума не доказана, но в литературных источниках и экспериментальных исследованиях обосновано, что они

позволяют получать наборы квазиоптимальных решений. Разные биоинспирированные методы оперируют, согласно своей природе, такими обозначениями как пчелы, светлячки, бактерии, кроты и др. Для внесения единообразия предлагается объединить их одним термином агент. Представим обобщенную схему последовательности шагов во всех методах биоинспирированного поиска рис. 2.1.

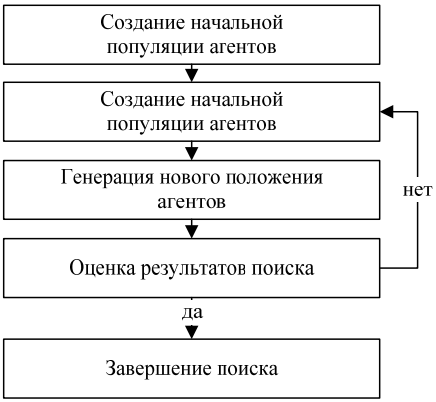


Рис 2.1. Обобщенная схема последовательности шагов во всех методах биоинспирированного поиска

Опишем представленный алгоритм более подробно. В первом блоке создается начальная популяция агентов, соответствующая рассматриваемому методу и с учетом принципов ее формирования для покрытия по возможности всей области поиска. Во втором блоке агенты мигрируют на основе операторов биоинспирированной оптимизации, определенных для каждого из них. В третьем блоке после их перемещения закрепляется их новое расположение. В четвертом блоке проверяем выполнение условий если да, то принимается лучшее из найденных положений агентов и 5-й блок завершение поиска, а если нет, то возврат к блоку 2 для продолжения итерационного поиска. Заметим, что на начальных этапах поиск ведется для нахождения новых решений, на последующих- поиск новых, а также их корректировка, а на последних производится только корректировка этих решений.

Рассмотренная обобщенная схема позволяет отметить, что все методы биоинспирированной оптимизации обладают модульной структурой, позволяющей моделировать из них различные вариации, а также проводить их комбинирование, интеграцию и гибридизацию.

Заметим, что все агенты обладают следующими основными свойствами, рис. 2.2 [26].

Рис. 2.2. Общие свойства агентов

Обладая этими свойствами, можно сказать, что все агенты формируются случайно, мигрируют произвольно, перемещаются независимо в основном друг от друга, способны обмениваться информацией между собой и не имеют центр управления.

Кроме всем известных преимуществ при реализации данных методов возникает одна существенная проблема, заключающаяся в обеспечении баланса между разнообразием агентов поиска, расширением области поиска и скоростью сходимости результатов. В качестве одних из эффективных механизмов при решении данной проблемы в биоинспирированных методах, по мнению авторов, выступают процедуры адаптации и самоорганизации, присущие живым системам.

2.2. Классификация методов биоинспирированного поиска

Классификация обозначает разновидность деления объема понятия по определенному основанию (признаку, критерию), при котором объем родового понятия (класс, множество) делится на виды (подклассы, подмножества), а виды, в свою очередь делятся на подвиды и т. д. Классификация методов биоинспирированного поиска (метаэвристик) как процесс предполагает упорядоченное и систематическое отнесение каждой метаэвристики к одному и только одному классу в

рамках системы взаимоисключающих и неперекрывающихся классов. Классификация сортирует метаэвристики в соответствии с их сходством, используя определенный набор признаков или критериев.

Классификация позволяет объективно подходить к выбору метаэвристик, поскольку для каждой из них имеются конкретные задачи, с которыми она справляется особенно хорошо. Знать и понимать эти взаимосвязи важно для целенаправленного применения метаэвристики. Одной из объективных проблем при классификации метаэвристик является бессистемность их обозначений. Большинство метаэвристик используют обозначения, основанные на их природной метафоре. Вместо использования общих терминов оптимизации. Это вызывает трудности при формулировании общих критериев классификации и извлечении необходимой информации из метаэвристики для применения этих критериев. Проблемой также является отсутствие руководящих принципов в отношении критериев классификации. Это приводит к множеству слабо детализированных схем классификации. Зачастую критерии классификации выбираются не по принципу их информативности и значимости, а с точки зрения простоты применения.

Классификация метаэвристик, основанная на природных метафорах, не позволяет однозначно различать метаэвристики.

В [41] была предложена классификация, которая включает эволюционные и роевые метаэвристики, а также метаэвристики, основанные на физических процессах и на когнитивных процессах, связанных с деятельностью человека. Предлагаемое множество критериев включает такие характеристики, как скорость сходимости алгоритма, диверсификация пространства поиска решений, механизм селекции решений, вычислительная трудоемкость, требуемый объем памяти, настройка параметров алгоритма, трудности программной реализации.

Одной из важных особенностей метаэвристик является баланс между скоростью сходимости и диверсификацией пространства поиска оптимальных решений. Диверсификация пространства поиска решений относится к способности поисковых агентов исследовать новые области пространства поиска, в то время как сходимости алгоритма делает акцент на возможностях этих агентов уточнять уже найденные «хорошие» решения. Поиск баланса между

ними зависит от используемого в метаэвристике механизма популяционного отбора, аттрактивности операторов поиска, настройки параметров и от числа итераций алгоритма.

Некоторые подходы к классификации метаэвристик основаны на использовании таких специфических характеристик как локальный поиск в окрестности, восхождение на холм, многократный запуск алгоритма, адаптивное программирование памяти. Метаэвристики роевого интеллекта иногда разделяются на инспирированные водной фауной, наземными животными, птицами, насекомыми и микроорганизмами.

Современная классификация метаэвристик должна быть многоуровневой [42]. Уровни классификации и используемые критерии представлены в табл. 2.1.

Таблица 2.1

Многоуровневая классификация метаэвристик

№	Уровень	Критерии
1	Природная метафора	Роевые алгоритмы; алгоритмы, основанные на физических и химических процессах; алгоритмы, основанные на когнитивных процессах и деятельности человека; эволюционные алгоритмы; алгоритмы, основанные на особенностях организмов, способных к фотосинтезу; прочие алгоритмы.
2	Структурный	Дискретные/траекторные; популяционные/одиночные; с памятью/без памяти.
3	Поведенческий	Получение новых решений: создаются из уже имеющихся решений или путем дифференциально-векторного движения.
4	Поисковый	Оценка баланса: скорость сходимости метаэвристики/диверсификация пространства поиска.
5	Компонентный	Локальный поиск; эволюционный механизм; механизм оптимизации роя частиц; механизм оптимизации колонии муравьев.
6	Специфические особенности	Механизмы расширения структуры фреймворков; вычислительная сложность метаэвристики.
7	Оценочный	Тип/размерность задачи

Рассмотрим подробнее каждый уровень и соответствующие критерии классификации.

Уровень соответствия природной метафоре. Разнообразие природных метафор может привести к мелкозернистой классификации. Необходим компромисс, чтобы сохранить классификацию простой, но информативной. В результате более чем 400 из существующих метаэвристик разделяются по критерию соответствия природной метафоре на шесть классов: (1) роевые алгоритмы (48%); (2) алгоритмы, основанные на физических и химических процессах (18%); (3) алгоритмы, основанные на когнитивных процессах и деятельности человека (12%); (4) эволюционные алгоритмы (7%); (5) алгоритмы, основанные на особенностях многоклеточных организмов, способных к фотосинтезу (5%); (6) прочие алгоритмы, которые настолько различны, что их невозможно сгруппировать в представительные классы (4 или более метаэвристики, около 10%). Для прояснения критериев классификации на этом уровне приведем описание основных характеристик каждого из шести классов.

Алгоритмы, основанные на роевом интеллекте, характеризуются коллективным поведением децентрализованных, самоорганизующихся агентов [43–45]. Первоначально этот термин был предложен в контексте роботизированных систем, однако с годами получил широкое распространение для обозначения коллективного разума в группе агентов, управляемых простыми правилами поведения. Речь идет о моделях коллективного поведения таких сообществ как колонии насекомых или птичьи стаи, где коллективный интеллект роя позволяет эффективно решать задачи оптимизации.

Роевые метаэвристики разделяются по следующим критериям классификации:

(1) алгоритмы, инспирированные полетом птиц и насекомых (17%, наиболее известными являются оптимизация роя частиц (*Particle Swarm Optimization, PSO*) [46] и искусственная пчелиная колония (*Artificial Bee Colony, ABC*) [47];

(2) алгоритмы, инспирированные механизмами добывания пищи и охоты наземных животных (14 %, наиболее известными являются алгоритм оптимизации колонии муравьев (*Ant Colony Optimization, ACO*) [48], алгоритм оптимизации стаей серых волков (*Grey Wolf Optimizer, GWO*) [49], львиный алгоритм оптимизации (*Lion Optimization Algorithm, LOA*) [50], алгоритм оптимизации кузнечика (*Grasshopper Optimization Algorithm, GOA*) [51];

(3) алгоритмы, инспирированные водными животными (7%, наиболее известными являются алгоритм стада криля (*Krill Herd, KH*) [52], алгоритм оптимизации китов (*Whale Optimization Algorithm, WOA*) [53];

(4) алгоритмы, инспирированные микроорганизмами (4%, наиболее известным является алгоритм поиска пищи бактериями (*Bacterial Foraging Algorithm, BFOA*) [54];

(5) другие роевые метаэвристики настолько различные, что их невозможно сгруппировать в представительные классы (6%) [37].

Внутри каждой подкатегории также имеются определенные различия, например метафора связана с поисками роем пищи или же речь идет о модели движения роя.

Алгоритмы, основанные на физических (14%) и химических (4%) процессах, наиболее известные из которых алгоритм поиска гармонии (*Harmony Search, HS*) [55], алгоритм интеллектуальных капель воды (*Intelligent Water Drops, IWD*) [56], моделирования отжига (*Simulated Annealing, SA*) [57], характеризуются тем, что они имитируют, например, гравитационные силы, электромагнетизм, электрические заряды и движение воды, а также химические реакции и движение частиц газов [39].

Алгоритмы, основанные на когнитивных процессах и деятельности человека, инспирированы социальными и политическими концепциями, процессами принятия решений, конкуренцией идеологий внутри общества, спортивными соревнованиями, мозговым штурмом. Наиболее известными являются иммунный алгоритм (*Immune Algorithm, IA*) [58], алгоритм империалистической конкуренции (*Imperialist Competitive Algorithm, ICA*) [59], глобальный алгоритм оптимизации мозгового штурма (*Global Brainstorm Optimization Algorithm, GBSO*) [39, 60].

Следующим классом метаэвристик по критерию соответствия природной метафоре являются эволюционные алгоритмы. Эволюционные алгоритмы основаны на принципах природной эволюции. Каждый агент в популяции представляет собой решение задачи и имеет соответствующее значение функции пригодности. В этих алгоритмах процесс воспроизводства и отбора повторяется в течение многих поколений, совокупность решений эволюционирует в направлении областей с более высокой пригодностью. Особенность селекции делает алгоритмы этого класса уникальными по сравнению с алгоритмами других категорий. Наиболее известными

эволюционными алгоритмами являются генетический алгоритм (*Genetic Algorithm, GA*) [3, 8, 26–31], алгоритм дифференциальной эволюции (*Differential Evolution, DE*) [61], алгоритм эволюционной стратегии (*Evolutionary Strategy, ES*) [37, 62].

Алгоритмы, основанные на особенностях многоклеточных организмов, способных к фотосинтезу, инспирированы растительным миром. В отличие от других метаэвристик, в них отсутствует связь между агентами. Одним из наиболее известных является алгоритм оптимизации лесов (*Forest Optimization Algorithm, FOA*) [63], инспирированный процессом размножения растений.

В категорию прочих входят метаэвристики, которые не относятся ни к одной из перечисленных выше категорий. Они настолько различны, что их невозможно сгруппировать в представительные классы. Например, алгоритм Инь-Ян. Эта категория метаэвристик неоднородна, ее включение в классификацию, возможно, в будущем послужит основой для создания новых подкатегорий, развитию классификации и облегчит анализ достижений в этой области [64].

Структурный уровень. Он включает критерии, относящиеся к общей структуре метаэвристик. Вначале метаэвристики классифицируются на дискретные и траекторные. Далее, большинство дискретных метаэвристик являются популяционными, а траекторные обычно основаны на одном решении. На следующем шаге классификации проводится различие между метаэвристиками, основанными на локальном поиске и так называемыми метаэвристиками конструктивного поиска. На заключительном шаге структурной классификации используется критерий наличия/отсутствия памяти в процессе поиска. Структурный уровень классификации дает важную информацию о метаэвристике, однако недостаточную для однозначной классификации.

Поведенческий уровень. Согласно этому уровню классификации метаэвристики группируются по их поведенческим особенностям безотносительно от их природной метафоры. С этой целью необходим четкий критерий классификации. В качестве такого критерия используются механизмы для создания новых решений или для изменения существующих решений задачи оптимизации. Согласно этому критерию, вначале проводится различие между процессом получения новых решений: они создаются из уже имеющихся решений или путем дифференциально-векторного движения [65].

При дифференциально-векторном движении новые решения создаются путем сдвига или мутации предыдущего решения. Репрезентативными примерами этой категории метаэвристик являются *PSO* и *DE*.

При создании новых решений используются механизмы, которые либо рекомбинируют несколько решений, либо основаны на стигмергии путем спонтанного непрямого взаимодействия между индивидами. Репрезентативными примерами этой категории метаэвристик являются *GA* и *ACO*.

В результате более чем 400 метаэвристик вначале разделяются на два класса:

(1) алгоритмы на основе дифференциально-векторного движения (66 %);

(2) алгоритмы, основанные на создании новых решений (34%).

В определении направления дифференциально-векторного движения может участвовать вся популяция решений (около 4%), только определенные решения (около 55%), только решения из некоторой окрестности или субпопуляции (около 7%).

Среди алгоритмов, основанных на создании новых решений, около 32% используют рекомбинацию нескольких решений, а 2% – стигмергию.

Представим поведенческие характеристики метаэвристик из различных категорий.

Особенность категории метаэвристик дифференциально-векторного движения заключается в том, что вычисляется направление дифференциального вектора. Одним из возможных критериев является использование для этих целей всех агентов в популяции. В этих алгоритмах все агенты имеют определенную степень влияния на движение других решений. Такая степень обычно взвешивается в соответствии с разницей в функции пригодности или расстоянием между решениями. Примером здесь является светлячковый алгоритм (*Firefly Algorithms, FA*) [66], в котором близкие к лучшему решению оказывают более сильное влияние, нежели более удаленные. Другим критерием класса метаэвристик с дифференциально-векторным движением является использование определенных репрезентативных решений. Чаще всего в этом качестве выбираются наилучшие решения, найденные алгоритмом. Примером является *PSO*. В этом оптимизаторе каждое решение или частица руководствуется глобальным текущим лучшим решением и лучшим

решением, полученным этой частицей во время поиска. Другим примером в этой категории является семейство алгоритмов *DE*. В них наилучшее решение комбинируется с дифференциальным вектором для расширения пространства поиска решений. Еще одним критерием класса метаэвристик с дифференциально-векторным движением является использование решений из некоторой окрестности или субпопуляции. Примерами алгоритмов в этой категории являются алгоритм оптимизации бабочки монарха (*Monarch Butterfly Optimization, MBO*) [67] и *BFOA*.

Особенность категории метаэвристик основанных на создании новых решений заключается в том, что они используют операторы рекомбинации или стигмергию. Наиболее распространенным вариантом является рекомбинация новых решений путем объединения некоторых из существующих решений с помощью оператора кроссинговера или путем комбинирования хороших решений. Самым популярным алгоритмом в этой категории являются эволюционный алгоритм *GA*, алгоритм *LOA*, алгоритм столкновения частиц (*Particle Collision Algorithm, PCA [68]*). Наиболее распространенным вариантом создания новых решений путем стигмергии является алгоритм *ACO*, моделирующий механизмом добывания пищи колонией муравьев [26, 48, 69, 70]. Метаэвристики, имеющие схожую природную метафору, могут значительно отличаться друг от друга по поведенческому критерию. Примером являются алгоритмы эхолокации дельфина (*Dolphin Echolocation Algorithm, DEA [71]*) и алгоритм дельфина (*Dolphin [72]*). Оба инспирированы одним и тем же животным (дельфином) и его механизмом эхолокация для обнаружения рыбы, однако поведенческие механизмы у них разные: *DEA* создает новые решения путем комбинирования, в то время как *Dolphin* аналогичен *PSO*. Другим примером является класс метаэвристик с дифференциально-векторным движением, который содержит более половины рассмотренных алгоритмов (55%) и включает алгоритмы из всех различных классов по критерию соответствия природной метафоре: социальное поведение человека алгоритм оптимизации анархического общества (*Anarchic Society Optimization, ASO [73]*), бактериальный алгоритм *BFOA*, алгоритм фейерверка (*Fireworks Algorithm, FWA [74]*), алгоритм опыления цветов (*Flower Pollination Algorithm, FPA [75]*).

На наш взгляд, поведенческий уровень классификации является наиболее информативным, а алгоритмы на основе дифференциально-векторного движения, входящие в этот класс, представляют более 60% всех метаэвристик. По этой причине именно метаэвристики дифференциально-векторного движения будут подробно рассмотрены в последующих главах.

Поисковый уровень. Он связан со скоростью сходимости метаэвристики при поиске оптимального решения и диверсификацией пространства поиска решений. Важность поиска баланса между скоростью сходимости алгоритма и диверсификацией пространства поиска решений является фундаментальной задачей, однако пока отсутствуют инструменты измерения баланса. На баланс между скоростью сходимости алгоритма и диверсификацией пространства поиска решений оказывают влияние механизм селекции, аттрактивность операторов поиска и число итераций алгоритма. Например, жадный механизм селекции приводит к преждевременной сходимости в точке локального оптимума. Аттрактивность операторов, используемых в метаэвристиках, означает дрейф имеющихся решений в направлении лучших решений, которые рассматриваются в качестве аттракторов. Этот механизм эффективен для поддержания разнообразия популяции решений. Необходима метрика для оценки баланса между скоростью сходимости метаэвристики и диверсификацией пространства поиска решений.

Компонентный уровень. Он связан с особенностями базовых алгоритмических структур и механизмов. Метаэвристики, использующие механизм локального поиска, классифицируются по критериям поиска в окрестности, поиска восхождением к вершине, предотвращения преждевременной сходимости в локальных оптимумах, мультистарта и адаптивного программирования памяти. Метаэвристики, использующие эволюционный механизм, классифицируются по критериям использования популяционного поиска или поиск, при котором исследуется пространство между двумя или более наилучшими решениями. Метаэвристики, использующие механизм оптимизации роя частиц, классифицируются по критериям направленного поиска по перспективным направлениям (например, градиентным) и поиска в переменной окрестности. В настоящее время многие метаэвристики используют комбинацию сразу несколько указанных выше алгоритмических компонентов.

Таким образом, компонентный классификационный уровень предоставляет информацию об их сходствах и различиях.

Специфический уровень. Он связан с особенностями и возможным расширением структуры метаэвристики. Поиск критериев, подходящих для данного уровня классификации, представляется несколько более сложным, нежели для предыдущих уровней, поскольку они должны быть связаны с возможностями фреймворка, т. е. программной платформы, определяющей структуру программного обеспечения, облегчающего разработку и объединение разных компонентов большого программного проекта. Некоторые метаэвристические фреймворки включают алгоритмы, в которых это расширение уже сделано. В других случаях расширение возможно, но не было включено или невозможно. Эта характеристика может быть использована в качестве критерия классификации. Также возможен поиск на этом уровне других критериев, основанных на особых или уникальных характеристиках метаэвристических фреймворков. Однако для этого требуются обширные знания о метаэвристиках, включая факторы, определяющие вычислительную сложность метаэвристики. Однако поиск критериев, подходящих для этого уровня классификации, является серьезной проблемой и нуждается в дальнейшем исследовании. Однако при классификации метаэвристических алгоритмов этот уровень, на наш взгляд, важен для обеспечения детального различия.

Оценочный уровень. Он связан с эффективностью метаэвристик для решения конкретных задач оптимизации. Возможно, что подходящими классификационными критериями на этом уровне могут быть, например, структура задачи и ее размерность. Однако для этого необходимо знать эффективность метаэвристик для решения конкретных классов задач. Оценочный уровень классификации требует дальнейших исследований, чтобы стать применимым в общей системе классификации метаэвристик. Понятно, что знание о производительности метаэвристики для различных классов задач облегчает ее выбор для решения конкретной задачи, а также помогает определить, какие метаэвристики демонстрируют идентичные результаты на одних и тех же классах задач.

Проиллюстрируем применение многоуровневой классификации на примере трех метаэвристик в их базовых версиях: генетический алгоритм [8, 28, 76], эволюция $(1 + \lambda)$ [62, 76–79] и табуированный

поиск [73–74], используя структурный, поведенческий, поисковый и компонентный уровни классификации.

Генетический алгоритм *GA* подходит для задач комбинаторной оптимизации, использует начальную популяцию в пространстве поиска решений, предполагает повторение следующих шагов до тех пор, пока не будет выполнен критерий остановки:

(1) родители выбираются с использованием турнирного отбора;

(2) потомство генерируется путем равномерного кроссинговера;

(3) потомство мутирует, случайно инвертируя один бит в решении;

(4) потомство оценивается и происходит замена поколений с учетом элитарного отбора.

GA использует дискретную структуру, память, а также три различные стратегии поиска рекомбинацию, мутацию и отбор. На компонентном уровне оператор мутации использует поиск в окрестности, оператор кроссинговера использует промежуточный поиск на основе популяции, популяционный отбор, программируемую память и мультизапуск. Смена поколений в популяции происходит по схеме восхождения на вершину фитнес-ландшафта.

Эволюционная стратегия $(1 + \lambda)$-*ES* также, как и *GA*, относится к группе эволюционных алгоритмов, использует память и дискретную структуру, подходящую для комбинаторной оптимизации. В $(1 + \lambda)$-*ES* случайным образом инициализируется только один родительский элемент. Следующие шаги повторяются до тех пор, пока не будет выполнен критерий остановки:

(1) λ потомков генерируются мутацией родителя;

(2) потомство оценивается, и лучшая особь родителя и потомства выбирается в качестве родителя следующего поколения.

В $(1 + \lambda)$-*ES* новые решения создаются путем адаптации ранее найденных решений, не используется оператор кроссинговера. На компонентном уровне популяция и оператор мутации используются для поиска в окрестности на основе популяции. Смена поколений в популяции происходит по схеме восхождения на вершину фитнес-ландшафта.

Табуированный поиск (*Tabu Search, TS [80,81]*) подходит для решения задач комбинаторной оптимизации, использует кратковременную память при формировании списка запретов. Поиск начинается со случайно инициализированного решения, для которого создается спи-

сок соседних решений. В табу список добавляется каждое просмотренное решение. *TS* предполагает повторение следующих шагов до тех пор, пока не будет выполнен критерий остановки:

(1) на каждой итерации выбирается лучшее решение в окрестности текущего решения в качестве нового текущего решения, даже если это приводит к увеличению стоимости решения;

(2) в кратковременной памяти, называемой списком табу, сохраняется недавно найденные решения, чтобы избежать зацикливания.

Поиск прекращается после определенного числа итераций или если после ряда последовательных итераций не было достигнуто каких-либо улучшений в наилучшем известном решении. *TS* – это метаэвристика, основанная на траектории одного решения, которая использует локальный поиск и память. Его поведение основано на дифференциальном векторном движении в управляемой окрестности. На компонентном уровне список табу использует адаптивное программирование памяти, поиск восхождением на вершину в ландшафте фитнес-функции.

Если необходимо оценить скорость сходимости и возможности диверсификации пространства поиска указанных метаэвристик, например, при оптимизации мультимодальных целевых функций, то система классификации показывает, что *GA* предоставляет больше возможностей для балансировки поиска, нежели *TS* и $(1 + \lambda)$-*ES*. При использовании соответствующего классификационного уровня эта оценка может быть улучшена.

2.3 Области применения методов биоинспирированного поиска

На практике наиболее распространенный вопрос заключается в том, какая метаэвристика лучше всего подходит и как ее можно успешно применить для решения конкретной задачи? Ответ на этот вопрос – непростая задача. В зависимости от требований по точности может существовать несколько метаэвристик, дающих приемлемые решения. Проблема выбора метаэвристики включает в себя пространство задач P, пространство алгоритмов A и отображение $P \times A$ на модель эффективности R.

Известны рекомендации по выбору метаэвристик. Они включают выбор критериев и проверку результатов путем статистического анализа и визуализации, а также настройку параметров алгоритмов.

Чтобы определить, какой алгоритм лучше всего подходит для этих инженерных задач, был применен подход, основанный на принципе Кондорсе для определения победителя при голосовании за кандидатов [82]. В этом контексте сравниваемые алгоритмы представляют кандидатов, а их решения для каждой из задач указывают на избирателей. Согласно принципу Кондорсе победителем на выборах объявляется кандидат, который превосходит остальных при парном сравнении. По результатам работы сравниваемых алгоритмов на трех инженерных задачах-бенчмарках были определены четыре лучших метаэвристики: алгоритм императорских пингвинов (*EPO*, 45 голосов), алгоритм охоты птицы-хищника (*AO*, 42 голоса), алгоритм хамелеона (*ChSA*, 34 голоса) и алгоритм африканских стервятников (*AVOA*, 32 голоса).

В последние годы метаэвристики стали активно применяться для решения широкого спектра задач. В интеллектуальных сетях метаэвристики используются для прогнозирования пиковой нагрузки, внезапных колебаний выходной мощности, обнаружения вторжений в сеть и оптимального планирования будущих требований клиентов. В сфере здравоохранения метаэвристики применяются в клинических исследованиях, персонализированной медицине, идентификации заболеваний и прогнозировании вирусных вспышек. В производстве, транспорте и промышленности применение метаэвристик в сочетании с другими методами машинного обучения выявило большие преимущества в управлении беспилотным транспортом, в использовании виртуальных помощников, в обнаружении неисправностей технических устройств, в поддержке принятия решений и оперативном управлении. В торговле метаэвристические алгоритмы машинного обучения нашли применение для прогнозирования поведения клиентов, интеллектуального складирования и инвентаризации, прогнозирования спроса и оптимизации цен. Интернет-платформы социальных сетей в настоящее время используют метаэвристики в сочетании с другими алгоритмами машинного обучения для персонализированной рекламы, обнаружения мошеннических учетных записей. В образовании метаэвристики используется для анализа успеваемости обучающихся и моделирования их поведения.

В табл. 1.2 представлены современные области применения различных классов метаэвристик для решения конкретных оптимизационных задач.

Таблица 2.2

Области применения метаэвристик

Область применения	Мета-эвристики	Задачи
1	*2*	*3*
Инженерное проектирование	*GSA, SA, GP, WOA*	Проектирование антенн; Оптимизация авиационных конструкций; Выбор параметров обработки деталей.
Обработка изображений и компьютерное зрение	*ACO, PSO, BA, ABC, BMO, CSA, EPO*	Сегментация изображений, обнаружение областей интереса на цифровых изображениях, цветовая экстракция.
Компьютерные сети и коммуникации	*GA, ABC, PSO, CS, EPO, MHSA, MBO, MMA*	Оптимальное распределение сети сенсорных датчиков, обнаружение сообществ в сети, обнаружение вредоносных *URL*-адресов и спама, криптоанализ, задача коммивояжера.
Энергетика и энерго-менеджмент	*PSO, SSO, GWO, KHA, BBO, WOA, FA, CSA, HHO, MFO*	Системы хранения энергии, управление домашней энергоустановкой, регулирование нагрузки в энергосистеме, управление мощностью, диспетчеризация, реконфигурация, оптимальное размещение конденсаторов, ветрогенераторов.
Анализ данных и машинное обучение	*WOA, GWO, SA, FA, PSO, GA, GSA, ABC, GOA, ALO, RIO*	Выбор информативных признаков для классификации объектов и ранжирования многомерных данных, кластеризация данных, обучение сверточных нейросетей, настройка параметров в *SVM*-методе машинного обучения.

Окончание таблицы 2.2

1	*2*	*3*
Робототехника	*GSA, SA, GP, WOA*	Планирование и оптимизация траектории роботов, автономная навигация полета БПЛА, разработка контроллеров для роботизированных платформ.
Медицинская диагностика	*ACO, PSO, BA, ABC, BMO, CSA, EPO, PBA*	Обработка медицинских изображений, прогнозирование заболеваний путем анализа больших наборов данных, диагноз онкологии, болезни Паркинсона с использованием данных микрочипов.
Информатика и другие области	*BA, SHO, LA, ESA, MPA, BBO, BNSS, FOA, DFO*	Оценка подверженности наводнениям, планирование работы магазина, оптимизации каркасных конструкций, добычи природного газа, оценка токсичности лекарственного средства, регулирование орошения, подводная акустическая классификация, укрепление почвы, оптимизация протонообменных процессов в топливных элементах

В качестве примера приведем краткое описание задач интеллектуального анализа данных, машинного обучения и энергетики, решенных в последние годы с использованием упомянутых метаэвристик.

Объект исследования – аккумуляторное оборудование, батареи и накопители тепловой энергии, имеющие большое значение при переключении пиковых нагрузок в энергетических системах. Для решения задачи оптимизации графиков работы энергетических систем, включающих батарею, воздушный тепловой насос, применяются метаэвристики, такие как генетический алгоритм (*GA*), оптимизация роя частиц (*PSO*) и поиск кукушки (*CS*). Показана эффективность метаэвристик при оптимизации режимов работы энергосистем [83].

Метаэвристика серых волков (*GWO*) применяется для оптимизации систем электроснабжения. Алгоритм тестируется на 23 тестовых

функциях и применяется для подключенного к сети синхронного генератора, приводимому в действие ветровой турбиной. Результаты сравниваются с конкурирующими алгоритмами. Предлагаемый подход значительно выигрывает в производительности [84].

Рассматривается задача оптимального распределения реактивной мощности как задача минимизации целевой функции, представляющей общие потери активной мощности в электрических сетях. Ограничения связаны с напряжением генератора, коэффициентами отводящих регулирующих трансформаторов и количеством реактивных шунтирующих компенсаторов. Цель исследования – найти наилучший вектор управляющих переменных, чтобы можно было реализовать снижение потерь мощности. Применяется биоэвристика охоты на горбатых китов с помощью пузырьковой сети (*WOA*). Приводится сравнение с алгоритмом роя частиц [85].

Решается задача оптимального распределения реактивной мощности. Используется метаэвристика оптимизации пламени мотылька (*MFO*). Исследуется наилучшая комбинация управляющих переменных, включая напряжение генераторов, настройку отводов трансформаторов. Реактивные компенсаторы подбираются таким образом, чтобы обеспечить минимальные общие потери мощности и минимальное отклонение напряжения. Эффективность алгоритма *MFO* сравнивается с другими методами оптимизации. Статистический анализ показал конкурентоспособные результаты, обеспечивая меньшие потери мощности и меньшее отклонение напряжения, нежели конкурирующие подходы [86].

Объект исследования – домашние системы управления энергопотреблением для минимизации затрат. Предлагается гибридная метаэвристика алгоритма дифференциальной эволюции (*DE*) и алгоритма оптимизации дождевого червя (*Earthworm Algorithm*, *EWA*). Моделирование показало, что *EWA* работает лучше с точки зрения снижения затрат, а *DE* – с точки зрения снижения пикового отношения к среднему [87].

Применяется алгоритм охоты пауков (*SSA*) для повышения энергоэффективности поездов. Эксперименты показали, что предлагаемые алгоритмом планы энергосбережения на железной дороге могут помочь машинистам локомотивов в планировании движения поездов [88].

Интеллектуальный анализ правил ассоциации, направленный на обнаружение правил ассоциации, которые удовлетворяют заранее определенной минимальной поддержке и достоверности из данной базы данных. Метаэвристика основана на оптимизации миграции животных для уменьшения количества ассоциативных правил: из данных удаляются правила, которые не имеют высокой поддержки и не нужны [89].

SVM считается одним из самых мощных методов машинного обучения. Предлагается гибридный подход, основанный на алгоритме оптимизации кузнечиков (*GOA*). Цель подхода – оптимизировать параметры модели *SVM* и одновременно найти подмножество лучших функций. Результаты экспериментов показали, что предложенный подход превосходит конкурирующие алгоритмы с точки зрения точности классификации при минимизации количества признаков [90].

Кластеризация текста – это процесс группировки значительных объемов текстовых документов в кластеры с релевантными документами. Для кластеризации текста используется алгоритм стада криля. Проверка алгоритма проводилась с использованием таких показателей как точность, полнота, *F*-мера и энтропия. [91].

Кроме того, существует большое количество программных фреймворков для эволюционных и роевых алгоритмов на разных языках, таких как *C++, Java, Matlab* и *Python*. Например, фреймворк *Evolutional Computation Framework* (*http://ecf.zemris.fer.hr/*) и *ParadisEO* (*https://en.wikipedia.org/wiki/Paradiseo*) на *C++*; *jMetal* (*https://jmetal.sourceforge.net/)* на *Java*, *jMetalPy* на *Python* (*https://pypi.org/project/metal-python/*); *PlatEMO* в *Matlab* (*https://github.com/BIMK/PlatEMO*) и другие.

ГЛАВА 3. МОДЕЛИ И МЕТОДЫ РОЕВОГО ИНТЕЛЛЕКТА

3.1. Понятие роевого интеллекта

Согласно известной классификации основными методами биоинспирированной оптимизации являются методы роевого интеллекта [26, 37 ,43–46, 92–100]. Впервые данный термин был введён в работах о клеточных роботах американским и китайским учеными Херардо Бени и Ван Цзином в 1989 году.

Роевой интеллект представляет собой коллективный разум, т. е. коллективное поведение отдельных автономных агентов, локально взаимодействующих между собой и с окружающей средой, обмениваясь информацией по примитивным правилам в самоорганизующейся системе без выраженного центра управления для достижения всех его целей рис. 2.3. Роевой интеллект можно сравнить с многоагентной системой [101].

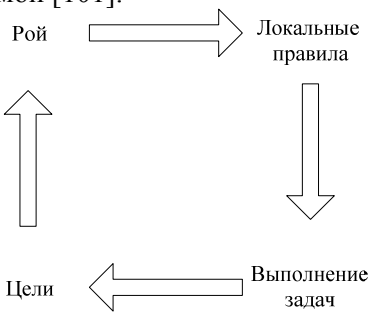


Рис. 3.1. Обобщенная схема роя

Исследование роевых моделей ведется с 90-х годов. За время их развития ученые пришли к единодушному мнению [26, 37, 43–46, 92–99], что данные модели эффективны при решении различных оптимизационных задач, а также динамической оптимизации процессов в распределенных нестационарных системах. Отметим, что модель роевого интеллекта.

Может быть построена и реализована при решении различных оптимизационных задач, если она может быть представлена в виде графа, а также, чтобы был задан конечный набор элементов и их возможных состояний с проведением их оценки.

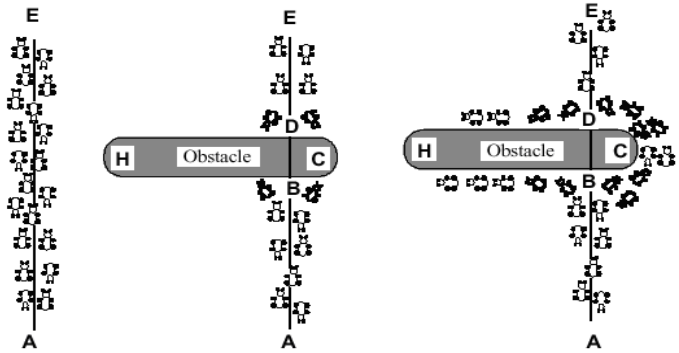
3.2. Модели и метод муравьиной оптимизации

Одним из самых ярких представителей роевого интеллекта являются методы муравьиной оптимизации. Ученые уже давно обратили внимание на то, как почти слепым муравьям удается находить

источники пищи прокладывая для этого кратчайшие пути. В 1959 г. Французский зоолог Пьер-Поль Грассе предложил теорию поведения колонии термитов, в 1983 г. Денеборг описал и проанализировал коллективное поведение муравьев в живой природе, в 1988 г. Мэйсон и Мандерский опубликовали статью о «самоорганизации» муравьёв и в 1989 г. Эблингом была написана работа о реализация модели поведения муравьев в поисках питания.

В природе муравьи являются социальными насекомыми, обладающие самоорганизацией и живущими колонией внутри муравейника. Они перемещаются в произвольном порядке и после нахождения источника пищи возвращаются в муравейник, откладывая по пути феромонные следы. Данные следы являются одним из коммуникативных средств общения между ними. Самоорганизация муравьиной колонии выражается во взаимодействии таких компонентов, как случайность, многократность и положительная и отрицательная обратные связи [26, 37, 38, 48, 69, 70, 96, 100, 102–105].

Рассмотрим поведение реальных муравьев в природе рис. 3.2. Здесь А это муравейник, а Е-источник пиши. Вначале один из муравьев находит источник пиши и возвращается в муравейник, оставляя по пути феромонные следы. Затем другие муравьи выбирают один из возможных путей АСЕ или АНЕ и обратно. По пути они также откладывают свои феромонные следы, закрепляя эффект (положительная обратная связь).



а) б) в)

Рис. 3.2. Пример поведения реальных муравьев

Заметим, что муравьи идут, как по короткому пути, так и по длинному, но феромонные следы, обладают свойством испаряться (отрицательная обратная связь). Поэтому с течением времени длинный путь исчезнет вследствие испарения феромона, и наоборот короткий путь становится все более привлекательным для других муравьев и в итоге муравьи прокладывают самый короткий маршрут от муравейника до источника пищи.

Идея использования данной модели поведения муравьёв в живой природе при поиске пищи, как метаэвристического метода оптимизации принадлежит итальянскому ученому доктору наук М.Дориго. В 1992 он в своей работе Optimization, Learning and Natural Algorithms описал первый муравьиный алгоритм известный как «муравьиная система».

Согласно М. Дориго при графовом представлении задачи каждый муравей в колонии обладает следующими свойствами, такими как [48, 102]:

1) имеет начальное, промежуточное и конечное состояния;

2) помечает пройденный маршрут перемещаясь по ребрам графа в любую доступную вершину из множества соседних (промежуточное состояние) при этом откладывая феромонные следы причем муравей может обновлять феромонный след, как при перемещении вперед, а также и назад и даже найдя решение;

3) присвоение конечного состояния – получение лучшего решения;

4) обладает памятью, т. е. хранит информацию о пройденном пути (таблица муравьиных маршрутов). Данная таблица используется для определения допустимых решений или их оценки;

5) переход из одного состояния в другое осуществляется с помощью вероятностных правил (функций значений). Они запоминаются в базу данных называемых таблицей муравьиных маршрутов.

Одним из важных достоинств этой модели является передвижение одновременно и независимо всей или части колонии муравьев, за счет коллективного поведения, т. е. непрямого обмена информацией (значений феромонных следов). Этот процесс можно сравнить с распределенным обучением, в котором отдельные муравьи, не адаптируются, а наоборот, адаптивно изменяют вид и восприятие задачи другими муравьями.

На рис. 3.3. представлена базовая архитектура муравьиного поиска.

В дальнейших своих работах М. Дориго модифицировал свою «муравьиную систему». Одним из таких улучшений стало введение элитных муравьев, которые откладывают большее количество феромона на «лучших» маршрутах. Другим стало введением обучающей таблицы с данными о сопоставлении каждому ребру величины, определяющий функцию приоритета перехода по этому ребру. При чем эта функция вычисляется из всех значений приоритетов доступных переходов, а также обновляется на каждой последующей итерации [106–108]. Следующей модификацией в работах Томас Штютцле (Tomas Stützle) и Хольгер Хооса стало установление минимального и максимального уровня феромона. При этом сначала устанавливается максимальный уровень феромона и на каждой итерации появляется только один элитный муравей. Еще одной модификацией в работах Бернд Бульнхаймер (Bernd Bullnheimer), Рихард Хартл (Richard F. Hartl) и Кристине Штраусс стало ранжирование муравьев согласно длинам, пройденных ими маршрутов, т. е. уровень феромонов здесь задается пропорционально занимаемым ими позициями [108].

Проведя анализ рассмотренных алгоритмов, авторы предлагают свой модифицированный муравьиный алгоритм. В качестве модификаций здесь предлагается учесть все достоинства предыдущих, ввести механизмы выделения и сохранения так называемых строительных блоков-структур (части решения), которые вносят существенный вклад в общую оценку функции качества.

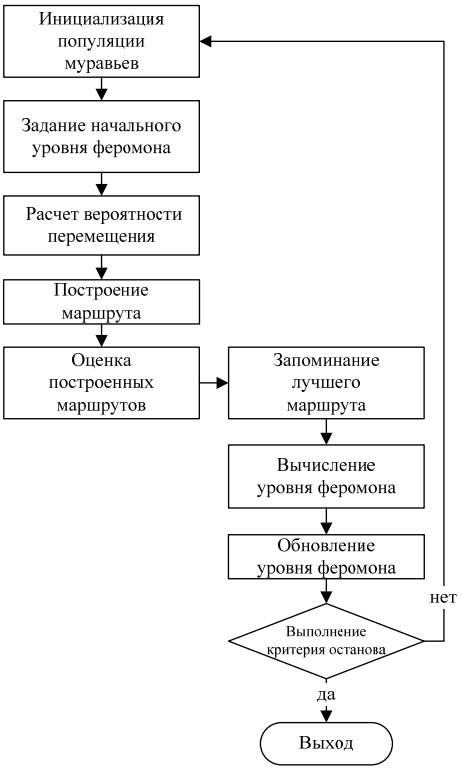


Рис. 3.3. Базовая архитектура муравьиного поиска

Данной структурой является вектор с номерами вершин графа, определяющий приоритет перехода. В нашем алгоритме предлагается использовать два подхода к формированию таких структур. Первый подход заключается в выделении таких структур на начальном этапе за счет имеющихся знаний о решаемой задаче. Второй подход- в динамическом формировании таких структур на каждой итерации. Обобщенная схема такого модифицированного алгоритма приведена на рис. 3.4 [38, 103–105].

Для наглядности рассмотрим работу данного алгоритма на примере решения классической задачи о коммивояжере [105].

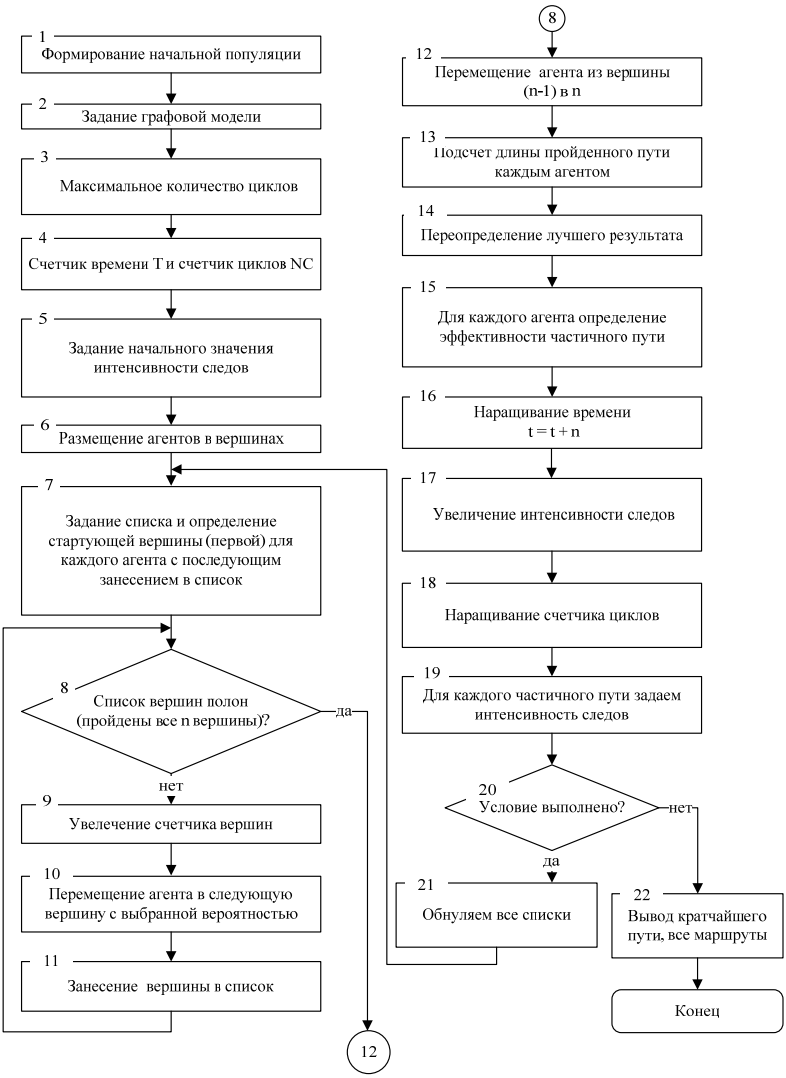


Рис. 3.4. Схема модифицированного алгоритма муравьиной оптимизации

Сформулируем кратко постановку задачи. Коммивояжеру необходимо посетить N городов, не заезжая в один и тот же город дважды, и вернуться в исходный пункт по маршруту с минимальной

стоимостью. Пусть задана модель задачи в виде графа G = (X, U), где |X| = n – множество вершин (городов), |U| = m – множество ребер (возможных путей между городами). Дана матрица чисел D(i, j), где i, j ∈ 1, 2,…, n, представляющих собой стоимость переезда из вершины xi в xj. Тогда требуется найти такую перестановку φ из элементов множества X, что значение целевой функции равно:

$$Fitness\ (\varphi) = D(\varphi(1), \varphi(n)) + \sum_i \{D(\varphi(i), \varphi(i+1))\} \to \min .$$

Здесь D(i, j)–определяется на основе следующего выражения:

$$d_{ij} = [(x_i\text{-}x_j)^2 + (y_i\text{-}y_j)^2] \qquad (3.1)$$

При использовании не полносвязного графа, для исключения перемещения по не существующему ребру в соответствующей ячейке матрицы указывается бесконечно большое число.

Сначала определим свойства муравья[105]. Каждый муравей из колонии имеет «память». Это список городов, которые ему необходимо посетить. Каждый муравей из колонии обладает «зрением», которое обратно пропорционально длине ребра и рассчитывается на основе следующей формулы:

$$\eta_{ij} = 1/D_{ij}. \qquad (2.2)$$

Данное свойство определяет его выбор, т. е. оценка расстояния между вершинами. Чем меньше расстояние до вершины, тем лучше «видимость» и тем самым осуществление перехода в нее.

Кроме расстояния учитывается и уровень феромона τij(t) на соответствующем ребре, согласно этому вероятность перехода муравья из одной вершины в другую определяется таким соотношением:

$$\begin{cases} P_{ij,k}(t) = \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum_{l \in J_{i,k}} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}(t)]^{\beta}}, j \in J_{i,k}, \\ P_{ij,k}(t) = 0, j \notin J_{i,k} \end{cases} \qquad (3.2)$$

где k – муравей: Ji,k (табу-лист); i и j – соответствующие начальная и последующая вершины графа; α, β – это параметры, задающие уровень феромона. Если параметр α = 0, то выбор пути происходит по кратчайшему ребру, если параметр β = 0, то муравей выбирает ребро с наибольшим уровнем феромона.

В процессе поиска вероятность выбора кратчайшего пути увеличивается, так как уровень откладываемого феромона повышается согласно (3.2) и рассчитывается на основе следующего выражения:

$$\Delta\tau_{ij,k}(t)=\begin{cases}\dfrac{Q}{L_k(t)},(i,j)\in T_k(t)\\ 0,(i,j)\notin T_k(t)\end{cases}, \qquad (3.3)$$

где Q – параметр, определяющий значение порядка длины кратчайшего пути, а Lk(t) – длина маршрута Tk(t).

Как известно, что стечением времени происходит уменьшение уровня феромона. Этот процесс называется испарением. Он определяется на основе следующего выражения:

$$\tau_{ij}(t+1)=(1-p)\cdot\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij,k}(t), \qquad (3.4)$$

где m – количество муравьев, p – коэффициент испарения, лежащий в пределах ($0 \le p \le 1$).

