ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

1(197)
2013

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с ноября 1995 г.

УЧРЕДИТЕЛЬ
Издательство "Новые технологии"

СОДЕРЖАНИЕ

Информация о журнале доступна по сети Internet по адресу **http://novtex.ru/IT.**
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

УДК 004.9

С. М. Авдошин, канд. техн. наук, проф.,
e-mail: savdoshin@hse.ru,
М. С. Горбатовский, науч. сотр.,
e-mail: gorbatovskiy@gmail.com,
А. В. Чернов, преподаватель,
Национальный исследовательский университет
"Высшая школа экономики", г. Москва

Интеллектуальная платформа для создания ситуационного центра обеспечения безопасности железнодорожной транспортной инфраструктуры[1]

Предлагается подход к созданию программной платформы для проактивного обеспечения безопасности на железнодорожной инфраструктуре. Обосновывается необходимость методов прогнозирования для предотвращения транспортных инцидентов, а также подход к созданию платформы с учетом огромных объемов разнородных данных, поступающих с транспортной инфраструктуры, и необходимости их обработки в режиме реального времени. Предлагаемый концептуальный подход к созданию платформы, опирающийся на технологии создания промышленных систем ситуационной аналитики, позволит существенно повысить качество прогнозирования и предотвращения нештатных ситуаций на железных дорогах.

Ключевые слова: *безопасность железных дорог, обработка потоков данных, ситуационное реагирование, прогнозирование, транспортный инцидент, анализ в режиме реального времени*

Введение

Развитие транспортной инфраструктуры России формирует основу для конкурентного развития как внутреннего рынка, так и внешнеэкономической деятельности страны. Железнодорожный транспорт обеспечивает существенную долю грузовых и пассажирских перевозок во внешнем и внутреннем сообщении, поэтому задача обеспечения безопасности перевозок и инфраструктуры имеет высокий приоритет на государственном уровне. С ростом интенсивности и объемов перевозок высокий уровень безопасности, основанный на упреждающих мерах по обнаружению и предотвращению нештатных ситуаций, является гарантией сохранения устойчивых конкурентных преимуществ.

Безопасность движения и эксплуатации железных дорог сегодня основана на расследовании вручную ранее произошедших инцидентов и возможных их причин. На его основе подразделениям и хозяйствам готовятся рекомендации по организационным и техническим мерам обеспечения безопасности. И, тем не менее, уровень аварийности на железных дорогах остается значительным, так как оперативный ежедневный контроль за качеством услуг и безопасности сегодня связан с рядом технологических ограничений. Прежде всего, с невозможностью собирать, оценивать и эффективно применять информацию из доступных источников для прогнозирования транспортных инцидентов и их предотвращения. Целью проведенной авторами работы явилось определение оптимальной архитектуры и методов анализа данных на железных дорогах для повышения качества прогнозирования инцидентов с учетом обилия источников данных, территориальной распределенности инфраструктуры и объемов непрерывно создаваемой информации, потребности в быстрой скорости реагирования.

В исследовании также решаются задачи оперативного управления безопасностью движения, в которых время на принятие решения измеряется секундами, минутами и редко часами. Также предложенная архитектура единого комплекса аналитических систем должна обеспечивать проактивное управление и прогнозирование на основе источников данных, уже имеющихся у железнодорожных предприятий России и стран СНГ. Это данные транзакционных систем, датчиков, текстовых сообщений, видеокамер, ручного ввода. Проактивный подход к управлению безопасностью должен следовать принципам принятия решений на основе всего объема доступной информации, истории и статистики ситуаций, быстрого обучения моделей прогнозирования событий и их адаптации к новым угрозам безопасности.

[1] Статья подготовлена в рамках государственного контракта № 07.514.11.4039 на выполнение научно-исследовательских работ по теме: "Исследование и разработка инновационных комплексных моделей интеллектуальной системы ситуационного реагирования и контроля безопасности железных дорог современной России", проводимых по федеральной целевой программе "Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007—2013 годы".

С учетом этих принципов, задачами проводимого авторами исследования является создание высокопроизводительных алгоритмов управления сверхбольшими наборами данных на железных дорогах. Во-первых, это архитектурные подходы, математические методы и алгоритмы, оптимизированные для достижения высокой производительности программных комплексов, работы с большими объемами данных, перекрестного анализа данных из нескольких источников, а также направленные на решение поставленных задач с минимальной степенью участия человека в работе промышленной системы. Во-вторых, это программные компоненты, технологии и инфраструктура, обеспечивающие высокую производительность и параллельность вычислений, записи и чтения, анализа данных. В-третьих, это специализированные программные и программно-аппаратные решения и технологии для обработки и анализа отдельных типов данных, таких как видео-, аудиоданные, геолокационные данные, изображения, текстовая информация.

В статье рассматриваются аспекты концептуальной архитектуры интеллектуальной системы, алгоритмы и методы прогнозирования показателей и контекстно-зависимого анализа, данных, учитывающих информационное окружение объектов и событий в конкретный момент времени, а также один из эффективных методов по машинному обучению системы и ее адаптации к новым условиям.

Концептуальная архитектура и компоненты системы

Главной проблемой информационных систем железных дорог России и стран СНГ является отсутствие возможности обеспечить необходимую производительность при обработке и анализе больших объемов неструктурированных и структурированных данных, а также обеспечить совместный анализ данных из разнородных источников для изучения взаимосвязей между событиями различной природы. Проблема также состоит в существующих ограничениях инфраструктуры консолидации первичных данных, поступающих из регионов России, в частности, вследствие недостаточной пропускной способности существующих сетей передачи данных.

Таким образом, требуется поддержка территориально-распределенных вычислений и хранения информации при централизованном управлении конфигурациями и исходным кодом всей системы. Железнодорожный транспорт России остро нуждается в новых архитектурных подходах и программных решениях, обеспечивающих обработку колоссальных массивов данных в режиме реального времени.

Объемы первичных данных, генерируемых на уровне станции, сегодня таковы, что даже на региональном уровне они консолидируются только в агрегированной форме, а данные, обрабатываемые в вычислительных центрах, по объемам в несколько раз меньше исходных первичных данных, что позволяет решать преимущественно традиционные задачи управления движением поездов. Большая часть задач по обеспечению безопасности на железной дороге связана с обработкой разноплановой первичной информации (телеметрии с датчиков, устройств и систем уровня АСУТП, текстовых сообщений от так называемых "ведомостей", телефонограмм, данных с видеокамер и тепловизоров, координат с GPS/Глонасс-приемников, изображений с видеокамер). Увеличение нагрузки на транспортную сеть, введение в эксплуатацию новых дорогостоящих скоростных поездов связано с новыми угрозами безопасности движения и необходимостью оперативного и упреждающего реагирования на первые признаки нештатной ситуации.

Для решения задач ситуационной аналитики на больших объемах разнородных данных на рынке уже имеются зарекомендовавшие себя технологические комплексы, однако применение их для решения специфических индустриальных задач на железнодорожном транспорте требует дополнительной проработки.

Система, которая обеспечивает сбор, обработку и оценку данных с железных дорог для поддержки оперативного принятия решений и своевременного предотвращения чрезвычайных и нештатных ситуаций, может иметь концептуальную архитектуру, приведенную на рис. 1.

В построении системы используется событийная архитектура, основанная на сборе наблюдений, обогащении их контекстной информацией и получении новых сложных событий верхнего уровня (*complex application level events*), запускающих механизм ответных действий со стороны приложений — потребителей информации. Данные с устройств линейного уровня поступают на распределенную

Рис. 1. Концептуальная архитектура системы

систему *in-memory* анализа потоков данных, фильтруются и обрабатываются на предмет наличия сигналов, позволяющих сделать прогноз. Если прогноз является существенным для принятия решений, он поступает в системы профильных подразделений в форме рекомендации или информационного сообщения. На его основе принимаются решения либо моделируются различные варианты решений. В частности, проводится расчет оптимального расписания движения с учетом новых возможных ограничений. Более долгосрочные прогнозы, доработка моделей угроз и прогнозирования, ретроспективный анализ данных проводятся на стратегическом уровне управления. Для этого целесообразно использовать хранилища сверхбольших объемов данных для запуска алгоритмов анализа исторических данных на достаточной глубине выборки.

Перечень предложенных классов систем решения задач ситуационной аналитики и реагирования на железных дорогах, предлагаемых в рамках концептуального подхода, представлен в таблице.

Предложенная концептуальная архитектура направлена на решение задачи создания программных комплексов, ориентированных на управление большими объемами данных, удовлетворяющих требованиям к скорости реагирования на железнодорожном транспорте и предназначенных для поддержки принятия решений на всех уровнях управления. При этом для эффективного применения новых математических методов анализа и управления в задачах оптимизации и имитационного моделирования при работе с большим объемом данных в распределенных системах потребуется создание новых моделей и разработка подходов к их реализации.

Предлагаемые компоненты решения по ситуационному анализу и реагированию

Компонент архитектуры	Класс системы	Назначение
Уровень операционного управления		
Потоковая обработка сверхбольших объемов данных	Система потоковой обработки данных (*Streams processing engine*)	Сбор, транзакционная обработка и анализ структурированных и неструктурированных цифровых данных со всех доступных источников на распределенной сети в режиме реального времени
Уровень тактического управления		
Имитационное моделирование и поиск оптимальных вариантов	Система транспортной оптимизации	Поиск оптимального решения (например, расписание поездов) с учетом имеющихся ограничений по выбранным критериям на больших объемах данных
Уровень стратегического управления		
Анализ, прогнозирование и планирование	Система интеллектуального анализа данных	Применение широкого спектра математических методов к исследованию данных, скоринг, выявление зависимостей и создание моделей прогнозирования
	Хранилище больших наборов данных	Хранение и ретроспективный анализ больших наборов разнородных данных. Многофакторный анализ данных для обновления моделей обнаружения и алгоритмов оптимизации

Алгоритмы и методы анализа данных

Математические алгоритмы и методы, используемые для прогнозирования нештатных ситуаций, как правило, построены на основе задач классификации и секвенциального анализа [1]. При построении классификационных моделей принимается во внимание необходимое число атрибутов, а затем проводится группировка в соответствии с заранее определенными классами. Это позволяет, с одной стороны, максимально эффективно использовать информационное наполнение потоков данных, с другой стороны, устраняет несущественные или ненужные для задачи детали. Далее к структурированным группам применяют методы секвенциального анализа, которые служат для выявления частых последовательностей и построения правил и прогнозов на их основе. Дадим формальные определения задачам классификации и секвенциального анализа.