Введение в процесс поиска «элитных» муравьев, которые откладывают большее количество феромона на «лучших» маршрутах. Это количество рассчитывается на основе следующего выражения:

$$\Delta\tau_e(t)=A^e\cdot\frac{Q}{L^*(t)}, \qquad (3.5)$$

где Ae – коэффициент, учитывающий влияние элитных муравьев. Например, если $Ae = 0$, то влияние муравья игнорируется, если $0 < Ae < 1$, то уровень феромона низкий, или он быстро испаряется. При $Ae = 1$, то влияние элитного муравья приравнивается другим муравьям, но если $1 < Ae < \infty$, то влияние этого муравья возрастает.

Зададим начальное расположение колонии муравьев. В каждой вершине располагается по одному муравью из которой он начинает свой путь. При этом число агентов m рассчитывается по следующей формуле:

$$m=\sum_{i=1}^{n}b_i(t), \qquad (3.6)$$

где i=1, ..., n – число агентов в вершине i в момент времени t.

Каждый агент выбирает путь согласно выражениям 3.2 и 3.3. В качестве ограничения вводится запрет на посещение пройденных вершин до построения всего маршрута. В конце маршрута выставляются соответствующие метки. Их уровень рассчитывается в соответствии с формулой:

$$\tau_{ij}(t+n)=\rho*\tau_{ij}(t)+\Delta\tau_{ij}, \qquad (3.7)$$

где ρ –коэффициент, отвечающий за уровень феромона. При этом данный коэффициент должен быть меньше 1, чтобы исключить неограниченное накопление уровня феромона. При этом $\Delta\tau ij$ рассчитывается исходя из следующей формулы:

$$\Delta\tau_{ij} = \sum_{i=1}^{m} \Delta\tau_{ij}{}^{k}, \tag{3.8}$$

где $\Delta\tau_{ij}{}^{k}$ – частичная приспособленность промежутка между ближайшими вершинами на протяжении всего маршрута, выбранного одним из агентов. Она рассчитывается на основе следующего выражения:

$$\Delta\tau_{ij}{}^{k} = \begin{cases} \dfrac{Q}{L_k}, \\ 0. \end{cases} \tag{3.9}$$

Здесь первое условие выполняется, если агент перемещается согласно выбранному направлению, если нет, то выполнение второго условия.

При равняем интенсивность следов $\tau_{ij}(t)$ в момент времени t=0, некоторой константе c.

Согласно введенному ограничению после прохождения всего маршрута в tabu-лист вводятся значения (пройденная дистанция) для каждого текущего агента. Затем эти значения обнуляются и текущие вершины становятся стартовыми. Зададим tabuk как динамически изменяющийся вектор, a tabuk(s), как список посещенных вершин на одной итерации Размерность данного списка равна числу вершин n рассматриваемого графа. С учетом этого и исходя из выражений 3.2 и 3.3 вероятность перехода агента из одной вершины в другую вычисляется по формуле [108]:

$$p_{ij}{}^{k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} * \left[\eta_{ij}\right]^{\beta}}{\sum_{k \in allowed_k} \left[\tau_{ij}(t)\right]^{\alpha} * \left[\eta_{ij}\right]^{\beta}}, \text{ где } j \in allowed_k \; u \; allowed_k = \{N - tabu_k\} \\ 0 \end{cases}, \tag{3.10}$$

где α, β – задаваемые параметры «памяти» и «зрения», allowedk – список доступных и не пройденных вершин.

Опишем работу данного алгоритма более подробно. Сначала задается графовая модель рассматриваемой задачи и формируется

начальная популяция агентов. Далее задаются параметры муравьиного алгоритма, такие как время, количество итераций. После этого агенты m случайно распределяются во всех вершинах n графа, при этом начальный уровень феромона приравнивается константе c. Затем определяются стартовые вершины и заполняется tabu-лист начальными значениями. Далее в блоках 8–11 реализуется первый цикл алгоритма в котором агенты проходят по всем вершинам согласно выражению 3.10 с занесением в tabu-лист. Затем переход агентов в следующие вершины с расчетом длины пройденного пути переопределения их положения, а также подсчета частичной ЦФ согласно выражению 3.7. На рис. 3.5 рассмотрен пример, иллюстрирующий прохождения муравья по вершинам графа с выбором последующего его движения между вершинами 7 и 3.

В блоках 16–18 наращиваем время, количество циклов и увеличиваем уровень феромона. Блоки 18–19 – наращивание счетчика циклов и обнуление интенсивности. Далее выполняется второй цикл алгоритма где для каждого частичного пути задается новый уровень феромона и проверяется условие NC < NCmax , если да то выполняется обнуление всех значений в tabu-листе и переход к шагу 7, в противном случае конец работы алгоритма с выводом кратчайшего пути.

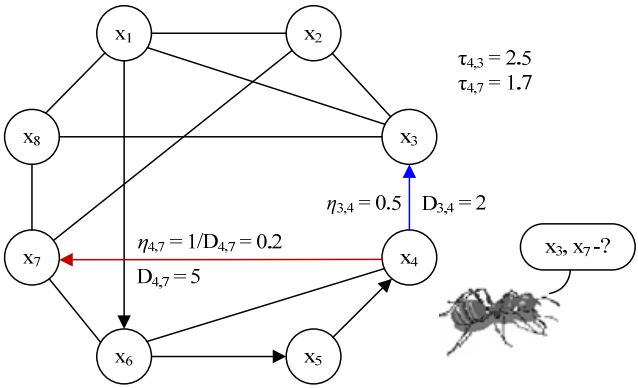


Рис. 3.5. Иллюстрация поведения муравья

Отметим, что время сложность данного алгоритма зависит от числа вершин графа и от выбранного распределения вероятностей, а также изменения параметров α и β и в лучшем случае составляет $O(n^2)$, где n – число вершин рассматриваемого графа.

Для подтверждения эффективности данного алгоритма был проведен ряд экспериментальных исследований, таких как [105].

1. Оценка зависимости эффективности алгоритма от количества вершин в графе. При проведении эксперимента последовательно произведем изменение вершин в графе от 10 до 29 с шагом 1. Исходные данные для проведения эксперимента: $\alpha = 5$, $\beta = 1$, $Q = 10$, количество агентов – 50, максимальное количество циклов алгоритма – 50. Полученную зависимость отобразим на графике (рис. 3.6.)

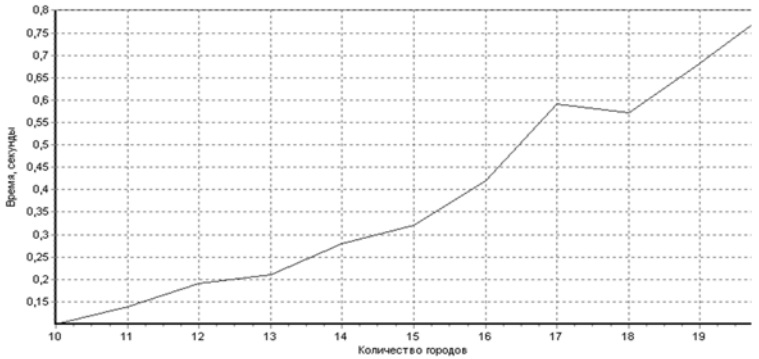


Рис. 3.6. График зависимости времени решения задачи от размерности графа

Анализ полученных экспериментальных данных позволяет сделать вывод, что увеличение количества вершин в графе от 10 до 29 приводит к тому, что время, затрачиваемое на получение эффективного решения, возрастает и принимает значения от 5,8 до 130 секунд соответственно.

2. **Оценка влияния изменения размера популяции на нахождение минимального пути в графе.** В ходе проведения эксперимента было исследовано изменение размера популяции от 30 до 80 агентов с шагом 1. Исходные данные для проведения эксперимента: $\alpha = 5$, $\beta = 1$, $Q = 10$, максимальное количество циклов алгоритма – 50. Количество вершин в анализируемом графе фиксировано и равно 10. Полученная зависимость приведена на графике рис. 3.7.

Исходя из экспериментальных данных, можно сделать вывод, что последовательное увеличение размера популяции приводит к увеличению временных затрат на реализацию рассматриваемого алгоритма. Из формулы (3.8) по мере увеличения m (числа особей, агентов в популяции) следует увеличение $\Delta\tau ij$ – эффективности выбора пути из вершины графа i в j.

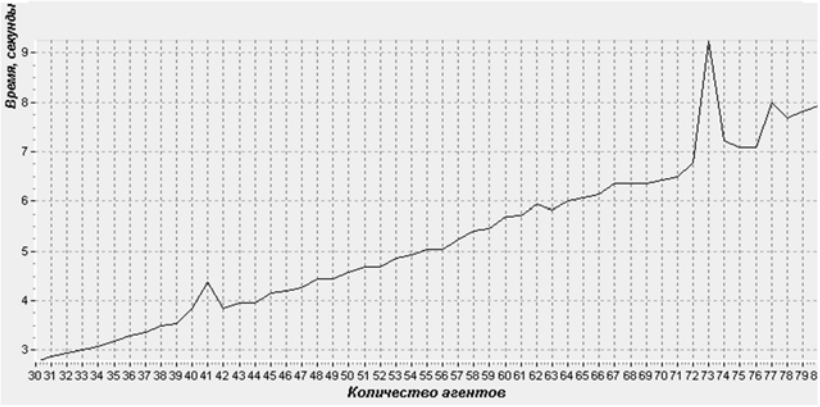


Рис. 3.7. График зависимости времени решения задачи от количества агентов

Из формулы (3.9) видно, что при постоянном Q и возрастающем $\Delta\tau ij$, значение Lk (минимальный путь, пройденный k-ым агентом) должен уменьшаться (обратная зависимость). Однако, учитывая вероятностный характер рассматриваемого алгоритма (формула 3.10), в результате проведенного моделирования при указанных входных параметрах график зависимости длины пути от изменения количества агентов будет иметь следующий вид (рис. 3.8).

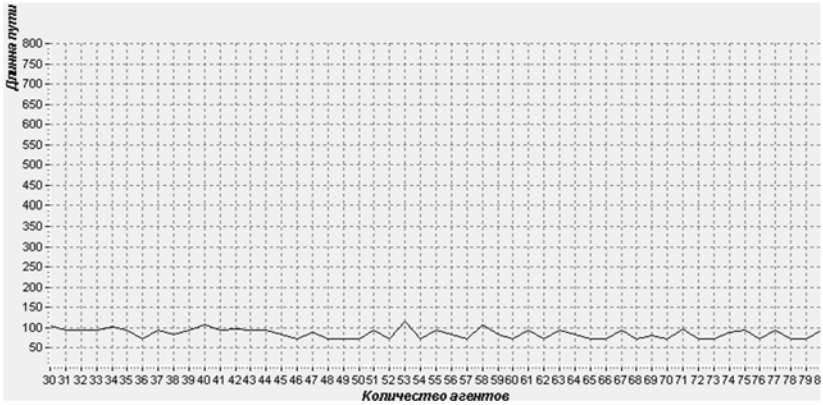


Рис. 3.8. График зависимости длины пути от изменения количества агентов

Таким образом, эффективным решением в данном случае будет количество агентов равное **36**, при этом длина пути составит наименьшее значение.

3. Оценка влияния изменения параметров α и β на минимальное значение пути. Исходя из формулы (3.10), при последовательном увеличении параметра α происходит увеличение значения $p_{ij}^{k}(t)$ (вероятности перехода агента из города i в j в момент времени t). Аналогично и при выборе входного параметра β в качестве варьируемого параметра, также происходит увеличение значения $p_{ij}^{k}(t)$.

Рассмотрим влияние увеличения параметра α (от 1 до 17 с шагом 1) на нахождение минимального цикла. Экспериментальные исследования будем проводить при прочих равных условиях: исследуемый граф содержит 10 вершин, β = 1, Q = 10, максимальное количество циклов алгоритма – 50, количество агентов равно 36. Полученные зависимости – функции времени и критерия пригодности при варьируемом параметре α примут вид рис. 3.9 и 3.10.

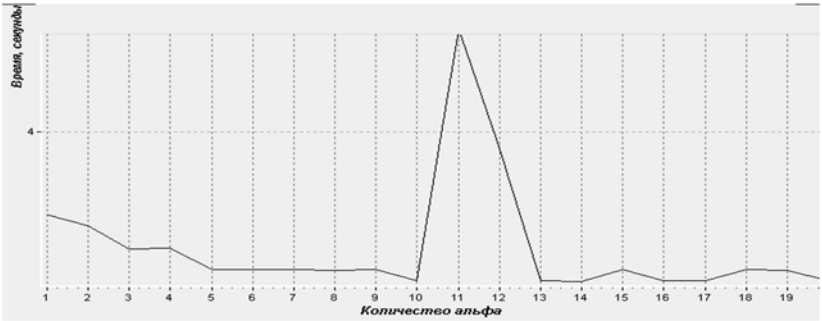


Рис. 3.9. График зависимости времени решения от изменения α

Исходя из результатов проведенных исследований, можно сделать вывод, что эффективное решение поставленной задачи достигается при увеличении значений α (на интервале от 15 до 17) и составляет 124.

Рассмотрим влияние увеличения параметра β (от 1 до 10 с шагом 1) на нахождение минимального цикла. Экспериментальные исследования будем проводить при прочих равных условиях: исследуемый граф содержит 10 вершин, α = 15, Q = 10, максимальное количество циклов алгоритма – 50, количество агентов равно 36.

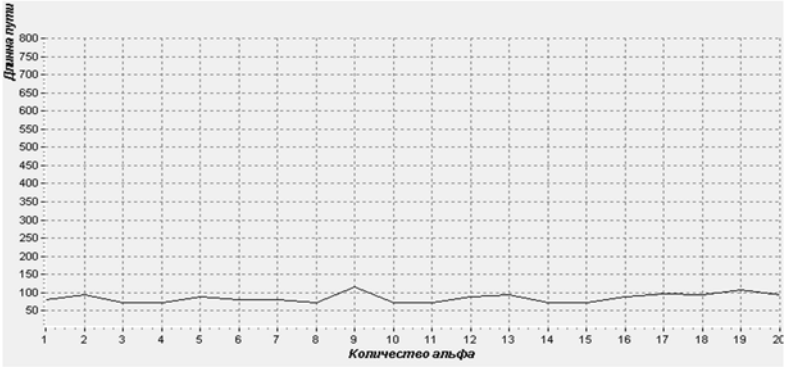


Рис. 3.10. График зависимости длины пути от изменения α

Полученные зависимости – функции времени и критерия пригодности при варьируемом параметре β примут вид рис. 3.11 и 3.12.

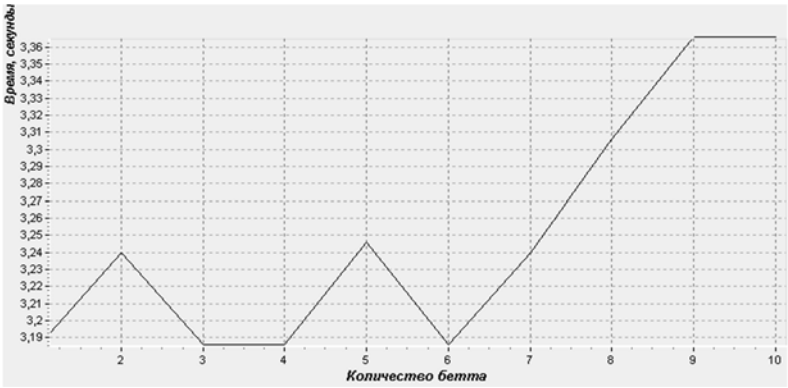


Рис. 3.11. График зависимости времени
решения от изменения β

Исходя из полученных результатов, можно сделать вывод, что достаточно хорошее решение поставленной задачи достигается при изменении β от 1 до 4 и составляет 124.

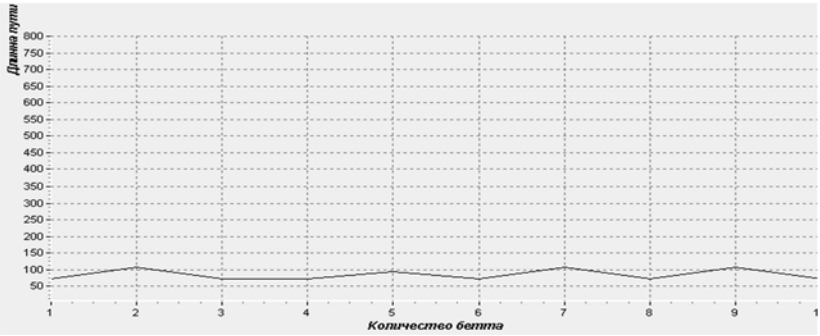


Рис. 3.12. График зависимости длины пути от изменения β

4. Оценка влияния изменения параметра Q на минимальное значение пути. Исходя из формулы (3.9), при последовательном увеличении Q и фиксированном значении L_k происходит увеличение τ_{ij}^k – эффективности выбора пути из вершины графа i в j. Рост величины τ_{ij}^k приводит к увеличению вероятности $p_{ij}^k(t)$, исходя из формулы (3.10). Таким образом, увеличение значения входного параметра Q приводит к накоплению большего количества меток на наиболее эффективных направлениях пути в графе. Экспериментальные исследования будем проводить при изменении Q от 1 до 30 и прочих равных условиях: исследуемый граф содержит 10 вершин, **α=15**, **β=3**, максимальное количество циклов алгоритма – 50, количество агентов равно **36**. Полученные зависимости – функции времени и критерия пригодности при варьируемом параметре β приведены на рис. 3.13 и 3.14.

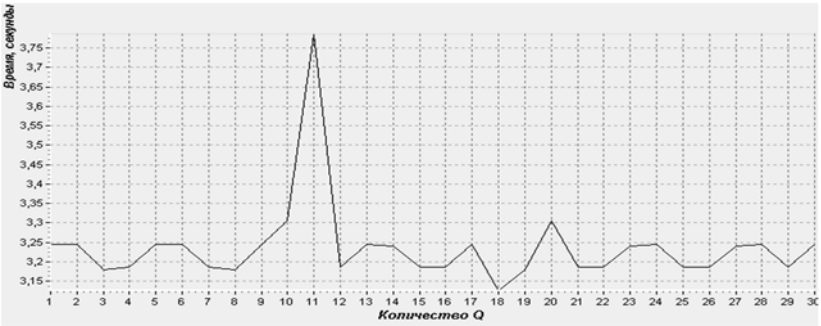


Рис. 3.13. График зависимости времени решения от изменения Q

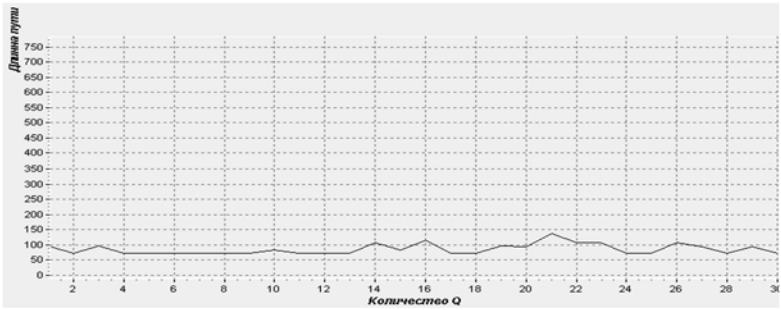


Рис. 3.14. График зависимости длины пути от изменения Q

Исходя из полученных результатов, можно сделать вывод, что изменение параметра Q не оказывает ощутимого влияния, как на функцию времени, так и критерия пригодности. Для избежания попадания в состояние стагнации необходимо выбирать значение Q в пределах от 21 до 30.

5. Оценка влияния изменения количества циклов повторения алгоритма на решение задачи. В ходе проведения экспериментальных исследований будем изменять значение допустимого количества циклов в пределах от 50 до 200 с шагом 1, оставляя неизменными все прочие условия: исследуемый граф содержит 10 вершин, $\alpha = 15$, $\beta = 3$, $Q = 30$, количество агентов равно 42. Результатом исследований является получение эффективного интервала решений при изменении количества итераций алгоритма от 125 до 135. Полученные зависимости – функции времени и критерия пригодности при варьируемом параметре β приведены на рис. рис. 3.15 и 3.16.

Рис. 3.15. График зависимости времени решения
от числа итераций

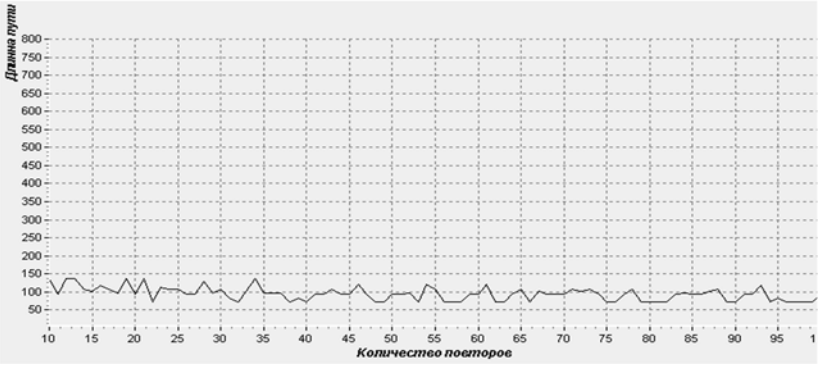


Рис. 3.16. График зависимости длины пути от изменения числа итераций алгоритма

Исходя из полученных результатов, можно сделать вывод, что увеличение числа итераций приводит к накоплению у агентов опыта в выборе наиболее эффективного пути и в итоге получения наилучшего результата.

В заключении отметим, что данный алгоритм позволяет получать наборы квазиоптимальных решений независимо от диапазона изменения варьируемых параметров, исключает зацикливание в локальных оптимумах. Это позволяет сделать вывод о гибкости и устойчивости рассматриваемого метода.

3.3. Модели и метод пчелиной оптимизации

Еще одной разновидностью методов роевого интеллекта является поведение пчелиного роя в живой природе. Ученые смоделировали некоторые аспекты жизнедеятельности колонии пчел. Основной идеей, положенной в основу функционирования этой модели, является совместное исследование перспективных областей пространства (элитные участки) допустимых решений и их окрестностей, что позволяет разнообразить популяцию решений на последующих итерациях и увеличивает вероятность обнаружения близких к оптимальным решений [26, 37, 47, 94, 96, 97, 100, 109, 110]. Рой пчел вылетает в разных направлениях и обрабатывает достаточно большую площадь участков. На тех участках, где находится большее количество нектара, задействуется и большее число пчел рис. 3.17.

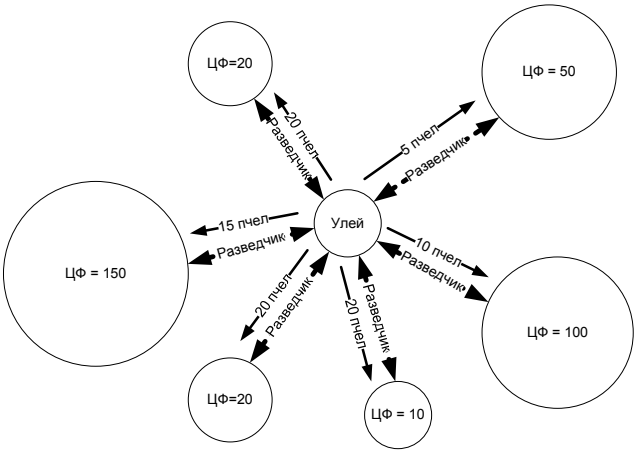


Рис. 3.17. Модель поведения роя пчел в живой природе

Исследование участков закреплено за пчелами разведчиками. После исследования участков они возвращаются в улей и передают с помощью танца и собранного нектара пчелам фуражирам и пчелам исследователям некоторую информацию, такую как в каком направлении находятся найденные участки, на каком расстоянии от улья и их уровень качества. Затем пчелы фуражиры и пчелы исследователи отправляются к самым перспективным участкам, где первые собирают нектар, а вторые исследуют окрестности этих участков. Причем на более перспективные отправляется и большее количество пчел. В то же время разведчики отправляются на поиск новых участков.

Первый алгоритм колонии пчел предложил в своей работе турецкий ученый Д. Карабога в 2005 г. рис. 3.18 [47].

Кратко опишем суть его работы. Сначала формируется начальная популяция пчел, которая сортируется в зависимости от значений ЦФ участков. Далее выбираются элитные участки и в них отправляются пчёлы разведчики для образования новых участков в окрестностях элитных, а в остальные участки отправляются рабочие пчелы. Работа данного алгоритма зависит от следующих основных параметров, таких как: число рабочих пчел и разведчиков, число элитных и других участков, их размер, а также заданное число итераций.

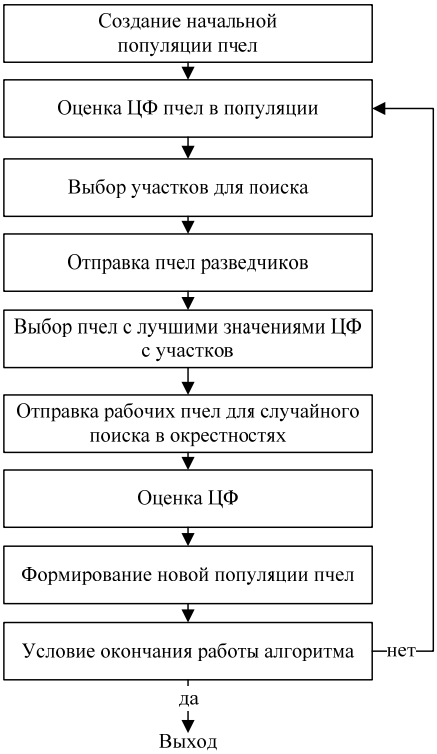


Рис. 3.18. Первый пчелиный алгоритм

В данном алгоритме пчелы исследуют не только элитные участки, но и их окрестности, что позволяет увеличить вероятность нахождения квазиоптимальных решений за счет повышения разнообразия решений в популяции. В 2009 г. D.T. Pham, M. Castellani в своей работе The Bees Algorithm – Modelling Foraging Behaviour to Solve Continuous Optimization Problems привели модификацию стандартного пчелиного алгоритма. Она основана на чётком распределении обязанностей пчел в колонии рис. 3.19 [111].

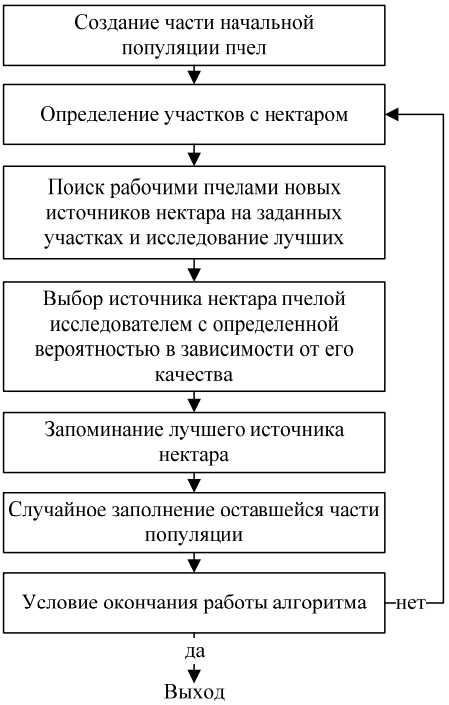


Рис. 3.19. Модификация стандартного пчелиного алгоритма

Проведя анализ рассмотренных алгоритмов, авторы предлагают свой модифицированный пчелиный алгоритм, который будет состоять из четырех ключевых операций рис. 3.20.

В качестве модификации добавлены блок случайного поиска, что позволит избежать попадания алгоритма в локальные оптимумы, а также блок динамического изменения размера окрестностей. Структурная схема данного алгоритма приведена на рис. 3.23 [38, 109, 110, 112]. Опишем последовательность производимых операций. Заметим в данном алгоритме также используются три группы пчёл: пчелы-разведчики, рабочие пчёлы и пчелы исследователи. Сначала формируется начальная популяция пчел. Далее определяется месторасположения источников нектара v_i. Производится отправка пчёл-исследователей и случайным образом для них задается начальная позиция x_i.

Рис. 3.20. Ключевые операции метода пчелиной оптимизации

Для пчел-исследователей участки для поиска выбираются на основе следующей формулы:

$$v_{i,j}=x_{i,j}+\phi_{i,j}*(x_{i,j}-x_{k,j}), \qquad (3.11)$$

где $\phi_{i,j}$ случайное число, находящиеся в интервале *[-1;1]*, k – индекс, который выбирается случайно из всего роя *(k = int(rand * m)+1)*, $j = 1,...,D$, D − размерность задачи.

Затем производится оценка ЦФ, задается радиус окрестности поиска и производится отправка рабочих пчёл, с определенной вероятностью, при этом рассматриваются, как элитные участки, так и их окрестности. Вероятность этого перехода рассчитывается на основе следующего выражения:

$$p_i = \frac{fit_i}{\sum_{j=1}^{S} fit_j}, \qquad (3.12)$$

где fit_i – значение ЦФ x_i.

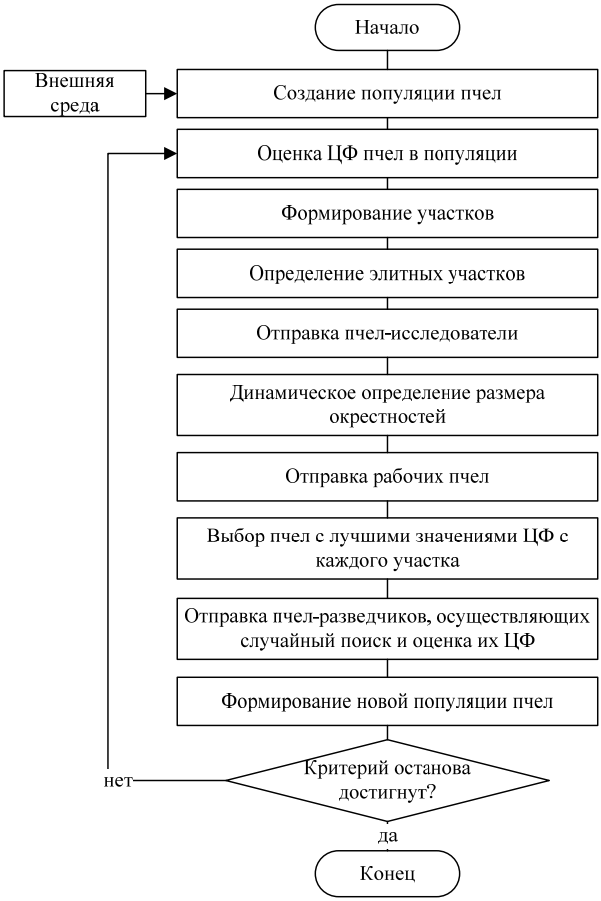


Рис. 3.21. Модифицированная схема алгоритма пчелиной оптимизации

Далее отбираются пчёлы с лучшими значениями ЦФ с каждого источника. При стагнации алгоритма для выхода с этого состояния производится отправка рабочих пчел, осуществляющих случайный поиск с оценкой значений их ЦФ. Случайный поиск осуществляется на основе следующего выражения:

$$x_{ij} = x_j^{min} + (x_j^{max} - x_j^{min}) * rand. \qquad (3.13)$$

После этого формируется новая популяция пчёл, состоящая из двух частей это пчелы как с элитных участков, так и со случайных. Далее процесс продолжается итерационно до достижения критерия остановки при этом участок с наибольшим количеством нектара соответствует нахождению оптимального значения ЦФ.

Данный метод пчелиной оптимизации обладает следующими характеристиками: многоагентность, самоорганизация, положительная и отрицательная обратные связи. Процесс поиска эффективных решений обеспечивается следующими тремя процедурами: Поиск ведется всеми агентами с распределенными обязанностями, что позволяет проводить исследовать все доступное пространства поиска. Проводить детальное исследование перспективных областей и динамике их окрестностей. При зацикливании алгоритма осуществлять случайный поиск.

В заключении подраздела отметим, что предложенный пчелиный алгоритм позволяет быстро динамически разбивать поисковое пространство на области с высоким значением ЦФ, легко распараллеливаться, и применяться для решения дискретных, непрерывных и динамических задач оптимизации.

3.4. Модели и метод светлячковой оптимизации

Следующей разновидностью методов роевого интеллекта является поведение роя светлячков в живой природе [26, 66, 113–115]. Данный метод основан на поведении жуков-лампирид (светлячков) в процессе спаривания. Эти жуки способны выделять особое вещество именуемое люциферином, которое и является коммуникативным средством общения для привлечения противоположного пола. С помощью этого вещества в темное время суток лампириды излучают свечение производя кратковременные вспышки, при этом чем больше светлячок выделяет люциферина, тем ярче свечение, тем привлекательнее он для других особей. Также уровень яркости свечения зависит от расстояния между ними. При этом если в пределах видимости светлячков не наблюдается ярких вспышек, то тогда они случайно передвигаются.

Впервые данную метаэвристику в 2008 году описал Синь-Шэ Янг в своей работе *Nature-Inspired Metaheuristic Algorithms и предложил алгоритм, основанный на моделировании поведения светлячков* рис. 3.22 [66].

Рис. 3.22. Первый светлячковый алгоритм

В этой модели сначала создается популяция светлячков, которая распределяется в пространстве поиска. Затем светлячкам задается определенный уровень люциферина, определяющий яркость свечения и являющийся ЦФ. Кроме яркости для выбора соседа также учитывается расстояние между ними. Этот процесс в алгоритме реализуется на основе вероятностного механизма. В качестве критерия окончания работы используется заданное число итераций.

На основе рассмотренной модели поведения роя святлячков и стандартного алгоритма, авторы предлагают свой модифицированный светлячковый алгоритм рис. 3.23. В качестве модификации предлагается в алгоритм ввести два новых блока. Первый позволит динамически изменять радиус окрестности поиска, а второй выбирать соседа на основе колеса рулетки или элитных правил в зависимости от знаний ЛПР [113–115].

Опишем работу предложенного алгоритма более подробно. Сначала создается начальная популяция светлячков. Задается их число; r_i– радиус окрестности поиска и определенный уровень люциферина – l_i. Затем происходит случайное распределение светлячков в пространстве поиска.

Далее определяем уровень люциферина $l_i(t)$ в зависимости от занимаемой позиции светлячка в пространстве поиска на основе выражения:

$$l_j(t+1) = (1-\rho)l_j(t) + \gamma J_j(t+1), \qquad (3.14)$$

где ρ –коэффициент, находящийся в пределах ($0 < \rho < 1$) и учитывающий ослабление яркости свечения (распад люциферина); γ – коэффициент привлекательности светлячка; J_j– значение целевой функции светлячка в текущий момент времени.

Затем с учетом заданного r_i– (радиуса окрестности поиска) находим соседей $N_i(t)$ для i-х светлячков. Заметим, что i-й светлячок выбирает соседа j, который имеет большую яркость свечения чем собственная из окрестности радиуса на основе вероятностного механизма по формуле:

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)}, \qquad (3.15)$$

где $j \in N_i(t)$, $N_i(t) = \{j: d_{i,j}(t) < r_d^i(t)\}$; $l_i(t) < l_j(t)$, $d_{i,j}(t)$ – эвклидово расстояние между агентами i и j в текущий момент времени; $l_j(t)$ –яркость свечения j-го светлячка; $r_d^i(t)$ – динамически изменяемый радиус окрестности поиска i-го светлячка. Пример такого перемещения показан на рис. 3.24.

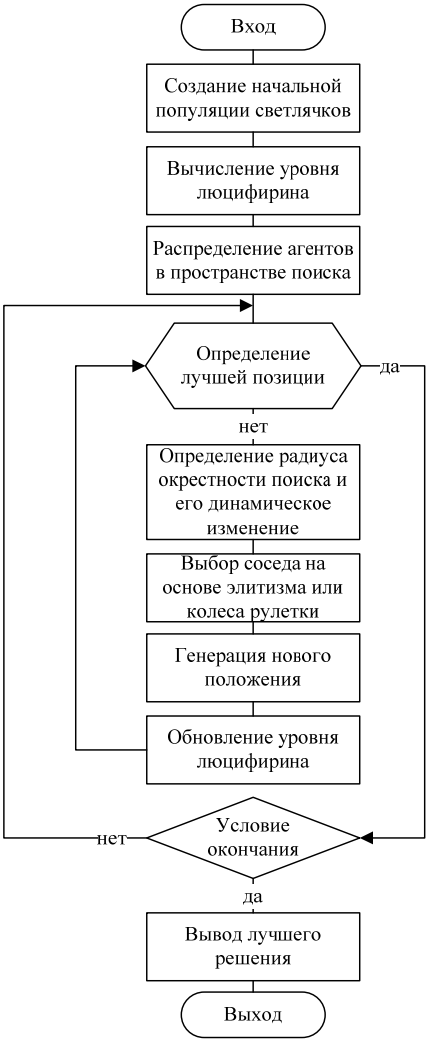


Рис. 3.23. Модифицированный алгоритм светлячковой оптимизации

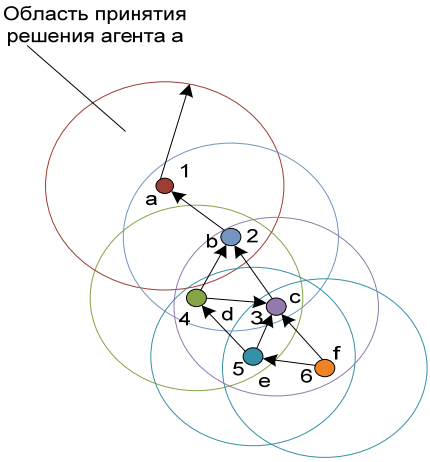


Рис. 3.24. Пример перемещения светлячков

Здесь a, b, c, d, e и f – ранжированные светлячки и их перемещение зависит от яркости свечения и радиуса окрестности поиска. Согласно рисунку видно, как могут двигаться светлячки. Если рассматривать светлячка f, то его яркость самая меньшая и он мог бы двигаться к любому соседу, но в его радиус окрестности входят только светлячки c и e поэтому перемещение может быть только в эти позиции. Далее, согласно алгоритму, генерируется следующее положение светлячка. Выбор соседа при этом производится на основе элитизма или колеса рулетки [3, 8, 36] и тогда новая позиция светлячка определяется на основе следующей формулы:

$$x_i(t+1) = x_i(t) + st * \left\{\frac{x_j(t)-x_i(t)}{\|x_j(t)-x_i(t)\|}\right\}, \qquad (3.16)$$

где st –величина шага, задаваемого ЛПР.

Затем производится процедура обновления радиуса окрестности поиска r_d^i, определяемая на основе представленной формулы:

$$r_d^i(t+1) = \min\left\{r_s, \max\left\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\right\}\right\}, \qquad (3.17)$$

где β –константа, an_t – параметр учета соседей.

В заключении параграфа отметим, что предложенный метод и разработанный на его основе модифицированный алгоритм позволяет достаточно быстро получать наборы квазиоптимальных решений, а за счет введения в его структуру новых процедур избегать попадания в локальные оптимумы.

3.5. Модели и метод бактериальной оптимизации

Одним из эвристических алгоритмов с децентрализованным управлением является алгоритм бактериальной оптимизации (Bacterial Foraging Optimization, BFO), который был впервые предложен в работах Пассино в 2002 году [26, 54, 116–119]. Приведем каноническое описание данного алгоритма, которое включает в себя следующие основные операции, представленные на рисунке 3.25.

Рис. 3.25. Основные операции канонического алгоритма бактериальной оптимизации

Бактерии формируют сложные устойчивые паттерны в некоторых питательных веществах, позволяющие им выжить в среде, если изначально поместить их вместе в ее центр. Поэтому при *инициализации колонии бактерий* (этап 1) задаются следующие свободные параметры алгоритма: число бактерий; размерность пространства поиска; количество шагов процедуры хемотаксиса; число шагов-повторений на одном шаге процедуры хемотаксиса; число шагов в процедуре репродукции; число событий в процедуре ликвидации-рассеивания; вероятность рассеивания бактерий [54].

Характеристики движения бактерий в поисках пищи определяются двумя способами: совместное плавание и кувыркание. Данный процесс называется *хемотаксисом* (этап 2). Считается, что бактерия «плавает», если она движется в нужном направлении, и «кувыркается», если движется в сторону ухудшения среды.

Для того чтобы бактерии достигали наиболее богатого пищей места, наиболее успешная в поиске бактерия привлекает других бактерий, чтобы вместе они быстрее сходились в богатом пищей месте. Для этого к исходной функции стоимости добавляется штрафная функция, основанная на относительном расстоянии каждой бактерии от наиболее приспособленной бактерии до текущей итерации. Когда все бактерии собираются в точке решения, эта штрафная функция становится равной нулю. Эффект *роения* (этап 3) заключается в том, что бактерии собираются в группы и двигаются концентрическими паттернами с высокой плотностью концентрации бактерий.

Исходный набор бактерий, пройдя несколько итераций хемотаксиса, достигает стадии *репродукции* (этап 4). На стадии репродукции половина лучших по результатам поиска пищи бактерий дублируется и заменяет собой половину наихудших бактерий, которые уничтожаются из-за их меньшей способности к поиску пищи. Это делает популяцию бактерий более устойчивой в процессе эволюции.

В процессе эволюции периодически происходят внезапные непредвиденные события, которое иногда приводят к резкому изменению плавного процесса эволюции. В случае с рассматриваемым алгоритмом подобное непредвиденное событие вызывает ликвидацию (элиминацию) множества бактерий и/или рассеивание их в новую среду. *Ликвидация и рассеивание* (этап 5) являются элементами поведения популяции на больших расстояниях. Применительно к оптимизации это помогает уменьшить вероятность попадания в «локальные ямы», что часто наблюдается в алгоритмах параллельного поиска.

Опишем одну из модификаций рассмотренного алгоритма бактериальной оптимизации. Для решения задачи семантического поиска знаний рассмотрим абстрактную многоуровневую онтологическую архитектуру семантических связей в междисциплинарном пространстве знаний. Связи между вершинами заданы множествами отношений C^1 и C^2 онтологий O_1 с подмножеством $R_1^1; R_2^1; \dots; R_i^1; \dots; R_N^1$ и O_2 с подмножеством $R_1^2; R_2^2; \dots; R_j^2; \dots; R_M^2$ соответственно. С целью упрощения вычислений будет исследоваться эквивалентность двух любых онтологий из пространства взаимосвязанных функциональных областей на метауровне знаний. При этом параллельность вычислений реализуется на основе кластерного подхода и алгоритма бактериальной оптимизации. Здесь значение S – это размерность онтологий равная числу

бактерий. Заметим, что для повышения эффективности поиска будут использованы эвристики на основе жадных процедур. На рисунке 3.26 представлены рассматриваемые онтологии, где вершины графов определяют значения атрибутов сравниваемых концептов.

Зададим пороговое значение *d*. Если данное значение больше или равно значению целевой функции $f(R^1, R^2)$, то существует эквивалентная семантическая близость между подмножествами предикатов, принадлежащих концептам P_i^1 и P_j^2.

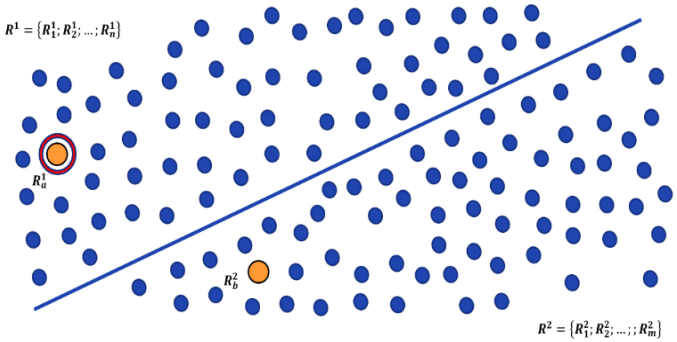


Рис. 3.26. Модель разделенного на подмножества пространства решений

Рассмотрим работу предложенного алгоритма более подробно. Сначала случайным образом выбираются наборы, т. е. пары подмножеств атрибутов с номерами *a* и *b*, где $a \in [1:N], b \in [1:M]$, для каждой бактерии колонии. Все выбранные пары будут составлять выборку. При этом одно подмножество назовем «верхним», а второе – «нижним».

На рисунке 3.27 рассмотрена процедура поиска. Здесь «верхнее» подмножество набора является эталоном-константой, а «нижнее» – текущим положением $X_{i,r,l} = X_{i,r,l}(t)$ бактерии $s_i \in S$ на *t*-м шаге хемотаксиса, *r*-м шаге репродукции и *l*-м шаге ликвидации и рассеивания, при этом $i \in [1:|S|, t \in [1:T], r \in [1:T^r], l \in [1:T^l]$, где T, T^r, T^l – общие числа шагов хемотаксиса, репродукции, ликвидации и рассеивания соответственно (свободные параметры алгоритма), а $|S|$ – четное число бактерии в колонии.

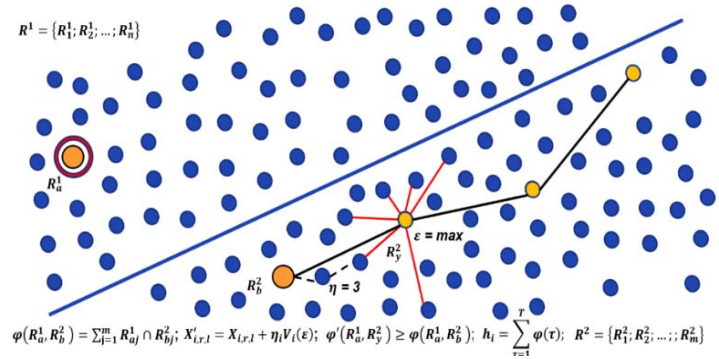


Рис. 3.27. Передвижение бактерии при росте ЦФ

Если в процессе движения происходит уменьшении значения ЦФ, то бактерия производит кувырок и возвращается в предыдущую вершину для изменения дальнейшей траектории движения (рис. 3.28).

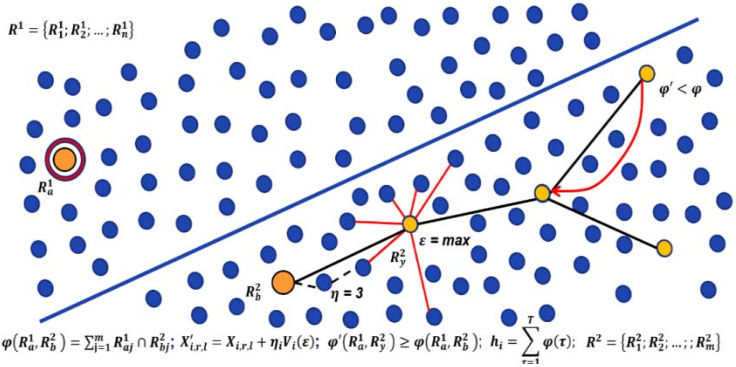


Рис. 3.28. Иллюстрация кувырка бактерии

При попадании бактерии в область локального оптимума во время реализации процедуры хемотаксиса производится изменение текущего положения бактерии и смена статуса подмножеств «верхнего» на «нижний» и наоборот (рис. 3.29).

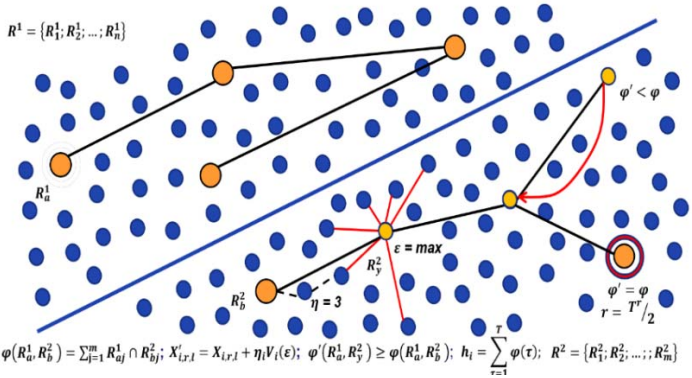


Рис. 3.29. Иллюстрация изменения статусов подмножеств

Для реализации этого механизма (рис. 3.30) в модифицированном алгоритме предусмотрена переменная **α**, имеющая значения «0» – означающее текущее положение бактерии или «1» – константу. Смена их значений является шагом процедуры репродукции ***r*** и означает смену статуса подмножеств, а также текущего положения бактерии.