Пусть n_C — набор классов, по которым нужно классифицировать данные. Алгоритм классификации строит модель, которая каждому неопределенному набору данных I соотносит класс C, к которому I, с определенной вероятностью, принадлежит. В частности, если имеется набор из N тренировочных примеров вида (x, y), где y — единичный класс, а x — набор из d атрибутов, каждый из которых может быть символьным или числовым, то в результате классификации из тренировочных примеров должна быть построена модель f, которая будет определять класс $y = f(x)$ будущих примеров с высокой точностью.

Будем называть транзакцией отклики с обобщенных датчиков железной дороги (ЖД), пришедшие на вход системы обработки данных за единицу времени. После классификации каждая транзакция T будет приписана к своему классу. Последовательностью S назовем упорядоченное множество вида:

$$S = \{..., i_p, ..., i_q, ...\},$$

где i — событие (объект); $p < q$. Говорят, что транзакция T содержит последовательность S, если $S \subseteq T$ и объекты, входящие в S, входят и в множество T с сохранением отношения порядка. При этом допускается, что в множестве T между объектами из последовательности S могут находиться другие объекты.

Поддержкой последовательности S называется отношение числа транзакций, в которое входит последовательность S, к общему числу транзакций.

Последовательность называется частой, если ее поддержка превышает минимальную поддержку, заданную пользователем:

$$Supp(S) > Supp_{\min}.$$

Таким образом, задачей секвенциального анализа является поиск частых последовательностей.

Решение задачи обработки данных и прогнозирования не ограничено лишь описанными двумя методами. Например, возможно применение элементов теории надежности систем, когда вместо деревьев решений (одного из методов классификации), строится дерево аварий, а для прогнозирования — дерево событий. Перейдем к рассмотрению алгоритмов и методов на основе описанных подходов.

Алгоритмы прогнозирования событий в режиме реального времени

Для задач потоковой классификации наиболее популярным на сегодняшний день является алгоритм построения деревьев Хеффдинга [2]. Это инкрементный индукционный алгоритм построения деревьев решений с возможностью обучения из потока данных в предположении, что генерация распределения примеров не меняется. Деревья Хеффдинга основаны на факте того, что малого количества выборок может быть достаточно для выбора оптимального атрибута ветвления.

Математически это закреплено в виде критерия Хеффдинга. Реальное значение некоторой величины диапазона R, с вероятностью $1 - \delta$ после n независимых просмотров не будет отличаться от оценочного значения более чем на

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}.$$

Деревья Хеффдинга имеют теоретические гарантии, выражающиеся в нижеследующей теореме. Назовем интенсиональным рассогласованием Δ_i между двумя деревьями решений DT_1 и DT_2 вероятность того, что путь трансфера примера через DT_1 отличается от пути этого примера через DT_2.

Теорема 1. Если HT_δ — дерево решений, полученное с помощью алгоритма Хеффдинга с вероятностью несовпадения оценок δ, а DT — асимптотическое дерево и p — вероятность появления листа, то $E[\Delta_i(HT_\delta, DT)] \leqslant \delta/p$.

Для секвенциальной задачи прогнозирования чаще всего используют математическое моделирование на основе регрессионных моделей и элементы теории вероятности [3]. При поиске аномалий в транзакциях используют дискретное вейвлет-преобразование [4].

Наибольший интерес представляет регрессионный алгоритм прогнозирования FTP-DS [7]. Он достаточно компактен и требует только одного просмотра на транзакцию. В соответствии с регрессионным анализом, если по имеющимся примерам (событиям с обобщенных датчиков ЖД) найти отношения между транзакциями, то можно предсказать частоту появления интересующих паттернов [5].

Прямолинейный отрезок для временного ряда транзакций $s(t)$, который имеет смысл частоты появления паттерна, можно записать в виде:

$$\hat{f} = \hat{\alpha} + \hat{\beta}(t).$$

Эта зависимость удовлетворяет принципу наименьших квадратов, т. е. регрессионные параметры $\hat{\alpha}$ и $\hat{\beta}$ должны минимизировать соотношение

$$D = \sum_{i=1}^{n} (f_i - \hat{\alpha} - \hat{\beta}(t))^2,$$

где f_i — частота в i-й точке. Для того чтобы наилучшим образом определить $\hat{\alpha}$ и $\hat{\beta}$, проводят следующие вычисления:

$$\bar{t} = \frac{1}{n}\sum t,\ \bar{f} = \frac{1}{n}\sum f;$$

$$S_{tt} = \sum(t - \bar{t})^2 = \sum t^2 - \frac{(\sum t)^2}{n};$$

$$S_{ff} = \sum(f - \bar{f})^2 = \sum f^2 - \frac{(\sum f)^2}{n};$$

$$S_{tf} = \sum(t - \bar{t})(f - \bar{f}) = \sum tf - \frac{(\sum t)(\sum f)}{n}.$$

Тогда $\hat{\alpha}$ и $\hat{\beta}$ можно вычислить следующим образом:

$$\hat{\alpha} = \bar{f} - \hat{\beta}\,\bar{t},\ \hat{\beta} = \frac{S_{tf}}{S_{tt}}.$$

Мера линейного отношения, или коэффициент определенности, записывается в следующем виде:

$$r^2 = \frac{(S_{tf})^2}{S_{tt}S_{ff}}.$$

Далее вычисленная частота появления паттерна сравнивается с минимальной поддержкой и принимается решение.

При использовании принципов из теории надежности систем возможно для конкретного показателя, например показателя аварийности ЖД, рассчитать его мгновенные значения по приходящим данным с обобщенных датчиков, а затем, для выдачи прогноза, по вычисленным вероятностям отказов либо промоделировать дерево событий, либо использовать алгоритмы секвенциального анализа. Расчет мгновенного интегрального пока-

зателя аварийности инфраструктуры ЖД (МИА ЖД) может быть выполнен следующим образом:

$$AO_j = \sum_{i=1}^{n} \alpha_i \times АД_i;$$

$$AO_j^{\text{reg}} = \overline{1..3};$$

$$МИА^{\text{reg}} = \sum_{j=1}^{k} \beta_j \times AO_j^{\text{reg}};$$

$$МИА_{\text{ЖД}} = \sum_{k=1}^{l} \gamma_k \times МИА^{\text{reg}},$$

где AO_j — аварийность j-го объекта; $АД_i$ — аварийность i-го датчика; α_i — весовой коэффициент, вклад каждого из i-х датчиков в аварийность всего объекта; AO_j^{reg} — аварийность j-х объектов региона; имеет три уровня опасности: первый и второй уровни находятся в диапазоне [0..1], а третий — бинарный; β_j — весовой коэффициент, вклад, вносимый определенным объектом (переезд, перегон и пр.) в интегральную аварийность региона, он должен быть посчитан на основе статистических данных частоты аварий на определенных объектах (например, за год), но можно задаться им самостоятельно, на основе экспертной оценки; $МИА^{\text{reg}}$ — мгновенная интегральная аварийность региона; γ_k — весовой коэффициент, рассчитывается на основе протяженности линий регионов; $МИА_{\text{ЖД}}$ — мгновенная интегральная аварийность всей железной дороги.

Математические алгоритмы и методы, используемые для прогнозирования нештатных ситуаций, могут основываться как на наборах "жестких" правил и зависимостей, так и на "гибких" поведенческих шаблонах, позволяющих обнаруживать аномалии в поведении объектов, причем эти методы необходимо применять в промышленных системах для работы с несколькими видами источников, дополняющих информационную картину событий. Поэтому для алгоритмов потокового анализа важно уметь адаптироваться к изменению во входных данных или к изменению параметров работы. Для этого, в частности, могут использоваться нейронечеткие методы [1]. Как правило, это методы, основанные на нейронных сетях и генетических алгоритмах. В этом случае алгоритмы нечеткой логики будут агрегировать в себе и методы потокового анализа, и нейронечеткие методы для "тонкой настройки" работы системы. Концептуальная схема блока потокового анализа данных показана на рис. 2.

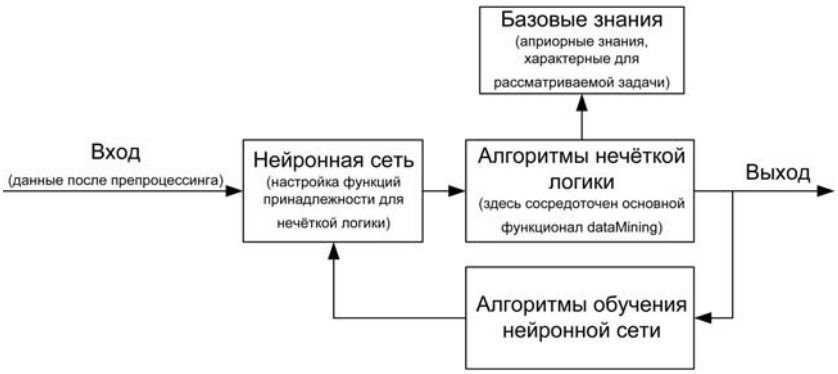


Рис. 2. Схема нейронечеткой сети анализа данных

Рис. 3. Концептуальная архитектура цикла контекстно-зависимого анализа данных

Цикл контекстно-зависимого анализа

Контекстно-зависимый анализ — один из наиболее перспективных подходов к обработке данных. Смысл технологии заключается в том, что пользователь (человек, система, устройство) приобретает целостное представление об окружающей среде и событии и использует его при дальнейшей деятельности. Для того чтобы составить комплексное знание об окружающей среде, необходим детальный анализ множества источников информации. Окружающая среда — это и есть контекст, т. е. любая информация, которая может описать состояние сущности. Сущность — это человек, место или объект, которые имеют какое-либо отношение к пользователю или к приложению (Dey et al., 1999). При этом контекстом часто считается именно скрытая, неочевидная информация. В этом и состоит особенность технологии. Система учитывает не только явные сигналы от пользователей, но и любую информацию, которую ей удалось получить о них, об окружающей среде.

Цикл контекстно-зависимого анализа — это непрерывный процесс выявления явных и неявных взаимосвязей, построения и проверки гипотез, уточнения на основе сделанных проверок понимания причинно-следственных связей между событиями. Авторами были сформулированы основные этапы этого цикла (рис. 3).

В процессе изучения контекста, информационного окружения событий и сущностей можно выделить следующие этапы.