«upper» subset	R_{a1}^1	R_{a2}^1	…	R_{MAX}^1	α = 1 (const)
«lower» subset	R_{b1}^2	R_{b2}^2	…	R_{MAX}^2	α = 0 (search)

Рис. 3.30. Процедура репродукции

Таким образом, обнаружение максимального подобия (эквивалентности) между наборами подмножеств атрибутов позволяет сделать вывод о высокой эквивалентной семантической близости концептов, которым данные атрибуты принадлежат.

Далее для устранения несоответствия различной размерности подмножеств атрибутов разных концептов обеих онтологий выбирается максимальная размерность (*MAX)*, а незанятые поля в конкретных подмножествах заполняются нулевыми значениями (*NULL)*.

Для определения траектории направления вектора движения бактерии $V_i(\varepsilon)$ по вершинам графа онтологии, соответствующей «нижнему» подмножеству, введем переменную ε_m, где $m \in [1: M]$ для онтологии O_2. Данное значение определяет локальную степень вершин графа (мощность окрестности) онтологии. При изменении

статуса онтологий, для O_l мощность окрестности обозначим переменной ε_n, где $n \in [1:N]$. Тогда движение бактерии осуществляется к вершине, имеющую большую локальную степень. При этом скорость передвижения бактерии будет зависеть от величины шага хемотаксиса, которая задается переменной $\eta \geq 1$, указывающей количество вершин, которые бактерия проходит за один шаг.

Значение данной ЦФ принимает следующий вид [26]:

$$\varphi_{i,r,l} = \varphi_{i,r,l}(T). \quad (3.18)$$

А новое текущее положение $X'_{i,r,l}$ бактерии s_i на $t+1$ шаге хемотаксиса определяется следующим выражением:

$$X'_{i,r,l} = X_{i,r,l} + \eta_i V_i(\varepsilon). \quad (3.19)$$

Заметим, что вектор $V_i(\varepsilon)$ остается неизменным и $V'_i(\varepsilon) = V_i(\varepsilon)$, когда происходит увеличение значения ЦФ. Это демонстрирует, что находится эквивалентность в наборах анализируемых онтологий. В противном случае бактерия совершает «кувырок», возвращаясь в вершину с максимальной ε, для определения новой траектории движения. Данная процедура позволяет находить множество локально-оптимальных решений. На рисунке 3.31 представлена структурная схема предложенного модифицированного алгоритма бактериальной оптимизации (МАБО). Здесь механизм репродукции позволяет обеспечивать процесс поиска локальных оптимумов.

Для реализации данного механизма зададим текущий уровень здоровья (health status) бактерии как сумму значений ЦФ во всех точках пройденной траектории [26]:

$$h_i = \sum_{\tau=1}^{T} \varphi_{i,r,l}(\tau), i \epsilon [1:|S|]. \quad (3.20)$$

Для проведения процедуры репродукции после вычисления значений всех h_i номера бактерий заносятся в список в порядке убывания значений их здоровья, чтобы на $r+1$ шаге исключить из рассмотрения половину наиболее слабых агентов, а каждого агента-бактерию из выживших дублировать копией с координатами, равными координатам дублируемого агента.

Например, если выжившая бактерия $s_j, j \in [1:|S|]$ имеет положение $X_{j,r,l}$, тогда после репродукции появится бактерия s_k, причем, $k = \frac{|S|}{2} + j$, $X_{j,r+1,l} = X_{j,r,l}$, $X_{k,r+1,l} = X_{j,r,l}$. Таким образом, после процедуры репродукции общее количество бактерий в колонии остается неизменным.

Чтобы преодолеть локальные оптимумы и найти глобальный максимум целевой функции в алгоритме используются процедуры ликвидации и рассеивания. Для определения момента запуска этих процедур введем переменную θ_z, где $z \in [1:Z]$, Z – число процедур

репродукции до начала ликвидации и рассеивания. При достижении максимального значения Z включаются процедуры ликвидации и рассеивания [26, 54, 116–119].

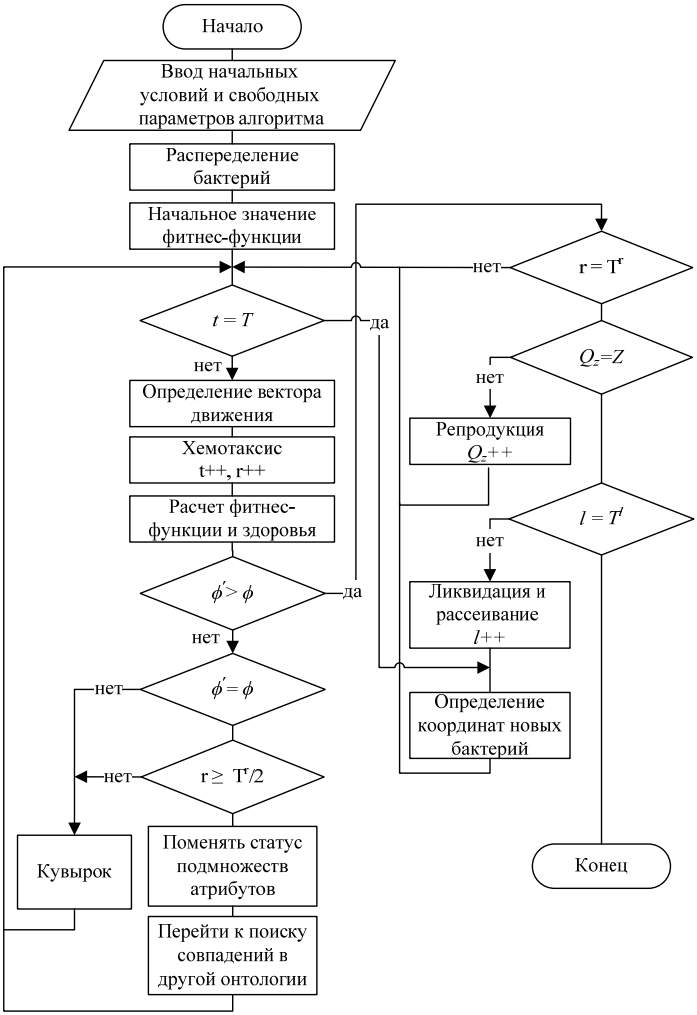


Рис. 3.31. Структурная схема модифицированного метода бактериальной оптимизации

Сначала случайным образом производится уничтожение выбранного количества $w < |S|$ бактерий, а вместо них создается аналогичное число бактерий с начальными координатами в новых случайно выбранных парах подмножеств атрибутов концептов рассматриваемых онтологий. Далее процесс протекает итерационно.

Отметим, что колония бактерий значительной размерности формирует масштабные пространственно-временные структуры со сложной системой отношений, позволяющие повысить эффективность семантического поиска знаний. Данный алгоритм позволяет проводить детальный поиск в локальных областях. При этом эффективность поиска повышается за счет реализации процедуры репродукции, а преодоление локальных оптимумов производится за счет выполнения механизмов ликвидации-рассеивания.

3.6. Модели и метод роя саранчи

Колонии насекомых предоставляют богатый набор метафор для разработки сбалансированных метаэвристик. Такие кооперативные образования представляют собой сложные системы, состоящие из агентов с различными задачами и специализированными паттернами поведения в зависимости от своего типа. В большинстве метаэвристик используются поисковые агенты с одинаковыми свойствами и паттернами поведения. В таких условиях операторы эти алгоритмов теряют аттрактивность, не позволяют улучшить разнообразие популяции и расширить пространство поиска оптимальных решений. Поэтому включение в алгоритм операторов, моделирующих индивидуальные поведенческие характеристики агентов популяции способствует появлению вычислительных механизмов, улучшающих баланс между скоростью сходимости алгоритма и диверсификацией пространства поиска.

Роевой алгоритм и алгоритм дифференциальной эволюции являются наиболее популярными метаэвристиками для решения сложных оптимизационных задач. Однако они имеют определенные недостатки, связанные с преждевременной сходимостью и трудностями преодоления локальных оптимумов.

В частности, в роевом алгоритме проблемы связаны с операторами, которые изменяют местоположение агентов роя и обновляют положение каждого агента роя на очередной итерации, что приводит к их перемещению в направлении лучшей в данный момент особи. В алгоритме дифференциальной эволюции новое решение выбирается для дальнейшего поиска только если оно улучшает предыдущее решение. В результате вся популяция концентрируется вокруг лучшего

решения или хаотично расходится в процессе поиска. Это способствует нарушению баланса между скоростью сходимости алгоритма и диверсификацией пространства поиска. Между тем, коллаборативное роевое поведение относительно простых агентов в популяции обеспечивает их выживание, используя ограниченную локальную информацию и несложные правила поведения.

Саранча является репрезентативным примером насекомых, которые могут сочетать роевое и индивидуальное поведение [51, 120–121]. Это различные паттерны поведения. Индивидуальное поведение предполагает, что саранча избегает контактов и, как следствие, рой распределяется по всему пространству поиска. Роевое поведение предполагает концентрацию саранчи вокруг особей, которым удалось найти источник пищи.

Используем для описания паттернов индивидуального и роевого поведения роя саранчи известную биологическая модель. Согласно этой модели, опишем вначале изменение положения отдельной саранчи при индивидуальном поведении. Пусть что $\boldsymbol{x}_i^k$ представляет собой текущее положение i-й саранчи в рое из N особей. Тогда новое положение $\boldsymbol{x}_i^{k+1}$ особи вычисляется по следующей формуле:

$$\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k + \Delta \boldsymbol{x}_i, \tag{3.21}$$

где $\Delta \boldsymbol{x}_i$ соответствует изменению положения i-й саранчи на (k+1)-й итерации в результате взаимодействия с другими особями роя.

Две саранчи при индивидуальном поведении не стремятся сблизиться, если между ними небольшое расстояние и, наоборот, сближаются, поддерживая сплоченность роя, если между ними значительное расстояние. Сила притяжения/отталкивания определяется как разность:

$$s(r) = kar \cdot e^{-\frac{r}{lim}} - e^{-r}. \tag{3.22}$$

Здесь r – расстояние между парой особей, kar – коэффициент притяжения/отталкивания, lim – пограничное значение расстояния между особями. Если $kar < 1$ и $lim > 1$, то это означает, что между особями небольшое расстояние и отталкивание сильнее нежели притяжение. Тогда сила воздействия j-й саранчи на i-ю саранчу определяется как

$$\boldsymbol{s}_{ij} = s(r_{ij}) \cdot \boldsymbol{d}_{ij}, \tag{3.23}$$

где $r_{ij} = |\boldsymbol{x}_j - \boldsymbol{x}_i|$ – расстояние между j-й и i-й особями роя, а $\boldsymbol{d}_{ij} = (\boldsymbol{x}_j - \boldsymbol{x}_i)/ r_{ij}$ – единичный вектор. Тогда общая сила притяжения/отталкивания роя для i-й саранчи определяется как суперпозиция всех парных взаимодействий:

$$\boldsymbol{S}_i = \sum_{j=1,\ j \neq i}^{N} \boldsymbol{s}_{ij}. \tag{3.24}$$

Изменение положения i-й саранчи $\Delta \boldsymbol{x}_i$ соответствует (3.4):

$$\Delta \boldsymbol{x}_i = \boldsymbol{S}_i. \qquad (3.25)$$

В отличие от индивидуального поведения при роевом поведении саранча стремительно концентрируется вокруг особей, которые нашли источники пищи. Для того чтобы смоделировать роевое поведение, вводится для каждой саранчи $\boldsymbol{x}_i$ индекс пищи f_i ($f_i \in [0, 1]$ Далее N особей популяции ранжируются по убыванию этого индекса, а затем среди них выбираются b особей ($b << N$) с наивысшими показателями пищи. Вокруг каждой из b особей в радиусе R_c случайным образом концентрируются подмножество саранчи.

Представим алгоритм роевого и индивидуального поведения саранчи (МАРС) [121].

Предположим, что все пространство поиска представляет собой плантацию, где саранча взаимодействует друг с другом. Каждое решение в пространстве поиска представляет положение саранчи на плантации и характеризуется значением функции пригодности, отражающей уровень пищевого индекса. Алгоритм реализует паттерны индивидуального и роевого поведения, которые управляются набором операторов, моделирующих эти поведенческие паттерны.

Популяция $\boldsymbol{L}^k(\{\boldsymbol{l}_1^k, \boldsymbol{l}_2^k, ..., \boldsymbol{l}_N^k\})$ из N особей эволюционирует из начального положения ($k = 0$) к заданному числу поколений ($k = gen$). Каждая саранча $\boldsymbol{l}_i^k$ ($i = 1..N$) представляет собой n-мерный вектор $\{l_{i1}^k, l_{i2}^k, ..., l_{in}^k\}$, в котором каждый элемент соответствует переменной решения задачи оптимизации. Множество переменных решений составляет допустимое пространство поиска $\boldsymbol{S} = \{\boldsymbol{l}_i^k \in \boldsymbol{R}^n \,|\, lb_d \le l_{id}^k \le ub_d\}$, где lb_d и ub_d соответствуют нижней и верхней границам размерности d соответственно. Уровень пищевого индекса, связанный с каждой саранчой, оценивается с помощью функции $f_i(\boldsymbol{l}_i^k)$.

В алгоритме МАРС на каждой итерации процесса эволюции применяются два оператора поведения: **A** – индивидуальный и **B** – роевой. Оператор **A** применяется для диверсификации пространства поиска решений, а оператор **B** – для уточнения решения в определенной области пространства.

Рассмотрим подробнее каждый из операторов.

Оператор **A**, реализующий паттерн индивидуального поведения саранчи, изменяет текущее положение $\boldsymbol{l}_i^k$ i-й ($i = 1..N$) саранчи на

величину $\Delta \boldsymbol{l}_i^k$: $\boldsymbol{p}_i = \boldsymbol{l}_i^k + \Delta \boldsymbol{l}_i^k$ с учетом значения функции пригодности и положения доминирующих особей роя. Сила притяжения/отталкивания между *j*-й и *i*-й особями, рассчитывается как

$$\boldsymbol{s}_{ij}^m = \rho\left(\boldsymbol{l}_i^k, \boldsymbol{l}_j^k\right) \cdot s\left(r_{ij}\right) \cdot \boldsymbol{d}_{ij} + rand(1, -1), \qquad (3.26)$$

где $s(r_{ij})$ определяется согласно (2); $\boldsymbol{d}_{ij} = (\boldsymbol{l}_j^k - \boldsymbol{l}_i^k)/\, r_{ij}$ – единичный вектор, направленный от $\boldsymbol{l}_i^k$ к $\boldsymbol{l}_j^k$; *rand*(–1, 1) – случайное число из интервала (–1, 1); $\rho(\boldsymbol{l}_i^k, \boldsymbol{l}_j^k)$ – функция доминирования между *j*-й и *i*-й особями. Для определения ρ все особи популяции $\boldsymbol{L}^k(\{\boldsymbol{l}_1^k, \boldsymbol{l}_2^k, ..., \boldsymbol{l}_N^k\})$ ранжируются по убыванию их функций пригодности. Лучшей особи присваивается ранг 0, худшая особь получает ранг *N*–1. Таким образом, функция $\rho(\boldsymbol{l}_i^k, \boldsymbol{l}_j^k)$ определяется следующим образом

$$\rho\left(\boldsymbol{l}_i^k, \boldsymbol{l}_j^k\right) == \begin{cases} e^{-(5 \cdot rank(l_i^k)/N)}, & \text{если } rank\left(l_i^k\right) < rank(l_j^k) \\ e^{-(5 \cdot rank(l_j^k)/N)}, & \text{если } rank\left(l_i^k\right) > rank(l_j^k) \end{cases}, \qquad (3.27)$$

где функция *rank*(α) указывает на ранг особи. Согласно (2.7) функция ρ принимает значения из интервала (0, 1), причем значение 1 достигается, когда одна из особей является лучшим элементом популяции, а значение близкое к 0 – когда обе особи обладают низкими показателями функции пригодности.

Наконец, общая сила притяжения/отталкивания, действующая на *i*-ю особь, вычисляется как суперпозиция всех парных взаимодействий:

$$\boldsymbol{S}_i^m = \sum_{j=1,\ j \neq i}^{N} \boldsymbol{s}_{ij}^m \quad . \qquad (3.28)$$

Изменение положения $\Delta \boldsymbol{l}_i$ определяется следующим образом:

$$\Delta \boldsymbol{l}_i = \boldsymbol{S}_i^m. \qquad (3.29)$$

После вычисления новых позиций $\boldsymbol{P}(\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N\})$ особей популяции $\boldsymbol{L}^k$, необходимо изменить значения функций пригодности $\boldsymbol{F}(\{\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_N.\})$. При этом допускаются только те изменения, которые гарантируют улучшение результатов поиска. Иными словами, если $f_i(\boldsymbol{p}_i) > f_i(\boldsymbol{l}_i^k)$, то принимается новая позиция $\boldsymbol{p}_i$, иначе – сохраняется позиция $\boldsymbol{l}_i^k$:

$$f_i = \begin{cases} p_i, \text{если } f(p_i) < f\left(l_i^k\right), \\ l_i^k, \text{в противном случае.} \end{cases} \qquad (3.30)$$

Оператор **B**, реализующий паттерн роевого поведения саранчи, направлен на уточнение решения в определенной области пространства поиска. Для его выполнения вначале проводится сортировка функций пригодности особей по убыванию. Результаты сортировки

сохраняются в множестве $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_N\}$. Среди них выделяются g наилучших особей, имеющих наибольшее значение функции пригодности. Они образуют подмножество $\boldsymbol{E}$ наиболее перспективных решений. Вокруг каждой особи с $f_i \in \boldsymbol{E}$ создается подпространство C_j радиусом, который определяется следующим образом:

$$e_d = \frac{\sum_{q=1}^{n}(ub_q - lb_q)}{n} \cdot \beta, \qquad (3.31)$$

где ub_q и lb_q – верхняя и нижняя границы в q-м измерении, n – размерность переменных оптимизационной задачи, $\beta \in [0, 1]$ – параметр алгоритма. Границы подпространства C_j моделируются следующим образом:

$$uss_j^q = b_{j,q} + e_d, lss_j^q = b_{j,q} - e_d\,, \qquad (3.32)$$

где $uss^q{}_j$ и $lss^q{}_j$ – верхняя и нижняя границы в q-м измерении подпространства C_j соответственно. Внутри этого подпространства случайно генерируются h ($h < 4$) новых особей, среди которых выбирается особь с наилучшим значением функции пригодности.

Таким образом, алгоритм МАРС имеет 5 настраиваемых параметров: коэффициент притяжения/отталкивания *kar*, пограничное значение расстояния между особями *lim*, количество наилучших решений *g*, размер популяции *N* и число поколений *gen*.

Алгоритм включает шаги инициализации, выполнения индивидуального и роевого операторов.

При инициализации ($k = 0$) формируется начальная популяция $\boldsymbol{L}^0(\{\boldsymbol{l}_1{}^0, \boldsymbol{l}_2{}^0, \ldots, \boldsymbol{l}_N{}^0\})$. Значения $(\{l_{i1}{}^0, l_{i2}{}^0, \ldots, l_{in}{}^0\}$ каждого отдельного измерения d распределены случайным образом и равномерно между предварительно заданной нижней начальной границей параметра lb_d и верхней начальной границей параметра ub_d:

$$l_{ij}^0 = lb_d + rand \cdot (ub_d - lb_d), \qquad (3.33)$$

где $i = 1..N$, $d = 1..n$. В процессе итеративного выполнения алгоритма индивидуальный оператор **A** и роевой оператор **B** исполняются до тех пор, пока не будет достигнуто число итераций $k = gen$. Псевдокод алгоритма МАРС имеет следующий вид:

1: *Input*: *kar*, *lim*, *g*, *N*, *gen*
2: Инициализация $L^0(k{=}0)$
3: *until* (*k*=*gen*)
4: $F \leftarrow$ оператор$A(L^k)$
5: $L^{k+1} \leftarrow$ оператор$B(L^k, F)$

6: $k = k + 1$

7: *end until*

В отличие от многих и роевых алгоритмов, данная метаэвристика на основе паттернов поведения саранчи позволяет избежать концентрации особей на текущих наилучших позициях, снижает вероятность преждевременной сходимости, поддерживает баланс между скоростью сходимости алгоритма и диверсификацией пространства поиска решений.

Алгоритм МАРС экспериментально проверялся на многомерных ($n = 30$) тестовых функциях Розенброка ($f_1(X)$ роз), сферической ($f_2(X)$ сф), Экли ($f_3(X)$ экл), Швефеля ($f_4(X)$ шв), квадратов ($f_5(X)$ ква), Растригина ($f_6(X)$ рас), Саломона ($f_7(X)$ сал):

- функция Розенброка $f_1(X) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$; $x_i \in [-30, 30]^n$, $x^* = (1,\ldots,1)$, $f_1(x^*) = 0$;
- сферическая функция $f_2(X) = \sum_{i=1}^{n} x_i^2$, $x_i \in [-100, 100]^n$, $x^* = (0,\ldots,0)$, $f_2(x^*) = 0$;
- функция Экли $f_3(X) = -20\exp\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20$; $x_i \in [-32, 32]^n$, $x^* = (0,\ldots,0)$, $f_3(x^*) = 0$;
- функция Швефеля $f_4(X) = \sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$, $x_i \in [-100, 100]^n$, $x^* = (0,\ldots,0)$, $f_4(x^*) = 0$;
- сумма квадратов $f_5(X) = \sum_{i=1}^{n} i x_i^2$, $x_i \in [-10, 10]^n$, $x^* = (0,\ldots,0)$, $f_5(x^*) = 0$;
- функция Растригина $f_6(X) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$, $x_i \in [-5{,}12; 5{,}12]^n$, $x^* = (0,\ldots,0)$, $f_6(x^*) = 0$;
- функция Саломона $f_7(X) = -\cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0{,}1\sqrt{\sum_{i=1}^{n} x_i^2} + 1$, $x_i \in [-100, 100]^n$, $x^* = (0,\ldots,0)$, $f_7(x^*) = 0$.

Здесь n – размерность функции, x^* – оптимальное решение, $f_i(x^*)$ – минимальное значение функции.

Эксперименты проводились в программной среде на языке программирования *C#*. При отладке и тестировании использовался компьютер *IBM PC* с процессором *Core i7* с ОЗУ-8 Гб.

В экспериментах использовались следующие настройки параметров алгоритма МАРС: $kar = 0{,}6$, $lim = 1$, $N = 50$, g = 20, $gen = 1000$.

По каждой функции проводилось 30 прогонов. Затем полученные результаты усреднялись. Определялись среднее значение по лучшим решениям (*Mean best*), медианное лучшее решение (*Med. best*) и стандартное отклонение от лучшего решения (*St. dev*).

Усредненные результаты алгоритма МАРС по 30 отдельным запускам, приведены в таблице 3.1.

Таблица 3.1

Результаты работы алгоритма МАРС

Функция	$f_1(X)$	$f_2(X)$	$f_3(X)$	$f_4(X)$	$f_5(X)$	$f_6(X)$	$f_7(X)$
Mean best	$1{,}5\cdot 10^{-02}$	$9{,}2\cdot 10^{-06}$	$3{,}4\cdot 10^{-05}$	$8\cdot 10^{-02}$	$10\cdot 10^{-04}$	$8\cdot 10^{-02}$	$8\cdot 10^{-02}$
Med. best	$1{,}9\cdot 10^{-02}$	$7{,}9\cdot 10^{-06}$	$2{,}6\cdot 10^{-05}$	$6\cdot 10^{-02}$	$5\cdot 10^{-04}$	$7\cdot 10^{-02}$	$6\cdot 10^{-02}$
St. dev	$6{,}3\cdot 10^{-03}$	$2{,}2\cdot 10^{-06}$	$3{,}7\cdot 10^{-06}$	$3\ 10^{-02}$	$2\ 10^{-04}$	$5{,}4\ 10^{-03}$	$2{,}2\ 10^{-03}$

3.7. Модели и метод колонии пауков

В [122] был представлен алгоритм оптимизации, моделирующий кооперативное поведение колонии социальных пауков, и биологическая модель их взаимодействия. Используем эту модель для представления паттернов поведения в распределенной самоорганизующейся сети и построения модифицированного алгоритма колонии пауков (МАКП) [123–124].

Основными функциями, характеризующими поведение пауков в колонии, являются строительство сетевой паутины, размножение и охота. При этом инструментом взаимодействия агрегации пауков является сеть. Сеть представляет собой канал связи через вибрации и постукивания. Вибрации используются пауками для синхронизации и декодирования своих действий, а также для авторизации особи, передающей сообщение. По интенсивности вибраций и постукиваний определяется вес паука и расстояние до него. Каждый шаг паука вызывает вибрацию. Поэтому, например, при охоте колония перестает двигаться, чтобы по сети почувствовать добычу и понять правильное направление движения. Колония использует определенную тактику и командную работу для защиты своих ресурсов и потомства с дифференциацией ролей.

Определим согласно биологической модели, следующие паттерны поведения пауков: размножение и кооперация. Альфа-самцы пауков отличаются большим весом в сравнении с остальными самцами. При размножении они стремятся двигаться по сети к ближайшей самке. В отличие от альфа-самцов, недоминирующие самцы, в основном, сосредотачиваются в центре колонии, чтобы воспользоваться ресурсами альфа-самцов.

С учетом этого МАКП включает следующую последовательность шагов.

Шаг 1. Инициализация популяции пауков $S = B \cup M$ размером N. Поскольку согласно биологической модели, в колонии преобладают самки, то их число N_b определяется по формуле:

$$N_b = floor[(0{,}9 - rand \cdot 0{,}25) \cdot N]$$

где *rand* – случайное число на интервале [0, 1], *floor*(·) функция преобразования действительных чисел в целые числа. Общее число самцов $N_m = N - N_b$.

В целом колония социальных пауков S размером N состоит из подмножеств $B = \{b_1, b_2, \ldots, b_{Nb}\}$ и $M = \{m_1, m_2, \ldots, m_{Nm}\}$.

Шаг 2. Случайная инициализация позиций пауков. Позиции самцов b_i и самок m_j генерируются случайно и равномерно. Для этого задаются нижний начальный параметр $q_k^{\text{ниж}}$ и верхний начальный параметр $q_k^{\text{верх}}$:

$$b_{i,k}^0 = q_k^{\text{ниж}} + rand(0{,}1) \cdot (q_k^{\text{верх}} - q_k^{\text{ниж}}),$$
$$m_{j,k}^0 = q_k^{\text{ниж}} + rand(0{,}1) \cdot (q_k^{\text{верх}} - q_k^{\text{ниж}}),$$

где $i = 1..N_b$, $j = 1..N_m$, $k = 1..n$, функция *rand*(0, 1) генерирует случайное число в интервале от 0 до 1.

Шаг 3. Диапазон взаимодействия в колонии социальных пауков определяется размерами пространства поиска по следующей формуле:

$$r = \frac{\sum_{k=1}^{n}(q_k^{\text{верх}} - q_k^{\text{ниж}})}{2 \cdot n}.$$

Шаг 4. Вероятность участия в процессе размножения зависит от веса паука и вычисляется по методу рулетки:

$$P_{s_i} = \frac{w_i}{\sum_{k \in T^g} w_k},$$

где T^g – множество пауков в радиусе r. При этом сохраняется первоначальная пропорция между самками и самцами в популяции S.

Шаг 5. Позиция самки b_i на итерации алгоритма (t+1) изменяется по формуле

$$b_i^{t+1}$$
$$= \begin{cases} b_i^t + \alpha \cdot Vib_{i,u} \cdot (m_u - b_i^t) + \beta \cdot Vib_{i,maxw} \cdot (m_{maxw} - b_i^t) + \delta \cdot \left(rand - \frac{1}{2}\right) \\ \text{с вероятностью } PF \\ b_i^t - \alpha \cdot Vib_{i,u} \cdot (m_u - b_i^t) - \beta \cdot Vib_{i,maxw} \cdot (m_{maxw} - b_i^t) + \delta \cdot \left(rand - \frac{1}{2}\right) \\ \text{с вероятностью } 1 - PF, \end{cases}$$

Здесь моделируются вибрации паутины. Вибрации зависят от веса и расстояния до паука, который их породил. Иными словами, особи рядом с пауком, создающим вибрации, улавливают их как более сильные, нежели особи, находящиеся вдали. Вибрация, которую i-й паук улавливает от паука j-го паука, определяется как $Vib_{i,j}$

$$Vib_{i,j} = w_j \cdot e^{-d_{i,j}^2},$$

где $d_{i,j} = \|s_i - s_j\|$ – евклидово расстояние между i-м и j-м пауками. Модификация алгоритма заключается в использовании в МАКП следующих вариантов вибраций:

- $Vibc_i = w_c \cdot e^{-d_{ci}^2}$. Это вибрации, улавливаемые i-м пауком от ближайшего к нему паука c, обладающим большим весом ($w_c > w_i$);
- $Vibb_{i,maxw} = w_{maxw} \cdot e^{-d_{bi,maxw}^2}$. Это вибрации, улавливаемые i-м пауком от паука $maxw$ с максимальным весом в колонии;
- $Vibf_i = w_f \cdot e^{-d_{fi}^2}$. Это вибрации, улавливаемые i-м пауком от ближайшей самки f.

Шаг 6. Выполняется оператор, моделирующий движение самца паука:

$$m_i^{t+1} = \begin{cases} m_i^t + \alpha \cdot Vib_{i,b} \cdot (s_b - m_i^t) + \delta \cdot \left(rand - \frac{1}{2}\right), \text{если } w_{N_{\text{ср}}} > w_i \\ m_i^t + \alpha \cdot \left(\frac{\sum_{h=1}^{N_m} m_h^t \cdot w_{N_b+h}}{\sum_{h=1}^{N_m} w_{N_b+h}} - m_i^t\right), \text{если } w_i > w_{N_{\text{ср}}} \end{cases},$$

где s_b – ближайшая к пауку самка, а величина $\left(\frac{\sum_{h=1}^{N_m} m_h^t \cdot w_{N_b+h}}{\sum_{h=1}^{N_m} w_{N_b+h}} - m_i^t\right)$ представляет средневзвешенный вес самцов M в колонии.

Оператор определяет различные поведенческие паттерны, что способствует диверсификации поиска оптимума. К тому же уменьшается влияние очень хороших и плохих решений на результаты поиска.

Шаг 7. Выполняется оператор размножения в определенном диапазоне *r*, который вычисляется как

$$P_{s_i} = \frac{w_i}{\sum_{k \in T^g} w_k},$$

где T^g – множество пауков в радиусе *r*, которые участвуют в размножении.

Шаг 8. Останов алгоритма происходит при достижении заданного максимального числа итераций. Иначе – возврат к шагу 4 алгоритма.

Алгоритм МАРС экспериментально проверялся на многомерных ($n = 30$) тестовых функциях Розенброка ($f_1(X)$ роз), сферической ($f_2(X)$ сф), Экли ($f_3(X)$ экл), Швефеля ($f_4(X)$ шв), квадратов ($f_5(X)$ ква), Растригина ($f_6(X)$ рас), Саломона ($f_7(X)$ сал).

В экспериментах использовался следующий параметр настройки алгоритма МАКП: $PF = 0{,}7$.

По каждой функции проводилось 30 прогонов. Затем полученные результаты усреднялись. Определялись среднее значение по лучшим решениям (*Mean best*), медианное лучшее решение (*Med. best*) и стандартное отклонение от лучшего решения (*St. dev*). Усредненные результаты алгоритма МАКП по 30 отдельным запускам, приведены в таблице 3.2.

Таблица 3.2

Результаты работы алгоритма МАКП

Функция	$f_1(X)$	$f_2(X)$	$f_3(X)$	$f_4(X)$	$f_5(X)$	$f_6(X)$	$f_7(X)$
Mean best	$8{,}8 \cdot 10^{-02}$	$5 \cdot 10^{-05}$	$4{,}2 \cdot 10^{-05}$	$9 \cdot 10^{-02}$	$9 \cdot 10^{-04}$	$8 \cdot 10^{-02}$	$8 \cdot 10^{-02}$
Med. best	$5{,}1 \cdot 10^{-02}$	$4\ 10^{-05}$	$1{,}5 \cdot 10^{-05}$	$5 \cdot 10^{-02}$	$7 \cdot 10^{-04}$	$7 \cdot 10^{-02}$	$6 \cdot 10^{-03}$
St. dev	$2{,}7\ 10^{-02}$	$9\ 10^{-06}$	$8{,}2 \cdot 10^{-06}$	$3\ 10^{-02}$	$2\ 10^{-04}$	$2\ 10^{-03}$	$3\ 10^{-03}$

ГЛАВА 4. МОДЕЛИ И МЕТОДЫ, ИНСПИРИРОВАННЫЕ ФАУНОЙ

4.1. Модели и метод обезьяньей оптимизации

Ученые давно стали изучать поведение различных животных, включая движения для добычи пищи. Это прыжки, бег, ползание, лазание и многие другие. Данные типы движения животных позволяют моделировать их поведение для создания метаэвристических алгоритмов.

Одними из них являются методы обезьяньего поиска. Заметим, что обезьяны ведут групповой образ жизни. Существует два метода обезьяньего поиска, рассмотрим их более подробно.

Алгоритм обезьяньего поиска (Monkey Search, MS) впервые был предложен Мучерино (A. Mucherino) и Шереф (O. Sheref) в 2007 г. [125]. В основе этого алгоритма лежит поведение обезьяны, передвигающейся по дереву в поиске пищи. При этом каждая обезьяна способна находить лучшие плоды, что соответствует нахождению экстремума ЦФ. В данном алгоритме рассматриваются бинарные деревья поиска, в которых от каждой ветки отходят две другие, содержащие решения на их концах.

Рассмотрим шаги данного алгоритма более подробно. Если в текущий момент времени рассматриваемая обезьяна находится на конце ветви, то затем она с равной долей вероятности может переместиться как по левой, так и по правой исходящим ветвям. В конце ветки вычисляется значение целевой функции. Если это значение лучше найденного ранее, то записываем его, и подобным образом обезьяна продолжает двигаться вверх. По достижении вершины дерева, которая определяется максимально допустимой высотой, движение обезьяны прекращается. При этом все ветви дерева, посещенные обезьяной, запоминаются. В том случае, если были исследованы не все пути в дереве, то каждый раз, когда обезьяна достигает вершины дерева, она возвращается к текущей лучшей точке и продолжает движение вверх. При этом допускается прохождение некоторых уже пройденных ветвей.

Заметим, что особое внимание следует уделить методу генерации ветвей поискового дерева, а именно генерации решений, находящихся по соседству с текущим расположением обезьяны. Эти решения возможно получить с помощью любых локальных методов. Чтобы уменьшить вероятность преждевременной сходимости алгоритма, ограничивается максимальное число путей, которое необходимо исследовать. По достижении этого числа, запоминается

наилучшее полученное решение. Далее из произвольной точки поискового пространства – «выращивается» новое дерево. Когда исследовано заданное число деревьев поиск останавливается.

Второй подход основан на поведении групп обезьян в поисках пищи и проживающих в гористой местности. В 2008 году Жао (R. Zhao) и Танг (W. Tang) опубликовали работу Monkey Algorithm for Global Numerical Optimization в которой предложили и описали обезьяний алгоритм, моделирующий поведение группы обезьян в процессе поиска пищи в горах [126]. Согласно проведенным исследованиям, было замечено, что с набором высоты пищи становится все больше. Они представили гористую местность исследуемой группой обезьян в виде ландшафта целевой функции, где самая высокая точка горы будет соответствовать нахождению оптимального решения. Описанный ими классический обезьяний метод состоит из трех основных операций: «Движение вверх», «Локальный прыжок» и «Глобальный прыжок» рис. 4.1 [26, 126, 127].

Рис. 4.1. Основные операции классического метода обезьяньей оптимизации

Согласно рис. 4.1, классический обезьяний алгоритм представляется следующим образом рис. 4.2.

Рис. 4.2. Классический обезьяний алгоритм

Кратко опишем его работу. Сначала, как и во всех биоинспирированных алгоритмах создается популяция обезьян и размещается случайным образом. Затем каждая обезьяна из своего начального положения движется вверх с запоминаем лучшего результата. После достижения вершины обезьяны совершают в разных направлениях заданную серию локальных прыжков для поиска более высокой горы. При ее нахождении набор высоты продолжается. После выполнения этого этапа поиска обезьяны совершают глобальные (длинные) прыжки для исследования других частей гор. Процесс повторяется итерационно в зависимости от заданного числа прыжков, при этом оптимуму соответствует самая высокая вершина

горы. Заметим, что набор высоты может выполняться на основе методов одномерного или локального поиска, прыжки реализуются на основе известных выражений [26]. Данный метод, как и другие биоинспирированные подходы позволяют эффективно решать различные классы оптимизационных задач включая Big data.

Проведя анализ рассмотренных подходов, авторы предлагают свою модификацию, состоящую из пяти ключевых операций рис. 4.3 [128, 129]. Введение двух дополнительных операций набора высоты после выполнения локальных и глобальных прыжков позволит более тщательно исследовать новые области поиска и их окрестности, что приведет к получению набора эффективных квазиоптимальных решений, а также избегать преждевременной сходимости алгоритма в локальных областях.

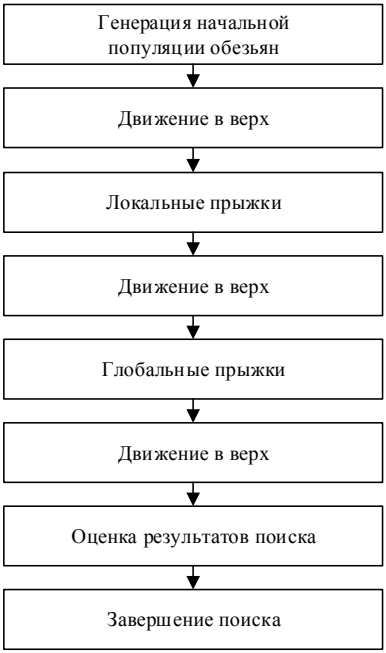


Рис. 4.3. Ключевые операции модифицированного метода обезьяньей оптимизации

С учетом выделенных ключевых операций и на основе рассмотренной модели поведения обезьяньей колонии авторы предлагают модифицированный обезьяний алгоритм, структурная схема которого приведена на рис. 4.4.

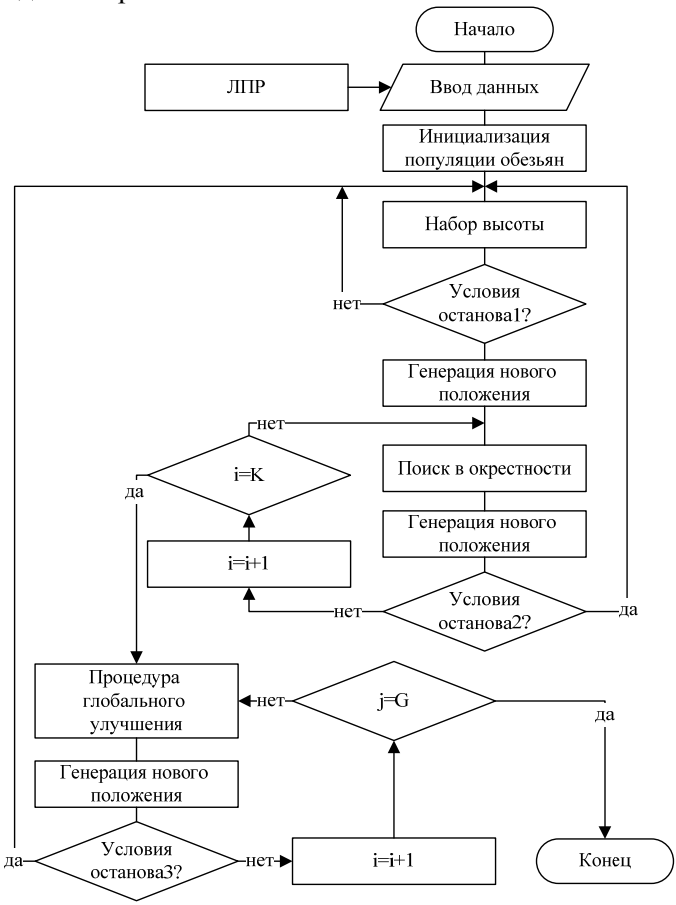


Рис. 4.4. Модифицированный алгоритм обезьяньей оптимизации

Опишем работу этого алгоритма более подробно. На предварительном этапе сначала задаются параметры решаемой задачи и обезьяньего алгоритма это количество обезьян, число итераций, количество и длины локальных и глобальных прыжков. Далее генерируется

начальная популяция обезьян p_i, $i \in [1:|P|]$ и задается их расположение на основе известных принципов [8]. Эта позиция и является альтернативным решением поставленной задачи. Далее, согласно алгоритму, реализуется первая процедура «движения в верх». Она может выполняться на основе локального или одномерного поиска. В предложенном алгоритме данная процедура выполняется на основе случайного одномерного поиска с учетом следующего выражения:

$$X' = (1- \alpha)\, X + \alpha R, \qquad (4.1)$$

где α – длина шага, R – направление движения, а t – число повторений.

Заметим, что новое положение обезьяны X_i запоминается, если $f(X_i') \leq f(X_i)$, если нет, то изменяем направление движения или длину шага. При этом процедура продолжает реализовываться до нахождения самой высокой точки. Это установленное положение становится текущим и запоминается.

Далее выполняется вторая процедура «локальные прыжки», т. е. поиск реализуется в окрестностях текущего положения. При этом новое положение обезьяны p_i вычисляется на основе следующей формулы:

$$X'_{i,j} = U_1((x_{i,j} - b); (x_{i,j} + b)),\ i \in [1:|P|],\ j \in [1:|X|]. \qquad (4.2)$$

Данное положение является случайной величиной, равномерно распределённой в интервале $[(x_{i,j} - b); (x_{i,j} + b)]$, где b параметр, с помощью которого задается длина локального прыжка. Данная процедура выполняется несколько раз, пока не будет выполнено условие $f(X_i') \leq f(X_i)$. Если полученное новое решение X_i' входит в область допустимых значений, то присваиваем $X_i' = X_i$ объявляем его текущим и запоминаем. В противном случае возврат к лучшему решению.

После этого, в отличие от классического метода [126], реализуется процедура «движения в верх» на основе выражения 4.1. Если в процессе реализации этой процедуры улучшения не происходит, то алгоритм попал в зону локального оптимума. Тогда, согласно алгоритму, для решения этой проблемы реализуется четвертая процедура глобального улучшения за счет выполнения «глобальных прыжков». При ее реализации используется текущее положение обезьяны p_i. и задается длина шага глобального прыжка v на основе выражения

$$v = U_1(v_{min}; v_{max}), \qquad (4.3)$$

где v_{min}; v_{max}– нижняя и верхняя граница этой величины. Заметим, что эта величина может принимать как положительное, так и

отрицательное значение. При положительном значении обезьяна движется в сторону центра тяжести x^c, а при отрицательном – в противоположную сторону. С учетом этого новое положение обезьяны p_i вычисляется на основе следующей формулы:

$$X'_{i,j} = x_{i,j} + v(x^c_j - x_{i,j}),\ i \in [1:|P|],\ j \in [1:|X|], \qquad (4.4)$$

где $x_j^c = \frac{1}{|S|}\sum_{i=1}^{S} x_{i,j}$ – текущее положение центра тяжести всей популяции обезьян по j-му координатному направлению.

Данная процедура, также как процедура локальных прыжков, выполняется несколько раз, пока не будет выполнено условие $f(X_i') \leq f(X_i)$. Если полученное новое решение X_i' входит в область допустимых значений, то присваиваем $X_i' = X_i$ объявляем его текущим и запоминаем. В противном случае возврат к лучшему решению.

После этого, в отличие от классического метода, реализуется еще одна процедура «движения в верх» на основе выражения 4.1. Если в процессе выполнения этой процедуры улучшения не происходит, то возврат к лучшему решению и конец работы алгоритма.

В заключении подраздела отметим, что поиск в представленном обезьяньем алгоритме происходит итерационно, причем нахождение оптимального решения может произойти после реализации любой процедуры поиска. В связи с этим в алгоритме после выполнения каждой процедуры происходит запись лучшего решения. Согласно проведенным исследованиям, алгоритм обезьяньего поиска быстро сходится и имеет высокую вероятность нахождения глобального оптимума при решении оптимизационных задач большой размерности.

4.2. Модели и метод поиска кукушки

Еще одним алгоритмом биоинспирированной оптимизации является поиск кукушки (*Cuckoo Search, CS*). Он был разработан Янгом и Дебом в 2009 году [130, 131]. Одним из главных достоинств данного алгоритма является малое число задаваемых параметров. За счет этого поиск кукушки является более универсальным и эффективным в сравнении с другими биоинспирированными алгоритмами. Поиск кукушки используется для непрерывной нелинейной оптимизации. Образ жизни кукушки в живой природе лежит в основе разработки данного алгоритма оптимизации.

Как и другие эволюционные подходы, алгоритма поиска кукушки начинается с определения размера популяции. В основе логики поведения агентов в алгоритме находится борьба за выживание. Часть

агентов умирают. Выжившие кукушки переселяются в лучшие места и начинают размножаться и откладывать яйца. Наконец, выжившие агенты формируют общество кукушек с близкими значениями приспособленности [131]. Основные операции канонического алгоритма оптимизации поиском кукушек представлены на рисунке 4.5.

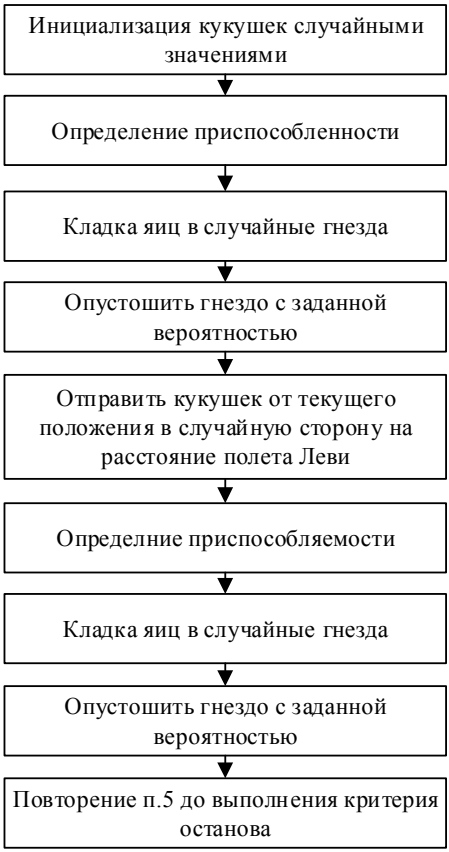


Рис. 4.5. Основные операции канонического алгоритма поиска кукушки

В алгоритме поиска кукушки возможные решения задачи оптимизации имитируются яйцами в гнезде, т. е. каждое яйцо кукушки представляет собой новое решение. Конечной целью алгоритма является

использование этих новых (и потенциально лучших) решений, связанных с паразитическими яйцами кукушки, для замены текущего решения, связанного с яйцами в гнезде [131]. Эта замена, осуществляемая итеративно, в конечном итоге приводит к решению задачи.

Остановимся более подробно на процессе подкладывания яиц кукушкой. Из всех гнезд случайным образом выбирается то гнездо, в которое предположительно будет отложено яйцо. Так как яйцо представляет собой решение, то оно может быть представлено качеством яйца, если яйцо кукушки более высокого качества, чем родительское, то оно будет заменено. В противном случае в гнезде останется родительское яйцо. По сути, последующая эволюция будет продолжатся от выжившего птенца. Это означает, что если выжил птенец родительского яйца, то эволюция продолжится с этого же места [131]. Дальнейшее развитие возможно только, если яйцо кукушки окажется более жизнеспособно и поиск решения задачи будет продолжен с нового места. Операция подкладывания яиц в чужие гнезда представляется деревом решений.

Вторая основная операция алгоритма после построения дерева решений – полет Леви, который является случайным блужданием (марковский статистический процесс), в котором длина скачка изменяется ступенчато, направление скачков изменяется случайным образом, а распределение вероятностей является частным случаем распределения Парето [131]. Определяется полет Леви как скачок в пространстве, причем скачок осуществляется изотропно в случайных направлениях. Полёт Леви – инструмент описания аномальных стохастических процессов.

Понятие полетов Леви используют в теории хаоса, при моделировании случайных или псевдослучайных природных явлений (например, полёта альбатроса, сочетающего длинные и короткие траектории). Примеры включают в себя анализ данных о землетрясениях, финансовая математика, криптография, анализ сигналов, турбулентное движение, а также множество применений в астрономии, биологии и физике.

Опишем одну из модификаций рассмотренного алгоритма поиска кукушки. На рисунке 4.6 представлена укрупненная структурная схема модифицированного алгоритма поиска кукушки (МАПК) для решения задачи оценки эквивалентной семантической близости [132].

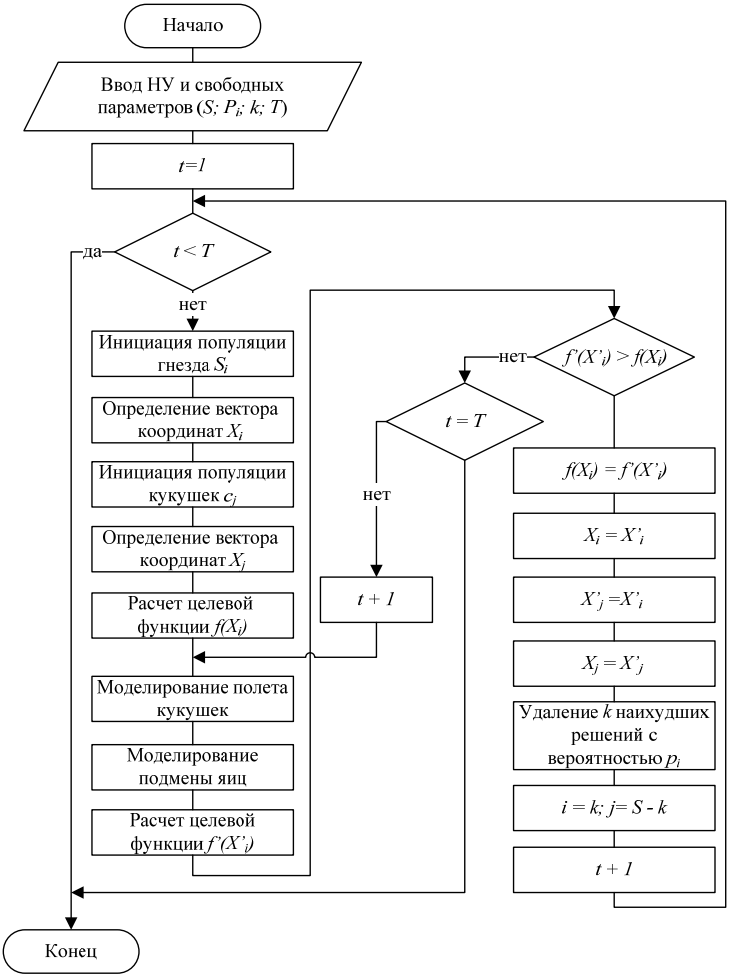


Рис. 4.6. Структурная схема модифицированного метода поиска кукушки

Здесь в каждом гнезде кроме настоящего находится чужое подложенное, которое имеет определенную степень сходства с оригинальным. Тогда оригинальному яйцу будет соответствовать элементу «верхнего» подмножества $R_a^1, a \in [1:N]$, при $(\alpha = 1)$, а чужое – «нижнему» подмножеству $R_b^2, b \in [1:M]$ при $(\alpha = 0)$. При этом ЦФ будет являться степень сходства данных подмножеств. Ее

необходимо максимизировать. Для этого необходимо подложить в гнездо такое «яйцо» $R_y^2, y \in [1:M], y \neq b$, где y – случайно выбранная вершина онтологии O_2, у которого степень сходства больше с «яйцом» R_a^1 будет больше, чем у R_b^2. Это позволяет повысить вероятность выживаемости за счет сходства [133]. Опишем работу метода более подробно.

1. Для реализации поиска используем шаги 1–4 из модифицированного метода обезьян.

2. Далее установим число итераций $t = 1, t \in [1:T]$, где T – максимум.

3. Сгенерируем из пар подмножества атрибутов рассматриваемых онтологий популяцию гнезд s_i, $i \in [1:S]$ из S хозяйских гнезд. Затем определим векторы координат гнезд X_i. При этом вероятность обнаружения в них чужих яиц будет задаваться случайно из множества $p_i \in (0:1)$.

4. Далее в онтологии O_2, сгенерируем новую популяцию кукушек c_j, $j \in [1:S]$ с координатами X_j.

5. Затем для каждого нового гнезда вычисляем ЦФ $f(X_i) = f(R_a^1 \cap R_b^2)$, а при достижении максимального числа итераций конец работы.

6. Моделируем полет случайно выбранной кукушки c_j в случайно выбранное гнездо s_i, меняя при этом в имеющемся решении подмножество R_b^2 на подмножество R_y^2, причем, $\forall y, b\ y \neq b$.

7. После моделирования получаем новое значение ЦФ $f'(X_i') = f(R_a^1 \cap R_y^2)$.

8. При чем если $f'(X_i') > f(X_i)$, тогда $f(X_i) = f'(X_i')$, а $X_i = X_i', X_j' = X_i', X_j = X_j'$, т.е. если условие выполняется, то происходит подмена яйца, а если нет то рассматривается другое гнездо.

9. Далее из популяции удаляется заданное число k, $k \in [1:S/2]$ наихудших решений-гнезд с вероятностью p_i, а на их место генерируются новые пары решений-гнезд $i = k$ и недостающее число новых кукушек $j = S - k$. Процесс продолжается итерационно.

Заметим, что начальные координаты гнезд и кукушек являются случайными, равномерно распределенными в пространстве вершин рассматриваемых онтологий. А траектория движения кукушки рассчитывается на основе полетов Леви [26, 134]:

$$X_j' = X_j + V \otimes L_{|X|}(\lambda), \qquad (4.5)$$

где $V = (v_z, z \in [1:|X|])$ – вектор размера шагов по соответствующим компонентам вектора X; $L_{|X|}(\lambda) - (|X| \times 1)$ – случайный вектор независимых вещественных случайных чисел, распределенных по закону Леви [26,134] (рис. 4.7).

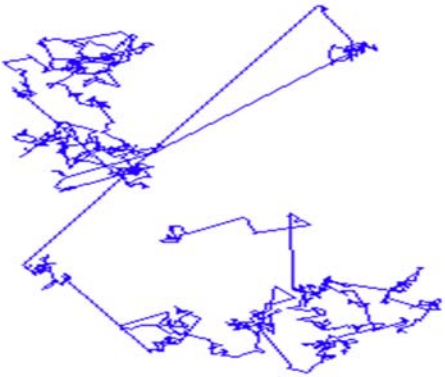


Рис. 4.7. Визуальная модель реализации полетов Леви

Здесь выражение (4.5) позволяет поддерживать траекторию агента, как совокупность малого числа больших (глобальных) перелетов и значительного числа – малых (локальных). В случае, если агент сразу не попадает в координаты свободного гнезда, тогда он перемещается в ближайшее из возможных (рис. 4.8).

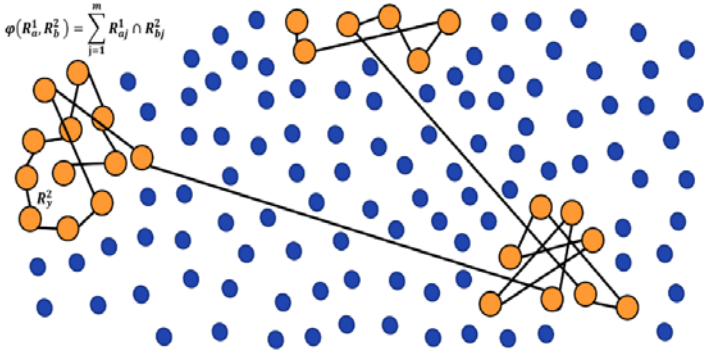


Рис. 4.8. Иллюстрация работы механизма поиска модифицированного кукушкиного метода

Отметим, что эффективность данного алгоритма обеспечивается за счет улучшенного механизма отбора решений и аттрактивности применяемых процедур.

4.3. Модели и метод летучих мышей

Алгоритм летучих мышей (*BA*) был предложен Янгом на основе эхолокационного поведения летучих мышей [135–136]. Эхолокация состоит из двух этапов: испускание громких частотно-модулированных звуковых импульсов и прием (прослушивание) эхо-звуков, которые отражаются от окружающих объектов. Летучие мыши используют свою гидролокационную систему для обнаружения добычи, уклонения от препятствий, определения местоположения ночлега.

В *BA* используются следующие параметры, значения которых корректируются в процессе поиска: частота, громкость и скорость импульсного излучения. Частота моделируется набором *d*-мерных векторов, каждый из которых связан с *i*-й мышью и которые случайным образом корректируются на *k*-й итерации по формуле:

$$\mathcal{F}_i^k = \mathcal{F}_{min} + \beta(\mathcal{F}_{max} - \mathcal{F}_{min}), \qquad (4.6)$$

где параметр $\mathcal{F}_{min}$ и $\mathcal{F}_{max}$ обозначают минимальную и максимальную частоты соответственно; β – это вектор случайных чисел, каждое из которых находится в интервале [0, 1]. Параметры громкости A_i и скорости импульсного излучения r_i, начальные значения которых A_i^0 и r_i^0 определяются при инициализации алгоритма. По мере развития процесса поиска оптимального решения значения этих параметров изменяются в соответствии со следующим выражением:

$$A_i^{k+1} = \alpha A_i^k,\ r_i^{k+1} = r_i^0(1 - \exp(-\gamma k)), \qquad (4.7)$$

где $\alpha < 1$ и $\gamma < 1$ – некоторые константы. При обновлении местоположения *i*-й летучей мыши на *k*-й итерации применяется следующий оператор:

$$\boldsymbol{x}_i^k = \boldsymbol{x}_i^{k-1} + \boldsymbol{v}_i^k, \qquad (4.8)$$

где $\boldsymbol{x}_i^{k-1}$ представляет положение *i*-й летучей мыши на предыдущей итерации (*k*–1), в то время как $\boldsymbol{v}_i^k$ обозначает скорость *i*-й летучей мыши, которая, в свою очередь, вычисляется следующим образом:

$$\boldsymbol{v}_i^k = \boldsymbol{v}_i^{k-1} + \mathcal{F}_i^k \cdot \left(x_i^{k-1} - \boldsymbol{x}_{best}\right), \qquad (4.9)$$

где $\boldsymbol{x}_{best}$ обозначает текущее глобально лучшее решение, найденное в процессе поиска, а $\mathcal{F}_i^k$ представляет частотный вектор, определяемый согласно (4.6).

Модификация процедуры *BA* включает локальную схему поиска, согласно которой на каждой итерации случайно выбранный индивид среди текущих лучших решений дополнительно уточняется путем выполнения случайного блуждания следующим образом [137, 138]:

$$\boldsymbol{x}_*^k = \begin{cases} \boldsymbol{x}_i^k + \varepsilon A_i^k, & \text{если } \left(rand > r_i^k\right), \\ \boldsymbol{x}_i^k, & \text{в противном случае,} \end{cases} \qquad (4.10)$$

где *rand* – случайное число, полученное из равномерно распределенного интервала [0, 1]. Причем новое решение $\boldsymbol{x}_*^k$ принимается

в качестве местоположения *i*-й летучей мыши только при соблюдении определенных условий, в частности:

$$\boldsymbol{x}_i^k = \begin{cases} \boldsymbol{x}_*^k, & \text{если } \left(rand < A_i^k \;\&\boldsymbol{x}_i^k < \boldsymbol{x}_*^k\right), \\ \boldsymbol{x}_i^k, & \text{в противном случае.} \end{cases} \quad (4.11)$$

В каждом поколении для модифицированной процедуры *BA* основным действием является обновление местоположения *i*-й летучей мыши на *k*-й итерации согласно формуле (4.8). В модифицированной процедуре *BA* применяется оператор скорости импульсного излучения r_i на основе *V*-образной передаточной функции согласно (4.7). *V*-образная передаточная функция в *BA* используется для установления положения летучей мыши согласно (4.3) следующим образом:

$$f = \arctan\left(\boldsymbol{x}_i^k\right). \quad (4.12)$$

В [139] был предложен гиперэвристический алгоритм *GEBA*, который является комбинацией метаэвристик летучих мышей и дифференциальной эволюции. Он включает вероятностный механизм их выбора в зависимости от текущего состояния решения и был разработан для решения задачи выбора значимых признаков классификации и оценки точности классификации. Выбор параметров настройки эвристик основан на экспериментальных данных. Если *n* – число признаков, характеризующих данные, то наилучшим будем считать решение, имеющее минимальное число признаков и высокую точность классификации.

Алгоритм начинается с генерации популяции случайных решений (подмножества классификационных признаков). Для оценки решений используется фитнесс-функция. Через $\boldsymbol{x}_{best}$ обозначается лучшее решение в популяции. Основной цикл алгоритма повторяется многократно. В алгоритме *GEBA* эмпирически установлена вероятность выбора решения по метаэвристике летучих мышей, равной 0,6, по метаэвристике дифференциальной эволюции – 0,4.

В качестве данных использовался мировой репозиторий данных и модельных задач машинного обучения *UCI*. Репозиторий содержит реальные данные по прикладным задачам в области биологии, медицины, физики, техники, социологии и др. Наборы данных (*data set*) именно этого репозитория чаще всего используются исследовательским сообществом для эмпирического анализа алгоритмов машинного обучения. Для проверки производительности предложенного алгоритма использовались 10 эталонных наборов данных *UCI*. При выборе наборов данных из репозитория учитывалось разнообразие областей, а также различие в числе признаков и экземпляров. В табл. 4.1 приводится краткое описание наборов данных.

Таблица 4.1

Описание используемых наборов данных *UCI*

№	Имя набора данных	Признаков	Экземпляров	Классов	Область данных
S1	Банковский кредит	24	1000	2	Бизнес
S2	Гидролокатор	60	20	2	Физика
S3	Диабет	6	144	3	Медицина
S4	Диагностика онкологии	32	596	2	Медицина
S5	Зоопарк	18	101	2	Фауна
S6	Ионосфера	34	351	2	Физика
S7	Лимфография	18	148	4	Медицина
S8	Реабилитация	309	126	2	Жизнь
S9	Сердце	13	270	2	Медицина
S10	Шахматы	36	3196	2	Игра

Каждое решение оценивалось с помощью фитнесс-функции на каждой итерации алгоритма. Данные случайным образом делились на две части: обучающие и тестовые наборы. Первая часть предназначена для обучения классификатора, а вторая – для тестирования. В качестве критерия оценки классификации используется функция потерь (ошибки классификации) для линейных моделей. Точность классификации определялась отношением правильно классифицированных экземпляров к их общему числу.

Использовались два популярных классификатора: *KNN* (*k* ближайших соседей) и *CART* (классификация и регрессия построением дерева решений). Согласно *KNN* для классификации каждого из объектов тестовой выборки необходимо последовательно выполнить следующие операции: вычислить расстояние до каждого из объектов обучающей выборки; отобрать *k* объектов обучающей выборки, расстояние до которых минимально; определить наиболее часто встречающийся класс среди *k* ближайших соседей в зависимости от расстояния между новым и обучающим экземплярами. Другим классификатором является *CART*, который строит бинарное дерево решений, где каждый узел дерева при разбиении имеет только двух потомков. На каждом шаге построения дерева правило, формируемое в узле, делит заданное множество наборов данных на часть, в которой выполняется

правило, и часть, в которой правило не выполняется. Оценочная функция, используемая алгоритмом *CART*, базируется на идее уменьшения неопределённости в узле. В алгоритме *CART* идея неопределенности формализована в индексе *Gini*.

Цель любого классификатора – получить наибольшую точность, то есть лучшим решением является то, которое минимизирует ошибку классификации, минимизирует количество выбранных признаков классификации и максимизирует точность классификации. Процесс классификации по алгоритму *GEBA* останавливался при достижении максимального числа итераций. В целях сравнения с конкурирующими алгоритмами максимальное число итераций k_{max} во всех экспериментах устанавливалось равным 50, значение параметра размера популяции – 20. Эксперименты проводились с использованием *Matlab* на процессоре *Core i5* с ОЗУ 8 Гбайт.

Для сравнения алгоритма *GEBA* с современными конкурирующими алгоритмами использовались следующие критерии: ошибка классификации (сравниваются максимальные, средние и минимальные значения фитнесс-функции); средний размер выборки; *T*-критерий суммы рангов Уилкоксона для проверки различий между выборками по уровню количественного признака; критерий Фридмана – непараметрический статистический тест с 5%-м уровнем значимости.

В ходе экспериментов производительность *GEBA* для решения задачи выбора значимых признаков классификации сравнивалась с алгоритмами *BA*, кукушки (*Cuckoo Search Algorithm, CSA)*, *GWO* и гибридным алгоритмом роя частиц и гравитационного поиска (*Gravitational Search Algorithm, GSA*) *PSO-GSA*.

В качестве настроек конкурирующих алгоритмов использовались следующие параметры, указанные в оригинальных работах.

Для *BA* скорости импульсного излучения *r* равна 0,1; громкость испускаемого летучей мышью звука *A* равна 0,25; параметр $\mathcal{F}_{min}$ = 0, $\mathcal{F}_{max}$ = 2.

Для *CSA* параметр *fl* = 2.

Для *PSO-GSA* начальное значение гравитационной постоянной g_0 = 1, а факторы $c_1' = -2\frac{t^3}{T^3} + 2, c_1' = -2\frac{t^3}{T^3}$, где *T*-максимальное число итераций, *t*-текущая итерация, коэффициент α равен 20.

Для *GEBA* параметр $\mathcal{F}_{min}$ = 0,1; $\mathcal{F}_{max}$ = 2. Алгоритм *GWO* не предполагает настройки параметров.

В табл. 4.2 представлены лучшие, средние и худшие значения фитнесс-функции, полученные конкурирующими алгоритмами *BA*, *CSA*, *GWO*, *PSO-GSA*, и алгоритмом *GEBA* за 50 итераций.

Таблица 4.2

Лучшие, средние и худшие значения фитнесс-функции,
полученные алгоритмами *BA, CSA, GWO, PSO-GSA* и *GEBA*

№	Классификатор *k* ближайших соседей					Классификатор *CART*				
	BA	*CSA*	*GWO*	*PSO-GSA*	*GEBA*	*BA*	*CSA*	*GWO*	*PSO-GSA*	*GEBA*
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
S1	**0,23**	0,28	0,26	0,26	0,27	0,24	0,25	0,24	0,24	**0,23**
	0,28	0,30	0,28	**0,23**	0,28	0,25	0,26	0,25	0,24	**0,23**
	0,29	**0,33**	0,30	0,29	0,29	0,27	**0,27**	0,26	0,26	0,25
S2	0,05	0,08	0,04	**0,03**	0,05	0,13	0,14	0,14	0,13	**0,12**
	0,07	0,09	0,08	**0,06**	0,07	0,16	0,16	0,17	**0,15**	0,16
	0,11	**0,12**	0,09	0,09	0,09	0,18	0,18	0,19	0,17	**0,20**
S3	**0,02**	0,03	**0,02**	0,03	**0,02**	**0,02**	**0,02**	**0,02**	0,03	**0,02**
	0,03	0,04	0,04	0,04	**0,02**	0,03	0,03	0,023	0,03	**0,02**
	0,05	0,06	0,06	**0,07**	0,05	0,06	0,03	0,034	**0,07**	0,03
S4	**0,00**	**0,00**	**0,00**	0,01	**0,00**	**0,00**	0,01	**0,00**	0,01	0,01
	0,00	**0,00**	**0,00**	0,01	**0,00**	**0,01**	0,02	**0,01**	0,02	**0,01**
	0,00	0,00	0,00	**0,01**	0,00	0,01	0,01	**0,02**	**0,02**	0,01
S5	**0,00**	0,01	**0,00**	**0,00**	**0,00**	**0,02**	0,04	0,03	**0,02**	**0,02**

Окончание таблицы 4.2

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
	0,01	0,02	0,01	**0,00**	**0,00**	0,03	0,05	0,04	0,03	**0,02**
	0,01	**0,04**	0,02	**0,04**	0,00	0,05	**0,07**	0,06	0,05	0,04
	0,05	0,07	0,06	**0,04**	0,05	**0,04**	0,06	**0,04**	**0,04**	0,05
S6	0,06	0,08	0,07	**0,05**	0,06	0,06	0,06	0,06	**0,05**	0,06
	0,08	**0,09**	0,08	0,07	0,07	0,07	0,07	**0,08**	0,07	0,07
	0,12	0,12	0,12	**0,11**	0,12	0,18	0,13	**0,09**	0,11	0,12
S7	0,13	0,16	0,13	**0,12**	0,13	0,14	0,15	0,14	0,14	**0,13**
	0,15	**0,19**	0,16	0,15	0,14	**0,18**	0,17	0,16	0,16	0,16
	0,29	0,32	0,33	0,23	**0,22**	**0,11**	0,12	0,13	**0,11**	0,12
S8	0,34	0,34	0,35	0,32	**0,31**	0,14	0,14	0,14	**0,13**	0,14
	0,35	**0,36**	**0,36**	**0,36**	**0,36**	**0,16**	0,15	**0,16**	**0,16**	0,15
	0,15	**0,15**	**0,15**	**0,15**	**0,15**	0,14	0,15	0,14	**0,13**	**0,13**
S9	0,17	0,20	0,18	0,17	**0,16**	0,16	0,17	0,16	0,16	**0,15**
	0,19	**0,23**	0,22	0,22	0,19	0,17	**0,19**	0,17	0,18	0,17
	0,02	0,05	**0,02**	0,03	**0,02**	**0,007**	0,018	**0,007**	**0,007**	**0,007**
S10	0,026	0,069	0,030	0,027	**0,024**	**0,008**	0,038	**0,008**	0,009	**0,008**
	0,041	**0,096**	0,043	0,032	0,027	0,014	**0,059**	0,009	0,014	0,009

Анализ результатов показывает следующее.

Для классификатора *KNN* алгоритм *GEBA* получает наилучшее среднее фитнесс-значение для 6 из 10 тестируемых наборов данных (*S3, S4, S5, S8 S9* и *S10*), в то время как *PSO-GSA* лидирует для 5 из 10 наборов данных (*S7, S1, S4, S6* и *S2*). Алгоритмы *BA*, *CSA* и *GWO* получили наилучшее среднее фитнесс-значение для 1 из 10 тестируемых наборов данных (*S10*). Для классификатора *KNN* алгоритм *GEBA* получает максимальные фитнесс-значения для 6 из 10 тестируемых наборов данных (*S3, S4, S5, S8, S9* и *S10*), как и алгоритмы *BA* и *PSO-GSA*. Алгоритм *CSA* на большинстве наборов данных дает наихудшие результаты.

Для классификатора *CART* алгоритм *GEBA* получает наилучшее среднее фитнесс-значение для 7 из 10 тестируемых наборов данных (*S1, S3, S4, S5, S7, S9* и *S10*), а лучший из конкурирующих алгоритмов *PSO-GSA* – на 3 из 10. Алгоритм *GEBA* получает максимальные фитнесс-значения для 6 из 10 тестируемых наборов данных, а наихудшие результаты на большинстве наборов данных дает *CSA*.

T-критерий суммы рангов Уилкоксона и критерий Фридмана с доверительным уровнем 0,95 показывают, что алгоритм *GEBA* является статистически значимым по сравнению с *BA*, *CSA* и *GWO*. Алгоритм *GEBA* занимает лидирующие позиции, полученными за 50 независимых запусков для каждого из 10 наборов данных, в сравнении с *PSO-GSA,* как для классификатора *KNN*, так и для *CART*.

Табл. 4.3 иллюстрирует устойчивость сравниваемых алгоритмов.

Таблица 4.3

Стандартное отклонение для алгоритмов *BA, CSA, GWO, PSO-GSA, GEBA*

№	Классификатор *k* ближайших соседей					Классификатор *CART*				
	BA	*CSA*	*GWO*	*PSO-GSA*	*GEBA*	*BA*	*CSA*	*GWO*	*PSO-GSA*	*GEBA*
S1	0,01	0,01	0,01	0,01	**0,006**	0,01	**0,007**	0,008	0,01	0,008
S2	0,014	**0,009**	0,014	0,014	0,011	0,016	**0,012**	0,014	0,013	0,019
S3	0,008	0,011	0,011	0,011	**0,005**	0,009	0,005	**0,003**	0,009	**0,003**
S4	**0,000**	**0,000**	**0,000**	0,001	**0,000**	0,002	0,002	**0,001**	0,002	**0,001**
S5	0,005	0,008	0,006	0,009	**0,000**	0,011	0,009	**0,008**	0,010	**0,008**
S6	0,011	0,007	**0,005**	0,007	0,006	0,005	**0,004**	0,006	0,007	0,005
S7	0,013	0,017	0,016	0,011	**0,007**	0,016	**0,012**	0,014	0,015	**0,012**
S8	0,016	0,012	**0,009**	0,046	0,050	0,013	0,009	**0,007**	0,012	0,009
S9	0,010	0,023	0,019	0,019	**0,008**	**0,008**	0,011	0,009	0,011	**0,008**
S10	0,005	0,011	0,005	0,003	**0,001**	0,001	0,011	0,001	0,002	**0,000**

Устойчивость *GEBA* выше, нежели у конкурирующих алгоритмов на 7 из 10 наборах данных (*S1, S3, S4, S5, S7, S9* и *S10*) при использовании классификатора *KNN*, и на 6 из 10 наборах данных (*S3, S4, S5, S7, S9* и *S10*) при использовании классификатора *CART*. Главное преимущество алгоритма *GEBA* заключается в том, что у него меньше корректируемых переменных, что влияет на производительность алгоритма оптимизации.

Полученные экспериментальные результаты позволяют утверждать, что *GEBA* позволяет повысить производительность и устойчивость его работы при решении задачи выбора значимых классификационных признаков, в сравнении с конкурирующими метаэвристиками. Результаты, приведенные в табл. 4.2–4.3, демонстрируют наилучшую среднюю точность алгоритма *GEBA* для классификаторов *KNN* и *CART* в большинстве используемых наборов данных, в том числе для самого большого набора данных *S*16.

Сравнительный анализ скорости сходимости алгоритмов показывает, что *GEBA* быстрее сходится к глобальным оптимумам, нежели конкурирующие алгоритмы *BA*, *CSA, GWO* и *PSO-GSA*. Основной причиной здесь является комбинирование модифицированных процедур *BA* и *DE* с использованием механизма их выбора в зависимости от текущего состояния решения, при соблюдении баланса между скоростью сходимости алгоритма и диверсификацией пространства поиска решений. Самую низкую скорость сходимости демонстрирует алгоритм *CSA*. Это согласуется с результатами, представленными в табл. 4.3.

4.4. Модели и метод бабочек-монархов

Алгоритм бабочки-монарха (*Monarch Butterfly Optimization, MBO*) – это метаэвристика, моделирующая миграционное поведение бабочек-монархов в природе [42, 67]. Направление движения особей бабочки-монарха в алгоритме *MBO* в основном определяется оператором миграции и специальным корректирующим оператором. *MBO* подходит для параллельной обработки операторов и способен обеспечить баланс между скоростью сходимости алгоритма и диверсификацией пространства поиска решений. Во всяком случае об этом свидетельствуют результаты сравнения производительности *MBO* с многими конкурирующими метаэвристиками на тестовых бенчмарках.

Для решения различных оптимизационных задач используются следующие идеализированные правила поведения бабочек-монархов:

(1) вся их популяция находится в двух областях их обитания;

(2) особи размножаются и мигрируют из одной области в другую;

(3) общая численность популяции поддерживается неизменной;

(4) особи с наилучшей функцией пригодности автоматически переходят в следующее поколение.

Упрощая и идеализируя процесс миграции, предположим, что бабочки-монархи находятся в одной области с апреля по август (5 месяцев), а затем мигрируют в другую область с сентября по март (7 месяцев). Обозначим общее количество бабочек-монархов через *NP*, количество бабочек-монархов в области 1 через *NP1* = *ceil*(*p***NP*), количество бабочек-монархов в области 2 через *NP2* = *NP*–*NP1*. Здесь *ceil*(*x*) означает округление *x* до ближайшего целого числа, большего или равного *x*; *p* – доля бабочек-монархов в области 1. Бабочки-монархи в области 1 образуют субпопуляцию 1, в области 2 – субпопуляцию 2. Тогда процесс миграции бабочек-монархов можно выразить следующим образом:

$$x_{i,k}^{t+1} = x_{r1,k}^{t}, \tag{4.13}$$

где $x_{i,k}^{t+1}$ указывает положение *k*-й особи бабочки-монарха x_i в поколении $t + 1$. Аналогично, $x_{r1,k}^{t}$ указывает на случайно выбранную *k*-ю особь бабочки-монарха x_{r1} из субпопуляции 1 в поколении *t*. Если $r \le p$, то *k*-я особь новой бабочки-монарха генерируется с помощью уравнения (4.8).

Здесь

$$r = rand * peri \tag{4.14}$$

указывает на период миграции, равный 1,2 (12 месяцев в году), где *rand* – случайное число, полученное из равномерного распределения. Напротив, если $r > p$, то *k*-я особь новой бабочки-монарха генерируется следующим образом:

$$x_{i,k}^{t+1} = x_{r2,k}^{t}, \tag{4.15}$$

Здесь $x_{r2,k}^{t}$ указывает на случайно выбранную *k*-ю особь бабочки-монарха x_{r2} из субпопуляции 2 в поколении *t*.

В алгоритме *MBO* направление миграции зависит от параметра *p*. Если значение *p* велико, то больше бабочек-монархов выбирается из субпопуляции 1, иначе больше бабочек-монархов выбирается из субпопуляции 2. Обычно значение *p* выбирается равным 5/12 в соответствии с периодом миграции.

Псевдокод оператора миграции имеет следующий вид:

begin
for *i*=1 *to NP1* **do**
for *k*=1 *to D* **do**
генерация *rand*;
if $r \le p$ **then**
случайный выбор бабочки-монарха из субпопуляции 1 (скажем, *r1*);
генерация *k*-й особи x_i^{t+1} согласно (4.13).
else
случайный выбор бабочки-монарха из субпопуляции 2 (скажем, *r2*);
генерация *k*-й особи x_i^{t+1} согласно (4.8).
end if
end for k
end for i
end

Кроме оператора миграции, положение бабочки-монарха также может быть изменено специальным корректирующим оператором. Положение *k*-й особи бабочки-монарха x_j в поколении $t + 1$ определяется следующим образом:

$$x_{j,k}^{t+1} = x_{best,k}^{t}, \tag{4.16}$$

если случайно сгенерированное число $rand \le p$. Здесь $x_{best,k}^{t}$ является положением лучшей особи в субпопуляциях 1 и 2 текущего поколения.

Напротив, если $rand > p$, то положение *k*-й особи бабочки-монарха x_j в поколении $t + 1$ обновляется как

$$x_{j,k}^{t+1} = x_{r3,k}^{t}, \tag{4.17}$$

где $x_{r3,k}^{t}$указывает на *k*-ю особь x_{r3}^{t}, случайно выбранную из субпопуляции 2. Здесь $r_3 \in \{1, 2, \dots, NP2\}$. При условии, что $rand > BAR$, положение *k*-й особи бабочки-монарха x_j в поколении $t + 1$ обновляется как

$$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha * (dx_k - 0{,}5), \tag{4.18}$$

где *BAR* указывает на скорость корректировки, а dx_k – шаг корректировки, который рассчитывается согласно полетам Леви:

$$dx = Levy(x_j^t), \tag{4.19}$$

α – весовой коэффициент, который рассчитывается как

$$\alpha = S_{max}/t^2, \tag{4.20}$$

где S_{max} – максимальный шаг корректировки, t – текущее поколение.

Большее значение α приводит к диверсификации пространства поиска решения, а меньшие значения α – к увеличению скорости сходимости.

Псевдокод специального оператора корректировки имеет следующий вид:

begin
for *j*=1 *to NP2* **do**
расчет *dx* согласно (4.12);
расчет весового коэффициента α согласно (4.20);
for *k*=1 *to D* **do**
генерация *rand*;
if $r \le p$ **then**
случайный выбор бабочки-монарха из субпопуляции 2 (скажем, *r3*);
генерация *k*-й особи x_j^{t+1} согласно (4.17).
if *rand* > *BAR* **then**
$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha * (dx_k - 0{,}5)$;
end if
end if
end for *k*
end for *j*
end

В целом алгоритм *MBO* включает следующие шаги.

begin

1. Инициализация. Установить счетчик поколений $t = 1$; случайным образом инициализировать популяцию из *NP* особей бабочки-монарха; установить максимальное число поколений равным *MaxGen*, число бабочек-монархов *NP1* в субпопуляции 1, и число бабочек-монархов в субпопуляции 2 *NP2*, максимальный шаг корректировки, равным *Max*, скорость корректировки *BAR*, период миграции *peri* и коэффициент миграции *p*.

2. Вычислить функцию пригодности каждой бабочки-монарха в соответствии с ее положением.

3. **while** (лучшее решение не найдено **or** $t <$ *MaxGen*) **do**

Ранжировать особей бабочки-монарха в соответствии с их функцией пригодности;

Разделить особей на субпопуляцию 1 и субпопуляцию 2);

for *i*= 1 to *NP*1 **do**

Создать новую субпопуляцию 1 с помощью оператора миграции.

end for *i*

for *j*= 1 to *NP*2 **do**

Создать новую субпопуляцию 2 с помощью специального оператора корректировки.

end for *j*

Объединить две вновь созданные субпопуляции в одну популяцию;

Вычислить функцию пригодности для особей с обновленными позициями;

$t = t+1$

4. **end while**

5. Вывод лучшего решения

В [67] представлены результаты сравнения производительности алгоритма *MBO* с пятью другими метаэвристиками (*ABC, ACO, BBO, DE, SGA*) с помощью 38 тестовых задач поиска оптимума для многомерных функций. Результаты показывают, что алгоритм *MBO* находит лучшие значения функций в большинстве задач в сравнении с конкурирующими алгоритмами.

Кроме того, алгоритм *MBO* прост и не содержит сложных вычислений и операторов. Это делает реализацию алгоритма *MBO* простой и быстрой. Несмотря на преимущества *MBO* обращают на себя внимание следующие проблемы: необходимость настройки параметров, используемых в алгоритме с помощью теоретического анализа или экспериментов; анализ алгоритма в таких приложениях, как сегментация изображений, планирование, проектирование и транспортная логистика; алгоритм уступает конкурирующим алгоритмам по показателю средней производительности и стандартному отклонению, а также для низкоразмерных функций. Необходимо также теоретически проанализировать сходимость *MBO* с помощью динамических систем и цепей Маркова.

4.5. Модели и метод червей

Алгоритм оптимизации червей (*Worm Optimization, WO*), представленный в [140], основан на поведении червя нематоды, имеющего всего 302 нейрона. Тем не менее, это позволяет червям выполнять несколько сложных действий, включая выживание, смену индивидуального и коллективного стилей поиска пищи, избежание

токсинов и другие. Алгоритм *WO* превзошел алгоритмы муравьиной колонии *ACO*, роя частиц *PSO* и генетического *GA* в *NP*-сложной задаче о коммивояжере.

Чтобы решить задачу оптимизации с помощью *WO*, ее необходимо представить в виде графа, содержащего вершины и дуги, в котором червь будет перемещаться от одной вершины к другой в процессе поиска оптимального решения.

WO начинается с отложения феромона. Первоначально определенное количество феромона τ_{ij} (обычно $\tau_{ij} = 0{,}01$) откладывается в каждой дуге графа. Черви перемещаются от узла к другому в соответствии с вероятностью, которая частично аналогична вероятности при оптимизации муравьиным алгоритмом [69]. Вероятность определяется тремя факторами: количеством феромона (τ), доступностью (η) и коэффициентом плохого решения (*ADF*). Вероятность перемещения из вершины *i* в вершину *j* для червя *k* вычисляется согласно:

$$P_{ij}^{k} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}ADF_{ij}}{\sum_{l\in\Psi}\tau_{il}^{\alpha}\eta_{il}^{\beta}ADF_{ij}}, \qquad (4.21)$$

где Ψ представляет собой множество еще не посещенных вершин, а величина η определяется в зависимости от того, является ли поведение червя коллективным или индивидуальным. Важными параметрами для направления поиска являются α (обычно $\alpha = 1{,}5$) и β (обычно $\beta \in [0{,}5;\ 4{,}5]$), которые являются степенными показателями в (4.15) и определяют важность количества феромонов в сравнении с величиной доступности η. Затем величина феромона обновляется в соответствии со следующим уравнением:

$$\tau_{ij} \leftarrow \tau_{ij} \cdot (1 + \rho), \qquad (4.22)$$

если дуга (i, j) используется червем *k*. Здесь ρ (обычно $\rho = 0{,}01$) параметр обновления феромона (величина, на которую увеличивается количество феромона при обнаружении хорошего решения).

Рассмотрим подробнее основные операторы алгоритма *WO*.

Оператор, моделирующий паттерны добывания пищи. Параметр *RMG* указывает на индивидуальное поведение червя (если *RMG* = 1) или на коллективное поведение (если *RMG* = 0). Если $RMG \in [0{,}4;\ 0{,}7]$, то это не исключает ни того ни другого стиля поведения. После инициализации *WO*, в соответствии с *RMG*, каждый червь помечается как склонный к коллективному или индивидуальному поведению. Если червь склонен к коллективному поведению, он будет привлечен феромоном; в противном случае он с равной вероятностью переместится в другие вершины.

Оператор, позволяющий избежать токсинов. После перемещения червя в другую вершину, необходимо оценить данное решение. С этой целью формируется список *ADF*, в котором хранятся до h плохих решений, где $h = \lceil\sqrt{Worm}\rceil$. Здесь *Worm* определяет количество червей в алгоритме на этапе инициализации.

Оператор, моделирующий передвижение червя. В процессе моделирования передвижения червя вводится вероятность *AIY* (обычно $AIY \in [0{,}1;\ 0{,}5]$) проведения локального поиска. Чем она выше, тем выше вероятность локального поиска. В этом случае две вершины из *Worm* выбираются случайным образом и меняются местами. Если это приводит к лучшему решению, то оно сохраняется; в противном случае локальный поиск продолжается до тех пор, пока либо не будет найдено лучшее решение, либо не будет достигнуто заданное максимальное количество итераций.

Что касается концентрации червей в пространстве поиска, то ее лучше оценивать по их количеству. Для каждого червя количество итераций обновляется до тех пор, пока оно не достигнет своего максимума (обычно от 5000 до 15000), что указывает на высокую концентрацию червей.

Известны и другие примеры применения алгоритма *WO*, в частности для задачи минимизации времени параллельной обработки изделий на нескольких машинах, оптимального размещения изделий на складе.

4.6. Модели и метод серых волков

Впервые алгоритм стаи серых волков был упомянут в своей работе в 2011 году с. Муро, Р. Эскобедо и др., а в 2014 году С. Мирамуничем и Дж. Р. Мирамуничем одновременно с С.Мирджалили и М. Мирджалили в своих работах был предложен мета-эвристический алгоритм оптимизации серых волков, который имитирует иерархическое поведение волков в живой природе при поиске и добычи пищи [49, 141–143]. Данный алгоритм изначально предназначался для решения векторных задач. В природе волки относятся к семейству собачьих и живут стаями в среднем от 7 до 10 особей, причем в стае действует четкая иерархия. Выделят вожака его помощников и остальную стаю, т. е. стая серых волков согласно иерархии делится на 4 типа (рис. 4.9).

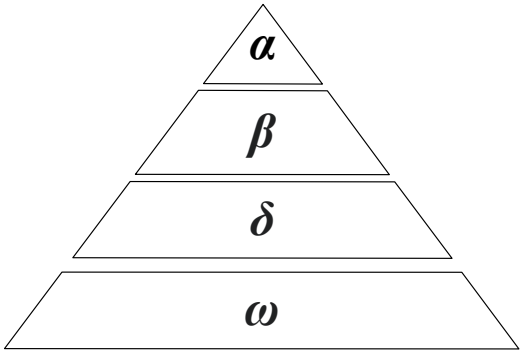


Рис. 4.9. Социальная иерархия серых волков

В данной иерархии Альфа – вожак стаи, причем это может быть как самец, так и самка, который избирается на общем собрании стаи. Заметим, что это может быть не самая сильная особь, но она должна удовлетворять таким требованиям как хитрость (умение избегать ловушек), выносливость (загон жертвы), умением управлять стаей (охота, ночлег и т. п.) и нести полную ответственность за ее жизнеобеспечение.

Следующим в иерархии являются Бета – прямые помощники вожака (кандидаты), которые участвуют в утверждении решений или другой деятельности стаи, а также управляют волками низшего уровня и дисциплинируют стаю.

Ниже в иерархии расположены Дельта. Они подчиняются как вожаку Альфе, так и помощникам Бета. Данный тип волков подразделяются на следующие типы.

Разведчики, которые следят за границами и предупреждают стаю в случаи опасности.

– Дозорные, которые несут ответственность за безопасность стаи.

– Старейшины, обычно ими являются опытные волки, например Альфа и Бета в прошлом.

– Охотники, которые обязаны помогать Альфам и Бетам при охоте.

И наконец на нижней ступени иерархии располагаются волки Омега. Они подчиняются всем остальным. Эта иерархия используется в алгоритме для обновления позиции волков и поиска глобального оптимума. Данный тип кажется не уместным в природе, но на самом деле они помогают сохранить иерархическую структуру и

собранность стаи. Выявлено, что при их отсутствии возникают проблемы внутри стаи. Основная идея классического алгоритма (рис. 4.10) основана на моделировании поведения стаи серых волков во время охоты [141–143].

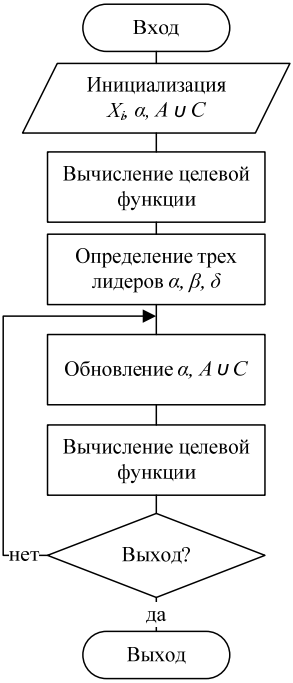


Рис. 4.10. Классическая схема алгоритма серых волков

В ходе охоты волки осуществляют следующие действия это поиск добычи, ее окружение и нападение. Опишем работу алгоритма более подробно. Сначала задается популяция серых волков, т. е. альтернативные решения задачи. Далее массив координат и ЦФ (значение добычи). Математическое моделирование процесса окружения добычи выполняется на основе следующих уравнений:

$$D = \left|\vec{C} \times \vec{X}_p(t) - \vec{X}(t)\right|. \quad (4.23)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \times \vec{D}, \quad (4.24)$$

где t – текущая генерация, $\vec{A}$ и $\vec{C}$ – векторы-коэффициенты, $\vec{X}_p$ и $\vec{X}$ – соответственно вектора положения жертвы и серого волка (рис. 4.11). Здесь вектора $\vec{A}$ и $\vec{C}$ рассчитываются согласно формулам.

$$\vec{A} = 2 \times \vec{a} \times \overrightarrow{r_1} - \vec{a}. \quad (4.25)$$

$$\vec{C} = 2 \times \overrightarrow{r_2}, \quad (4.26)$$

где $\vec{a}$ – компонент, который в течении генераций линейно уменьшается с 2 до 0, а r_1 и r_2 – случайные вектора, находящиеся в интервале от 0 до 1, которые вычисляются для волков на каждой генерации и позволяют волкам перейти в любую позицию между двумя конкретными точками.

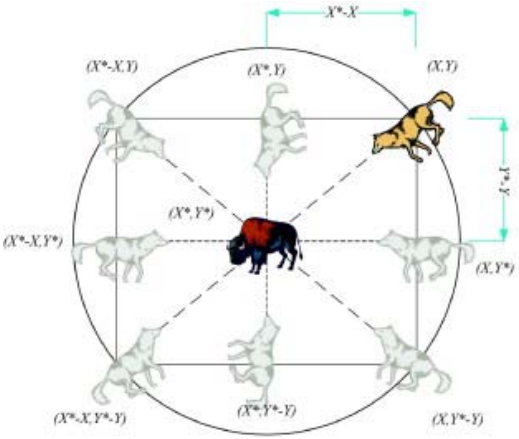


Рис. 4.11. Начальные позиции волков

На основе формулы 4.25. устанавливается баланс между поиском добычи и ее окружением, а согласно формуле 4.26. осуществляется случайный поиск. Данный компонент при C > 1 усиливает влияние добычи на расстояние, а при C < 1 наоборот уменьшает его.

Так как вначале решения поставленной задачи не известно положение оптимального решения, то генерация нового положения волков будет зависеть от трех лучших волков и рассчитывается последующим формулам:

$$\bar{D}_\alpha = |\bar{C}_1 \times \bar{X}_\alpha - \bar{X}|,\ \bar{D}_\beta = \left|\bar{C}_2 \times \bar{X}_\beta - \bar{X}\right|, \bar{D}_\delta = |\bar{C}_3 \times \bar{X}_\delta - \bar{X}|. \quad (4.5)$$

$$\bar{X}_1 = \bar{X}_\alpha - \bar{A}_1 \times (\bar{D}_\alpha), \bar{X}_2 = \bar{X}_\beta - \bar{A}_2 \times \left(\bar{D}_\beta\right), \bar{X}_3 = \bar{X}_\delta - \bar{A}_3 \times (\bar{D}_\delta). \quad (4.22)$$

$$\bar{X}(t+1) = \frac{\bar{X}_1 + \bar{X}_2 + \bar{X}_3}{3}, \quad (4.28)$$

где $\bar{X}_i$ – текущая позиция соответствующего волка, а $\bar{X}(t+1)$ – новая.

Заметим, что можно рассматривать N-мерное пространство, тогда волки двигаются в гиперкубе вокруг наилучшего решения (рис. 4.12).

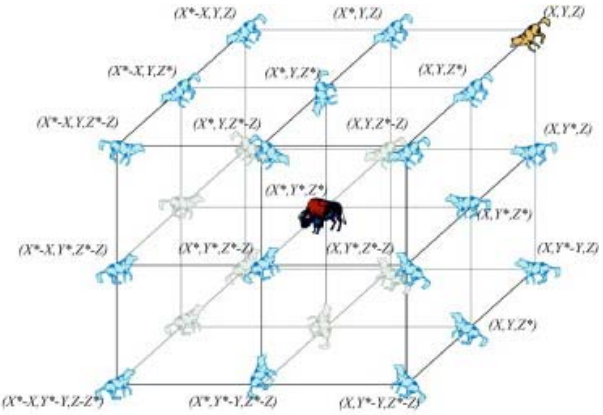


Рис. 4.12. Обновленные позиции волков

Серый волк в положение (X, Y) переходит в новую позицию согласно от расположения добычи (X*, Y*). При этом определение положения вокруг наилучшего волка производится за счет изменения значения векторов $\vec{A}$ и $\vec{C}$ по отношению к текущей позиции. Например, (X*-X,Y*) достигается с помощью обозначения $\vec{A} = (1,0)$ и $\vec{C} = (1,1)$.