Наблюдения. Под наблюдениями понимаются любые новые (первичные) данные, поступающие из всех доступных источников. Данные могут быть сгенерированы как устройством (например, видеопоток), так и пользователем (например, сообщения диспетчеров).

Извлечение существенной информации. Этап извлечения существенной информации, или контекста, из потока данных и вопросов является ключевым при работе с большими объемами данных, поскольку позволяет провести предварительную очистку и трансформацию данных перед их загрузкой в хранилище, а также исключить из первичного потока данные, потребность в хранении которых отсутствует или минимальна (например, при анализе видеопотока). В потоке сигналов, формируемых каким-либо источником, выделяются дискретные события, которые представляются в виде последовательного набора атрибутов (значение сигнала, время, режим работы устройства и т. д.).

Контекстный анализ данных. На этом этапе обработка событий происходит в системах, построенных по принципу событийной архитектуры и контекстно-зависимого анализа, позволяющего соотносить новую информацию с уже имеющейся, оценивать ее значимость, формировать аналитические результаты для быстрого реагирования в режиме реального времени. Это позволяет формировать более полное и достоверное понимание явлений благодаря сбору и анализу относящейся к их окружению информации.

Хранимая информация. Хранение больших наборов данных имеет несколько важных особенностей по сравнению с классическими подходами, применяемыми при работе со значительно меньшими объемами данных. Выделяют три ключевых области, характеризующих обработку больших наборов хранимых данных:

- инфраструктурная, ее составляют базы данных, способные к эффективному управлению большими объемами данных (сегодня здесь набирают популярность технологии NoSQL);
- аналитическая, предоставляющая методы обработки данных (в решаемой авторами задаче используется платформа Hadoop и метод MapReduce);
- облачная, естественным образом снижающая затраты на создание и поддержку (ключевым здесь является поддержка выбранным провайдером стандартизованных методов доступа, таких как HTTP, REST, SOAP).

Публикация. На данном этапе потребитель получает результат аналитической обработки. Результат в виде рекомендации диспетчеру или управляющего воздействия поступает либо на рабочие места сотрудников диспетчерских и ситуационных центров в виде уведомлений руководству, либо в форме сигнала другому приложению выполнить некоторые действия.

Реализация функций обнаружения аномальных событий и оптимизации на практике связана с невозможностью досконально описать правила, на основе которых будут обнаруживаться признаки чрезвычайной ситуации. В связи с этим модель обнаружения и прогнозирования должна ориентироваться не только на "жесткие" правила, но и на статистические методы обнаружения нормального и аномального поведения объектов системы. Также должен быть организован цикл обучения системы и корректировки моделей в случае низкого качества прогнозов и появления дополнительных факторов, определяющих развитие ситуации.

Цикл машинного обучения системы

В промышленной эксплуатации созданные модели применяют по отношению ко всем коммуникациям с клиентами. Несколько технологий анализа потоковых данных (*streams processing engine*) позволяют загружать в промышленную среду модели, созданные с помощью различных пакетов для исследования данных, таких как SPSS, SAS, R. Это позволяет применять сотни миллионов моделей в ходе взаимодействия с огромным числом объектов транспортной инфраструктуры и индивидуализировать этот процесс. Скорость применения таких моделей измеряется миллисекундами.

Предлагаемую архитектуру планируется реализовать на базе промышленного пакета исследования данных и хранилища больших объемов данных на базе Hadoop в силу наличия вариантов готовой интеграции с системами потокового анализа данных. Для того чтобы приступить к выполнению процедуры анализа данных, системы потокового анализа должны сначала наполнить хранилище данных информацией, соответствующей требованиям системы прогнозирования. Далее в пакетном режиме выполняется изучение данных с применением универсальных методов анализа (анализ связей между переменными, параметрические и непараметрические методы, регрессии, общие линейные модели и т. д.) и специальных методов (метод временных рядов, прогнозирование количественных и категориальных исходов, моделирование сложных взаимосвязей, нейронные сети).

Полученная на выходе модель загружается через *API* в систему анализа потоковых данных (*streams processing engine*). В случае падения качества модели дается сигнал к новому циклу автоматизированного анализа данных, в котором на исторических данных рассчитывается более надежная модель системы. Результаты анализа данных передаются в модель исследования и прогнозирования, а затем создаются автоматические задания, поддерживающие

Рис. 4. Непрерывный цикл прогнозной аналитики

процесс принятия решения. Запуск заданий на исследование данных и принятие решений по подготовленным алгоритмам реализуется в двух вариантах (рис. 4).

В первом случае задания, сформированные в системе прогнозирования и анализа данных, через API выгружаются в систему обработки потоковых данных. Это позволяет обнаруживать аномалии, выполнять прогнозирование и аналитику в режиме реального времени с низкими задержками и без использования хранилища данных.

Во втором случае задания запускают для обработки хранимой информации в пакетном режиме, когда требуется углубленный анализ исторической информации. Это делается, например, когда нужно повысить качество модели или автоматически обновить ее с учетом новых факторов.