В природе волки расходятся находят добычу и окружают её. Но в нашем случае эта информация отсутствует, поэтому для моделирования стадии охоты будем считать, что Альфа, Бета и Дельта имеют знания о потенциальном месторасположении добычи. В связи с этим, запоминаются три лучших решения, а остальные члены

стаи должны улучшать свои позиции. Далее моделируется процесс атаки добычи. Этот процесс регулируется изменением снижением значения $\vec{a}$, что приводит к снижению колебания $\vec{A}$. Вектор $\vec{A}$ лежит в интервале [-2a,2a]. Например, когда $\vec{A}$ находится в интервале [-1,1], то новая позиция волка будет между своей текущей позицией и позицией жертвы, при этом когда значение |A|>1, то волки осуществляют поиск, а когда |A|<1, то окружение жертвы (рис. 4.13).

Рис. 4.13. Поиск и окружение добычи

В отличие от классического алгоритма в качестве модификации предлагается ввести следующие процедуры. Это процедура случайного поиска добычи волками Омега, а также задание и динамическое изменение количества лидеров-волков в стае. Реализация данных процедур позволит учесть специфику решаемых задач и избегать зацикливания в локальных оптимумах [144–147]. Структурная схема предложенного алгоритма представлена на рисунке 4.14.

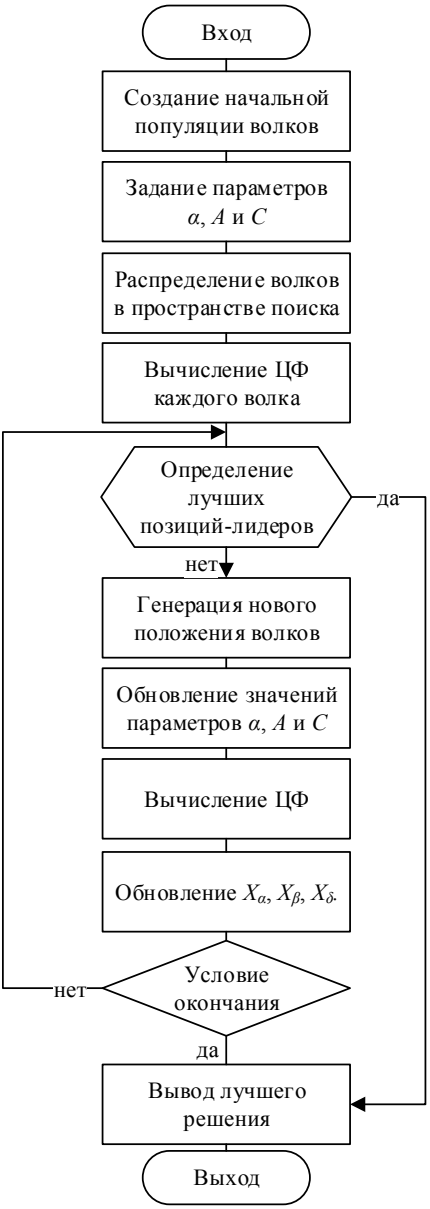


Рис. 4.14. Модифицированный алгоритм серых волков

В заключении подраздела отметим, что алгоритм стаи серых волков, как и другие биоинспирированные методы, легко распараллеливается за счет использования нескольких стай, что позволяет говорить об его эффективности.

4.7. Модели и метод колонии белых кротов

В 2012 году предложенный Taherdangkoo и др. в своей работе [148, 149] был предложен мета-эвристический алгоритм оптимизации, основанный на моделировании поведения белых кротов в живой природе. Данный алгоритм изначально предназначался для оптимизации числовых функций. За основу данного метода взято моделирование коллективного поведения колонии кротов при рытье лабиринтов и исследовании подземных ходов и их окрестностей в поисках пищи.

Белые кроты – подземное млекопитающее живущие колонией в жарком климате в зеленых зонах. Их подземные лабиринты могут достигать в длину до трех километров и отличаются фиксированной температурой в 32–36°C и влажностью. Белые кроты пойкилотермны и хладнокровны. В составе своей колонии имеют самку-вожака (королеву) управляющую колонией и способную производить потомство, рабочих кротов, выполняющих функции рытья лабиринтов и поиска пищи, а также кротов-охранников для защиты колонии от врагов. В связи с проживанием в тяжелых условиях окружающей среды колония белых кротов выработала и развила некоторое количество физиологических адаптационных признаков, помогающих им выживать и размножаться.

Рассматриваемый метод не в точности воспроизводит природную экосистему, а имеет небольшие отличия и упрощения [148, 149]. В предложенном методе, как и в природе движение белых кротов осуществляется случайным образом из центра к источникам пищи и их окрестностям. При их движении также производится рытье подземных ходов. В отличие от естественного поведения для упрощения механизма поиска этим процессом занимаются как рабочие кроты, так и кроты-охранники. После создания начальной популяции белых кротов начинается случайное движение охватывающее все пространство поиска. При этом число белых кротов должно превышать минимум в два раза число источников

пищи (альтернативное решение). В связи с этим к каждому источнику пищи отправляется пара белых кротов.

В процессе поиска после нахождения источника пищи кроты возвращаются в центр к королеве. Здесь, как и в муравьином методе, запоминается пройденный путь, а как в пчелином методе сохраняются все данные об источнике пищи.

После первичного поиска пищи агенты сохраняют пройдённый путь, качество источника пищи и возвращаются в колонию, где передают все сохраненные характеристики о источниках пищи королеве. Она производит сортировку источников в соответствии полученными характеристиками. Сортировка производится с вероятностью P по следующему выражению:

$$P_i = \frac{FS_i \cdot R_i}{\sum_{j=1}^{n} F_i}, \tag{4.29}$$

где FS_i – лучший источник, R_i –минимальное путь, N – число источников половина от количества кротов. Заметим, что отобранное решение с высокой долей вероятности будет перспективным. Далее каждый из пары кротов нашедшие лучший источник формируют из случайных пар две группы с собой во главе. При этом одна группа обрабатывает найденный источник, а другая проводит поиск в его окрестностях в разных направлениях в секторах 45° друг относительно друга, т. е. рассматривается восемь направлений поиска. Данный подход позволяет производить параллельную обработку как основного источника пищи, так и поиск новых источников в его окрестности, что позволяет повысить скорость поиска. Данный процесс продолжается итерационно пока не будут рассмотрены все перспективные источники с их окрестностями. На рисунке 4.15 представлен классический алгоритм оптимизации на основе поведения белых кротов [148, 149]. Работу данного алгоритма можно разделить на три ключевых этапа. На первом этапе создается начальная популяция кротов, которая распределяется в пространстве поиска. Далее поиск ведется парами кротов в случайном направлении. На втором этапе производится обновление приоритетных направлений поиска и перераспределение кротов в соответствии с ними. На третьем этапе выполняется обновление окрестностей и производится поиск в них.

Рис. 4.15. Основные этапы стандартного метода оптимизации белых кротов

В алгоритме производится отбор кротов с высокими значениями ЦФ. Этот процесс реализуется на основе моделирования механизма защиты колонии кротов от врагов (захватчиков). Врагами (захватчиками) на каждой итерации объявляются кроты с низким значением ЦФ. Они удаляются из популяции, при чем число удаляемых кротов зависит от коэффициента и рассчитывается по следующей формуле:

$$B_i^k = \varphi \cdot B_i^{k-1}. \quad (4.30)$$

где $\varphi \geq 1$ – коэффициент, учитывающий влияние внешней среды, задается ЛПР, B_i^k – число удаленных агентов, i – источник пищи, k – индекс итерации.

Как известно из зоологии в природе кроты способны излучать и принимать отраженные сигналы высокочастотных ультразвуковых волн для оценки положения, размера, расстояния и типа объекта. В процессе движения т. е. поиска источников пищи на пути кротов могут возникать препятствия (локальные оптимумы). Тогда движение кротов может происходить по следующим сценариям (рис. 4.16).

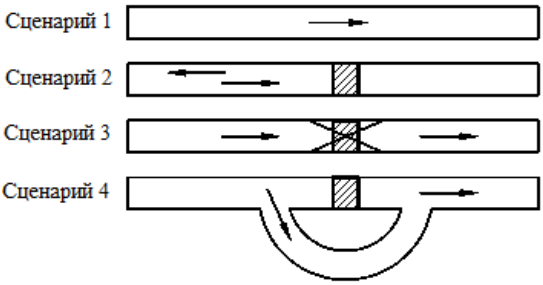


Рис. 4.16. Возможные сценарии движения кротов

Здесь сценарий 1 показывает прямолинейное движение кротов. При возникновении препятствия движение заблокировано – сценарий 2. Сценарий 3 – движение возможно за счет разрушения препятствия и сценарий 4 – при невозможности разрушения препятствия рытье обходного туннеля. Заметим, что при движении кроты посылают сигналы и принимают их отражение, что позволяет им выбирать тот или иной сценарий движения согласно (рис. 4.17).

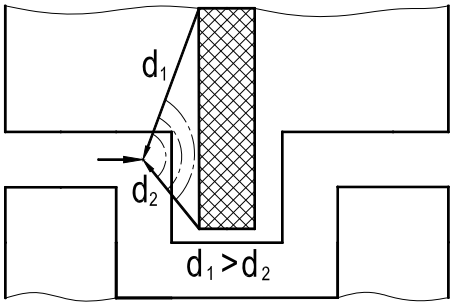


Рис 4.17. Обход препятствия

Здесь стрелочкой показано движение крота при встрече с препятствием. Он уловил отраженный сигнал и оценил, что расстояние $d_1 > d_2$, поэтому обход препятствия осуществляется с минимальной стороны.

С учетом этого авторы предлагают модификацию метода оптимизации на основе поведения колонии белых кротов, которая основана на выполнении 5 ключевых этапов в один, из которых вводится модифицированный механизма обхода препятствий [150–151]. Здесь крот принимает значение текущей позиции, как целевую точку и радиус препятствия. Если расстояние до цели меньше или равно радиусу препятствия, то изменяется направление движения для обхода препятствия и крот переводится в новую позицию.

Рассмотрим работу этого механизма более подробно. Кроты использует сенсоры, аналогичные биосонарам, для восприятия окружающей среды, а отраженные сигналы S_i, полученные от i-го сенсора, используются для определения расположения препятствий. Формализуем этот процесс.

Здесь каждый сенсор получает информацию о расстоянии d_i и угле θ_i до ближайшего препятствия. При этом сигнал от i -го сенсора представляется в виде координат $S_i = (d_i, \theta_i)$ для определения ориентации в пространстве. Тогда поворот крота определяется углом α который определяется через функцию ориентации от всех сенсоров

$$\alpha = f(S_1, S_2, \dots, S_n) \text{ или}$$
$$\alpha = f(d_1, \theta_1, d_2, \theta_2, \dots, d_n, \theta_n). \quad (4.31)$$

Учитывая динамику реальной среды, введем понятие адаптивной навигации, которая позволяет корректировать угол направления агента в реальном времени в зависимости от обнаруженных препятствий. Пусть $\theta_т$ текущий угол направления движения крота. Тогда, адаптивная навигация определяется на основе следующего выражения:

$$\theta_н = \theta_т + k \cdot \alpha, \quad (4.32)$$

где $\theta_н$ – новый угол направления движения, α – угол ориентации, полученный на основе выражения (4.32), k – коэффициент, учитывающий изменения внешней среды.

Кроме использования механизма динамического обхода препятствий, также вводится понятие специального лабиринта, предназначенного для защиты колонии от врагов. В случае прорыва его

блокирует один из кротов, чтобы было время у остальной колонии для блокировки других тоннелей.

С учетом этих модификаций авторы предлагают следующий модифицированный алгоритм оптимизации на основе поведения белых кротов (рис. 4.18) [151].

Опишем работу представленного алгоритма более подробно.

Сначала генерируется популяция белых кротов X:

$$X = [M_1, M_2, \cdots, M_N]. \quad (4.33)$$

где $M_1, M_2, \cdots, M_N$ – колония кротов; N – мощность популяции. Далее производится распределение кротов по всем возможным направлениям поиска при этом им устанавливается одинаковый приоритет. Затем задается начальный шаг поиска источников пищи в границах близлежащих окрестностей. Он описывается на основе следующего выражения:

$$x_i = x_i^{Min} + \beta\left(x_i^{Max} - x_i^{Min}\right);$$

$$i = 1,2,\cdots,S, \quad (4.34)$$

где x_i объём пищи в источнике i; β нормировочный коэффициент, лежащий в интервале [0, 1]; S – число источников пищи. При этом основными критериями при оценке источника пищи являются температура и влажность, определяемые с учетом коэффициента затухания qϵ[0,1]. Данные критерии позволяют оценить перспективность перемещения процесса поиска от источников пищи в их окрестности. Эти критерии определяются на основе следующих выражений:

$$H(x) = \rho(x)C(x)\frac{\Delta T(x,t)}{\Delta t};$$
$$(\rho\mathrm{C}) = f_s(\rho\mathrm{C})_s + f_a(\rho\mathrm{C})_s + f_w(\rho\mathrm{C})_w; \quad (4.35)$$
$$f_s + f_s + f_w = 1,$$

где H – значение температуры земли на глубине x; ΔT – среднее значение температуры; Δt – среднее значение времени цикла, $\rho(x)$ – плотность грунта, а $C(x)$ – удельная теплоемкость.

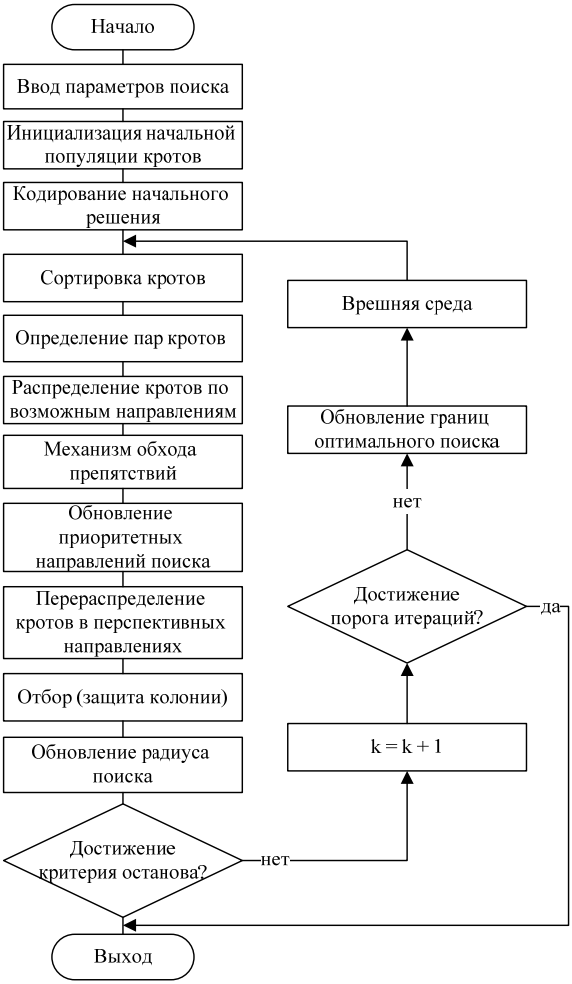


Рис. 4.18. Модифицированный алгоритм оптимизации на основе поведения колонии белых кротов

Заметим, что переменные ρ и C влияют друг на друга, и изменяют свои значения в соответствии с изменением расстояния при перемещении кротов. Для упрощения процедуры поиска значения данных переменных в алгоритме являются постоянными величинами, находящимися в интервале [2, 4].

Отношение $\Delta T(x,t)/\Delta t$ определяет скорость изменения температуры в течение единицы времени на каждой итерации алгоритма. Параметр $(f,\%)$ описывает общие характеристики для каждого крота, при этом индексы отображают среду *s* – песок, *a* – воздух и *w* – вода. Для поддержания баланса между параметрами ρ и C сумма значений индексов должна равняться единице *(s+ a+ w=1*).

Отметим, что на скорость поиска влияет коэффициент влажности A, который рассчитывается на основе выражения:

$$\mathrm{A}_i^t = q\left[1 - exp\left(\frac{-\alpha t}{\Delta T}\right)\right], \qquad (4.36)$$

где α – задается случайно в интервале [0, 1] или имеет зафиксированное эмпирически найденное значение. Рекомендованное значение α = 0.95 и t – время цикла. Данный коэффициент зависит от коэффициента затухания q, обновляемого на каждой итерации. Например, при q = 0 поиск в перспективных направлениях и их окрестностях проходит со снижением интенсивности, а при q = 0,5 начинает возрастать.

Далее поиск ведется в выбранном направлении с использованием механизма обхода препятствий, с учетом заданных границ. Затем определяются приоритетные направления поиска производится отбор кротов и обновление радиуса поиска для каждой пары. После этого этапа поиск продолжается до получения набора квазиоптимальных решений или достижения критерия останова. При переходе алгоритма на следующую итерацию обновляются границы приоритетного направления, и корректируются параметры поиска.

Отметим, что поиск в предложенном методе выполняется итерационно, и получение глобального оптимума возможно на любой из итераций. Поэтому необходимо запоминать лучшие решения во время всего поиска. Предложенный метод позволяет сократить время поиска за счет параллельной обработки несколькими группами кротов.

Для подтверждения эффективности разработанного алгоритма был проведен вычислительный эксперимент решения задачи эвакуации при чрезвычайных ситуациях [151]. Анализ результатов вычислительного эксперимента выявил улучшение качества (значение целевой функции) эвакуации при использовании, модифицированного MRA по сравнению с традиционным MRA (рис. 4.19).

Наблюдалось сокращение времени эвакуации, уменьшение количества коллизий с препятствиями и повышение успешных эвакуаций при ЧС в среднем на 5%.

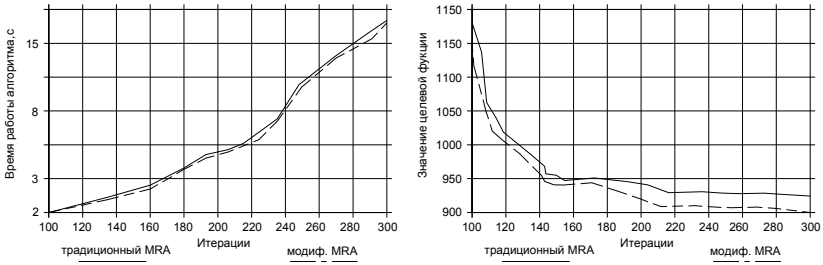


Рис. 4.19. Качественные показатели модифицированного MRA и традиционного MRA

4.8. Модели и методы искусственного косяка рыб

Алгоритм искусственного косяка рыб (*Artificial Fish Swarm Algorithm, AFSA*) впервые описан Ли Сяо Лэем в 2002 году [152]. Алгоритм искусственного косяка рыб (AFSA) – это стохастический популяционный алгоритм, который моделирует разумное коллективное поведение рыб в природе. В своей естественной среде обитания рыбы занимаются поиском мест с большим количеством пищи либо индивидуального поиска или в составе косяка (стаи). Соответственно имитируя поведение рыб в разных ситуациях, можно предложить несколько моделей поведения искусственных рыб: случайное (свободное) движение, поиск пищи, поведение в составе косяка рыб, преследование, скачкообразное движение. Искусственная рыба (*Artificial Fish, AF*), используя одну из перечисленных моделей поведения ведет поиск места (области пространства поиска) наиболее богатого пищей (соответствующей наилучшему значению целевой функции). При этом среда, в которой живет AF, является аналогом пространства поиска (области допустимых решений задачи оптимизации), а область, имеющая максимальную плотность пищи – аналог области, содержащей экстремум.

В стандартном алгоритме AFSA используются две важные величины: диапазон, в пределах которого они могут видеть других рыб (d_{vis})и величина шага при перемещении (d_{step}). Искусственные

рыбы исследуют проблемную среду в пределах своего диапазона видимости. Затем они начинают движение к цели, случайным образом выбирая величину шага на каждой итерации. В стандартном AFSA начальные значения этих параметров оказывают большое влияние на конечный результат, поскольку эти величины остаются постоянными до конца работы алгоритма.

Как показано на рисунке 4.20 [153], искусственная рыба воспринимает окружающую среду с помощью зрения. Обозначим текущее положение i-той искусственной рыбы в пространстве поиска как x_i, $i \in N$, где N – это число рыб в косяке. Текущее положение AF может быть отображено вектором $X = (x_1, x_2, \ldots, x_i, \ldots, x_n)$. Внешняя граница на рисунке соответствует области видимости AF, а x_v – это предполагаемая позиция в пределах диапазона видимости, куда искусственная рыба хочет переместиться. Если плотность корма в позиции x_v выше, чем в текущей позиции, AF перемещается на один шаг в направлении x_v, что приводит к перемещению AF из позиции x в следующую позицию x_{next}. Если текущее положение AF лучше, чем в позиции x_v, она выбирает другое направление движения в своем диапазоне видимости. Величина диапазона видимости может определяться выражением:

$$d_{vis} = \delta \max_i (u_i - l_i),$$

где $(u_i - l_i)$ – максимальная дистанция визуального контакта, а δ – поправочный коэффициент, который обычно уменьшается в процессе выполнения алгоритма.

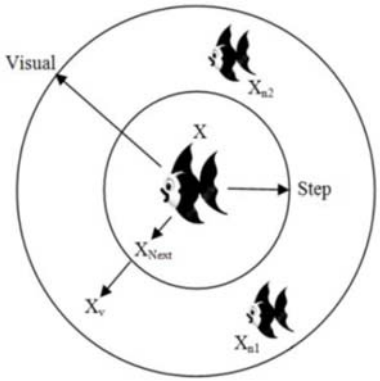


Рис. 4.20. Искусственная рыба и окружающая среда

Другие искусственные рыбы, находящиеся в пределах диапазона видимости рыбы x_i, обозначаются следующим образом:

$$V_j = \{j \neq j: \| x_i - x_j \| \leq d_{vis}\}. \quad (4.37)$$

Величина шага соответствует дистанции самого большого перемещения движения. Расстояние между двумя искусственными рыбками, находящимися в позициях X_i и X_j, рассчитывается как евклидово расстояние:

$$Dist_{(ij)} = \| x_i - x_j \|.$$

Оценка плотности пищи текущей позиции x_i является целевой функцией $f(X)$ алгоритма оптимизации. Можно считать, что AF находится плотной (наполненной пищей) среде, если в диапазоне ее видимости достаточно много других рыб. Для оценки плотности заполнения окружающего пространства рыбой используют можно использовать отношение числа рыб, находящихся в диапазоне видимости рыбы x_i к общему числу рыб в косяке:

$$\frac{|V_i|}{N} > \theta .$$

Если равенство выполняется, то плотность пищи достаточная, в противном случае – плотность низкая.

Модель искусственной рыбы включает в себя переменные и функции. Переменными являются x_i (текущее положение искусственной рыбы), d_{step} (шаг максимальной длины), d_{vis} (область видимости), Num_{try} (максимальное количество попыток для поиска лучшей позиции в диапазоне видимости) и коэффициент плотности скопления пищи ε ($0 < \varepsilon < 1$). Функции включают вышеперечисленные модели поведения: случайное (свободное) движение, поиск пищи, движение в составе косяка рыб, преследование, скачкообразное движение (прыжками). На каждой итерации алгоритма оптимизации (здесь и далее в этом разделе имеется в виду задача минимизации значений ЦФ) искусственная рыба ищет места с лучшими значениями целевой функции в пространстве поиска, выполняя одно из следующих действий.

Случайное движение

Как и обычные рыбы, находящиеся в своей естественной среде, искусственные рыбы могут двигаться хаотично, случайным образом перемещаясь из одной позиции в другую. Алгоритм AFAS использует модель случайного движения обычно в том случае, когда искусственная рыба не видит других «рыб» в пределах своего диапазона видимости или же при попадании алгоритма в локальный

оптимум, когда на протяжении определенного числа итераций не происходит улучшение значения целевой функции. Отсутствие улучшения значения целевой функции на протяжении m поколений можно оценить, например, с помощью следующего выражения:

$$\arg\min_x f_{t-m}(x) - \arg\min_x f_t(x) < \mu ,$$

где $f_t(x)$ – значение целевой функции «искусственной рыбы» x в поколении t; m – положительный поправочный коэффициент; μ – отрицательный поправочный коэффициент. При выполнении данного равенства считается, что процесс оптимизации попал в локальный оптимум и наблюдается стагнация.

Примерный алгоритм случайного поведения искусственной рыбы в рамках данной модели можно описать следующим образом.

1. Задание размера косяка AF. Выбор индивида AF x_i из текущей популяции (косяка). Задание нижней l_k и верхней u_k границ видимого диапазона AF.

2. На интервале значений $U \in [0, 1]$ случайным образом с помощью равномерного распределения задаем значение коэффициентов r и p.

3. Если значение коэффициента $r < 0.5$, то новая позиция y_i искусственной рыбы x_i определяется исходя из выражения: $x_i(k) + p\min(d_{vis}u_k - x_i(k)) \rightarrow y_i(k)$. В противном случае новая позиция y_i определяется выражением: $x_i(k) - p\min(d_{vis}x_i(k) - l_k) \rightarrow y_i(k)$. Здесь, как и ранее d_{vis} – это область видимости для текущего x_i индивида AF.

4. Если не все AF из текущего множества (косяка) проверены, то возвращаемся к п. 1, в противном случае – конец алгоритма.

Преследование

Данная модель описывает поведение «рыбы» x_i, которая в диапазоне своей видимости видит другую «рыбу» x_j, которая находится в месте большей концентрации (плотности) корма. В этом случае наша исходная «рыба» x_i начинает преследовать «рыбу» x_j. Модель поведения преследования может быть описана следующим образом:

$$\arg\min_j \left(f(x_j);\ j \in V_i \right),$$

$$x_j + r\left(x_{j^*} - x_j \right) \rightarrow y_j$$

где y_j – это новая позиция индивида x_i; j^* - индекс «рыбы», находящейся от «рыбы» x_i в пределах диапазона видимости и имеющей лучшее значение ЦФ среди индивидов, составляющих данный косяк. «Рыбы», находящиеся за пределами диапазона видимости в

данной модели поведения не участвуют. То есть «рыба» x_i выполняет преследование «рыбы» x_j только в том случае, когда «рыба» x_{j*} не только имеет лучшее значение целевой функции, но и находится от «рыбы» x_i в границах диапазона видимости.

Движение в составе косяка рыб

В случае, когда отдельная «рыба» движется в составе косяка AF, используется данная модель поведения. При этом каждая отдельная «рыба» в составе косяка движется по направлению к центру косяка из c_k «рыб», находящихся в пределах ее диапазона видимости. Такое поведение может быть описано следующим образом:

$$\frac{1}{|V_i|}\sum_{j\in V_i} x_j \to c_i ,$$

$$x_i + r(c_i - x_i) \to y_i,$$

где y_j – это новая позиция «рыбы» x_i.

Как и в предыдущей модели, те «рыбы», которые находятся вне пределов диапазона видимости рассматриваемой «рыбы» x_i в алгоритме не рассматриваются. Модель движения в составе косяка рыб применяется в тех случаях, когда диапазон видимости «рыбы» x_i одновременно не является пустым и не является заполненным, а также значение целевой функции центроида c_i выше значения целевой функции x_i.

Поиск пищи

Модель движения в режиме «поиск пищи» используется, если рассматриваемая в данный момент времени «рыба» x_i видит другую «рыбу» x_j или позицию в пределах диапазона видимости, где плотность корма выше. Тогда «рыба» x_i движение в данном направлении. Поведение «рыбы» в модели «поиск пищи» может быть описано следующим образом:

$$x_i + r(x_j - x_i) \to y_i,$$

где j – это выбранное случайным образом целое число, $j \in V_i$.

Модель «поиск пищи» предполагается движение в направлении случайно выбранной «рыбы», находящихся в границах диапазона видимости. Как и в предыдущих моделях, если «рыба» не находится в пределах видимости любой другой «рыбы», то такая «рыба» в работе алгоритма не участвует. Данная модель применяет в трех случаях:

а) диапазон видимости «рыбы» заполнен;

б) диапазон видимости «рыбы» не заполнен и значение функции $f(c_i)$ хуже значения $f(x_i)$;

в) диапазон видимости «рыбы» не заполнен и значение функции $f(x_{j*})$ хуже значения $f(x_i)$.

Скачкообразное движение (движение прыжками)

С точки зрения естественных систем эта модель имитирует поведение «летучих» рыб. Данная модель описывает поведение AF, которая выпрыгивает из воды и погружается в случайно выбранном месте. Такое поведение используется в случае, когда процесс оптимизации зашел в локальный оптимум и наблюдается стагнация. С помощью такого «скачкообразного» поведения конкретная «рыба» x_i перемещается из одной области пространства поиска в другую, что помогает найти выход из локального оптимума.

Алгоритм такого скачкообразного движения можно описать следующим образом.

1. Задание размера косяка AF. Выбор индивида AF x_i из текущей популяции (косяка). Задание нижней l_k и верхней u_k границ видимого диапазона AF.

2. На интервале значений $U \in [0, 1]$ случайным образом с помощью равномерного распределения задаем значение коэффициентов r и p.

3. Если значение коэффициента $r < 0.5$, то новая позиция искусственной рыбы x_i определяется исходя из выражения: $x_i(k) + p(u_k - x_i(k)) \rightarrow x_i(k)$. В противном случае новая позиция определяется выражением: $x_i(k) - p(x_i(k) - l_k) \rightarrow x_i(k)$.

4. Если не все AF_i из текущего множества (косяка) проверены, то возвращаемся к п. 1, в противном случае – конец алгоритма.

Обобщенный стандартный алгоритм AFSA

1. Задание размера начального множества решений (косяка AF). Выбор индивида AF x_i, $i \in N$, из текущего множества согласно выражению (4.31). Задание нижней l_k и верхней u_k границ видимого диапазона AF.

2. Если значение $V_j = 0$, то применяем алгоритм «**случайного движения**». Иначе,

2.1. Если диапазон видимости текущей «рыбы» заполнен, то применяем алгоритм «**поиска пищи**». Иначе,

2.1.1. Если $f(c_i) < f(x_i)$, тогда применяем алгоритм «**движения в составе косяка рыб**». Иначе применяем алгоритм «**поиска пищи**».

2.1.2. Если $f(x_{j*}) < f(x_i)$, тогда применяем алгоритм «**преследования**». Иначе применяем алгоритм «**поиска пищи**».

2.2. Вычисляем $\arg\min \{f(x_i), f(y_i)\} \rightarrow y_i$. Если диапазон видимости текущей «рыбы» заполнен, то применяем алгоритм «**поиска пищи**». Иначе,

3. Выбор следующего индивида AF из текущего множества.

4. Вычисляем arg min $\{f(x_i), f(y_i)\} \rightarrow y_i$, для $i \in N$.

5. Если улучшение значения ЦФ не происходит (алгоритм в фазе стагнации поиска), то тогда случайным образом выбираем целое число $j \in [1, N]$ и применяем алгоритм «**скачкообразного движения**». В результате получаем новую позицию x_j.

6. Если заданное число итераций алгоритма не пройдено, то увеличиваем номер текущей итерации и возвращаемся к п. 2. Иначе – конец работы алгоритма.

Необходимо отметить, что в результате выполнения описанного обобщенного алгоритма AFSA рассматриваемая на каждой итерации алгоритма i-тая «рыба» перемещается в новую позицию только в том случае, если значение ЦФ в этом случае улучшается y_i.

В литературе описаны примеры применения алгоритма AFSA для решения различных задач машинного обучения [154–156]: кластеризации данных [157], сегментации изображений [158]; задачи составления расписаний [159]; настройки беспроводных сенсорных сетей [160] и т. д. Достоинством стандартного алгоритма AFSA является его нечувствительность к исходному положению «рыб», гибкость и способность восстанавливаться после сбоев. Недостатки - высокая вычислительная сложность, а также сложность реализации. При этом результаты, которые показал стандартный алгоритм AFSA, незначительно превосходят результаты аналогичных алгоритмов, например алгоритма оптимизации на основе роя частиц (PSO). Ключевое отличие AFSA от PSO состоит в том, что движение искусственных рыб зависят исключительно от их текущего положения и взаимодействия с другими членами группы, тогда как движение частиц в PSO строится на основе истории их перемещений.

Для устранения недостатка, связанного с зависимостью результатов алгоритма от правильности задания значений основных параметров были разработаны различные модификации стандартного алгоритма.

Модифицированный алгоритм AFSA с использованием весовых коэффициентов

Рассмотрим усовершенствованный алгоритм искусственного косяка рыб, в который введен новый параметр, называемый весом перемещения. Вес перемещения может использоваться для адаптивной настройки основных параметров алгоритма: диапазона видимости и величины шага и, следовательно, для управления перемещениями

искусственной рыбы на пути к цели. Кроме того, использование веса перемещения позволяет поддерживать равновесие между глобальным и локальным поиском. Ранее в работах [161–163] описан похожий подход к алгоритму оптимизации роя частиц (PSO), суть которого в использовании специального параметра, называемого инерционным весом. При этом инерционный вес может быть положительной константой, положительной линейной или нелинейной функцией. Использование данного параметра позволили существенно улучшить производительность алгоритма оптимизации.

В AFSA поведение искусственной рыбы – это поиск в проблемной среде в пределах диапазона видимости, а затем движение к цели со случайным значением шага. Как уже было отмечено выше, в стандартном AFSA определение начальных значений шага и диапазона видимости существенно влияет на конечный результат, поскольку значения этих параметров остаются постоянными на протяжении всего времени выполнения алгоритма. Такой подход имеет свои достоинства и недостатки. Например, если мы задаем большие начальные значения для косяка AF, это приводит к увеличению скорости приближения к глобальному оптимуму и улучшению возможностей алгоритма по выходу из локальных оптимумов. Но при этом точность и устойчивость алгоритма снижаются. Выбор меньших значений для этих параметров приведет к тому, что AF будет лучше выполнять локальный поиск.

На самом деле, данный алгоритм лучше работает при решении задач глобальной оптимизации, но после достижения глобального оптимума он не может выполнить соответствующий локальный поиск из-за того, что диапазон видимости больше, чем должно быть. Таким образом, при больших значениях величины диапазона видимости вряд будет происходить улучшение текущего решения. Использование небольших значений этих параметров делает алгоритм более устойчивым и точным, но, одновременно, приводит к тому, что алгоритм движется к цели медленнее и не может выйти за пределы локальных оптимумов.

Следовательно, для достижения лучших результатов алгоритма необходимо вначале выбирать более высокие начальные значения для диапазона видимости и величины шага, а затем постепенно снижать их в процессе выполнения алгоритма. В этом случае «рыба» быстрее приближается к цели и с большей вероятностью выходит за пределы локальных оптимумов. Впоследствии, по мере приближения к глобальному оптимуму, искусственная рыба может

более подробно исследовать окружающую среду при меньших значениях диапазона видимости и величины шага.

Чтобы контролировать значения диапазона видимости и величины шага и поддерживать баланс между глобальным и локальным поиском, был предложен параметр [153], называемый весом перемещения (*Movement Weight*, *MW*). *MW* может быть положительной константой, меньшей единицы, линейной или нелинейной функцией. На каждой итерации диапазона видимости и величины шага задаются в соответствии со следующими уравнениями:

$$d_{vis_i} = MW \times d_{vis_{i-1}}$$

$$d_{step_i} = MW \times d_{step_{i-1}}$$

где i - текущая итерация алгоритма. Для получения более точных значений диапазона видимости и величины шага на текущей итерации используются различные методы расчета *MW*.

Линейное значение MW. В этом случае величина *MW* представляется положительной функцией, либо линейно возрастает, либо линейно убывает в области поиска. В первом случае значение функции изменяется от минимума до максимума и вычисляется в соответствии с учетом номера текущей и конечной итерации. Во втором [164] в начале алгоритма величина MW имеет максимальное значение, а затем в процессе выполнения алгоритма она линейно уменьшается до минимума в зависимости от номера итерации. MW на каждой итерации вычисляется в соответствии с уравнением:

$$MW_i = MW_{\min} + \frac{i_{\max} - i}{i_{\max}} \left(MW_{\max} - MW_{\min} \right).$$

В случае, когда используется линейно возрастающая функция [165] значение MW линейно увеличивается в процессе выполнения алгоритма. Величина MW на каждой итерации определяется при этом следующим выражением:

$$MW_i = MW_{\min} - \frac{i_{\max} - i}{i_{\max}} \left(MW_{\max} - MW_{\min} \right).$$

Случайное значение MW. При этом величина *MW* [166] – это случайное число, заданное на интервале между минимальным и максимальным значениями *MW*. На каждой итерации значение *MW* вычисляется на основе следующего уравнения:

$$MW_i = MW_{Min} + Rand \times (MW_{Max} - MW_{Min}),$$

где *Rand* – случайное число, заданное на интервале от 0 до 1 с нормальным распределением.

Результаты проводившихся экспериментов [167] подтвердили прогнозы. В качестве начальных значений для величин d_{vis} и d_{step} принимались величины равные 40% и 25% длины диапазона переменных целевой функции соответственно. Величина *MW* играет ключевую роль в точности и качестве конечных результатов. Малое значение *MW* может привести к резкому снижению значений диапазона видимости и величины шага, что приводит к тому, что искусственная рыба становится фактически «неподвижной» и «слепой» до достижения глобального оптимума.

Фактически, искусственная рыба не может эффективно исследовать окружающую среду, если значение диапазона видимости мало. Это также относится и к величине шаг. На самом деле, когда искусственная рыба находится достаточно далеко от глобального оптимума, а величина шага слишком мала, искусственная рыба не может эффективно перемещаться в области поиска. Следовательно, выбор небольшого значения *MW* делает искусственную рыбу неподвижной.

Поиск косяком рыб

Алгоритм поиска, основанный на поведении косяка рыб (*Fish School Search, FFS*), впервые был предложен в 2008 г. Фило (*B. Filho*), Нето (*L. Neto*) и др. [168].

Под косяком рыб (*fish school*) понимают некоторое множество агентов (рыб), которые передвигаются в пространстве поиска с постоянной скоростью и сохраняют при этом одинаковое расстояние между собой. Такая модель поведения повышает эффективность поиска пищи, защиту от хищных рыб и, кроме этого, позволяет уменьшить энергетические затраты.

В *FFS* алгоритме все рыбы находятся в заданной области пространства поиска (аквариуме), их задачей является поиск пищи (т.е. нахождение решения оптимизационной задачи). Вклад каждой рыбы в процессе поиска оптимального решения определяется ее весом. Таким образом, главной особенностью данного алгоритма является наличие веса у каждого агента, благодаря чему отпадает необходимость нахождения глобального оптимума.

Операторы, применяемые в данном алгоритме, делятся на две группы:

- **оператор кормления** (*feeding operator*), определяющие степень исследования различных областей аквариума;
- **оператор плавания**, задающий методы миграции агентов.

Рассмотрим основные принципы работы оператора кормления. Пусть w_i, $i \in [1:|S|]$ – вес агента s_i. В общем случае вес агента прямо пропорционален разности значений целевой функции на последующей и текущей итерациях, и задается формулой

$$w_i' = w_i + \frac{\varphi(X_i') - \varphi(X_i)}{\max(\varphi(X_i'), \varphi(X_i))}, i \in [1:|S|].$$

Вес агента ограничивается величиной $w_{max} > 0$. Поэтому для каждого агента s_i справедливо условие:

$$w_i \in [1:w_{max_{fr}}],\ i \in [1:|S|].$$

где $w_{max_{fr}}$ – свободный параметр алгоритма.

В процессе инициализации новой популяции присваивается вес $w_{max} / 2$.

В алгоритме *FFS* выделяют три разновидности операторов плавания – оператор индивидуального плавания, оператор плавания на основе коллективного инстинкта и оператор на основе коллективного желания.

Данные виды плавания выполняются на интервалах $(t, \tau]$, $(\tau, \theta]$, $(\theta, t']$ соответственно, причем $t < \tau < \theta < t' + 1$, $t' = t + 1$.

При **индивидуальном плавании** *(individual swimming)* направление перемещения агентов задается случайным образом. Если в результате этого перемещения агент выходит за пределы области допустимых значений D, то действие не выполняется. Также не происходит перемещение агента s_i в том случае, если в новом положение X_i^t значение целевой функции не превышает значения в предыдущем положение X_i^t, т.е. выполняется неравенство $\varphi(X_i^\tau) < \varphi(X_i^t)$. V_i^{ind} – шаг перемещения представляет собой случайную величину в интервале $[0; v_{max}^{ind}]$ и определяется формулой:

$$V_i^{ind} = \mathrm{U}|X|\ (0; 1)\ v_{max}^{ind},\ i \in [1:|S|].$$

Для плавного перехода от диверсификации популяции агентов на начальных этапах поиска к интенсификации поиска на его конечных этапах необходимо в процессе поиска линейно уменьшать величину v_{max}^{ind}.

Операцию индивидуального плавания можно рассматривать как локальный поиск в окрестностях текущего положения агента, т. к. данный процесс может включать в себя несколько итераций.

Плавание на основе коллективного инстинкта (*collective-instinct swimming*) выполняется обычно после того, как завершилась предыдущая операция. С формальной точки зрения для реализации данной операции используется следующее выражение:

$$X_i^{\theta} = X_i^{\tau} + \frac{\sum_j V_j^{ind}(\tau)\left(\varphi\left(X_j^{\tau}\right) - \varphi\left(X_j^{t}\right)\right)}{\sum_j \left(\varphi\left(X_j^{\tau}\right) - \varphi\left(X_j^{t}\right)\right)}, i \in \left[1 : |S|\right].$$

Второе слагаемое в приведенной формуле – шаг миграции общий для всех агентов. Данное выражение показывает, что в процессе плавания на основе коллективного инстинкта каждый из агентов подвержен влиянию со стороны всех остальных агентов.

После выполнения всеми агентами плавания на основе коллективного инстинкта определяем координаты центра тяжести косяка X_c:

$$X_c^{\theta} = \frac{\sum_i w_i^{\theta} X_i^{\theta}, i \in \left[1 : |S|\right]}{w_{\Sigma}^{\theta}},$$

где $w_{\Sigma}^{\theta} = w_{\Sigma}(\theta) = \sum_{i=1}^{|S|} w_i^{\theta}$ – суммарный вес популяции на данной итерации.

Таким образом, в случае успешного выполнения первого и второго оператора популяция стремится к своему центру, что повышает качество поиска оптимального решения. В противоположном случае популяция расширяется, что повышает уровень ее диверсификации.

Следующей операцией является **плавание на основе коллективного желания (воли)** (*collective volition swimming*). При этом, если общий вес косяка (целевая функция популяции) в ходе выполнения предыдущих этапов увеличился, все агенты смещаются к центру тяжести популяции, если же общий вес уменьшился, то в противоположном направлении.

Плавание на основе коллективного желания выполняется в соответствие с правилом

$$X_i' = X_i^{\theta} \pm v^{vol}(X_i^{\theta} - X_c^{\theta}),\ i \in [1 : |S|],$$

причем тип арифметической операции выбирается в зависимости от результатов операции сравнения значений весов: сумма, если выполняется неравенство $w_{\Sigma}^{\theta} > w_{\Sigma}^{\theta-1}$, и разность – в

противоположном случае. Здесь $w_{\Sigma}^{\theta+1}$ - общий вес популяции после предыдущей операции; параметр v^{vol} – это размер шага перемещения агента:

$$v^{vol} = v_{vol}^{\max} U_1(0;1),$$

где $v_{max}^{vol} > 0$ – максимальная длина шага.

Можно отметить плотный алгоритм *FFS (Density FFS, DFSS)*, который является модификацией алгоритма FFS. В нем дополнительно вводятся оператор памяти и оператор разбиения.

Оператор памяти (*memory operator*) реализуется на основе множества векторов памяти M_i, $i \in [1:|S|])$ и множества агентов популяции. Целью применения данного оператора является формализация текущего и предшествующих воздействий на каждого агента популяции со стороны других агентов.

Оператор разбиения (*partitioning operator*), используя информацию о памяти агентов, формирует множество подпопуляций на основе текущей популяции $S = (s_i: i \in [1:|S|])$.

Помимо описанных методов широкое распространение получили гибридные методы, основанные на интеграции алгоритма *FFS* с другими роевыми методами, например, волевой алгоритм роя частиц (*Volititive PSO, VPSO*).

4.9. Модель и алгоритм прыжков лягушки с перемешиванием

Алгоритм, имитирующий поведение лягушек *(Shuffled Frog-Leaping Algorithm, SFLA*) был разработан Юсуфом (*Eusuff*) и Лэнси (*Lansey*) в 2003 году [169]. Он применяется для решения многих нелинейных, недифференцируемых и мультимодальных задач оптимизации [170]. Юсуф и др. [171] продемонстрировали возможности *SFLA* для калибровки моделей подземных вод и решения задач оптимизации для распределительных сетей водоснабжения. В работе также было проведено сравнение результатов, получаемых с помощью *SFLA* с результатами генетического алгоритма (GA). Проведенное сравнение показало, что *SFLA* может быть эффективным инструментом для решения задач комбинаторной оптимизации. Чанг и Лэнси [172] разработали общую крупномасштабную модель водоснабжения для минимизации общей стоимости системы путем интеграции математического представления системы водоснабжения с применением *SFLA*. В литературе описаны примеры успешного использования данного алгоритма для решения различных задач оптимизации, таким как распределение водных

ресурсов [173–181], ремонт опор моста [182], планирование расписания работы [183] и задача коммивояжера [184].

Алгоритм перемешанных прыжков лягушки представляет собой гибрид двух других методов оптимизации: метода роя частиц (PSO) и алгоритма перемешанной комплексной эволюции (*Shuffled Complex Evolution, SCE*). Основная идея алгоритма перемешанной комплексной эволюции состоит в том, что начальная популяция разделяется на несколько самостоятельно развивающихся подпопуляций, между которыми периодически происходит обмен. В алгоритме применяется вероятностный отбор родителей, а для предотвращения преждевременной сходимости используется механизм случайной генерации новых решений.

Главным преимуществом алгоритма, имитирующего поведение лягушек, является высокая скорость сходимости [185]. Этот алгоритм сочетает в себе преимущества генетических алгоритмов и алгоритма роя частиц, основанного на социальном поведении.

В работе [171] авторы дали определение меметической эволюции и провели ее сравнение с эволюцией в генетике. «*Мем*» – это информационный шаблон, который, распространяясь, воздействует на сознание людей и животных и изменяет их поведение. Распространять мемы может любой индивид, который ими обладает. Таким образом, любой шаблон можно считать мемом в случае, когда данная идея или информационный шаблон влияют на кого-то, и этот шаблон повторяется или передается кому-то еще. В противном случае шаблон не считается мемом. Соответственно вся передаваемая при этом информация называется *меметической*. Примерами мемов являются песни, идеи, крылатые фразы, мода на одежду и методы изготовления каких-либо предметов.

Меметический алгоритм (MA), происходит от слова «мем» и представляет собой популяционный метод решения задач оптимизации [171]. Каждый мем содержит мемотипы (*memotypes*), которые напоминают гены в хромосоме. Гены и мемы передаются от одного человека к другому различными путями, и их назначение различно. Мемы используются в основном для повышения коммуникабельности среди их носителей (применительно к алгоритму *SFLA* – это лягушки). Гены передают характеристики ДНК от родителей к потомству.

Юсуфф и др. [171] предположили, что меметическая и генетическая эволюция подчиняются одним и тем же основным принципам.

Тем не менее, меметическая эволюция – гораздо более гибкий механизм, чем генетическая эволюция. Это связано с тем, что в процессе генетической эволюции, что гены могут передаваться из поколения в поколение, от родителей к потомкам, а это означает, что для их распространения может потребоваться достаточно длительный срок. В свою очередь мемы могут передаваться между индивидами в течение нескольких минут. Репликация генов ограничена относительно небольшим числом потомков одного родителя, в то время как количество индивидуумов, способных принять мем от одного родителя, практически не ограничено. Следовательно, распространение мемов происходит гораздо быстрее, чем распространение генов [169].

В основу модели *SFLA* положено поведение лягушек в процессе поиска пищи. При этом происходит взаимодействие агентов из различных подпопуляций между собой, в процессе которого они обмениваются информацией с целью приближения к оптимальному решению.