Заключение

Предложенный подход к оперативному обеспечению безопасности на железнодорожной транспортной инфраструктуре, основанный на прогнозировании и проактивных действиях по предотвращению инцидентов и нештатных ситуаций, имеет следующие преимущества:

- учитываются особенности текущей инфраструктуры железных дорог России и СНГ, связанные с географической распределенностью, большим набором систем и различных типов данных, больших объемов данных при ограниченных возможностях каналов их передачи;
- обеспечивается своевременный мониторинг и реагирование на события за счет минимизации задержек в анализе и представлении информации;
- создается прецедент выстраивания взаимосвязей между существующими методиками ретроспективного факторного анализа транспортных инцидентов и контекстными особенностями ситуации в реальный момент времени;
- обеспечивается учет максимального числа факторов, определяющих моментальный контекст события, и выбор оптимального процесса реагирования на это событие с учетом данного контекста.

Представленная архитектура, методы обработки, анализа и представления информации должны обеспечить существенно большую эффективность по предотвращению транспортных инцидентов на железных дорогах, чем существующие подходы.

На данный момент в железнодорожном транспорте России и СНГ не было аналогичных проектов, а конкретные алгоритмы будут реализованы в ближайшее время в рамках текущих исследовательских работ.

Список литературы

1. **Барсегян А. А., Куприянов М. С., Степаненко В. В., Холод И. И.** Технологии анализа данных. СПб.: БХВ-Петербург, 2007. С. 58—89.

2. **Domingos P., Hulten G.** Mining high speed data streams // Proc. of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining, 2000. P. 71—80.

3. **Gaber M. M., Zaslavsky A., Krishnaswamy S.** Mining Data Streams: A Review // Newsletter ACM SIGMOD Record. June 2005. Vol. 34, Is. 2. P. 18—26.

4. **Papadimitriou S., Brockwell A., Faloutsos C.** AWSOM: Adaptive, Hands-Off Stream Mining // VLDB: Proc. of the 29th International Conference on Very Large Data Bases. 2003. Vol. 29. P. 560—571.

5. **Wei-Guang Teng, Ming-Syan Chen, Philip S. Yu.** A regression-based temporal pattern mining scheme for data streams // VLDB: Proc. of the 29th International Conference on Very Large Data Bases. 2003. Vol. 29. P. 93—104.

6. **Chen Y., Dong G., Han J., Wah B. W., Wang J.** Multi-Dimensional Regression Analysis of Time-Series data streams // VLDB: Proc. of the 28th International Conference on Very Large Data Bases. 2002. P. 323—334.

УДК 004.934

Я. Н. Имамвердиев, канд. техн. наук, зав. отд.,
Л. В. Сухостат, аспирант,
Институт информационных технологий НАНА,
Баку, Азербайджан,
e-mail: yadigar@lan.ah.az, lsuhostat@hotmail. com

Метод оптимизации показателя распознавания в мультибиометрических системах

Эффективные методы слияния информации являются актуальной задачей в мультибиометрических системах. Рассматривается метод оптимального слияния значений соответствия путем максимизации площади под ROC-кривой. Для оптимизации целевой функции в работе применяется метод роя частиц. Эксперименты проводили с использованием трех открытых баз данных значений соответствия: NIST BSSR1, XM2VTS-Benchmark и BANCA. Предложенный метод существенно улучшает проверку идентичности в мультибиометрических системах.

Ключевые слова: *мультибиометрическая система, слияние значений соответствия, площадь под ROC-кривой, метод роя частиц*

Введение

Биометрические технологии широко используются в системах различного назначения для аутентификации и идентификации человека. Унимодальные биометрические системы (т. е. системы, использующие только одну биометрическую характеристику) имеют ряд недостатков, создающих трудности при крупномасштабных применениях биометрических систем [1]. Среди них можно выделить высокую чувствительность к атакам спуфинга (обман системы фальшивыми биометрическими образцами) [2], отсутствие или нахождение биометрических характеристик у некоторой части людей в нежелательном качестве для работы биометрических систем, высокие значения вероятностей ложного допуска (*False Accept Rate*, *FAR*) и ложного отказа (*False Reject Rate*, *FRR*).

Мультибиометрические системы (МС) повышают устойчивость к атакам спуфинга и уменьшают ошибки *FAR* и *FRR* по сравнению с унимодальными биометрическими системами. Но разработка и внедрение МС в свою очередь создают ряд проблем. К ним относятся архитектура системы, методология слияния, выбор биометрических компонентов на основе их точности и различия, оценка их качества, надежности и компетентности [3—5].

Для усовершенствования возможностей распознавания личности с минимальной ошибкой в МС возникает задача эффективного слияния информации с учетом различных критериев [3—5].

Слияние информации может быть осуществлено на разных уровнях биометрической системы [1, 6]: на уровне сенсоров; на уровне признаков, на уровне значений соответствия; на уровне рангов и на уровне решений. Наиболее часто используется слияние на уровне значений соответствия. Для слияния значений соответствия существуют три группы методов [7]:

1. Методы на основе преобразований [8]: правила суммы, произведения, min, max и т. д. По результатам экспериментов [9], среди них наилучшим является правило суммы.

2. Методы на основе плотности распределения. Они основаны на критерии правдоподобия и требуют явной оценки плотности распределения. По заключению работы [10] "произведение отношений правдоподобия является наиболее точным, но и наиболее трудным для реализации" и "трудность этой реализации заключается не в самом слиянии, а в моделировании распределений".

3. Методы на основе классификаторов [11] — значения соответствия, полученные от разных модулей сравнения, принимаются как вектор признаков, и строится классификатор для распознавания подлинных и злоумышленных значений соответствия.

Наряду с этими методами были также предложены подходы на основе оптимизации различных метрик качества классификации [12, 13]. Общепринятой метрикой для измерения качества бинарных классификаторов в машинном обучении является *ROC*-кривая (*receiver operator characteristic* — рабочая характеристика (приемника)) [14]. В существующих работах процессы оптимального слияния и оценка качества классификации проводятся в отдельности, так как *ROC*-кривая и разработка оптимального слияния не имеют явной, структурированной связи, которую можно было бы легко вычислить.

В этой работе предлагается метод оптимизации показателей качества распознавания МС непосредственно в ходе разработки функции слияния. Для аналитического представления зависимости показателей распознавания от функции слияния предлагается параметрическая линейная модель, позволяющая рассматривать широкий класс сложных стратегий слияния. В качестве целевой функции предлагается использовать площадь под *ROC*-кривой (*area under ROC-curve*, *AUC*) [15] ввиду ее хорошей способности адекватного представления качества классификации. Так как не удается в явном виде выразить зависимость теоретической *AUC* от показателей качества классификации, то эмпирическая *AUC* аппроксимируется суммой недифференцируемых функций [16], поэтому для ее оптимизации используется метод роя частиц (*particle swarm optimization*, *PSO*) [17].

Площадь под *ROC*-кривой

ROC-кривая показывает зависимость числа верно классифицированных положительных примеров (*true positive rate*, *TPR*) и от числа неверно классифицированных отрицательных примеров (*false positive rate*, *FPR*) [18].

Пусть $y \in \{0, 1\}$ — бинарная метка классов, $x \in R^n$ — вектор признаков, $g_0(x)$ и $g_1(x)$ — функции плотности распределения для классов 0 и 1 соответственно. Объект с вектором признаков x классифицируется в класс 1, если значение функции соответствия $F(x)$ больше или равно пороговому значению τ, и в класс 0 — в противном случае. Характеристики качества классификации *TPR* и *FPR* определяются так:

$$FPR(\tau) = \int\limits_{F(x) \geqslant \tau} g_0(x)dx,\ TPR(\tau) = \int\limits_{F(x) \geqslant \tau} g_1(x)dx. \quad (1)$$

ROC-кривая графически отображает зависимость

$$ROC = \{(FPR(\tau),\ TPR(\tau) | \tau \in R)\},$$

которая показана на рис. 1. *AUC* является агрегированной характеристикой качества классификации, чем больше ее значение, тем эффективнее работает классификатор. Теоретически *AUC* изменяется от 0 до 1,0, так как любой адекватный классификатор должен дать большее число *TPR*, чем *FPR*, то нецелесообразно рассмотрение *ROC*-кривой, находящейся ниже прямой $y = x$, и обычно говорят об изменениях от 0,5 ("случайное предположение") до 1,0 ("идеальный" классификатор). Идеальная *ROC*-кривая проходит через точки (0, 0), (0, 1), (1, 1).

Используя выражение (1), *AUC* можно записать так:

$$AUC(F) = \int\limits_{-\infty}^{\infty} TPR(c)dFPR(c). \quad (2)$$

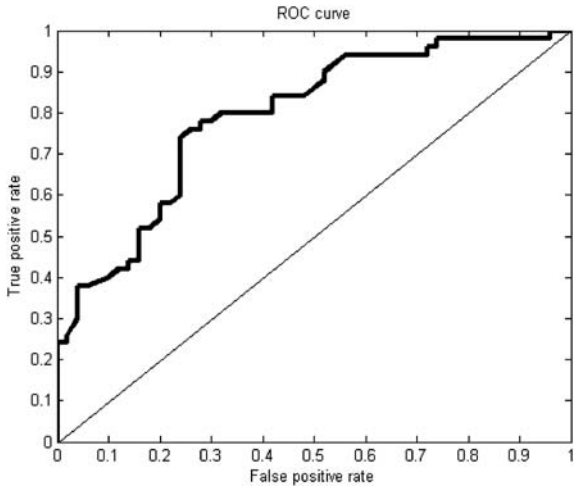


Рис. 1. Пример *ROC*-кривой. Диагональная линия является случайным классификатором

Чем дальше $g_0(x)$ от $g_1(x)$, тем *AUC* ближе к 1. *AUC* также зависит от функции $F(x)$, которая определяется из анализа данных. Наилучшее значение *AUC* можно получить только при использовании адекватной $F(x)$ для функций плотности $g_0(x)$ и $g_1(x)$.

Формулу (2) можно выразить по-другому [16]:

$$AUC(F) = P(F(X_1) \geqslant F(X_0)), \quad (3)$$

где X_0, X_1 — независимые n-мерные случайные векторы из классов 0 и 1 соответственно. Для заданных наблюдений $\{x_{0i}: i = 1, ..., n_0\}$ класса 0 и $\{x_{1j}: j = 1, ..., n_1\}$ класса 1 эмпирическая *AUC* выражается формулой

$$\overline{AUC}(F) = \frac{1}{n_0 n_1} \sum_{i=1}^{n_0} \sum_{j=1}^{n_1} H(F(x_{1j}) - F(x_{0i})), \quad (4)$$

где $H(z)$ — функция Хевисайда: $H(z) = 1$, если $z \geqslant 0$, иначе $H(z) = 0$. Для случая дискретной $F(x)$ функция $H(z)$ заменяется на $H^*(z)$, которая определяется так: $H^*(z) = 1$ при $z > 0$, $H^*(z) = \frac{1}{2}$ при $z = 0$ и $H^*(z) = 0$ для $z < 0$.