Метод, имитирующий поведение лягушек, моделирует поведение стаи особей при поиске пищи. Внутри каждой стаи производится одно и то же количество итераций локального поиска. Локальный поиск аналогичен применяемому в методе частиц в стае (члены стаи сообщают друг другу информацию о наилучших позициях в стае или в популяции в целом). На основании локального поиска лягушка в стае улучшает свое положение, чтобы получить больше пищи (достигнуть наилучшего решения). Таким образом, на каждой итерации особь с наихудшим значением целевой функции будет двигаться к лучшему решению на основе полученной информации.

Представим группу лягушек, прыгающих по болоту. На болоте в разных местах разбросаны камни. Лягушки хотят как можно быстрее найти камень с максимальным количеством доступной пищи. Для этого они улучшают свои мемы. Лягушки взаимодействуют друг с другом и вырабатывают свои мемы, обмениваясь информацией. Лягушки меняют свое положение, изменяя размер шага в зависимости от развития мемов.

SFLA можно рассматривать как меметический алгоритм, в котором лягушки (решения) улучшаются в процессе меметической эволюции. Отдельные лягушки в *SFLA* являются носителями мемов и представлены посредством меметического вектора. Каждый мем, связанный с лягушкой, состоит из нескольких мемотипов и является решением задачи оптимизации, в то время как мемотипы являются переменными для принятия решения и напоминают гены

хромосомы в генетическом алгоритме. *SFLA* не изменяет физические характеристики индивидуума; он постепенно улучшает идеи, которыми обладает каждая лягушка в виртуальной популяции. Виртуальная популяция лягушек (набор разнообразных решений) используется для моделирования пула мемов аналогично популяции, представляющей пул хромосом в ГА. Набор лягушек представляет собой популяцию (множество) решений. Множество возможных решений делится на подмножества, которые называются *мемплексами* (*memeplexes*). Каждый мем является единицей культурной эволюции. Термин «мемплекс» введен для обозначения группы взаимоподдерживающих мемов, которые образуют организованную систему идей, такую как религия или научная теория. Мемплексы можно воспринимать как набор параллельно развивающихся сообществ (подмножеств) лягушек, пытающихся достичь какой-либо цели. Каждое подмножество лягушек или мемплекс развивается в направлении достижения своей цели. Прыжки лягушек улучшают качества индивидуальных мемов и повышают эффективность достижения цели. В рамках каждого мемплекса лягушки находятся под влиянием идей других лягушек. Таким образом, они участвуют в меметической эволюции. Обмен информацией между мемплексами происходит в ходе процедуры перемешивания, которое выполняется в промежутке между разными этапами меметической эволюции и является одним из шагов алгоритма. В таблице 4.4 приведены характеристики *SFLA* в терминах, принятых в генетической и меметической эволюции соответственно.

Таблица 4.4

Характеристики *SFLA*

Генетический алгоритм	**Меметический алгоритм**
Переменная задачи	Мемотип
Решение (хромосома)	Мем (лягушка)
Целевая функция	Количество пищи
Начальная популяция	Случайное множество лягушек
Селекция	Распределение лягушек по мемплексам
Поиск нового решения	Прыжки лягушки

Выполнение *SFLA* начинается с создания начальной популяции (набора мемов-лягушек, каждый из которых является решением задачи оптимизации) размера N. Генерация начальной популяции выполняется случайным образом.

На первом этапе все лягушки (сгенерированная случайным образом начальная популяция) распределяются по нескольким m непересекающимся подпопуляциям (мемплексам), таким образом, чтобы каждая лягушка была назначена одному мемплексу. Обычно N кратно m, поэтому число агентов (лягушек) в каждой подпопуляции одинаково (например, $N = 200$, $m = 20$).

Полученные мемплексы эволюционируют независимо друг от друга, осуществляя поиск в пространстве решений в разных направлениях. Обмен информацией между мемплексами происходит в процессе выполнения процедуры перемешивания. После завершения процедуры перемешивания поиска оптимальных решений мемплексами (эволюция) продолжается. Выполнение алгоритма продолжается до тех пор, пока не будут выполнен заданный критерий остановки. В общем виде схема выполнения SFLA может быть представлена следующим образом.

1. Ввод исходных параметров алгоритма.
2. Создание начального множества решений.
3. Разделение лягушек по мемплексам.
4. Поиск лучших решений (прыжки лягушки).
5. Выполнение процедуры перемешивания.
6. 6 Проверка выполнения условия остановки алгоритма. Если условие не выполнено, то возврат к п.4, в противном случае, переход к п. 7.
7. Сохранение полученного решения.

Создание начальной популяции

Каждое возможное решение задачи оптимизации в *SFLA* называется лягушка. Каждая лягушка содержит N переменных (мемотипов) при решении N-мерной задачи оптимизации. Решение представлено в виде массива размером $1 \times N$:

$$Frog = X = x_1, x_2, \ldots, x_i, \ldots, x_N \qquad (4.38)$$

где X – возможное решение; x_i – i-я переменная решения (мемотип) решения X; N – число переменных.

Значения переменных (x_1, x_2, …, x_N) определяют мемотипы. *SFLA* начинается со случайной генерации матрицы размером M ≈ N, где M и N – размер популяции и число переменных соответственно:

$$P = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_j \\ \vdots \\ X_M \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,i} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,i} & \cdots & x_{2,N} \\ & & & \vdots & & \\ x_{j,1} & x_{j,2} & \cdots & x_{j,i} & \cdots & x_{j,N} \\ & & & \vdots & & \\ x_{M,1} & x_{M,2} & \cdots & x_{M,i} & \cdots & x_{M,N} \end{bmatrix} \quad (4.39)$$

где X_j – j-е решение, $x_{j,i}$ – i-я переменная j-го решения; M – размер популяции.

Распределение лягушек по мемплексам

Для оценки ценности каждой лягушки вычисляется значение функции приспособленности. Затем лягушки сортируются в порядке возрастания или убывания значений их функции приспособленности для задач минимизации или максимизации соответственно. На рисунке 4.21 показан пример сортировки лягушек в соответствии со значениями функции приспособленности F(X) в задаче максимизации.

После сортировки лягушки распределяются по Z мемплексам, по Y лягушек в каждом мемплексе. Порядок разбиения начальной популяции на мемплексы может быть разным, можно использовать, например, случайное разбиение или разбиение на основе лексикографического порядка, когда 1-й элемент назначается в 1-е подмножество, 2-й – во второе; m-й элемент в подмножество m; m+1 – й элемент – в 1-е множество; m+2 – й элемент – во 2-е множество и т. д.

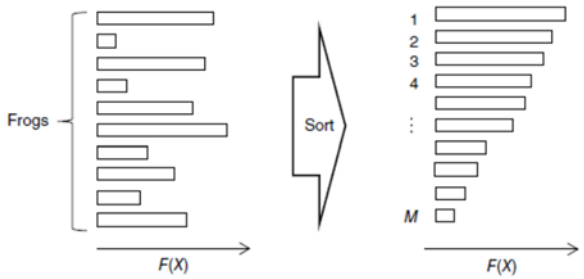


Рис. 4.21. Сортировка лягушек в соответствии с функцией приспособленности F(X) в задаче максимизации

Пусть лягушка с наилучшим значением функции приспособленности становится первым членом первого мемплекса; лягушка, занявшая второе место, становится первым членом второго мемплекса и т.д. Распределение отсортированных лягушек продолжается до тех пор, пока Z-я отсортированная лягушка не станет первым членом последнего Z-го мемплекса. На следующем шаге $(Z + 1)$-я лягушка становится вторым членом первого мемплекса и так далее. На рисунке 4.22 показан пример распределения лягушек по мемплексам. Значения параметров Y и Z задаются пользователем. Следовательно, размер популяции (M) равен $Z \times Y$.

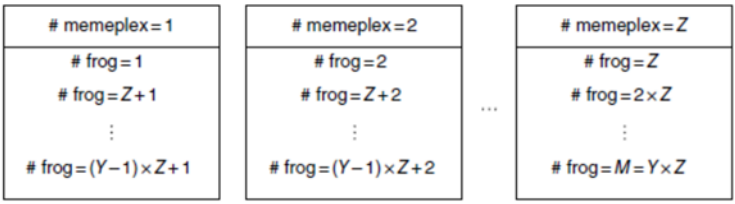


Рис. 4.22. Распределение лягушек по разным мемплексам; Z – количество мемплексов; Y – количество лягушек, распределенных в каждый мемплекс

Прыжки лягушек

После того, как популяция лягушек с возможными решениями классифицируется на несколько параллельных сообществ (мемплексов), на лягушек влияют идеи других лягушек в пределах каждого мемплекса, что приводит к так называемой меметической эволюции. Меметическая эволюция улучшает качество индивидуального мема и способствует достижению цели отдельной лягушкой. Лягушки с лучшими мемами (идеями) вносят больший вклад в разработку новых идей, чем лягушки с плохими идеями. Это гарантирует, что меметическая эволюция (то есть распространение превосходных идей) с большей вероятностью отберет лучших особей для продолжения поиска оптимума.

Цель лягушек – двигаться к оптимуму, совершенствуя свои мемы. Для этого в каждом мемплексе выбирается подмножество для обмена идеями, называемое субмемплексом. На самом деле, в подмемплексе есть $Q < Y$ лягушек. Иерархия системы может быть описана следующим образом (рис. 4.23):

Уровень 1. Общая популяция, включающая все имеющиеся решения, разделенная на Z множеств (мемплексов).

Уровень 2. Отдельное множество (мемплекс), содержащее Y решений (лягушек) из общего числа Z множеств.

Уровень 3. Некоторое подмножество (субмемплекс) из Q лягушек (решений), выделенное из отдельного Z_i - го множества решений.

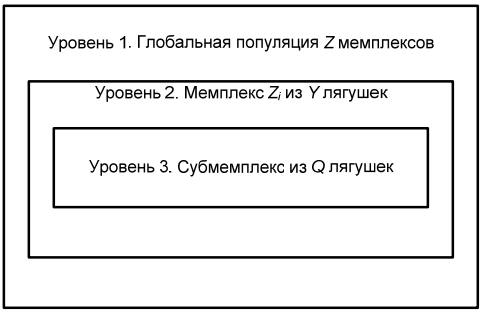


Рис. 4.23. Показано представление мемплекса и субмемплекса во всей популяции

Лягушки, входящие в состав подсемейства, ранжируются в соответствии с значением их целевой функции. Выбор лягушек для формирования субмемплекса осуществляется в соответствии с элитной стратегией, согласно которой лягушки с лучшим значением целевой функции имеют более высокую вероятность выбора. Вероятность, с которой Q лягушек выбираются из мемплекса для формирования подмемплекса, соответствует следующему распределению:

$$P_j = \frac{2 \times (Y + 1 - j)}{Y \times (Y + 1)},\ j = 1, 2, \ldots, Y \qquad (4.40)$$

где P_j – вероятность того, что j-я лягушка из мемплекса будет выбрана в субмемплекс, j – индекс текущей лягушки в мемплексе; Y – общее количество лягушек в мемплексе; индекс $j = 1$ соответствует лучшей, а индекс $j = Y$ – худшей лягушке в мемплексе. Из уравнения (4.40) следует, что лягушка, имеющая наибольшее значение ЦФ в текущем мемплекса ($j = 1$) имеет вероятность, равную $2/(Y + 1)$, быть выбранной, а наименее приспособленная лягушка ($j = Y$) из мемплекса выбирается с вероятностью $2/(Y^2 + Y)$.

Таким образом, мы отбираем Q лягушек для формирования субмемплекса и ранжируем по мере убывания значения их целевой функции: наиболее приспособленная лягушка получает 1-й ранг, а худшая лягушка – Q-й ранг.

После этого выполняется процедура локального поиска. В процессе выполнения локального поиска каждая подпопуляция эволюционирует отдельно в течение заданного числа итераций i_{max}. На каждой итерации происходит обновление только одного самого худшего решения в каждом субмемплексе. При этом худшая лягушка обменивается мемами с лучшей лягушкой из имеющихся в текущем субмемеплексе или, если обмен не дает улучшения, с лучшей лягушкой во всей глобальной популяции. Лучшие и худшие лягушки в каждой подгруппе и лучшая лягушка в мире во всей популяции лягушек называются *Mbest*, *Mworst* и *Pbest* соответственно. Процедура локального поиска может быть описана следующим образом:

Шаг 1. Улучшение худшего решения в выбранном субмемплексе. Для обновления текущего худшего решения в разных модификациях алгоритма могут использоваться разные выражения, например:

<u>*Вариант 1*</u>. Мем *Mworst* обновляем следующим образом:

$$d_i = Rand \times (x_{Mbest,i} - x_{Mworst,i}),\ D_{min} \le d_i \le D_{max},\ i = 1, 2, \ldots, N \quad (4.41)$$

$$x_i' = x_{Mworst,i} + d_i,\ i = 1, 2, \ldots, N \quad (4.42)$$

$$X^{(new)} = (x_1', x_2', \ldots, x_i', \ldots, x_N') \quad (4.43)$$

где d_i – размер шага для i-й переменной, соответствующей худшему решению в субмемплексе; *Rand* – случайные значения в диапазоне [0, 1]; $x_{Mbest,i}$ – i-я переменная, соответствующая лучшему решению в субмемплексе; $x_{Mworst,i}$ – i-я переменная, соответствующая худшему решению в субмемплексе; D_{min} и D_{max} – минимальное и максимальное допустимые значения размера шага соответственно; $x_i = i$-я переменная нового решения; и $X^{(new)}$ – новое решение.

<u>*Вариант 2*</u>. Обновляем решение, которому соответствует худшее значение целевой функции x_{worst}:

$$x_{worst} + r(x_{best} - x_{worst}) \rightarrow x_{worst},$$

где $r \in [0, 1]$ – это случайное число, заданное на основе равномерного распределения; x_{best} – решение с лучшим значением целевой функции.

<u>*Вариант 3*</u>.

$$x_{worst}^j(t+1) = \frac{U_i(-0,5;0,5)v_{\max}\left(\left\|x_{best}^j - x_{worst}^j\right\|\right)}{\left\|x_{best}^j - x_{worst}^j\right\|_E},\ j \in [1, N],$$

где v_{max} – максимальное значение величины шага перемещения.

Если новое решение $X^{(new)}$, полученное в результате выполнения шага 1 имеет значение ЦФ лучше, чем худшее решение *Mworst*, то

новое решение заменяет *Mworst*. Если же после выполнения шага 1 значение x_{worst} не улучшилось, то переходим к шагу 2.

Шаг 2. Поиск улучшения среди всех решений популяции. Выбираем для обмена вместо лучшего решения в текущем субмемплексе *Mbest* лучшее решения во всей популяции *Pbest* и пытаемся снова улучшить худшее решение.

Вариант 1. Новое решение $X^{(new)}$ генерируется следующим образом:

$$d_i = Rand \times (x_{Pbest,i} - x_{Mworst,i}),\ D_{min} \le d_i \le D_{max},\ i = 1, 2, \ldots, N \quad (4.44)$$

$$x_i' = x_{Mworst,i} + d_i,\ i = 1, 2, \ldots, N \quad (4.45)$$

$$X^{(new)} = (x_1', x_2', \ldots, x_i', \ldots, x_N') \quad (4.46)$$

в случае, когда $x_{Pbest,i}$ – i-я переменная, соответствующая наилучшему решению во всей популяции.

Вариант 2. Повторяем обновление текущего значения x_{worst} следующим образом:

$$x_{worst} + r(x_g - x_{worst}) \rightarrow x_{worst},$$

где r – это новое случайное число, заданное на основе равномерного распределения; x_g – решение с лучшим значением целевой функции среди всех имеющихся подпопуляций.

В случае, если и после этого значение $X^{(new)}$ не лучше *Mworst*, переходим к шагу 3.

Заменяем текущее значение *Mworst* случайно сгенерированным новым решением следующим образом:

$$x_i' = Rnd\ (x_i^{(L)}, x_i^{(U)}),\ i = 1, 2, \ldots, N \quad (4.47)$$

$$X^{(new)} = (x_1', x_2', \ldots, x_i', \ldots, x_N') \quad (4.48)$$

в котором *Rnd* (a, b) = случайное значение из диапазона [a, b], $x_i^{(L)}$ и $x_i^{(U}$ = нижнее и верхнее допустимые значения i-й переменной принятия решения соответственно.

После обновления значения *Mworst* все элементы мемплекса ранжируются в соответствии с значениями их ЦФ. Для создания нового субмемплекса случайным образом выбирается новый набор решений, и худшее решение из этого набора обновляется, как описано выше. Этот процесс выполняется для всех мемплексов определенное пользователем количество раз μ.

Процедура перемешивания

После определенного количества меметических эволюций (μ) мемплексы принудительно перемешиваются, образуя новые мемплексы в процессе перетасовки. Это перемешивание повышает качество мемов, поскольку на него влияют идеи лягушек из разных регионов болота (разные мемплексы).

Процедура перемешивания (миграция лягушек) ускоряет поиск за счет обмена информацией и гарантирует, что не произойдет смещение поиска в сторону какой-либо частной цели [170].

Перемешивание приводит к сближению культур (популяций), которые развиваются изолированно, до того момента, пока они не объединяются для обмена идеями или информацией. Перемешивание объединяет все мемплексы в общую популяцию возможных решений, из которых создаются следующие мемплексы. Кроме того, в процессе перемешивания обновляется список лучших мемплексов.

Процедуру меметической эволюции можно представить в виде следующего алгоритма.

1. Находим решение с лучшим значением ЦФ x_g во всех имеющихся подпопуляциях.

2. В каждой j-й подпопуляции находим решения с лучшим x_j^{best} и худшим x_j^{worst} значением ЦФ.

3.1. С помощью правила 1 выполняем обновление найденного худшего решения в подпопуляции x_j^{worst}. Если результате обновления значение x_j^{worst} улучшилось, то переходим к п. 4, в противном случае переходим к п. 3.2.

3.2. С помощью правила 2 выполняем обновление найденного худшего решения в подпопуляции x_j^{worst}. Если результате обновления значение x_j^{worst} улучшилось, то переходим к п. 4, в противном случае переходим к п. 3.3.

3.2. С помощью правила 3 выполняем обновление найденного худшего решения в подпопуляции x_j^{worst}.

4. Проверяем текущее значение счетчика итераций. Если заданное число итераций алгоритма не пройдено, то возвращаемся к п. 1; в противном случае – конец процедуры.

С учетом приведенной процедуры локального поиска общий алгоритм перемешанных прыжков лягушки можно записать в следующем виде.

1. Создание начальной популяции решений размера N.

2. Находим решение с лучшим значением ЦФ x_g среди всех решений начальной популяции.

3. Разделяем начальную популяцию размера N на m непересекающихся подпопуляций (мемплексов). Предполагаем, что N кратно m, поэтому число агентов (лягушек) в каждой подпопуляции одинаково.

4. Выполнить процедуру меметической эволюции для каждой j-й подпопуляции, $j \in [1, N]$.

5. Выполняется перемешивание (*shuffling*) агентов (решений) путем объединения всех агентов в одну популяцию. При этом все решения упорядочиваются по убыванию значения целевой функции. Затем выполняется новое разбиение. Порядок формирования новых подмножеств, как и в п. 3. может быть различным: случайным образом или в соответствии с значениями их ЦФ: наилучшее решение помещается в первую подмножество, следующее – во второе и т. д. Данная операция производится с целью обмена информацией между подпопуляциями с целью эффективного приближения к глобальному экстремуму. Процедура поиска завершается при прохождении заданного числа итераций.

6. Если все подмножества просмотрены, то переходим к следующему поколению алгоритма.

Структурная схема алгоритма перемешанных прыжков лягушки показана на рисунках 4.24, 4.25.

Рис. 4.24. Схема алгоритма перемешанных прыжков лягушки

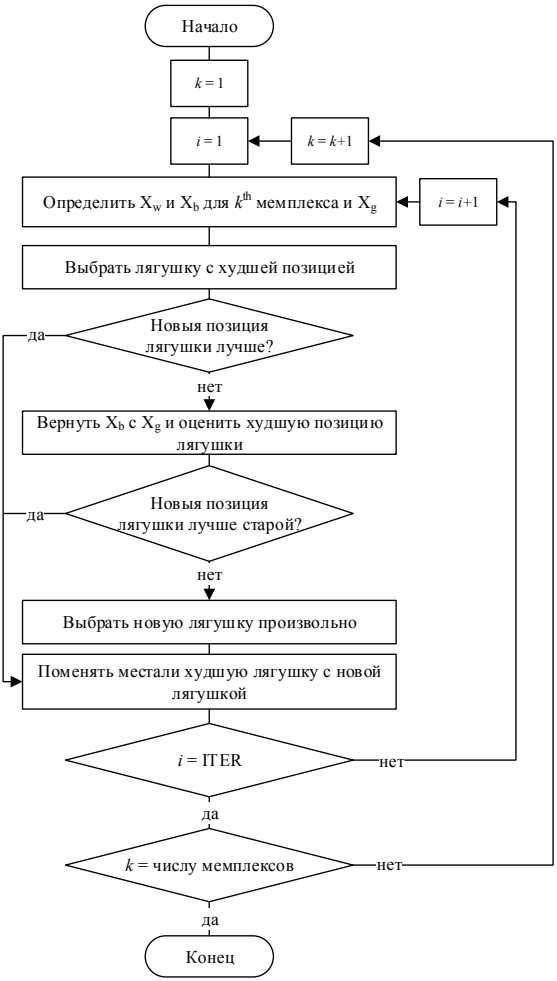


Рис. 4.25. Схема алгоритма меметической эволюции (локального поиска)

Алгоритм перемешанных прыжков лягушки

Шаг 1. Задать: размер популяции N; число подпопуляций (*memeplexes*) M, максимальное число итераций локального поиска в каждой подпопуляции i; максимальное число глобальных итераций *ITER*. Присвоить счетчику глобальных итераций начальное значение $j = 1$.

Шаг 2. Генерировать начальную популяцию лягушек (решений) на множестве D с помощью равномерного распределения. Для каждой лягушки подсчитать значение целевой функции.

Шаг 3. Упорядочить элементы популяции по убыванию значений целевой функции.

Шаг 4. Сформировать из популяции M стай (подмножеств) лягушек: наилучшее (с наибольшим значением целевой функции) решение поместить в первую стаю, следующее – во вторую и т. д., M-е поместить в M-ю стаю, $(M + 1)$-е снова в первую стаю и т. д. Результатом являются M стай, содержащих по m лягушек каждая, так что $N = M \times m$.

Шаг 5. В рамках каждой стаи провести заданное число итераций локального поиска (прыжков лягушки):

Шаг 5.1. Присвоить значение переменной Num = 1 (номер стаи).

Шаг 5.2. Присвоить $i = 1$ (счетчик числа итераций локального поиска).

Шаг 5.3. В рамках стаи с номером Num найти:

– наилучшее решение x_{best} (положение лягушки) в стае, которому соответствует наибольшее значение целевой функции;

– наихудшее решение x_{worst}, которому соответствует наименьшее значение целевой функции.

Найти наилучшее глобальное решение в популяции x_g.

Шаг 5.4. Найти новое положение наихудшей лягушки:

$$x_{worst} + r(x_{best} - x_{worst}) \to x_{worst}.$$

Шаг 5.5. Если $f(x_{worst}^{new}) > f(x_{worst})$, то заменить решение x_{worst} на x_{worst}^{new} и перейти к шагу 5.7. Иначе попытаться снова улучшить положение наихудшей лягушки:

$$x_{worst} + r(x_g - x_{worst}) \to x_{worst}.$$

Шаг 5.6. Если $f(x_{worst}^{new}) > f(x_{worst})$, то заменить решение x_{worst} на x_{worst}^{new} и перейти к шагу 5.7. Иначе генерировать решение x_{worst}^{new} случайно на множестве D с помощью равномерного распределения, заменить им решение x_{worst} и перейти к шагу 5.7.

Шаг 5.7. Если пройдены не все итерации, то увеличить $i = i + 1$ и перейти к шагу 5.3. Если все итерации пройдены, то локальный поиск в стае завершить. Сравнить номер Num с числом M:

– если $Num = M$, то завершить локальный поиск во всех стаях и перейти к шагу 6;

– если $Num < M$, то положить $Num = Num + 1$ и перейти к шагу 5.2.

Шаг 6. Перемешать элементы всех подпопуляций в одну общую популяцию с номером j.

Шаг 7. Если $j = ITER$, процесс завершить, в качестве ответа принять решение x_g. В противном случае, положить $j = j + 1$ и перейти к шагу 3.

Псевдокод *SFLA*

Begin
Input the parameters of the algorithm and initial data
Let Z = the number of memeplexes; Y = the number of frogs in each memeplex; μ = the number of memetic evolutions; Q = submemeplex size; and $M = Y \times Z$
Generate M initial possible frogs (solutions) randomly and evaluate their fitness values
While (the termination criteria are not satisfied)
Sort the frogs according to their fitness values
Divide frogs into Z memeplexes each of which has Y frogs
For s = 1 to Z
For j = 1 to μ
Choose Q frogs from memeplex s randomly and make a submemeplex
Improve the worst frog of the submemeplex (*MWorst*) according to the best frog of the submemeplex (*MBest*) and update its fitness value
If there is no improvement in the worst frog
Improve the worst frog of the submemeplex (*MWorst*) according to the best frog in the population (*PBest*) and update its fitness value
End if
If there is no improvement in the worst frog
Generate a random frog to replace the worst frog of the submemeplex and evaluate its fitness value
End if
Next j
Next s
Combine all memeplexes.
End while
Report the best solution
End

4.10. Модель и алгоритм оптимизации горбатых китов

Алгоритм оптимизации китов, (*Whale Optimization Algorithm, WOA*) был предложен Льюисом и Мирджалили [53]. Он имитирует поведение горбатых китов в поисках пищи и методы их охоты. В том же году Триведи и др. [187] усовершенствовали алгоритм WOA, добавив новую технологию под названием WOA Adaptive Technology.

В 2017 году Ху [188] и др. предложили улучшенный алгоритм оптимизации китов путем добавления его весов инерции (WOAs). Новый алгоритм был протестирован с использованием 27 функций и применен для прогнозирования ежедневного индекса качества воздуха. Предложенный алгоритм показал эффективность по сравнению с другими алгоритмами.

Этот алгоритм использовался для решения задачи в системе IEEE Bus-30 [189], а также для определения оптимального расположения конденсаторов и их размеров в радиальной распределительной сети, что позволило улучшить производительность, стабильность и надежность системы [190], для оптимизации возобновляемых ресурсов с целью снижения потерь в системах распределения электроэнергии [191], для улучшения мужской фертильности [192].

Были разработаны различные гибридные алгоритмы с использованием алгоритма WOA, например алгоритм оптимизации кита с алгоритмом имитации отжига для решения задачи классификации [193], алгоритм оптимизации кита и метод сопряженного градиента [194].

Задача поддержания надлежащего баланса между исследованиями, разведкой и эксплуатацией является очень важной и сложной в биоинспирированных алгоритмах, особенно когда это связано с решением задач стохастической оптимизации. Хоф и Ван дер Гюхт [195] полагают, что киты обладают веретенообразными клетками, которые в точности соответствуют клеткам мозга человека. Эти клетки отвечают за социальное поведение, эмоции и суждения человека. Другими словами, люди отличаются от других существ наличием веретенообразных клеток. Основная причина сообразительности китов заключается в том, что у них больше этих клеток, чем у человека. Киты могут учиться, мыслить, судить, взаимодействовать с окружающей средой и испытывать эмоции, как это делает человек, хотя они конечно менее интеллектуальны. Наблюдения показывают, что киты, особенно косатки, могут вырабатывать

свой особый язык или диалект для эффективного общения и взаимодействия друг с другом.

Киты – самые крупные животные в мире, встречаются киты длиной до 30 метров и весом 180 тонн. В мире существуют разные виды китов, такие как косатки, горбатые киты, синие киты. Некоторые виды китов являются хищниками, киты не спят (у китов спит только половина мозга), потому что для того, чтобы дышать, им необходимо подниматься на поверхность океана.

Еще одной выдающейся особенностью китов является их социальное поведение [53]. Некоторое количество китов живет поодиночке, нок большинство из них объединяются в группы, образуя сообщества. Некоторые виды китов, особенно косатки, могут жить в семье на протяжении всей своей жизни. Горбатые киты, известные как *Megaptera novaeangliae*, являются одними из самых крупных из когда-либо известных усатых китов. Выросший горбатый кит почти такого же размера, как школьный автобус. Как правило, они любят охотиться за крилем или небольшими стаями рыб [53].

Исследования, проведенные в отношении поведения этих видов китов, показывают, насколько они разумны, поскольку действия и поведение горбатых китов были результатом поверхностного наблюдения за категориями китов [196]. Особый интерес представляет особая техника охоты горбатых китов. Хорошо известным охотничьим поведением, свойственным им, является их способ питания, который можно описать как способ кормления пузырьковыми сетями [53, 188]. Горбатые киты используют уникальную стратегию охоты на мелких рыбешек у поверхности воды. Они высматривают косяки рыб или криль [53]. Техника охоты, используемая этими китами, является одним из самых интересных методов поиска пищи.

В ходе исследований было зафиксировано триста случаев кормления с помощью пузырьковой сети у девяти различных горбатых китов. Также были выявлены два типа движения, связанные с пузырьками, которые были названы «восходящими спиралями» и «двойными петлями». В большинстве случаев горбатые киты ныряют на глубину примерно 12 метров после предыдущего движения, а затем начинают по спирали всплывать на поверхность, создавая пузыри вокруг добычи. Этот процесс состоит из трех основных этапов, а именно: петли захвата, коралловой петли и взмаха хвостом.

Необходимо отметить, что способ кормления кита с помощью пузырьковой сети, является особым способом кормления, характерным только для горбатых китов. На рисунке 4.26 показано пищевое поведение горбатых китов с использованием пузырьковой ловушки.

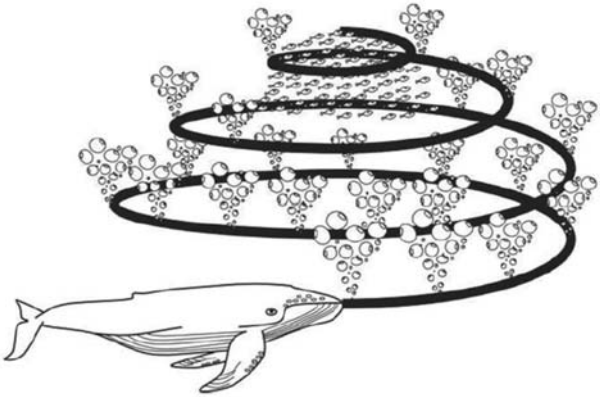


Рис. 4.26. Схема плавания кита, использующего пузырчатую сеть в поисках пищи [188]

Окружение добычи (этап исследования пространства)

Важно определить схему, по которой киты окружают добычу, и разработать на ее основе алгоритма оптимизации. Рассмотрим с этой целью поведение горбатых китов. Одной из особенностей горбатых китов является знание местонахождения добычи и окружение ее в пространстве поиска [53]. Оптимальное расположение не может быть известно заранее. Предполагается, что выбранная добыча является наилучшим возможным или близким к оптимальному решением. Если наилучший шаблон поиска определен, следующей задачей будет попытка других поисковых агентов улучшить свои позиции по отношению к лучшему поисковому агенту.

Следующие уравнения, представленные в работах [53, 197], математически описывают поведение китов:

$$D = |C \cdot X^*(t) - X(t)|, \quad (4.49)$$

$$X(t+1) = X^*(t) - A \cdot D, \quad (4.50)$$

где t – это текущая итерация; A и C – векторы коэффициентов; X^* –лучшее решение с точки зрения текущего вектора местоположения, которое должно присутствовать на всех итерациях, если

данное решение не является предпочтительным; X – текущий вектор местоположения; | | – абсолютное значение.

Параметры A и C описываются выражениями:

$$A = 2a \cdot r_1 - a \quad (4.51)$$

$$C = 2 \cdot r_2 \quad (4.52)$$

В этих уравнениях составляющие a линейно уменьшаются от 2 до 0 на последовательных итерациях, а r_1 и r_2 являются случайными векторами, которые случайным образом принимают значения в интервале [0, 1]. Изменение a показано в виде:

$$a = 2 - 2\frac{t}{t_{\max}}, \quad (4.53)$$

где t_{max} – максимальное количество итераций.

Значение X^* необходимо пересчитывать и обновлять после каждой итерации, т.к. необходимо проверить, получено ли на текущей итерации лучшее решение. Та же идея распространяется и на n-мерное пространство поиска, и, следовательно, поисковые агенты могут перемещаться по гиперкубу вокруг наилучшего ответа, полученного на данный момент.

Атака с помощью пузырьковой сети (этап эксплуатации пространства)

Такое поведение горбатых китов моделируется с помощью следующих двух методов:

а) *механизм сжатия окружности.* При таком поведении значение a уменьшается в уравнении (4.51). При этом диапазон изменения A также уменьшается с уменьшением a. Используя произвольные значения для A в диапазоне [-a, a], который является случайным значением, и что на всех итерациях значение a уменьшается от 2 до 0, при этом значения A находятся в интервале [-1, 1]. То есть последнее местоположение поискового агента находится где-то посередине между первоначальным местоположением поискового агента и местоположением существующего наилучшего поискового агента;

Новым местоположением исследуемого элемента можно считать любое положение между лучшим на данный момент элементом и исходным положением элемента. На рисунке 4.27 показано возможное положение (X, Y) в направлении (X*, Y*), этого можно достичь в пространстве двух осей, установив $0 \le A \le 1$ следующим образом. Это объясняет уменьшение опоясывающего механизма.

б) **местоположение, обновляемое по спирали**. В этом случае для определения местоположения вычисляется расстояние между китом, расположенным в точках (X, Y), и добычей, расположенной в точках (X*, Y*), как показано на рисунке 4.28.

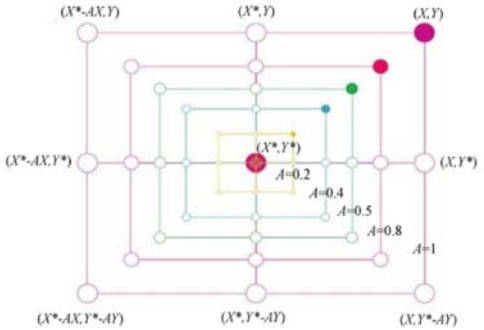


Рис. 4.27. Уменьшение опоясывающего механизма

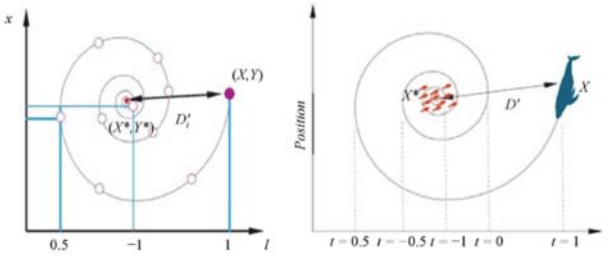


Рис. 4.28. Обновление местоположения по спирали

Движение горбатых китов по спирали представляет собой движение улитки и моделируется с помощью уравнения, учитывающего местоположение кита и добычи:

$$X(t+1) = D' \cdot e^{bl} \cos(2\pi l) + X^*(t) \qquad (4.54)$$

где $D' = |X^*(t) - X(t)|$ – это расстояние между i-м китом и добычей, которое является функцией текущего наилучшего решения; b – это константа, которая определяет контур и кривую логарифмической спирали или процесса скручивания; l – это случайное число в интервале [-1, 1], отражающее покомпонентное увеличение.

Когда горбатые киты нападают на свою жертву, они движутся по спиральной траектории, окружая ее. Горбатые плавают вокруг своей добычи по сужающемуся кругу, имеющему форму спирали. Чтобы

смоделировать это поведение, предполагается, что вероятность выбора для первой (механизма сжатия окружности) и второй (обновление местоположения по спирали) модели равны 0,5:

$$X(t+1) = \begin{cases} X^*(t) - A \cdot D, \text{ если } p < 0.5 \\ D' \cdot e^{bl} \cos(2\pi l) + X^*(t), \text{ если } p \geq 0.5 \end{cases} \quad (4.55)$$

где p – произвольное число в интервале [0, 1].

Модель поведения кита при поиске добычи

Как уже отмечалось выше, у китов, особенно у горбатых китов, есть уникальные способы охоты за добычей. Для моделирования поиска добычи можно использовать такой же метод, основанный на изменении вектора A.

Горбатые киты случайным образом перемещаются в процессе поиска своей добычи в окрестностях своего текущего местоположения. В процессе хищничества некоторые киты. Поэтому мы будем использовать вектор A со значениями $| A | \geq 1$ (больше 1 или меньше 1) заданными случайным образом. Это заставит текущего кита перемещаться дальше от места нахождения остальных китов, чтобы иметь возможность найти больше подходящей пищи. Данное действие подходит для реализации стратегии глобального поиска, поскольку расширяет область поиска для всего алгоритма оптимизации (WOA). Уравнение модели представлено в виде следующей системы уравнений:

$$D = | C \cdot X_{rand} - X | \vec{D} = \left| \vec{C}.\overrightarrow{X_{rand}} - \overrightarrow{X_{(1)}} \right| \quad (4.56)$$

$$X(t+1) = X_{rand} - A \cdot D \; \overrightarrow{X_{(t+1)}} = \overrightarrow{X_{rand}} - \vec{A}.\vec{D} \quad (4.57)$$

где X_{rand} – собой вектор положения текущего кита, произвольно выбранный по отношению к основной популяции китов.

Поскольку X_{rand} представляет собой выбор случайного кита из сообщества, существует несколько возможных вариантов расположения для любого решения с 1 < A, они показаны на рисунке 4.29 [192]. При этом мы стремимся случайным образом улучшить положение текущего агента на этапе исследования пространства поиска, это позволяет алгоритму WOA проводить полное исследование пространства поиска решений.

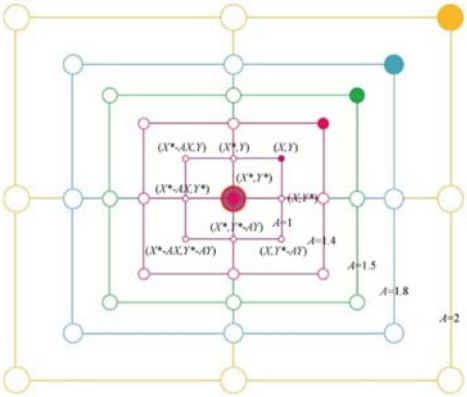


Рис. 4.29. Механизм работы алгоритма WOA на этапе исследования пространства

Алгоритм оптимизации кита

Алгоритм WOA основан на наборе случайных решений, с задания которых начинается каждый процесс. Решения (киты) пытаются улучшить свое текущее местоположение относительно лучшего решения, при $A < 1$, существующего в текущий момент, либо относительно случайно выбранного решения, при $A > 1$ [186].

Движение к лучшему решению может осуществляться либо с использованием движения по спирали, либо кругового движения на основе значения P. Алгоритм завершает работу, в случае выполнения условия остановки.

Блок-схема WOA показана на рис. 4.30 и 4.31.

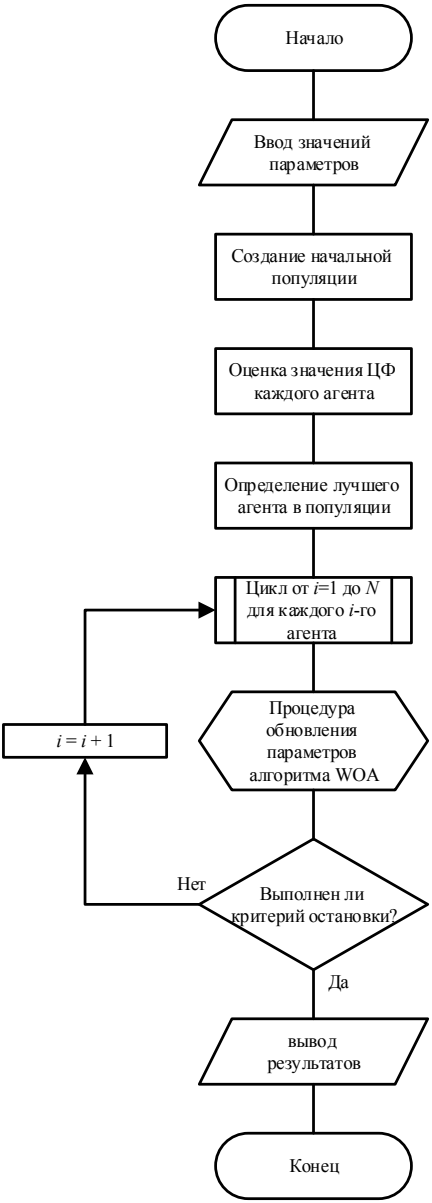


Рис. 4.30. Общая структура алгоритма WOA

Рис. 4.31. Процедура обновления параметров алгоритма WOA

Пример использования алгоритма оптимизации китов

Теоретическая оценка алгоритма *WOA*, позволяет сказать, что это интегрированный алгоритм оптимизации, поскольку он способен выполнять этапы исследования и эксплуатации (использования).

Оценка алгоритм проводилась для задачи минимизации на тестовой функции Химмельблау:

$$f(x_1, x_2) = \min [(x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2]$$

Эта модель была реализована в среде MATLAB с использованием алгоритма *WOA* [197]. Было задано множество из 30 агентов поиска (n = 30), а максимальное число итераций было установлено равным 1000. Наилучшее решение, полученное с помощью *WOA*, равно: [3,0015; -1,9969], а наилучшее оптимальное значение целевой функции, найденное с помощью *WOA*, равно 0,00015167. Пространство параметров и пространство целевой функции показаны на рис. 4.32 и 4.33.

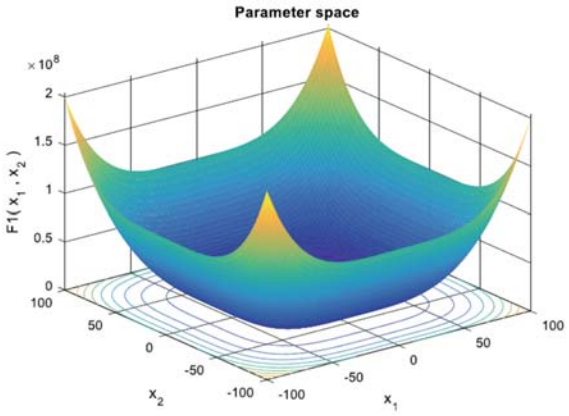


Рис. 4.32. Пространство параметров

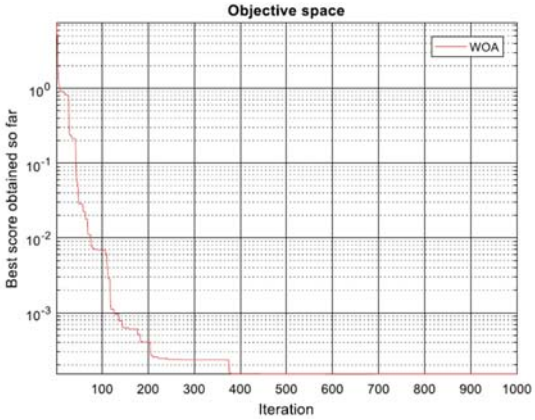


Рис. 4.33. Пространство ЦФ

В алгоритме *WOA* вектор поиска *A* может обновляться в лучшую сторону за счет легкого перехода от задачи исследования к задаче эксплуатации. Отметим, что алгоритм *WOA* имеет только два внутренних ключевых параметра, которые необходимо изменить для перехода к задаче эксплуатации (при $A \geq 1$) – это *A* и *C* [53].

Особенности алгоритма WOA.

1. Алгоритм можно легко реализовать, он достаточно гибкий и не требует большого числа параметров.

2. Имеется возможность свободно перемещаться по пространству решений в задачах исследования и эксплуатации за счет изменения всего лишь одного параметра.

3. Благодаря своим достоинствам и небольшому числу параметров алгоритм используется для решения задач с логарифмической спиральной функцией, охватывающей граничную область в пространстве исследования.

4. Текущее положение индивидов (решений) на этапе исследования улучшается с использованием случайно выбранных решений, а не лучшего из имеющихся на данный момент решений [194].

4.11. Модель и алгоритм оптимизации кузнечиков

Алгоритм оптимизации кузнечиков, (*Grasshopper Optimisation Algorithm, GOA*) был предложен Сареми, Мирджалили и Льюисом [51]. Он имитирует поведение кузнечиков в поисках пищи. Кузне-

чики обычно встречаются на сельскохозяйственных угодьях и в травах и обычно относят их к одним из жевательных травоядных существ или насекомых. Они очень гибкие и обладают вытянутыми задними крошечными лапками, позволяющими прыгать с одного места на другое по сельскохозяйственным угодьям и траве. В некоторых случаях, поскольку они питаются травой, их обычно считают вредителями, поскольку они повреждают посевы на сельскохозяйственных угодьях. На рисунке 4.34 показано графическое изображение кузнечика, а на рисунке 4.35 представлен неполный жизненный цикл метаморфоз кузнечика.

У них есть способ общения между собой, и они могут собираться вместе. Хотя в природе кузнечиков обычно можно видеть по отдельности, они объединяются в одни из самых крупных стай среди всех существ [198]. Способность кузнечиков собираться в рой делает их уникальными травоядными насекомыми. Размер роя может достигать континентальных масштабов и стать кошмаром для фермеров. Когда подобный рой попадает на поля они могут полностью уничтожить урожай на сельскохозяйственных угодьях. Уникальность стаи кузнечиков заключается в том, что роевое поведение наблюдается как у нимф (личинок), так и у взрослых особей [199]. Миллионы личинок кузнечиков прыгают и движутся, как вращающиеся цилиндры. На своем пути они поедают почти всю растительность. После такого поведения, когда они становятся взрослыми, они образуют рой в воздухе. Именно так кузнечики мигрируют на большие расстояния.

Рис. 4.34. Типичный кузнечик

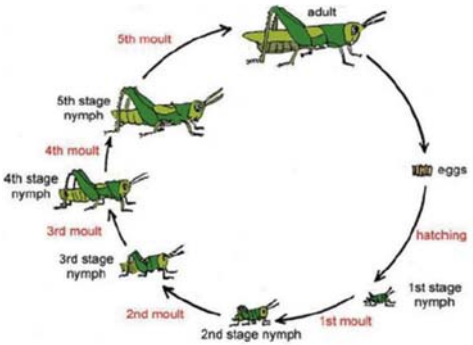


Рис. 4.35. Жизненный цикл кузнечиков с неполным превращением

Основная характеристика стаи кузнечиков в фазе личинок – медленное движение и мелкие шаги кузнечиков. Напротив, дальние и резкие движения являются важной особенностью стаи взрослых особей. Поиск источника пищи – еще одна важная особенность роя кузнечиков. Как отмечалось ранее, вдохновленные природой алгоритмы логически делят процесс поиска на два этапа: исследование и эксплуатацию. При разведке (исследовании) новых областей поисковым агентам рекомендуется двигаться резко, в то время как во время эксплуатации они склонны перемещаться в окрестностях локального участка пространства решений. Эти две функции, а также поиск цели выполняются кузнечиками естественным путем. Следовательно, если мы найдем способ математически смоделировать такое поведение, мы получим еще один алгоритм оптимизации, вдохновленный природой [51, 199].

Когда кузнечик движется к своей цели, он выполняет два типа движений. Чтобы разработать алгоритм кузнечика для решения реальной задачи, важно внимательно наблюдать и имитировать схему навигации травоядного насекомого.

Алгоритм кузнечика, модель, обозначения и формулировка

Для обозначения параметров математической модели используются следующие переменные:

- X_i – местоположение или положение i-го кузнечика;
- S_i – взаимодействие социальных насекомых;
- G_i – сила тяжести i-го кузнечика во время навигации;
- A_i – адвекция ветра;

• d_{ij} – евклидово местоположение или расстояние между двумя заданными кузнечиками;

• f – интенсивность притяжения;

• l – привлекательная шкала длины;

• ub_d; lb_d – верхняя и нижняя граница в d-м измерении соответственно;

• c – понижающий коэффициент для сокращения зон или окрестностей с точки зрения комфорта, отталкивания и притяжения.

• c_{max}; c_{min} – максимальное и минимальное значение коэффициента c соответственно;

• lc – текущая итерация;

• L – максимальное количество итераций.

Математическая модель, используемая для моделирования роевого поведения кузнечиков, представлена следующим образом [51, 199]:

$$X_i = S_i + G_i + A_i, \tag{4.58}$$

где X_i определяется положением i-го кузнечика, S_i – социальное взаимодействие, G_i – сила тяжести, действующая на i-го кузнечика, а A_i – адвекция ветра.

Для обеспечения случайного поведения уравнение можно записать в следующем виде: $X_i = r_1 S_i + r_2 G_i + r_3 A_i$, где r_1, r_2 и r_3 – случайные числа в интервале [0, 1].

Функция S_i, определяющая социальное взаимодействие, моделируется с учетом вышеупомянутых компонентов системы (воздействие гравитационной силы и адвекция ветра) следующим образом:

$$S_i = \sum_{j=1, j\neq i}^{N} s(d_{ij})\hat{d}_{ij} \tag{4.59}$$

где расстояние d_{ij} между i-м и j-м кузнечиком, которое может быть рассчитано как Евклидово расстояние с помощью формулы $d_{ij} = |x_j - x_i|$, единичный вектор $\hat{d}_{ij} = \frac{x_j - x_i}{d_{ij}}$ и $s(r)$ – функция, определяющая силу социального взаимодействия, вычисляется следующим образом [51, 200]:

$$s(r) = fe^{\frac{-r}{l}} - e^{-r} \tag{4.60}$$

где f и l – сила притяжения и масштаб длины притяжения соответственно.

Функция s проиллюстрирована на рис. 4.36, чтобы показать, как она влияет на социальное взаимодействие (взаимное привлечение и отталкивание) кузнечиков. Как видно из рис. 4.36, существующее пространство комфорта (зона внутри окружности) разбито на три

зоны – «до», «внутри» и «после». Анализ сил, обусловленных расположением кузнечиков, проводится путем сравнения с наиболее близкими кузнечиками в рое. При этом существуют силы притяжения, отталкивания и нейтральная силы. Наглядная иллюстрация действия функции *s* представлена на рис. 4.37, где показана концептуальная модель зоны комфорта, зоны притяжения и зоны отталкивания сближающихся (роящихся) кузнечиков.

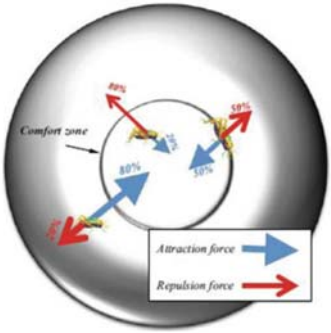


Рис. 4.36. Зоны комфорта, притяжения и отталкивания в стае кузнечиков [1]

На этом рисунке видно, что рассматриваются расстояния от 0 до 1.5. Отталкивание происходит в интервале [0 2,079]. Когда кузнечик находится на расстоянии 2,079 единиц от другого кузнечика, нет ни притяжения, ни отталкивания. Это называется зоной комфорта или комфортной дистанцией.

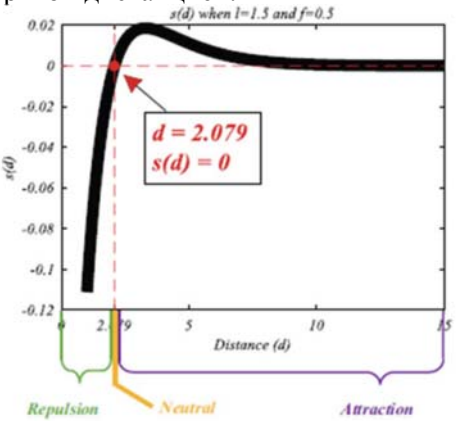


Рис. 4.37. Кривая, связанная с уравнением [51]

На рисунке также видно, что притяжение увеличивается с 2,079 единицы расстояния почти до 4, а затем постепенно уменьшается. Исходя их полученных данных, можно сделать вывод, что травоядное насекомое (кузнечик) будет сохранять нейтральное положение без приложения какой-либо силы, сохраняя при этом расстояние 2,079. При постепенном смещении движения кузнечика в сторону точки выгоды насекомое сталкивается с дополнительной силой притяжения, пока не занимает положение около 5,0. На этом уровне происходит уменьшение величины силы притяжения в результате увеличения расстояния. В алгоритме оптимизации кузнечика (GOA) предполагается, что функциональный диапазон обычно находится в интервале от 0 до 3,3. Функциональный диапазон имеет определенный уровень ограничений, поскольку силы недостаточно сильны в ситуациях, когда кузнечики разделены большими расстояниями. Самый простой способ преодолеть такое ограничение – нормализовать расстояние между соответствующими кузнечиками [51, 199]. Функция, представленная в уравнении 10.3, была разработана для работы с двумя ключевыми параметрами, которые служат для управления, а именно l и f. Изменение параметров l и f в уравнении (4.54) приводит к различному социальному поведению искусственных кузнечиков. Чтобы увидеть влияние этих двух параметров на рис. 4.38 приведен график изменения функции s при изменении значений переменных l и f. На этом рисунке видно, что параметры l и f существенно меняют зону комфорта, область притяжения и область отталкивания. График показывает, что в диапазоне $1 \leq d < 2{,}0790$ существует тенденция отталкивания, а в диапазоне значений $2{,}0790 \leq d$ существует вероятность сильного притяжения. Следует отметить, что области притяжения или отталкивания очень малы для некоторых значений (например, $l = 1{,}0$ или $f = 1{,}0$). Из всех вариантов значений наилучший результат был получен для следующих значений: $l = 1{,}5$ и $f = 0{,}5$. На рисунке 4.38 показаны кривые, полученные при изменении параметров функции.

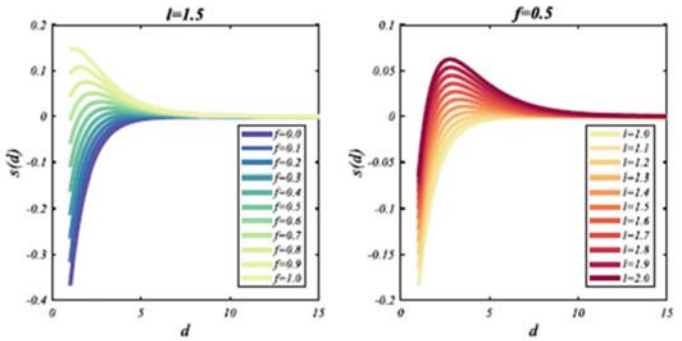


Рис. 4.38. Поведение функции *s* при изменении параметров *l* и *f*

Концептуальная модель взаимодействия между кузнечиками и зоной комфорта с использованием функции *s* представлена на рис. 4.39. Можно отметить, что в упрощенной форме это социальное взаимодействие было движущей силой в некоторых ранних моделях роя саранчи [198].

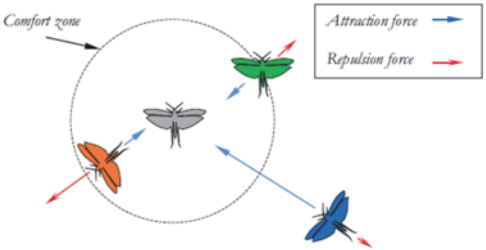


Рис. 4.39. Примитивные корректирующие шаблоны между особями в стае кузнечиков

Хотя функция *s* способна разделить пространство между двумя кузнечиками на область отталкивания, область комфорта и область притяжения, эта функция возвращает значения, близкие к нулю, при расстояниях больше 10. Следовательно, эта функция не может оперировать большими значениями сил взаимодействия между кузнечиками при больших расстояниях между ними. Для решения этой проблемы на карту нанесли расстояние между кузнечиками на интервале [1, 4]. Вид функции *s* в этом интервале показан на рис. 4.40.

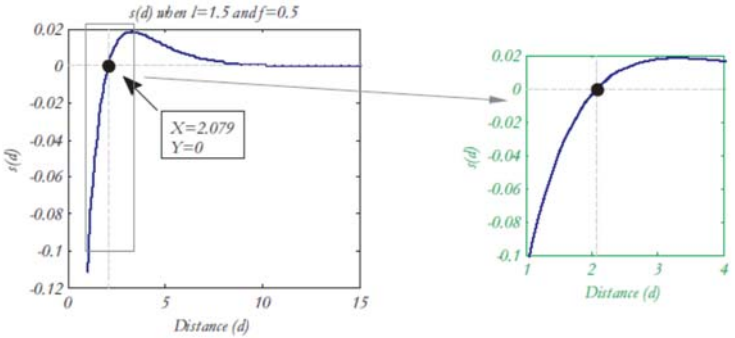


Рис. 4.40. Функции при $l = 1{,}5$ и $f = 0{,}5$(слева); диапазон изменения функции s, при x в интервале [1, 4] (справа)

Компонент G_i из уравнения (4.58) рассчитывается следующим образом:

$$G_i = -g\hat{e}_g, \tag{4.61}$$

где g – гравитационная постоянная, а $\hat{e}_g$ – это единичный вектор, направленный к центру Земли.

Компонент A_i в уравнении. (4.58) рассчитывается следующим образом:

$$A_i = u\hat{e}_w, \tag{4.62}$$

где u – постоянный снос, а $\hat{e}_w$ – единичный вектор направления ветра.

У личинок кузнечиков (нимф) нет крыльев, поэтому их движения тесно связаны с направлением ветра. Подставив выражения для S, G и A в уравнение (4.58), мы можем расширить это уравнение следующим образом:

$$X_i = \sum_{j=1, j\neq i}^{N} s\left(\left|x_j - x_i\right|\right)\frac{x_j - x_i}{d_{ij}} - g\hat{e}_g + u\hat{e}_w, \tag{4.63}$$

где $s(r) = fe^{\frac{-r}{l}} - e^{-r}$ и N — количество кузнечиков.

Поскольку личинки кузнечиков приземляются на землю, их положение не должно опускаться ниже порога. Однако мы не будем использовать это уравнение в алгоритме роевого моделирования и оптимизации, поскольку оно не позволяет алгоритму исследовать и использовать пространство поиска вокруг решения. Фактически, модель, используемая для роя, находится в свободном пространстве. Следовательно, уравнение. (4.63) может использоваться для

моделирования взаимодействия кузнечиков в стае. Эксперименты показали, что уравнение (4.63) сближает исходную случайную популяцию до тех пор, пока они не образуют единый регулируемый рой. По прошествии 10 единиц времени все кузнечики достигают зоны комфорта и больше не двигаются. Это показывает, что данная математическая модель способна имитировать рой кузнечиков в различных типах пространства (2D, 3D и гиперпространство).

Однако эту математическую модель нельзя использовать непосредственно для решения оптимизационных задач главным образом потому, что кузнечики быстро достигают зоны комфорта, а рой не сходится к заданной точке.

Сареми и др. определили параметры модели, показывающей процесс исследования кузнечиков и процесс эксплуатации для получения глобального оптимума с использованием метаэвристического алгоритма [51]. Они разработали математическую модель для решения задач оптимизации, описывающую обновление положения кузнечика, которая представляет собой модифицированную версию уравнения (4.63):

$$X_i^d = c\left(\sum_{j=1, j\neq i}^{N} c\frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j - x_i}{d_{ij}}\right) + \hat{T}_d \qquad (4.64)$$

где ub_d и lb_d – верхняя и нижняя границы D-го измерения, $s(r) = fe^{\frac{-r}{l}} - e^{-r}$, $\hat{T}_d$ – значение D-го измерения в точке где находится цель (лучшее решение, найденное на данный момент), а c – уменьшающий коэффициент для сокращения зоны комфорта, зоны отталкивания и зоны притяжения.

Переменная s почти аналогична компоненту S в уравнении (4.58). Однако в этом случае мы не учитываем гравитацию (без компонента G) и предполагаем, что направление ветра (компонент A) всегда направлено к цели $\hat{T}_d$.

Уравнение (4.64) показывает, что следующая позиция кузнечика определяется на основе его текущего положения, положения цели и положения всех остальных кузнечиков. Необходимо учитывать, что первый компонент этого уравнения учитывает расположение текущего кузнечика относительно других кузнечиков. Фактически, мы учли статус всех кузнечиков, чтобы определить местоположение поисковых агентов вокруг цели. В этом состоит отличие алгоритма *GOA*

от *PSO* как наиболее известного метода роевого интеллекта. В *PSO* для каждой частицы есть два вектора: вектор положения и вектор скорости. В алгоритме *GOA* для каждого поискового агента существует только один вектор позиции. Другое различие между этими двумя алгоритмами заключается в том, что *PSO* обновляет положение частиц относительно текущего положения, личного рекорда и глобального рекорда. А *GOA* обновляет позицию поискового агента на основе его текущей позиции, лучшего решения в популяции и позиции всех других поисковых агентов. Таким образом, в *PSO* ни одна из частиц не способствует обновлению положения других частиц, тогда как в *GOA* необходимо, чтобы все поисковые агенты участвовали в определении следующей позиции каждого агента.

Отметим, что адаптивный параметр c дважды использовался в уравнении (4.64) по следующим причинам:

- первый параметр c (слева) очень похож на инерционный вес w в *PSO*. Он уменьшает перемещения кузнечиков вокруг цели. Другими словами, этот параметр уравновешивает задачи разведки и эксплуатации для всего роя в области, расположенной вокруг цели;
- второй параметр c уменьшает зону притяжения, зону комфорта и зону отталкивания между кузнечиками. Рассматривая компонент $c\frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)$ в уравнении (4.64), элемент $c\frac{ub_d - lb_d}{2}$ линейно уменьшает пространство, которое кузнечики должны исследовать и использовать. Компонент $s(|x_j - x_i|)$ определяет, следует ли отпугнуть кузнечика (от исследования) или привлечь (эксплуатация) к цели.

Следует отметить, что второй (внутренний) c способствует уменьшению сил отталкивания/притяжения между кузнечиками пропорционально количеству итераций, тогда как первый (внешний) c уменьшает охват поиска вокруг цели по мере увеличения количества итераций.

Таким образом, первый член уравнения (4.64), сумма, позволяет учитывать положение других кузнечиков и реализует взаимодействие кузнечиков в стае. Второй член уравнения $\hat{T}_d$, позволяет моделировать стремление кузнечиков продвигаться к источнику пищи. Параметр c моделирует замедление кузнечиков, приближающихся к источнику пищи и в конечном итоге поедающих ее. Чтобы обеспечить более случайное поведение, и в качестве альтер-

нативы оба параметра в уравнении (4.58) можно умножить на случайные величины. Кроме того, отдельные члены выражения также могут быть умножены на случайные значения, чтобы обеспечить случайное поведение при взаимодействии кузнечиков, либо при поиске и движении к источнику пищи.

Предложенные математические формулы должны позволить исследовать и использовать пространство поиска. Однако должен существовать механизм, требующий от поисковых агентов настройки уровня разведки на эксплуатацию. В природе кузнечики сначала передвигаются и ищут пищу локально в ближайших окрестностях своего расположения, поскольку у личинок у них нет крыльев. Затем они свободно перемещаются в воздухе и исследуют более обширную область. Однако в алгоритмах стохастической оптимизации задача исследования стоит на первом месте из-за необходимости поиска перспективных областей пространства поиска. После обнаружения перспективных регионов эксплуатация вынуждает поисковых агентов вести локальный поиск, чтобы найти точное приближение к глобальному оптимуму.

Для поддержания баланса между задачами разведки и эксплуатации параметр c необходимо уменьшать пропорционально количеству итераций. Счетчик итераций увеличивается по мере того, как внешний c уменьшает зону исследования вокруг цели $\hat{T}_d$. Этот механизм способствует расширению области эксплуатации по мере увеличения количества итераций. Внутренний c способствует уменьшению величины сил притяжения или отталкивания между сближающимися кузнечиками. Его величина изменяется пропорционально количеству итераций. Ожидается, что решения модели будут сходиться и расходиться в сторону лучшего глобального решения. Системный параметр c должен быть реструктурирован или изменен, чтобы обеспечить увеличение добычи и сокращение разведки пропорционально увеличению числа итераций. Коэффициент c уменьшает зону комфорта пропорционально количеству итераций и рассчитывается следующим образом:

$$c = c_{M\,\text{ax}} - l\frac{c_{Max} - c_{Min}}{L} \tag{4.65}$$

где c_{Max}, c_{Min} – максимальное значение и минимальное значение соответственно, l – текущая итерация, а L – максимальное количество итераций. В работе [51] для c_{max} и c_{min} были использованы значения 1 и 0,00001 соответственно.

В работе [51] были проведены эксперименты как на стационарных, так и на мобильных целях, чтобы увидеть, как рой движется к ним и ведет их поиск. Из результатов видно, что рой постепенно сходится к неподвижной цели как в 2D, так и в 3D пространствах. Такое поведение обусловлено сокращением зоны комфорта за счет фактора *c*. Также видно, что рой правильно преследует и мобильную цель. Это связано с последним компонентом уравнения. (4.63) $\hat{T}_d$, в котором кузнечики притягиваются к цели. Можно отметить интересную закономерность – постепенное приближение кузнечиков к цели на каждой итерации связано с уменьшением коэффициента *c*. Такое поведение помогает алгоритму *GOA* не слишком быстро сходиться к цели и, следовательно, не попасть в ловушку локального оптимума. Однако на последних этапах оптимизации кузнечики будут максимально приближаться к цели, что очень важно при эксплуатации.

Приведенные результаты показывают, что предложенная математическая модель требует, чтобы кузнечики постепенно двигались к цели в ходе выполнения итераций. Однако в реальном пространстве поиска цели нет, поскольку мы не знаем точно, где находится главная цель – глобальный оптимум. Поэтому нам приходится задавать цель для кузнечиков на каждом этапе оптимизации. В *GOA* предполагается, что во время оптимизации целью является наиболее приспособленный кузнечик (тот, у кого наилучшее целевое значение). Это помогает *GOA* сохранить наиболее перспективную цель в пространстве поиска на каждой итерации и требует от кузнечиков движения к ней. Целью работы алгоритма является нахождение лучшего значения (лучшее приближение) к глобальному оптимуму в пространстве поиска.

Псевдокод алгоритма *GOA* показан на рис. 4.41. *GOA* начинает оптимизацию с создания набора случайных решений. Поисковые агенты обновляют свои позиции на основе уравнения. (4.64). Положение лучшей из известных на данный момент целей обновляется на каждой итерации. Коэффициент *c* рассчитывается по формуле (4.65) и расстояния между кузнечиками нормируются в интервале [1, 4] на каждой итерации. Обновление позиции выполняется итеративно до тех пор, пока не будет достигнут конечный критерий. В конечном итоге положение и значение функции приспособленности лучшей цели выводятся алгоритмом как наилучшее приближение к глобальному оптимуму.

```
Initialize the swarm Xi (i = 1, 2, ..., n)
Initialize cmax, cmin, and maximum number of iterations
Calculate the fitness of each search agent
T=the best search agent
while (l < Max number of iterations)
    Update c using Eq. (2.8)
    for each search agent
            Normalize the distances between grasshoppers in [1,4]
            Update the position of the current search agent by the equation (2.7)
            Bring the current search agent back if it goes outside the boundaries
    end for
    Update T if there is a better solution
    l=l+1
end while
Return T
```

Рис. 4.41. Псевдокод алгоритма *GOA*

Область применения алгоритма *GOA*

Достоинства алгоритма оптимизации кузнечика были продемонстрированы при решении реальных комбинаторных задач. Разработанный метаэвристический алгоритм *GOA* может использоваться в различных областях деятельности для планирования, оценки, моделирования и управления, в том числе:

- решение транспортных задач;
- дизайн и оптимизация макета;
- задачи, связанные с планированием и инвентаризацией;
- структурная оптимизация;
- оптимизация мощности потока;
- проблема составления временного графика использования оборудования;
- оптимизация ретрансляционных узлов в беспроводных сенсорных сетях;
- регулирование частоты нагрузки в энергосистемах и многое другое.

Пример применения алгоритма оптимизации кузнечика

Для тестирования использовалась задача минимизации функции Букина

$$f(x,y)=100\sqrt{\left|y-0.01x^2\right|}+0.01\left|x+10\right|$$

Эта модель была реализована в MATLAB с использованием алгоритма оптимизации кузнечика (GOA), предложенного Мирджалили [51]. Для иллюстрации подхода было рассмотрено 100 поисковых агентов (n = 100) и максимальное количество итераций было установлено равным 1000. Наилучшее решение, полученное с помощью GOA: [10 1] и лучшее оптимальное значение целевой функции, найденное с помощью GOA. составляет 0,2361. Пространство параметров, история испытаний, траектория первого кузнечика и тест сходимости показаны на рис. 4.42, 4.43, 4.44 и 4.45 соответственно.

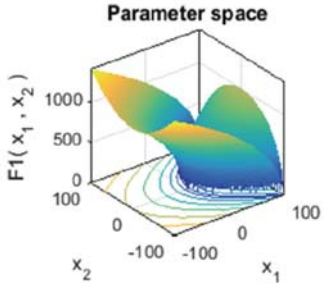


Рис. 4.42. Пространство параметров

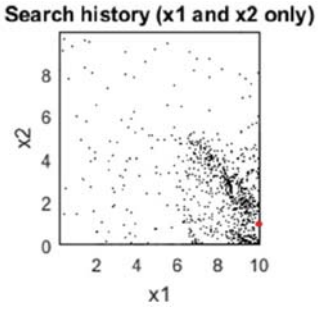


Рис. 4.43. Иллюстрация истории поиска

Рис. 4.44. Иллюстрация траектории полета первого кузнечика

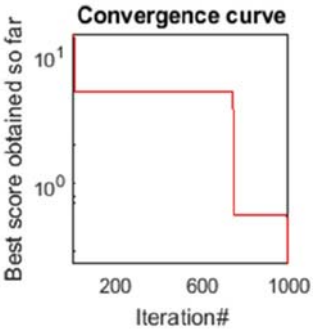


Рис. 4.45. Кривая сходимости

4.12. Модель и алгоритм оптимизации «мотылек – пламя»

Алгоритм оптимизации «мотылек – пламя», (*Moth-Flame Optimization Algorithm, MFOA*) был предложен Мирджалили [202] в 2015 году. Данная модель имитирует технику движения бабочек в ночное время, называемую поперечной ориентацией для навигации. Мотыльки летают ночью в зависимости от лунного света, сохраняя при этом фиксированный угол, чтобы найти свой путь. Было решено взять такое поведение бабочек за основу и предложен новый метод оптимизации.

MFOA сочетает в себе популяционный алгоритм и стратегию локального поиска, благодаря чему он способен к решению задач глобального исследования и локальной эксплуатации [203]. Подобно другим метаэвристикам, *MFOA* прост, гибок и легко реализуется, и может быть использован для решения широкого круга задач [204]. Благодаря этим достоинствам MFO успешно применялся

для решения различных задач оптимизации, например, планирования [205], обратной задачи и оценки параметров [206, 207], классификации [208], экономических [209], медицинских [210], энергетических энергия [211] и обработка изображений [212].

Мотыльки – удивительные насекомые, которые имеют внешнее сходство с широко распространенным семейством бабочек. Исследования показали, что в природе существует более 160 000 разновидностей мотыльков, которые были идентифицированы и подтверждены в литературе. Одной из интересных особенностей этих насекомых является их жизненный цикл. Жизненный цикл бабочек состоит из двух этапов: первый этап – стадия личинки, а второй – стадия взрослой особи. Личинки превращаются в мотыльков с помощью коконов. То есть в жизни мотылька имеет место процесс трансформации, который включает в себя последовательность процессов до момента достижения стадии взросления, на которой формируется взрослая особь. Жизненный цикл мотылька представлен на рис. 4.46.

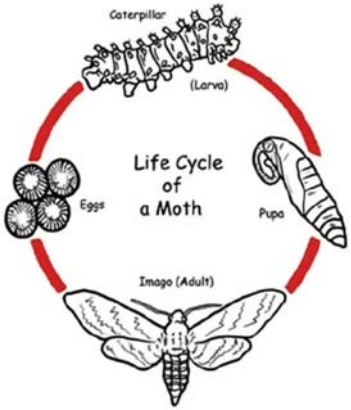


Рис. 4.46. Жизненный цикл мотылька

С практической точки зрения самым интересным в жизни мотыльков является их уникальный метод ночной навигации. Они были созданы для того, чтобы использовать лунный свет для полета по темным окрестностям. Для навигации они использовали механизм, называемый поперечной ориентацией. Такой метод ориентации позволяет мотыльку летать, сохраняя постоянный угол наклона по отношению к Луне в процессе навигации, при этом мотылек сохраняет при движе-

нии прямую траекторию. Это делает схему навигации и коммуникации уникальным и эффективным механизмом для перемещения на большие расстояния по прямой траектории [213, 214].

На рис. 4.47 показана концептуальная модель поперечной ориентации. Поскольку Луна находится далеко от мотылька, этот механизм гарантирует полет по прямой. Тот же метод навигации может быть использован людьми. Предположим, что Луна находится на южной стороне неба, а человек хочет отправиться на восток. Если при ходьбе он следить за тем, чтобы Луна все время оставалась слева, он сможет двигаться на восток по прямой линии.

Несмотря на эффективность модели поперечной ориентации, в природе мы наблюдаем, что мотыльки летают вокруг источников света по спирали. На самом деле, искусственное освещение обманывает мотыльков, и поэтому они ведут себя именно так. Это с недостатком модели поперечной ориентации, который делает эту модель полезной для ориентации при движении по прямой, только в том случае, когда источник света находится достаточно далеко. Когда мотыльки видят искусственный свет, созданный человеком, они стараются сохранить одинаковый угол со светом, чтобы лететь по прямой. Однако, поскольку такой свет находится очень близко по сравнению с Луной, поддержание аналогичного угла к источнику света приводит к бесполезной и смертельной спиралевидной траектории полета бабочек [215]. Концептуальная модель такого поведения показана на рис. 4.48. Как мы видим в результате такого полета бабочка в конечном итоге приближается к свету.

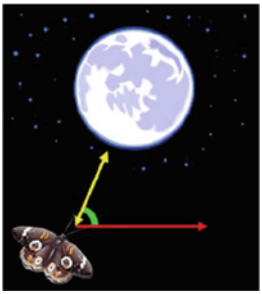


Рис. 4.47. Поперечная ориентация, демонстрируемая мотыльком во время полета [202]

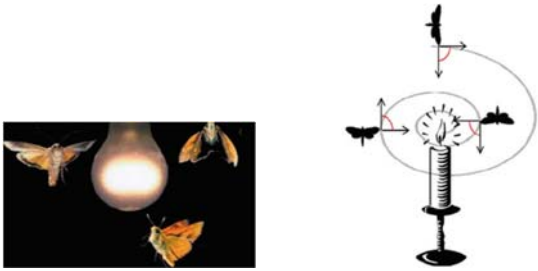


Рис. 4.48. (а) – полет мотылька вокруг источника света;
(б) – траектория полета вокруг источника света [205]

Алгоритм MFOA

Алгоритм оптимизации «Мотылек – пламя» (MFOA) – это простой и понятный в реализации метаэвристический алгоритм, который представляет собой математическую модель данного поведения. Он был разработан Мирджалили [202]. Метод оптимизации «мотылек-пламя», как утверждают Шехаб и др. [203], состоит из двух основных этапов. Первый – это качественный этап, а второй - количественный. Алгоритм «Мотылек – пламя» – это очень надежный алгоритм для решения задач оптимизации, хотя у него есть свои ограничения, которые связаны с процессом определения целевой функции на первом этапе и разработкой точных ограничений для соответствия заданной цели (целям). Баланс между знаками равенства и неравенства при создании ограничений очень важен в процессе оптимизации «Мотылек – пламя».

Решение задачи с использованием алгоритма оптимизации «Мотылек – пламя» может быть получено в процессе построения пути для мотыльков. Здесь следует отметить, что и мотыльки, и пламя являются решением проблемы. Разница между ними заключается в том, как мы их обрабатываем и обновляем на каждой итерации. В процессе поиска в окрестностях каждый мотылек является поисковым агентом, перемещающимся по пространству в процессе поиска, в то время как пламя – это наилучшее местоположение мотыльков, полученное на данный момент.

Мотыльки – это настоящие поисковые агенты, которые перемещаются по пространству поиска, тогда как пламя – лучшая позиция для мотыльков, которая может быть получена на данный момент. Другими словами, пламя можно рассматривать как флажки или бу-

лавки, которые мотыльки бросают при поиске в пространстве поиска. Поэтому каждый мотылек ищет флаг (пламя) и обновляет его в случае нахождения лучшего решения. Благодаря этому механизму мотылек никогда не потеряет свое лучшее решение.

В описываемом алгоритме MFOA [202] предполагается, что мотыльки являются кандидатами на решение, а переменными задачи являются положение мотыльков в пространстве. Следовательно, мотыльки могут летать в одномерном, двухмерном, трехмерном или гипермерном пространстве, меняя векторы своего положения. Первым шагом является случайная генерация мотыльков в пределах окрестности или пространства решений. Поскольку алгоритм MFOA является популяционным алгоритмом, мы представляем набор мотыльков в матричной форме следующим образом [202]:

$$M_{MFO} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,d} \\ \cdots & \cdots & \ddots & \cdots \\ m_{n,1} & m_{n,1} & \cdots & m_{n,d} \end{bmatrix} \quad (4.66)$$

где n – количество мотыльков в пространстве поиска; d – количество переменных (размерность задачи).

Затем вычисляется значение функции пригодности для каждого мотылька, и наилучшая из полученных позиций помечается меткой *flame* (пламя). Мы также предполагаем, что для всех мотыльков существует матрица для хранения соответствующих значений функции [202]:

$$OM_{MFO} = \begin{bmatrix} OM_{1,1} \\ OM_{2,1} \\ \vdots \\ OM_{n,1} \end{bmatrix} \quad (4.67)$$

где n – количество бабочек.

Отметим, что значение функции пригодности – это возвращаемое значение функции приспособленности (целевой) для каждого мотылька. Вектор положения (например, первая строка в матрице M) каждого мотылька передается в функцию пригодности, а выходные данные функции пригодности присваиваются соответствующему мотыльку в качестве его функции приспособленности (например, OM_1 в матрице OM).

Еще одним ключевым компонентом предлагаемого алгоритма является пламя. Рассмотрим матрицу, аналогичную матрице мотыльков, следующим образом:

Поскольку представлена матрица мотыльков, для сбалансированности алгоритма не менее важно представить матрицу пламени,

состоящую из того же количества строк и столбцов, что и матрица мотыльков, представленная в уравнении 4.66. Это представлено в уравнении 4.68 [202].

$$F_{MFO} = \begin{bmatrix} F_{1,1} & F_{1,2} & \cdots & F_{1,d} \\ F_{2,1} & F_{2,2} & \cdots & F_{2,d} \\ \cdots & \cdots & \ddots & \cdots \\ F_{n,1} & F_{n,1} & \cdots & F_{n,d} \end{bmatrix} \quad (4.68)$$

где n – количество бабочек, а d – количество переменных (размерность).

Затем происходит процесс обновления, после чего процесс будет повторяться до тех пор, пока не будут достигнуты критерии для завершения процесса.

Не менее важно подобрать размеры матриц значений функции пригодности для мотылька и пламени. Из уравнения (4.68) видно, что размерность массивов M и F равна. Для пламени мы также предполагаем, что существует массив для хранения соответствующих значений пригодности следующим образом [202]:

$$OF_{MFO} = \begin{bmatrix} OF_{1,1} \\ OF_{2,1} \\ \vdots \\ OF_{n,1} \end{bmatrix} \quad (4.69)$$

где n – количество мотыльков.

Математическая формулировка задачи в алгоритме *MFOA* состоит из трех кортежей, которые аппроксимируют глобальный оптимум задач оптимизации и записывается следующим образом [202]:

$$MFOA = (I, P, T) \quad (4.70)$$

где I – функция, которая генерирует случайную популяцию мотыльков при поиске в окрестностях текущей позиции и соответствующие значения пригодности.

Методическая модель этой функции выглядит следующим образом [202]:

$$I: \varnothing \rightarrow \{M, OM\} \quad (4.71)$$

Функция P, которая является основной функцией, задает перемещение мотыльков по пространству поиска. Эта функция получает на входе значения из исходной матрицы M и, в конечном итоге возвращает ее обновленную [202]:

$$P: M \rightarrow M \quad (4.72)$$

Функция *T* определяет уровень выполнения условия завершения в системе. Функция *T* возвращает значение *true*, если критерий завершения удовлетворен, и *false*, если критерий завершения не удовлетворен [202]:

$$T \rightarrow M\{true, false\} \quad (4.73)$$

При использовании функций *I*, *P* и *T* общая структура алгоритма *MFOA* определяется следующим образом:

M=I();
while T(M) is equal to false
M=P(M);
end
The function I has to generate initial solutions and calculate the objective function values. Any random distribution can be used in this function. What we implement is as follows:
for i = 1 : n
for j= 1 : d
M(i,j)=(ub(i)-lb(i)) rand()+lb(i);*
end
end
OM=FitnessFunction(M);

Важно отметить, что общая структура *MFOA* может быть использована с использованием любой из трех функций *P*, *T* и *I*. Как можно видеть, есть еще два массива, называемые *ub* и *lb*. Эти матрицы определяют верхнюю и нижнюю границы переменных следующим образом [202]:

$$ub = [ub_1, ub_2, ..., ub_{n-1}, ub_n] \quad (4.74)$$

$$lb = [lb_1, lb_2, ..., lb_{n-1}, ub_n] \quad (4.75)$$

где ub_i указывает верхнюю границу *i*-й переменной, а lb_i – нижнюю границу *i*-й переменной.

После инициализации функция *P* выполняется итеративно до тех пор, пока функция *T* не вернет значение *true*. Функция *P* – это основная функция, которая перемещает мотыльков по пространству поиска. Как упоминалось выше, основой этого алгоритма является поперечная ориентация. Чтобы математически смоделировать это поведение, мы обновляем положение каждого мотылька относительно пламени, используя следующее уравнение [202]:

$$M_i = S(M_i, F_j) \quad (4.76)$$

где *S* – спиральная функция, M_i – положение $\underline{i}$-го мотылька, а F_j – *j*-е пламя.

В качестве основного механизма обновления бабочек была выбрана логарифмическая спираль. Однако здесь можно использовать любые типы спиралей при соблюдении следующих условий:

- начальная точка спирали должна начинаться с мотылька;
- конечная точка спирали должна соответствовать положению пламени;
- колебание дальности спирали не должно выходить за пределы пространства поиска.

Учитывая эти моменты, мы определяем логарифмическую спираль для алгоритма *MFOA* следующим образом [202]:

$$S(M_i, F_j) = D_i E^{bt}\cos(2\pi t) + F_j \quad (4.77)$$

где D_i представляет собой возможное расстояние i-го мотылька до j-го пламени, b – константа, которая определяет форму логарифмической спирали, а t – случайное число в диапазоне [-1, 1]. D рассчитывается следующим образом [202]:

$$D_i = |F_j - M_i| \quad (4.78)$$

где M_i – i-й мотылек, F_j – j-е пламя, а D_i – расстояние от i-го мотылька до j-го пламени.

Уравнение (4.78) моделирует спиральную траекторию полета мотылька. Как видно из этого уравнения, следующее положение мотылька определяется относительно пламени. Параметр t в спиральном уравнении определяет, насколько следующее положение мотылька должно быть близко к пламени (t = -1 – самое близкое положение к пламени, а t = 1 – самое дальнее). Следовательно, вокруг пламени во всех направлениях можно предположить гиперэллипс, и следующее положение бабочки будет внутри этого пространства. Спиральное движение является основным компонентом предлагаемого метода, поскольку оно определяет, как мотыльки меняют свое положение относительно огня. Уравнение спирали позволяет мотыльку летать «вокруг» пламени, а не обязательно в пространстве между ними. Таким образом, обеспечивается выполнение задач исследования и эксплуатации пространства поиска. Логарифмическая спираль, пространство вокруг пламени и возможное положение мотылька на кривой с учетом различных t показаны на рис. 4.49.

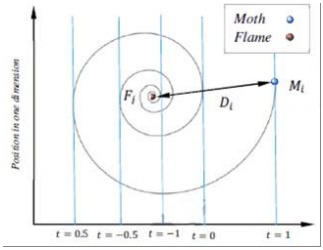


Рис. 4.49. Логарифмическая спираль, пространство вокруг пламени и положение относительно *t*

На рис. 4.50 показана концептуальная модель изменения положения мотылька относительно пламени. Отметим, что хотя вертикальная ось показывает только одно измерение (1 переменную задачи), тем не менее, рассматриваемый метод можно использовать для изменения всех переменных задачи. Возможные положения (черные пунктирные линии на рис. 4.50), которые можно выбрать в качестве следующего положения мотылька (синяя горизонтальная линия) вокруг пламени (зеленая горизонтальная линия), показывают, что мотылек может исследовать и использовать пространство поиска вокруг пламени в одном измерении. Исследование пространства поиска происходит, когда следующая позиция находится за пределами пространства, находящегося между бабочкой и пламенем, как это показано стрелками, обозначенным цифрами 1, 3 и 4. Задача эксплуатации выполняется, когда следующая позиция находится внутри пространства между мотыльком и пламенем. На рисунке 4.48 этому случаю соответствует позиция, обозначенная стрелкой под номером 2.

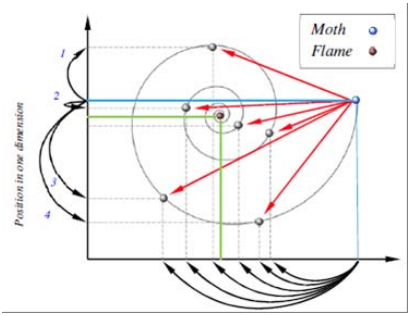


Рис. 4.50. Некоторые из возможных положений, которых может достичь мотылек относительно пламени, используя логарифмическую спираль

С точки зрения применения данной модели можно отметить несколько интересных фактов:

- поиск (мотылек) может сходиться в любой точке вблизи пламени, что достигается путем изменения значения t;
- чем меньше значение t, тем ближе расстояние до огня;
- частота обновления положения мотылька по обе стороны от пламени увеличивается по мере приближения его к огню.

Описанная процедура обновления позиции обеспечивает решение задачи эксплуатацию в окрестностях пламени. Чтобы повысить вероятность нахождения лучших решений, мы рассматриваем лучшие решения, полученные на данный момент с помощью пламени. Таким образом, матрица F в уравнении (4.68) всегда включает в себя n последних лучших решений, полученных на данный момент. Мотылькам необходимо обновлять свои позиции относительно этой матрицы в процессе оптимизации. Число r называют константой сходимости. Чтобы улучшить процесс эксплуатации, данном алгоритме предполагается, что t – это случайное число из интервала $[r, 1]$, где r линейно уменьшается от -1 до -2 в течение итерации.

При использовании этого метода мотыльки стремятся ориентироваться по текущему положению пламени на каждой итерации. При этом может возникнуть следующая проблема: обновление положения в уравнении (4.77) происходит путем перемещения мотыльков к огню, однако это приводит к тому, что алгоритм MFOA достаточно быстро сходится к точке локального оптимума. Чтобы избежать этого, каждый мотылек должен обновлять свое положение, используя только один из языков пламени в уравнении (4.77). На каждой итерации после обновления списка языков пламени они сортируются на основе значений их функции пригодности. Затем мотыльки меняют свое положение относительно текущего расположения пламени. Первый мотылек всегда обновляет свою позицию относительно лучшей позиции пламени, тогда как последний мотылек обновляет свою позицию относительно худшего пламени в списке. На рис. 4.51 показано, как устанавливается соответствие каждого мотылька каждой позиции пламени в соответствующем списке позиций пламени.

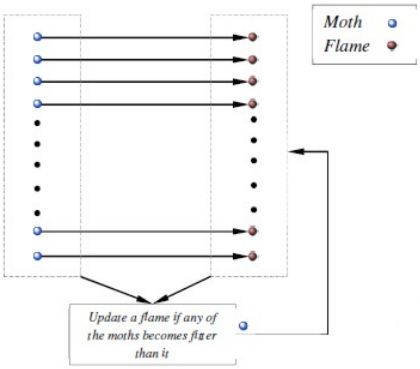


Рис. 4.51. Процедура присвоения каждому мотыльку одной позиции пламени

Следует отметить, что процедура установления соответствия между искусственными мотыльками и позициями пламени была разработана специально для решения задачи моделирования процесса поперечной ориентации в алгоритме MFOA, и, возможно, не соответствует реальному поведению мотыльков в природе. Причина, по которой каждому мотыльку присваивается определенная позиция пламени, заключается в том, что это должно помочь избежать преждевременного схождения алгоритма в точке локального оптимума. На самом деле, если всех мотыльков привлечет одно пламя, все они соберутся в одной точке в пространстве поиска, потому что они могут лететь только по направлению к пламени. А использование вышеописанной процедуры распределения позволяет организовать перемещение мотыльков вокруг разных позиций пламени, что дает возможность более детального исследования пространства поиска и приводит, в конечном итоге, снижению вероятности застоя в районе локальных оптимумов.

Таким образом, выполнение данной процедуры гарантирует, что исследование пространства поиска выполняется вокруг позиций из имеющихся на данный момент благодаря следующим обстоятельствам:

- мотыльки обновляют свои позиции в гиперсферах вокруг лучших решений, полученных на данный момент;
- изменение позиций пламени происходит относительно лучших решений в каждой итерации, и мотылькам необходимо соответственно обновлять свои позиции относительно обновленного

списка позиций пламени. При этом обновление положения мотыльков может происходить вокруг различных языков пламени, то есть дает механизм, обеспечивающий внезапное перемещение мотыльков в пространстве поиска, и способствует исследованию.

Еще одна проблема здесь заключается в том, что обновление положения мотыльков относительно *n* различных мест в пространстве поиска может ухудшить некоторые из имеющихся перспективных решений. Для решения данной проблемы используется адаптивный механизм количества позиций пламени с помощью которого, количество позиций языков пламени адаптивно уменьшается по мере прохождения итераций. Для этого используется следующее уравнение, описывающее число позиций языков пламени (F_N) [202]:

$$F_N = round[(N - l)\ (N - 1/T)] \tag{4.79}$$

где F_N – это номер позиции пламени, l – текущее количество итераций, N – максимальное количество языков пламени, а T – максимальное количество итераций.

Как правило, на начальных итерациях алгоритма имеется N языков пламени. Однако мотыльки обновляют свои позиции только относительно лучшего пламени на последних итерациях. Постепенное уменьшение количества позиций языков пламени позволяет поддерживать баланс между задачами исследования и использования пространства поиска. В общем виде данная процедура (вычисление функции P) может быть представлена следующим образом:

Update flame no using Equation (4.73)
OM=FitnessFunction(M);
if iteration==1
F=sort(M);
OF=sort(OM);
else
F=sort(Mt-1, Mt);
OF=sort(Mt-1, Mt);
end
for i = 1 : n
for j= 1 : d
Update r and t
Calculate D using Equation (3.13) with respect to the corresponding moth
Update M(i,j) using Eqs. (3.11) and (3.12) with respect to the corresponding moth
end
end

Как было отмечено выше, процедура вычисления функции *P* выполняется до тех пор, пока функция *T* не вернет значение *true*. После завершения процедура вычисления функции *P* лучший мотылек соответствовать наилучшему полученному приближению к оптимуму.

Блок-схема алгоритма приведена на рис. 4.52.

Рис. 4.52. Блок-схема алгоритма MFOA

Пример использования алгоритма оптимизации «Пламя-мотылек»

Рассмотрим пример использования алгоритма оптимизации «Пламя-мотылек» [202] для минимизации функции Гольдштейна-Прайса:

$$f(x, y) = 1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)]$$
$$[30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)],$$

реализованный в MATLAB.

Чтобы проиллюстрировать этот подход, было рассмотрено 30 поисковых агентов (n = 30), а максимальное число итераций было установлено равным 100 [202]. Наилучшее решение, полученное с помощью MFOA, равно: [1,7999 0,19996], а наилучшее оптимальное значение целевой функции, найденное с помощью MFOA, равно 84. Пространство параметров и тест на сходимость показаны на рисунках 4.53 и 4.54 соответственно [202].

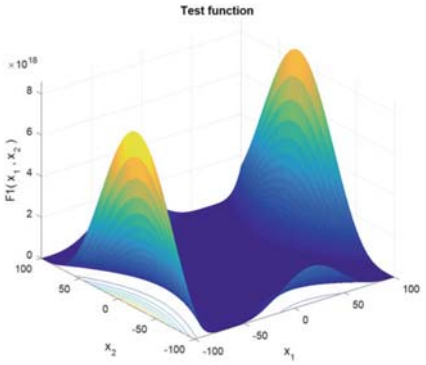


Рис. 4.53. Тестовая функция

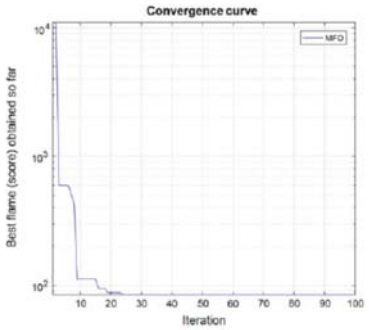


Рис. 4.54. Кривая сходимости

ЗАКЛЮЧЕНИЕ

Подводя итоги, отметим, что моделирование процессов, происходящих в живой природе, является эффективной методологической основой для создания интеллектуальных искусственных систем. Описанные модели и алгоритмы, построенные на их основе, могут быть использованы для решения задач искусственного интеллекта, оптимизации, принятия решений, конструирования, проектирования, технологической подготовки производства и др. При этом полученные результаты показывают перспективность данного направления исследований.

Основные результаты монографии можно сформулировать следующим образом. Рассмотрены основные принципы эволюции в живых и искусственных системах. Рассмотрены и описаны основные модели эволюций проанализированы подходы к построению архитектур искусственных систем на их основе. Сформулированы ряд положений и основные принципы теории эволюционного моделирования. Рассмотрены основные положения эволюционного и генетического поиска приведено их формальное описание. Предложены различные типы генетических алгоритмов. Описаны основные операторы генетического поиска. Рассмотрены общие положения теории биоинспирированного поиска. Представлена их классификация и области применения. Описано понятие роевого интеллекта, рассмотрены и проанализированы основные модели и методы, основанные на роевом интеллекте. Приведены и рассмотрены алгоритмы муравьиных колоний, пчелиного роя, светлячковой, бактериальной оптимизации, а также алгоритмы роя саранчи и колонии пауков. Также представлены другие модели и методы метаэвристической оптимизации, инспирированные фауной такие как обезьяньего поиска, кукушки, летучих мышей, бабочек монархов, червей, серых волков (GWO) и белых кротов. Необходимо заметить, что основным преимуществом данных методов является их модульная структура, что позволяет использовать их достоинства и недостатки для проектирования новых вариаций алгоритмов, а также проводить их комбинирование, интеграцию и гибридизацию. Также данные методы, на основе проведенных исследований легко распараллеливаются и демонстрируют эффективную работу при обработке больших массивов данных.

Теория биоинспирированного поиска это не законченное фундаментальное исследование. Она постоянно совершенствуется и дополняется. Авторы сделали первую попытку объединить некоторые эволюционные, генетические и бионические принципы для повышения качества управления метаэвристическим поиском при решении сложных задач оптимизации и принятия решений. Мы надеемся, что внесли определенный вклад в теорию и практику биоинспирированного поиска, сделав небольшой, но важный шаг в этом направлении. Необходима дальнейшая разработка новых моделей, инспирированных природными системами и широкое проведение экспериментальных исследований. Надеемся, что монография позволит интенсифицировать исследования в этой области.

СПИСОК ЛИТЕРАТУРЫ

1. Рапопорт, Г. Н. Искусственный и биологические интеллекты / Г. Н. Рапопорт, А. Г. Герц // Общность структуры, эволюция и процессы познания. – Москва: Комкнига, 2005. – 310 с.

2. Люггер, Дж. Искусственный интеллект / Дж. Люггер // Стратегии и методы решения сложных проблем. – Москва: Издательский дом «Вильямс», 2003.

3. Емельянов, В. В. Теория и практика эволюционного моделирования / В. В. Емельянов, В. В. Курейчик, В. М. Курейчик. – Москва: Физматлит, 2003. – 432 с.

4. Капра, Ф. Паутина жизни / Ф. Капра // Новое научное понимание живых систем – Москва: Гелиос, 2002. – 336 с.

5. Голицын, Г. А. Информация и биологические принципы оптимальности / Г. А. Голицын, В. М. Петров // Гармония и алгебра живого. – Москва: КомКнига, 2005. – 125 с.

6. Большая советская энциклопедия. – Т. 29. – Москва: Советская энциклопедия, 1978.

7. Дарвин, Ч. Происхождение видов путем естественного отбора / Ч. Дарвин. – Москва: Тайдекс Ко, 2003.

8. Гладков, Л. А. Генетические алгоритмы: учебник / Л. А. Гладков, В. В. Курейчик, В. М. Курейчик. – Москва: Физматлит, 2010. – 368 с.

9. Курейчик, В. М. Поисковая адаптация / В. М. Курейчик, Б. К. Лебедев, О. Б. Лебедев. – Москва: Физматлит, 2006. – 272 с.

10. Hugo de Vries. The Mutation Theory // Experiments and Observations on the Origin of Species in the Vegetable Kingdom The origin of species by mutation. Open Court Publishing Company, 1909.

11. Шмальгаузен, И. И. Проблемы дарвинизма / И. И. Шмальгаузен. – Ленинград: Наука, 1969.

12. Шмальгаузен, И. И. Пути и закономерности эволюционного процесса / И. И. Шмальгаузен. – 2-е изд. – Москва, 1983.

13. Северцев, А. С. Теория эволюции / А. С. Северцев. – Москва: Владос, 2005.

14. Хедрик, Ф. Генетика популяций / Ф. Хедрик. – Москва: Техносфера, 2003.

15. Дубинин, Н. П. Избранные труды / Н. П. Дубинин // Проблемы гена и эволюции. – Т. 1. – Москва: Наука, 2000.

16. Дульнев, Г. Н. Введение в синергетику / Г. Н. Дульнев – Санкт-Петербург: Проспект, 1998.

17. Курейчик, В. В. Анализ и обзор моделей эволюции / В. В. Курейчик, В. М. Курейчик, П. В. Сороколетов. // Известия Российской академии наук. Теория и системы управления. – 2007. – №5. – С. 114–126.

18. Ламарк, Ж. Б. Философия зоологии / Ж. Б. Ламарк. – Т. 1, 2. – Москва, Ленинград: Академия,1939.

19. Лахути Д. Г. Эволюционная эпистемология и логика социальных наук / Д. Г. Лахути, В. Н. Садовского, В. К. Финна // Карл Поппер и его критики. – Москва: Эдиториал УРСС, 2000.

20. Kimura, M. The Neutral Theory of Molecular Evolution. – Cambridge.: Cambridge University Press, 1983.

21. Майр, Э. Зоологический вид и эволюция / Э. Майр. – Москва, 1968.

22. Букатова, И. Л. Эволюционное моделирование и его приложения / И. Л. Букатова. – Москва: Наука, 1991.

23. Курейчик, В. В. Концепция эволюционных вычислений, инспирированных природными системами / В. В. Курейчик, В. М. Курейчик, С. И. Родзин // Известия ЮФУ. Технические науки. – 2009. – №4 (93). – С. 16–24.

24. Курейчик В. В. Теория эволюционных вычислений: монография / В. В. Курейчик, В. М. Курейчик, С. И. Родзин. – Москва: Физматлит, 2012. – 260 с.

25. Курейчик, В. В. О правилах представления решений в эволюционных алгоритмах / В. В. Курейчик, С. И. Родзин // Известия ЮФУ. Технические науки. – 2010. – №7 (108). – С. 13–21.

26. Карпенко, А. П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой: учебное пособие / А. П. Карпенко. – Москва: МГТУ им. Н.Э. Баумана, 2014. – 446 с.

27. De Jong, K. Evolutionary Computation: Recent Development and Open Issues // Proceedings1st International conf., Evolutionary Computation and Its Application. – Moscow, 1996. Pp. 7–18.

28. Holland, John H. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence. USA: University of Michigan, 1975. 183 p.

29. Goldberg, David E. Genetic Algorithm in Search, Optimization and Machine Learning. USA.: Addison-Wesley Publishing Company. Ind. 1989. 196 p.

30. Lawrence, Davis. Handbook of Genetic Algorithms. USA.: Van Nostrand Reinhold. New York, 1991. 100 p.

31. Kureichik, V. V., Kureichik, V. M. Genetic Search-Based Control. Automation and Remote Control. 2001. V. 62. 10. Pp. 1698–1710.

32. Растригин, Л. А. Статистические методы поиска / Л. А. Растригин. – Москва: Наука, 1968.

33. Курейчик, В. В. Концептуальная модель представления решений в генетических алгоритмах / В. В. Курейчик, П. В. Сороколетов // Известия Южного федерального университета. Технические науки. – 2008.– №9 (86). – С. 7–12.

34. Родзин, С. И. Теоретические вопросы и современные проблемы развития когнитивных биоинспирированных алгоритмов оптимизации (обзор) / С. И. Родзин, В. В. Курейчик // Кибернетика и программирование. – 2017. – №3. – С. 51–79.

35. Родзин, С. И. Состояние, проблемы и перспективы развития биоэвристик / С. И. Родзин, В. В. Курейчик // Программные системы и вычислительные методы. – 2016. – №2. – С. 158–172.

36. Курейчик, В. В. Бионспирированные методы в оптимизации: монография / В. В. Курейчик, В. М. Курейчик, Л. А. Гладков [и др.]. – Москва: Физматлит, 2009. – 384 с.

37. Курейчик, В. В. Вычислительные модели эволюционных и роевых биоэвристик (обзор) / В. В. Курейчик, С. И. Родзин // Информационные технологии – 2021. – Т. 27. №10. – С. 507–520.

38. Курейчик, Вл. Вл. Биоиспирированный поиск при проектировании и управлении / Вл. Вл. Курейчик, В. В. Курейчик // Известия ЮФУ. Технические науки. – 2012. – №11 (136). – С. 178–183.

39. Курейчик, В. В. Вычислительные модели биоэвристик, основанных на физических и когнитивных процессах (обзор) / В. В. Курейчик, С. И. Родзин // Информационные технологии. – 2021. – Т. 27. №11. – С. 563–574.

40. Курейчик, В. В. Биоэвристики, инспирированные фауной (обзор) / В. В. Курейчик, С. И. Родзин // Информационные технологии. – 2023. – Т. 29. №11. – С. 559–573.

41. Fausto, F., et. al. From ants to whales: metaheuristics for all tastes. Artif. Intell. Rev. 2019. Vol. 53 (1). Pp. 753–810. – URL: http://doi:10.1007/s10462-018-09676-2.

42. Родзин, С. И. Современное состояние биоэвристик: классификация, бенчмаркинг, области применения / С. И. Родзин // Известия ЮФУ. Технические науки. – 2023. – №2. – С. 280–298. URL: https://izv-tn.tti.sfedu.ru/index.php/izv_tn/article/view/793/981.

43. Abraham, A., Ramos V., Grosan G. Swarm Intelligence in Data Mining. Berlin. Heidelberg: Springer Verlag, 2007. 267 p.

44. Hassanien, E., Emary E. Swarm Intelligence. Principles Advances, and Applications. CRC Press. 2015. 228 p.

45. Mourelle, M., Nedjah L. De. Swarm intelligent systems. Berlin: Heidelberg: Springer Verlag, 2006. 217 p.

46. Kennedy, J., Eberhart, R. (1995). "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks. Vol. IV. Pp. 1942–1948. doi:10.1109/ICNN.1995.488968

47. Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Erciyes University, Engineering Faculty, Computer Engineering Department. 110 p.

48. Dorigo, M., Maniezzo V., Colorni A. (1996). The Ant System: Optimization by a colony of cooperating objects // IEEE Trans. on Systems, Man, and Cybernetics. Part B. №26 (1). Pp. 29–41.

49. Muro, C., Escobedo R., Spector L., Coppinger R. Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations. Behav. Process. 2011. Pp. 88,192–7.

50. Maziar Yazdani, Fariborz Jolai. Lion Optimization Algorithm (LOA). A nature-inspired metaheuristic algorithm Journal of Computational Design and Engineering. Vol. 3. Issue 1, January 2016. Pp. 24–36. https://doi.org/10.1016/j.jcde.2015.06.003

51. Saremi, S., Mirjalili S., Lewis A. Grasshopper optimization algorithm: Theory and application. Advances in Engineering Software. 105. 2017. Pp. 30–47. URL: https://www.researchgate.net/publication/350204693_Grasshopper_Optimization_Algorithm_Theory_Variants_and_Applications [accessed May 28 2024].

52. Amir Hossein Gandomi and Amir Hossein Alavi.Krill herd: a new bio-inspired optimization algorithm. Communications in nonlinear science and numerical simulation. 2012. 17(12). Pp. 4831–4845. – URL: https://www.researchgate.net/publication/350204693_Grasshopper_Optimization_Algorithm_Theory_Variants_and_Applications [accessed May 28 2024].

53. Mirjalili, S., Lewis A. The Whale Optimization Algorithm. Advances in Engineering Software. 2016. 95. Pp. 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

54. Passino, K. "Biomimicry of bacterial foraging for distributed optimization and control" Control Systems. IEEE. Vol. 22. Jun 2002. Pp. 52–67.

55. Zong Woo Geem, Joong Hoon Kim, and Gobichet-tipalayam Vasudevan Loganathan. A new heuristic optimization algorithm: harmony search. Simulation Engineering Applications of Artificial Intelligence. 2001. 76(2). Pp. 60–68. – URL: https://www.researchgate.net/publication/350204693_Grasshopper_Optimization_Algorithm_Theory_Variants_and_Applications [accessed May 28 2024].

56. Hamed Shah-Hosseini. Intelligent water drops algorithm: a new optimization method for solving the multiple knapsack problem. International Journal of Intelligent Computing and Cybernetics. June 2008. 1(2). Pp. 193–212

57. Kirkpatrick, S., Gelatt C. D., Vec-chi M. P. Optimization by simulated annealing. Science. 1983. 220(4598). Pp. 671–680.

58. Dasgupta, D. Artificial Immune Systems and Their Applications. Berlin, Germany: Springer-Verlag, 1999.

59. Esmaeil Atashpaz-Gargari and Caro Lucas. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition – In 2007IEEE congress on evolutionary computation, 2007. Pp. 4661–4667. – URL: https://www.researchgate.net/publication/350204693_Grasshopper_Optimization_Algorithm_Theory_Variants_and_Applications [accessed May 28 2024].

60. Cheng, Shi, Yuhui Shi. Brain Storm Optimization Algorithms. Concepts, Principles, and Applications Berlin, Germany: Springer-Verlag, 2017.

61. Storn, R., Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization. 1997. 11(4). Pp. 341–359, 1997. – URL: https://www.researchgate.net/publication/350204693_Grasshopper_Optimization_Algorithm_Theory_Variants_and_Applications [accessed May 28 2024].

62. Hans-Georg Beyer, Hans-Paul Schwefel. Evolution strategies–a comprehensive introduction. Natural computing. 2002. 1(1). Pp. 3–52.

63. Ghaemi, M., & Feizi-Derakhshi M.-R. (2014). Forest optimization algorithm. Expert Systems with Applications. 41(15). Pp. 6676–6687. doi: 10.1016/j.eswa.2014.05.009

64. Dragoi, E., Dafinescu V. Review of Metaheuristics Inspired from the Animal Kingdom. Mathematics. 2021. Vol. 9(19). P. 2335. http://doi:10.3390/math9182335.

65. Molina, D, et. al. Comprehensive taxonomies of nature- and bio-inspired optimization: inspiration versus algorithmic behavior, critical analysis, and recommendations. – URL: https://arxiv.org/abs/2002.08136v3.

66. Xin-She Yang. Firefly algorithms for multimodal optimization. In International symposium on stochastic algorithms. Springer, 2009. Pp. 169–178.

67. Wang, G.-G., Deb S., Cui Z. Monarch butterfly optimization. Neural Comput.&Applic. 2019. Vol. 31. Pp. 1995–2014.

68. Sacco, W. F., Oliveira C. R. (2005). A new stochastic optimization algorithm based onparticle collisions. Transactions of the American Nuclear Society, Vol. 92. Pp. 657–659. – URL: https://www.researchgate.net/publication/300172453_A_new_Metropolis_optimisation_method_the_cross-section_particle_collision_algorithm_some_preliminary_results [accessed May 28 2024].

69. Родзин, С. И. Биоэвристики: теория, алгоритмы и приложения / С. И. Родзин, Ю. А. Скобцов, С. А. Эль-Хатиб. – Чебоксары: Среда, 2019. – 224 с. URL: https://phsreda.com/e-articles/54/Action54-22141.pdf

70. Чивилихин, Д. С. Модифицированный муравьиный алгоритм для построения конечных автоматов по сценариям работы и темпоральным формулам / Д. С. Чивилихин, В. И. Ульянцев, А. А. Шалыто // Автоматика и телемеханика. – 2016. – №3. – С. 137–151.

71. Kaveh, A. (2021). Dolphin Echolocation Optimization. In: Advances in Metaheuristic Algorithms for Optimal Design of Structures. Springer, Cham. https://doi.org/10.1007/978-3-030-59392-6_6

72. Wu, Tq., Yao M. & Yang Jh. Dolphin swarm algorithm. Frontiers Inf Technol Electronic Eng. 2016. 17. Pp. 717–729. https://doi.org/10.1631/FITEE.1500287

73. Bozorgi, A., Bozorg-Haddad O., Chu X. (2018). Anarchic Society Optimization (ASO) Algorithm. In: Bozorg-Haddad, O. (eds) Advanced Optimization by Nature-Inspired Algorithms – Studies in Computational Intelligence. Vol. 720. Springer, Singapore. https://doi.org/10.1007/978-981-10-5221-7_4

74. Tan Y., Shi Y., Tan K.C. (Eds.). (2010). Fireworks Algorithm for Optimization: ICSI 2010, Part I, LNCS 6145. Pp. 355–364. Springer-Verlag Berlin Heidelberg 2010 November 2017.

75. Yang, X.-S. (2012). Flower pollination algorithm for global optimization. International conference on unconventional computing and natural computation. Pp. 240–249. – URL: https://www.researchgate.net/publication/340302086_Review_of_Flower_Pollination_Algorithm_Applications_and_Variants [accessed May 28 2024].

76. Скобцов, Ю. А. От генетических алгоритмов к метаэвристикам / Ю. А. Скобцов // Информатика и кибернетика. – 2021. – №1-2 (23-24). – С. 101–107.

77. Басин, А. О. Правило «одной пятой» с возвратами для настройки размера популяции в генетическом алгоритме (1 + (λ,λ)) / А. О. Басин, М. В. Буздалов, А. А. Шалыто // Моделирование и анализ информационных систем. – 2020. – Т. 27. №4. – С. 488–508.

78. Шерстнев, П. А. Применение эволюционных алгоритмов при проектировании интерпретируемых моделей машинного обучения в задачах классификации / П. А. Шерстнев, Е. С. Семенкин // Системы управления и информационные технологии. – 2022. – №1 (87). – С. 17–20.

79. Мех, М. А Сравнительный анализ применения методов дифференциальной эволюции для оптимизации параметров нечетких классификаторов / М. А. Мех, И. А. Ходашинский // Известия Российской академии наук. Теория и системы управления. – 2017. – №4. – С. 65–75.

80. Fred, Glover. Tabu Search – Part 1. ORSA Journal on Computing. 1989. Т. 1. Вып. 2. doi:10.1287/ijoc.1.3.190

81. Fred, Glover. Tabu Search – Part 2. ORSA Journal on Computing. 1990. Т. 2. Вып. 1. doi:10.1287/ijoc.2.1.4

82. Петровский, А. Б. Групповой вербальный анализ решений / А. Б. Петровский. – Москва: Наука, 2019. – 287 с.

83. Wittaya Julklang, Boris Golman Effect of process parameters on energy performance of spray drying with exhaust air heat recovery for production of high value particles. 15 May 2015. Pp. 285–295. DOI: 10.1016/j.apenergy.2015.04.029.

84. Mohammed, H. Qais, Hasanien Hany M., Alghuwainem Saad. (2018). Accepted Manuscript: Augmented Grey Wolf Optimizer for Grid-connected PMSG-based Wind Energy Conversion Systems, springer. DOI: 10.1016/j.asoc.2018.05.006.

85. Düzgün Akmaz, Mehmet Salih Mamis¸ Müslüm Arkanb, Mehmet Emin Tagluk. (2017). Electric Power Systems Research: Transmission line fault location using traveling wave frequencies and extreme learning machine. Pp. 570–575, Autumn. DOI: 10.1016/j.epsr.2017.09.001.

86. Rebecca Ng Shin Mei, Mohd Herwan Sulaiman, Zuriani Mustaffa, Hamdan Daniyal. (2017). Applied Soft Computing: Optimal reactive power dispatch solution by loss minimization using moth-flame optimization technique. Autumn. Pp. 210–222. DOI: 10.1016/j.asoc.2017.05.057.

87. Nadeem Javaid, Ihtisham Ullah, Syed Shahab Zarin, Mohsin Kamal, Babatunji Omoniwa, Abdul Mateen. (2019). Differential-Evolution-Earthworm Hybrid Meta-Heuristic Optimization Technique for Home Energy Management System in Smart Grid. January. Pp. 15–31 DOI: 10.1007/978-3-319-93554-6_2.

88. James, J.Q. Yu, Victor O.K. Li (2015). Applied Soft Computing: A social spider algorithm for global optimization. Pp. 614–627. 16 February. DOI: 10.1016/j.asoc.2015.02.014.

89. Le Hoang Son, Francisco Chiclana, Raghavendra Kumar, Mamta Mittal, Manju Khari, Jyotir Moy Chatterjee, Sung Wook Baik, Knowledge-Based Systems: ARM–AMO: An efficient association rule mining algorithm based on animal migration optimization. Pp. 68–80, 15 August 2018. (DOI: 10.1016/j.knosys.2018.04.038).

90. Ibrahim Aljarah, Ala' M. Al-Zoubi, Hossam Faris, Mohammad Hassonah, Simultaneous Feature Selection and Support Vector Machine Optimization Using the Grasshopper Optimization Algorithm – Springer, June 2018. (DOI: 10.1007/s12559-017-9542-9).

91. Laith Mohammad Abualigah, Ahamad Tajudin Khader, Essam Said Hanandeh, Hybrid clustering analysis using improved krill herd algorithm – Springer, May 2018. (DOI: 10.1007/s10489-018-1190-6).

92. Карпенко, А. П. Роевой интеллект и его применение в системах высокой доступности / А. П. Карпенко, И. Н. Синицын // Системы высокой доступности – 2022. – Т. 18. №4. – С. 44–55.

93. Ходашинский, И. А. Методы повышения эффективности роевых алгоритмов оптимизации / И. А. Ходашинский // Автоматика и телемеханика – 2021. – №6. – С. 3–45.

94. Курейчик, В. В. Роевой алгоритм в задачах оптимизации / В. В. Курейчик, Д. Ю. Запорожец // Известия ЮФУ. Технические науки. – 2010. – №7 (108). – С. 28–32.

95. Vladimir Kureichik Jr, Elmar Kuliev, Vladimir Kureichik. Mechanisms of swarm intelligence and evolutionary adaptation for solving PCB design tasks. Proceedings off International Seminar on Electron Devices Design and Production (SED). Prague. 2019. Pp. 109–113.

96. Курейчик, Вл. Вл. Обзор и анализ методов и моделей, инспирированных природными системами / Вл. Вл. Курейчик, В. В. Курейчик // Информатика, вычислительная техника инженерное образование. – 2013. – №2 (13). – 13 с.

97. Зайцев, А. А. Обзор эволюционных методов оптимизации на основе роевого интеллекта / А. А. Зайцев, В. В. Курейчик, А. А. Полупанов // Известия ЮФУ. Технические науки. – 2010. – №12 (113). – С. 7–12.

98. Бова, В. В. Многоуровневый алгоритм решения задач транспортной логистики на основе методов роевого интеллекта / В. В. Бова, В. В. Курейчик, А. А. Лежебоков // Вестник Ростовского государственного университета путей сообщения. – 2013. – №3 (51). – С. 113–118.

99. Кулиев, Э. В. Подход к исследованию окрестностей в роевых алгоритмах для решения оптимизационных задач / Э. В. Кулиев, А. А. Лежебоков, А. Н. Дуккардт // Известия ЮФУ. Технические науки. – 2014. – №7 (156). – С. 15–25.

100. Ахмедова, Ш. А. К. Кооперативные бионические методы оптимизации / Ш. А. К. Ахмедова, Е. С. Семенкин. – Красноярск, 2017. – 160 с.

101. Тарасов, В. Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика / В. Б. Тарасов. – Москва: Эдиториал УРСС, 2002. – 352 с.

102. Dorigo, M. (1992). Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, Italy.

103. Запорожец, Д. Ю. Муравьиный алгоритм определения критических связей в СБИС / Д. Ю. Запорожец, Д. В. Заруба, В. В. Курейчик // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). – 2014. – №2. – С. 107–112.

104. Кажаров, А. А. Муравьиные алгоритмы для решения транспортных задач / А. А. Кажаров, В. М. Курейчик // Известия Российской академии наук. Теория и системы управления. – 2010. – №1. – С. 32–45.

105. Балюк, Л. В. Вероятностный генетический алгоритм Ant Colony и его приложение для решения задач (на примере задачи о коммивояжере) / Л. В. Балюк, В. В. Курейчик // «Интеллектуальные системы» (IEEE AIS'04) и «Интеллектуальные САПР (CAD-2004)»: труды Международных научно-технических конференций. Т. 1.– Москва: Физматлит, 2004. – С. 53–65.

106. Colorni, A., Dorigo M., Maniezzo, V. F. Varela and P.Bourgine. Distributed Optimization by Ant Colonies. Proceedings of the First European Conference on Artificial Life. Paris, France, Elsevier Publishing. 1991. Pp. 134–142.

107. Colorni, A., Dorigo M., Maniezzo V., The Ant System: Optimization by a colony of cooperating agents. Tech.Rep.IRIDIA/94-28. Université Libre de Bruxelles, Belgium, 1996.

108. Dorigo, M., Stützle, T. (2004). Ant Colony Optimization. MIT Press.

109. Zaruba, D., Zaporozhets D., Kureichik V. Artificial bee colony algorithm-a novel tool for vlsi placement. Advances in Intelligent Systems and Computing. 2016. Т. 450. С. 433–442.

110. Курейчик, В. В. Эволюционная оптимизация на основе алгоритма колонии пчёл / В. В. Курейчик, Е. Е. Полупанова // Известия ЮФУ. Технические науки. – 2009. – №12 (101). – С. 41–46.

111. Pham, D. T., Castellani, M. (2009). The Bees Algorithm – Modelling Foraging Behaviour to Solve Continuous Optimisation Problems. Proc. ImechE, Part C. 223(12). Pp. 2919–2938.

112. Курейчик, Вл. Вл. Биоинспирированный алгоритм разбиения схем при проектировании – СБИС / Вл. Вл. Курейчик, В. В. Курейчик // Известия ЮФУ. Технические науки. – 2013. – №7 (144). – С. 23–29.

113. Курейчик, В. В. Метод светлячковой оптимизации для решения комбинаторно-логических задач на графах / В. В. Курейчик, В. В. Бова, В. В. Курейчик // Информационные технологии в науке, образовании и управлении. – 2021. – №1 (17). – С. 20–24.

114. Курейчик, В. В. Компоновка фрагментов СБИС на основе модели поведения роя светлячков / В. В Курейчик., Д. Ю. Запорожец, Д. В. Заруба // Международная конференция по мягким вычислениям и измерениям. – 2016. – Т. 1. – С. 463–466.

115. Курейчик, В. В. Алгоритм параметрической оптимизации на основе модели поведения роя светлячков / В. В. Курейчик, Д. В. Заруба, Д. Ю. Запорожец // Известия ЮФУ. Технические науки. – 2015. – №6 (167). – С. 6–15.

116. Бова, В. В. Моделирование поведения субъекта в интернет-сервисах на основе модифицированного алгоритма бактериальной оптимизации / В. В. Бова, Ю. А. Кравченко, Э. В. Кулиев [и др.] // Информационные технологии. – 2019. – Т. 25. №7. – С. 397–404.

117. Zaporozhets, D., Zaruba D. Bacterial foraging optimization for vlsi fragments placement. Advances in Intelligent Systems and Computing. 2018. Т. 679. С. 341–348.

118. Zaruba, D., Zaporozhets D., Kuliev E. Parametric optimization based on bacterial foraging optimization. Advances in Intelligent Systems and Computing. 2017. Т. 573. С. 54–63.

119. Кравченко, Ю. А. Метод интеллектуального принятия эффективных решений на основе биоинспирированного подхода / Ю. А. Кравченко, Э. В. Кулиев, О. А. Логинов [и др.] // Известия КБНЦ РАН. – 2017. – №6 (80). Ч. 2– С. 162–169.

120. Остапенко, Р. О. Формирование базы правил нечёткого классификатора с помощью метаэвристического алгоритма «саранчи» / Р. О. Остапенко, И. А. Ходашинский // Доклады ТУСУР. – 2022. Т. 25. №2. DOI: 10.21293/1818-0442-2022-25-2-31-36

121. Курейчик, В. В. Модель коллаборативного поведения роя саранчи для оптимизации многомерных многоэкстремальных функций / В. В. Курейчик, С. И. Родзин // Известия вузов. Северо-Кавказский регион. Технические науки. – 2023. – №1. – С. 10–16.

122. Cuevas, E., et. al. A swarm optimization algorithm inspired in the behavior of the social spider. Jour. Expert Systems with Applications. 2013. no. 16. Pp. 6374–6384.

123. Rodzin, S., Rodzina L. Spider Colony Optimization Algorithm: A Bio-Heuristic for Global Optimization Problem. Lecture Notes in Networks and Systems book series. 2023. Vol. 722. Pp. 661–669.

124. Курейчик, В. В. Метаэвристический оптимизатор на основе модели поведения колонии социальных пауков / В. В. Курейчик, С. И. Родзин // Вестник компьютерных и информационных технологий. – 2022. – Т. 19. №6 (216). – С. 12–20.

125. Mucherino, A., Seref O., Seref O., Kundakcioglu O. E., & Pardalos P. (2007). Monkey search: a novel metaheuristic search for global optimization. AIP Conference –Proceedings, 953, 162.

126. Zhao, R., Tang W. Monkey algorithm for global numerical optimization. Journal of Uncertain Systems. 2008 №2. Pp. 165–176.

127. Hodashinsky, I. A., Samsonov S. S. Design of fuzzy rule-based classifier using the monkey algorithm. Business Informatics. 2017. No.1 (39). Pp. 61–67.

128. Kuliev, E., Kureichik V., Kureichik V. Monkey search algorithm for ece components partitioning. Journal of Physics: Conference Series. 1015 (4), статья № UNSP 042026. 2018. Pp. 1–10.

129. Курейчик В.В. Модель адаптивного поведения «обезьян» для решения задачи компоновки блоков ЭВА / В. В. Курейчик, Э. В. Кулиев, В. В. Курейчик, // Информатизация и связь. – 2018. – №4. – С. 31–37.

130. Yang, X.-S., S. Deb (December 2009). Cuckoo search via Lévy flights. World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). IEEE Publications. Pp. 210–214.

131. Yang, X.-S., Deb S. (2010). Engineering Optimisation by Cuckoo Search. Int. J. Mathematical Modelling and Numerical Optimisation. Vol. 1. No. 4. Pp. 330–343.

132. Kravchenko, Yu. A., Bova V. V. Assessment of Ontological Structures Semantic Similarity Based on a Modified Cuckoo Search Algorithm. IOP Conference Series: Materials Science and Engineering. 2020. Vol. 734. № 1. № paper 012018.

133. Liao, Q., Shi H., Shi W., Zhou S. Parameter Estimation of Nonlinear Systems by Dynamic Cuckoo Search. Neural Computation. 2017. Vol. 29. №4. Pp. 1103–1123.

134. Barthelemy, P., Bertolotti J., Wiersma D. S. (2008). A L´evy flight for light', Nature. Pp. 453, 495–498.

135. Xin-She, Yang. A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, 2010. Pp. 65–74.

136. Xin-She, Yang. Bat algorithm: literature review and applications. arXiv preprint arXiv:1308.3900, 2013. – URL: https://www.researchgate.net/publication/350204693_Grasshopper_Optimization_Algorithm_Theory_Variants_and_Applications [accessed Jun 11 2024].

137. Кулиев, Э. В. Роевой алгоритм поисковой оптимизации на основе моделирования поведения летучих мышей / Э. В. Кулиев, А. А. Лежебоков, Ю. А. Кравченко // Известия ЮФУ. Технические науки. – 2016. – №7 (180). – С. 53–62.

138. Кулиев, Э. В. Интеллектуальная подсистема поддержки принятия решений на основе биологически правдоподобных алгоритмов самоорганизации / Э. В. Кулиев, М. П. Кривенко, М. М. Семенова [и др.] // Известия ЮФУ. Технические науки. – 2021. – №4 (221). – С. 105–116.

139. Родзин, С. И. Биоинспирированная гиперэвристика для отбора значимых признаков в задачах классификации больших данных / С. И. Родзин // Вестник компьютерных и информационных технологий. – 2021. – Т. 18. №5. – С. 35–44.

140. Arnaout, J.-P. Worm Optimization for the Traveling Salesman Problem. In book: Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling. 2016.

141. Seyedali, Mirjalili Seyed Mohammad Mirjalili, andAndrew Lewis. Grey wolf optimizer. Advances in engineering software. 2014. Vol. 69. Pp. 46–61.

142. Hossam, Faris, Ibrahim Aljarah, Mohammed Azmi Al-Betar, and Seyedali Mirjalili. Grey wolf optimizer: a review of recent variants and applications. Neural computing and applications. 2018. Vol. 30(2). Pp. 413–435.

143. Hatta, N. M. Zain Azlan Mohd, Sallehuddin Roselina, Shayfull Z., Yusoff Yusliza. Recent studies on optimization method of grey wolf optimizer (gwo): areview (2014–2017). Artificial Intelligence Review. 2019. Vol. 52(4)/ Pp. 2651–2683. – URL: https://www.researchgate.net/publication/350204693_Grasshopper_Optimization_Algorithm_Theory_Variants_and_Applications [accessed Jun 14 2024].

144. Kureichik. V. V., Kuliev E. V., Kursitys I. O., Ignatyeva S. V. Solving the problem of two-dimensional packing based on the wolf pack algorithm. Information Innovative Technologies. International Scientific. Practical Conference. Moscow, 2021. Pp. 145–152.

145. Kuliev, E. V., Kureichik V. V., Kursitys I. O. Decision making in vlsi components placement problem based on grey wolf optimization. IEEE East-West Design and Test Symposium, EWDTS 2019. P. 8884371.

146. Zaporozhets, D. Y., Kureichik V. V., Kursitys I. O. Wolf pack algorithm for solving vlsi design tasks. Journal of Physics: Conference Series. Cep. "International Conference "Information Technologies in Business and Industry. O 1 - Microprocessor Devices, Telecommunication and Networking". 2019. P. 022009.

147. Кулиев, Э. В. Принятие решений в задаче размещения компонентов СБИС на основе модели поведения стаи волков / Э. В. Кулиев, В. В. Курейчик, И. О. Курситыс // Международная конференция по мягким вычислениям и измерениям. – 2018. – Т. 1. – С. 712–715.

148. Taherdangkoo, M., Shirzadi M. H., Bagheri M. H. Novel meta-heuristic algorithm for numerical function optimization: blind, naked mole-rats (BNMR) algorithm. Scientific Research and Essays. 2012. Pp. 3566–3583.

149. Taherdangkoo, M., Yazdi M., Rezvani, M. H. Mohammad Hossein Shirzadi, M. H. Bagheri, M. A robust clustering method based on blind, naked mole-rats (BNMR) algorithm. Swarm and Evolutionary Computation. 2013. Pp. 1–11.

150. Данильченко, В. И. Метаэвристика на основе поведения колонии белых кротов / В. И. Данильченко, Е. В. Данильченко, В. М. Курейчик // Известия ЮФУ. Технические науки. – 2021. – №6 (223). – С. 132–140.

151. Курейчик, В. В. Модифицированный эволюционный алгоритм mole-rat с адаптивным механизмом динамического обхода препятствий в условиях чрезвычайных ситуаций / В. В. Курейчик, В. И. Данильченко // Информационные технологии. – 2024. – Т. 30. №7. – С. 342–349.

152. Li, L. X., Shao Z. J., Qian J. X. An Optimizing Method Based on Autonomous Animate: Fish Swarm Algorithm. Proceeding of System Engineering Theory and Practice. 2002. Pp. 32–38.

153. Azizi, R. Empirical Study of Artificial Fish Swarm Algorithm. International Journal of Computing, Communications and Networking. Vol. 3. No. 1. January-March 2014. – URL: http://warse.org/pdfs/2014/ijccn013120143.pdf

154. Yazdani, D., Golyari S., Meybodi M. R. A New Hybrid Algorithm for Optimization Based on Artificial Fish Swarm Algorithm and Cellular Learning Automata. 5th International Symposium on Telecommunication (IST). Tehran (2010)/ Pp. 932–937.

155. Yazdani, D., Toosi, A.N., Meybodi, M. R. Fuzzy Adaptive Artificial Fish Swarm Algorithm. 23th Australian Conference on Artificial Intelligent. Adelaide 2010. Pp. 334–343.

156. Hu, J., Zeng X., Xiao J. Artificial Fish Swarm Algorithm for Function Optimization. International Conference on Information Engineering and Computer Science. 2010. Pp. 1–4.

157. Luo, Y., Wei W., Wang S. X. The Optimization of PID Controller Parameters Based on an Improved Artificial Fish Swarm Algorithm. 3rd International Workshop on Advanced Computational Intelligence. 2010. Pp. 328–332.

158. Xiao, L. A Clustering Algorithm Based on Artificial Fish School. 2nd International Conference on Computer Engineering and Technology. 2010. Pp. 766–769.

159. Li, C.X., Ying Z., JunTao S., Qing S.J. Method of Image Segmentation Based on Fuzzy C-means Clustering Algorithm and Artificial Fish Swarm Algorithm. International Conference on Intelligent Computing and Integrated Systems (ICISS). Guilin (2010). Pp. 254–257.

160. Bing, D., Wen D. Scheduling Arrival Aircrafts on Multi-runway Based on an Improved Artificial Fish Swarm Algorithm. International Conference on Computational and Information Sciences. 2010. Pp. 499–502.

161. Song, X., Wang C., Wang J., Zhang B. A Hierarchical Routing Protocol Based on AFSO Algorithm for WSN. International Conference on Computer Design and Applications. 2010. Pp. 635–639.

162. Shi, Y., Eberhart R. C. A Modified Particle Swarm Optimization. IEEE International Conference on Evolutionary Computation Proceedings. Anchorage (1998). Pp. 69–73.

163. Chatterjee, A., Siarry P. Nonlinear Inertia Weight Variation for Dynamic Adaption in Particle Swarm Optimization. Computer and Operations Research, Elsevier Publishers. 2006. Pp. 859–871.

164. Zhan, Z. H., Zhang J., Li Y., Chung H. S. H. Adaptive Particle Swarm Optimization. IEEE Transaction on System, Man and Cybernetics, Part B: Cybernetics. 2009. Pp. 1362–1381.

165. Eberhart, R. C., Shi Y. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. Congress on Evolutionary Computing. 2000. Pp. 84–89.

166. Zheng, Y., Ma L., Zhang L., Qian J. Empirical Study of Particle Swarm Optimizer with an Increasing Inertia Weight. IEEE Congress on Evolutionary Computation (CEC 03). 2003.

167. Eberhart, R. C., Shi Y. H. Tracking and Optimizing Dynamic Systems with Particle Swarms. Proceeding Congress on Evolutionary Computation. 2001.

168. Bastos-Filho Carmelo, Lima Neto Fernando, Lins Anthony, Nascimento Antonio, Lima Marilia. A novel search algorithm based on fish school behavior. IEEE International Conference on Systems, Man and Cybernetics, 2008. Pp. 2646–2651. DOI 10.1109/ICSMC.2008.4811695

169. Eusuff Muzaffar, M., Lansey Kevin E. Optimization of water distribution network design using the shuffled frog leaping algorithm. Journal of Water Resources Planning and Management. 2003. Vol. 129. №3. Pp. 210–225. – URL: https://doi.org/10.1061/(asce)0733-9496(2003)129:3(210)

170. Zhang, X., Hu X., Cui G., Wang Y., Niu Y. An improved shuffled frog leaping algorithm with cognitive behavior. In: Proceedings of the 7th World Congress Intelligent Control and Automation. 2008.

171. Eusuff, M. M., Lansey K. E., Pasha F. Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. Engineering Optimization. 2006. Vol. 38(2). Pp. 129–154.

172. Chung, G., Lansey K. Application of the shuffled frog leaping algorithm for the optimization of a general large-scale water supply system. Water Resources Management. 2008. Vol. 23(4). Pp. 797–823.

173. Eslamian, M., Hosseinian S. H., Vahidi B. Bacterial foraging-based solution to the unit commitment problem. IEEE Trans. Power Syst. 2009. Vol. 24(3). Pp. 1478–1488.

174. Seifollahi-Aghmiuni, S., Bozorg-Haddad O., Omid M. H., Marico M. A. Long-term efficiency of water networks with demand uncertainty. Proceedings of the Institution of Civil Engineers: Water Management. 2001. Vol. 164 (3). Pp. 147–159.

175. Zhao, Y., Dong Z. C., Li Q. H. ANN based on SFLA for surface water quality evaluation model and its application. 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), Changchun, China, December 16–18, Piscataway, NJ: Institute of Electrical and Electronics Engineers (IEEE). 2011. Pp. 1615–1618.

176. Seifollahi-Aghmiuni, S., Bozorg-Haddad O., Omid M. H., Marico M. A. Effects of pipe roughness uncertainty on water distribution network performance during its operational period. Water Resources Management. 2013. Vol. 27(5). Pp. 1581–1599.

177. Mahmoudi, N., Orouji H., Fallah-Mehdipour E. Integration of shuffled frog leaping algorithm and support vector regression for prediction of water quality parameters. Water Resources Management. 2016. Vol. 30(7). Pp. 2195–2211.

178. Balamurugan, R. Application of shuffled frog leaping algorithm for economic dispatch with multiple fuel options. International Conference on Emerging Trends in Electrical Engineering and Energy Management (ICETEEEM), Chennai, Tamil Nadu, India, December 13–15, Piscataway, NJ: Institute of Electrical and Electronics Engineers (IEEE). 2012.

179. Fallah-Mehdipour, E., Bozorg-Haddad O., Marico M. A. Extraction of multi-crop planning rules in a reservoir system: Application of evolutionary algorithms. Journal of Irrigation and Drainage Engineering. 2013. Vol. 139(6). Pp. 490–498.

180. Orouji, H., Bozorg-Haddad O., Fallah-Mehdipour E., Marico M. A. Estimation of Muskingum parameter by meta-heuristic algorithms. // Proceedings of the Institution of Civil Engineers, Water Management. 2013. Vol. 165(1). Pp. 1–10.

181. Bozorg-Haddad, O., Hamedi F., Fallah-Mehdipour E., Orouji H., Marico M. A. Application of a hybrid optimization method in Muskingum parameter estimation. Journal of Irrigation and Drainage Engineering. 2015. Vol. 141(12). 04015026.

182. Elbehairy, H., Elbeltagi E., Hegazy T. Comparison of two evolutionary algorithms for optimization of bridge deck repairs. Comput. Aided Civ. Infrastructure. Eng. 21. 2006. Pp. 561–572.

183. Rahimi-Vahed, A., Mirzaei A. H. Solving a bi-criteria permutation flow-shop problem using shuffled frog-leaping algorithm. In: Soft Computing. Springer-Verlag, New York. 2007.

184. Luo, X.-H., Yang Y., Li X. Solving TSP with shuffled frog-leaping algorithm. Proc. ISDA 3. 2008. Pp. 228–232.

185. Elbeltagi, E., Hegazy T., Grierson D. Comparison among five evolutionary-based optimization algorithms. Adv. Eng. Inf. 2005. Vol. 19(1). Pp. 43–53.

186. Trivedi, I. N., Pradeep J., Narottam J., Arvind K. and Dilip L. Novel Adaptive Whale Optimization Algorithm for Global Optimization. Indian Journal of Science and Technology. 2016 Vol. 9. Pp. 1–6. https://doi.org/10.17485/ijst/2016/v9i38/101939

187. Hu, H., Bai Y., and Xu T. A whale optimization algorithm with inertia weight. WSEAS Transactions Comput 15. 2016. Pp. 319–326.

188. Touma, H. J. Study of the Economic Dispatch Problem on IEEE 30-Bus System Using Whale Optimization Algorithm. International Journal of Engineering Technology and Sciences (IJETS). 2016. Vol. 5. Pp. 11–18.

189. Andrea, R., Blesa M., Blum C. and Michael S. Hybrid Metaheuristics: An Emerging Approach to Optimization, Springer, Berlin.

190. Mirjalili, S. Dragonfly Algorithm: A New Meta-Heuristic Optimization Technique for Solving Single-Objective, Discrete, and Multi-Objective Problems. Neural Computing and Applications, 27. 2016. Pp. 1053–1073. https://doi.org/10.1007/s00521-015-1920-1

191. Manoharan, N., Dash S. S., Rajesh K. S. and Panda S. Automatic Generation Control by Hybrid Invasive Weed Optimization and Pattern Search Tuned 2-DOF PID Controller. International Journal of Computers, Communications & Control, 12. 2017. Pp. 533–549. https://doi.org/10.15837/ijccc.2017.4.2751

192. Mafarja, M. M. and Mirjalili S. Hybrid Whale Optimization Algorithm with Simulated Annealing for Feature Selection. Neurocomputing. 2017. Vol. 260. Pp. 302–312. https://doi.org/10.1016/j.neucom.2017.04.053

193. Khaleel, L. R. and Mitras B. A. Hybrid Whale Optimization Algorithm with Modified Conjugate Gradient Method to Solve Global Optimization Problems. Open Access Library Journal. 2020. Vol. 7: e6459. https://doi.org/10.4236/oalib.1106459

194. Hof, P. R., and Van der Gucht E. Structure of the cerebral cortex of the humpback whale, Megaptera novaeangliae (Cetacea, Mysticeti, Balaenopteridae). The Anatomical Record. 2008. Vol. 290 (1). Pp. 1–31.

195. Goldbogen, J. A., Friedlaender A. S., Calambokidis J., Mckenna M. F., Simon M. and Nowacek D. P. Integrative approaches to the study of baleen whale diving behavior, feeding performance, and foraging ecology. BioScience. 2013. Vol. 63 (2). Pp. 90–100.

196. Bentouati, B., Chaib L., and Chettih S. A hybrid whale algorithm and pattern search technique for optimal power flow problem. In 2016 8th international conference on modelling, identification and control (ICMIC). 2016. Pp. 1048–1053. IEEE.

197. Adhirai, S., Mahapatra R. P. and Singh P. The whale optimization algorithm and its implementation in MATLAB. International Journal of Computer and Information Engineering. 2018. Vol. 12 (10). Pp. 815–822.

198. Simpson, S. J., McCaffery A., Haegele B. F. A behavioral analysis of phase change in the desert locust. Biol Rev. 1994. Vol. 74. Pp. 461–80.

199. Rogers, S. M., Matheson T., Despland E., Dodgson T., Burrows M., and Simpson S. J. Mechanosensory-induced behavioral gregarization in the desert locust Schistocerca gregaria. Journal of Experimental Biology. 2003. Vol. 206 (22). Pp. 3991–4002.

200. Saremi, S., Mirjalili S. and Dong J. S. Grasshopper optimization algorithm: Theory, literature review, and application in hand posture estimation. In Nature-inspired optimizers. 2020. Pp. 107–122. Cham: Springer.

201. Topaz, C. M., Bernoff A. J., Logan S., and Toolson W. A model for rolling swarms of locusts. The European Physical Journal Special Topics. 2008. Vol. 157 (1). Pp. 93–109.

202. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems. 2015. Vol. 89. Pp. 228–249.

203. Shehab, M., Abualigah L., Al Hamad H., Alabool H., Alshinwan M., and Khasawneh A. M. Moth–flame optimization algorithm: Variants and applications. Neural Computing and Applications. 2020. Vol. 32 (14). Pp. 9859–9884.

204. Jangir, N., Pandya M. H., Trivedi I. N., Bhesdadiya R., Jangir P., Kumar A. Moth-flame optimization algorithm for solving real challenging constrained engineering optimization problems. In: 2016 IEEE students' conference on electrical, electronics and computer science (SCEECS). IEEE, 2016. Pp. 1–5.

205. Elsakaan, A. A., El-Sehiemy R. A., Kaddah S. S., Elsaid M. I. An enhanced moth-flame optimizer for solving non-smooth economic dispatch problems with emissions. Energy. 2018. Vol. 157. Pp. 1063–1078.

206. Allam, D., Yousri D., Eteiba M. Parameters extraction of the three diode model for the multi-crystalline solar cell/module using moth-flame optimization algorithm. Energy Convers Manag. 2016. Vol. 123. Pp. 535–548.

207. Hazir, E., Erdinler E. S., Koc K. H. Optimization of cnc cutting parameters using design of experiment (doe) and desirability function. J For Res. 2018. Vol. 29(5). Pp. 1423–1434.

208. Zawbaa, H. M., Emary E., Parv B., Sharawi M. Feature selection approach based on moth-flame optimization algorithm. In: 2016 IEEE congress on evolutionary computation (CEC). IEEE. 2016. Pp. 4612–4617.

209. Trivedi, I., Kumar A., Ranpariya A. H., Jangir P. Economic load dispatch problem with ramp rate limits and prohibited operating zones solve using Levy flight moth-flame optimizer. In: 2016 international conference on energy efficient technologies for sustainability (ICEETS). IEEE. 2016. Pp. 442–447.

210. Wang, M., Chen H., Yang B., Zhao X., Hu L., Cai Z., Huang H., Tong C. Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. Neurocomputing. 2017. Vol. 267. Pp. 69–84.

211. Yousri, D., AbdelAty A. M., Said L. A., AboBakr A., Radwan A. G. Biological inspired optimization algorithms for coleimpedance parameters identification. AEU-Int J Electron Commun. 2017. Vol. 78. Pp. 79–89.

212. Gaston, K. J., Bennie J., Davies T. W., Hopkins J. The ecological impacts of nighttime light pollution: a mechanistic appraisal. Biological reviews. 2013. Vol. 88. Pp. 912–927.

213. Frank, K. D., Rich C., Longcore T. Effects of artificial night lighting on moths, Ecological consequences of artificial night lighting. 2006, Pp. 305–344.

Научное издание

Гладков Леонид Анатольевич
Кравченко Юрий Алексеевич
Курейчик Владимир Викторович
Родзин Сергей Иванович

**ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ:
МОДЕЛИ И МЕТОДЫ
МЕТАЭВРИСТИЧЕСКОЙ ОПТИМИЗАЦИИ**

Монография

Чебоксары, 2024 г.

Компьютерная верстка *А. Д. Федоськина*
Дизайн обложки *М. С. Фёдорова*

Подписано в печать 01.08.2024 г.
Дата выхода издания в свет 12.08.2024 г.
Формат 60×84/16. Бумага офсетная. Печать офсетная.
Гарнитура Times. Усл. печ. л. 13,25. Заказ К-187. Тираж 500 экз.

Издательский дом «Среда»
428005, Чебоксары, Гражданская, 75, офис 12
+7 (8352) 655-731
info@phsreda.com
https://phsreda.com

Отпечатано в ООО «Типография «Новое Время»
428034, Чебоксары, Мичмана Павлова, 50/1