На практике максимизация эмпирической *AUC* методами, основанными на градиентном подходе [19], представляет трудности, так как она является суммой недифференцируемых функций. Одним из путей преодоления этой трудности является использование аппроксимации функции Хевисайда гладкими функциями. В работе [20] используется стандартная функция нормального распределения, а в работе [21] — сигмоидная функция.

Модель оптимизации *AUC*

Нашей целью является нахождение оптимальной функции слияния в смысле максимизации *AUC* в классе функций слияния. Как известно, биометрическое распознавание эквивалентно бинарной классификации, решается вопрос принадлежности индивидуума к одному из двух классов: законный пользователь (*genuine-user*) или злоумышленник (*impostor*). Пусть имеется обучающая выборка $(X_1, y_1), ..., (X_m, y_m)$, где $X_i \in R^n$ — n-мерный вектор признаков (значения соответствия из классификаторов), $y_i \in \{0, 1\}$, $i = 1, ..., m$ — метка классов.

Пусть $F: R^n \to R$ — функция слияния значений соответствия, которая отображает вектор признаков к скалярному значению для принятия решения о классификации. Предположим, что $g(X)$ дает непрерывные значения, и для классификации используется пороговое значение τ, чтобы присвоить каждому входному вектору признаков метку *законного пользователя* и *злоумышленника*. Для заданного τ

метку класса, присвоенную входному вектору признаков X_i можно выразить так:

$$cls(F(X_i)) = \begin{cases} 1(genuine - user), \text{ if } F(X_i) \geqslant \tau \\ 0\ (impostor), \text{ if } F(X_i) < \tau. \end{cases} \quad (5)$$

В качестве функции слияния будем использовать следующую параметрическую линейную модель:

$$F_b(X) = b_1\xi_1(X) + ... + b_r\xi_r(X), \quad (6)$$

где $r \geqslant 1$, $\xi_1(\cdot)$, ..., $\xi_r(\cdot)$ — известные функции; b_1, ..., b_r — неизвестные параметры, которые оцениваются на основе обучающей выборки. На функции $\xi_1(\cdot)$, ..., $\xi_r(\cdot)$ не налагаются никакие ограничения. Так как целесообразно нормализовать значения соответствия к интервалу [0, 1], то функции $\xi_1(\cdot)$, ..., $\xi_r(\cdot)$ можно определить в этом интервале. Выбирая разные функции $\xi_1(\cdot)$, ..., $\xi_r(\cdot)$, можно моделировать различные стратегии слияния значений соответствия. Если выберем $r = n$, $\xi_1(X) = x_1$, ..., $\xi_2(X) = x_2$, ..., $\xi_n(X) = x_n$, где x_1, x_2, ..., x_n — компоненты n-мерного вектора признаков X, то получим взвешенную сумму значений соответствия. Выбирая подходящим образом параметрические функции в модели (6), можно смоделировать обобщенную аддитивную модель [22], а также различные функции для классификации.

Нашей целью является нахождение таких значений коэффициентов b_1, ..., b_r, чтобы они давали наилучшую эмпирическую *ROC*-кривую для выбранных функций $\xi_1(\cdot)$, ..., $\xi_r(\cdot)$ и заданной обучающей выборки:

$$B = \text{argmax}(\overline{AUC}\ (F_b)). \quad (7)$$

Известно, что вероятность ошибки минимизируется правилом Байеса, которое можно выразить с помощью строго возрастающей функции от отношения правдоподобия. Согласно лемме Неймана—Пирсона [23], *ROC*-кривая для произвольной функции слияния всюду находится ниже *ROC*-кривой для отношения правдоподобия. Таким образом, оптимальную функцию слияния, максимизирующую *AUC*, можно определить в виде:

$$F(x) = m(\Lambda(x)), \quad (8)$$

где $\Lambda(x) = g_1(x)/g_0(x)$ и m — строго возрастающая функция. Это показывает, что максимизация *AUC* эквивалентна минимизации вероятности ошибки в смысле правила Байеса.

Используя методику, аналогичную [16], можно доказать следующее:

$$AUC(F_b) < AUC(\Lambda). \quad (9)$$

Это неравенство является строгим и никакой функцией F_b не удается достичь равенства. Поэтому можно выполнить только супремизацию взамен максимизации [24]. Это свойство не предпочтительно при построении итеративного алгоритма для максимизации, и для преодоления этой трудности предлагается следующая схема регуляризации:

$$B = \text{argmax}\left(\overline{AUC}\ (F_b) - \lambda \sum_{i=1}^{r} b_i^2 \right). \quad (10)$$

Цель работы — нахождение оптимальной функции слияния, представленной параметрической линейной моделью (6) с параметрами, максимизирующими *AUC* с регуляризацией (10) при заранее выбранных функциях $\xi_1(\cdot)$, ..., $\xi_r(\cdot)$.

Метод *PSO*

Метод *PSO* является одним из эффективных методов решения задач глобальной оптимизации. *PSO* впервые был предложен в 1995 г. Дж. Кеннеди и Р. Эберхартом [17] для оптимизации нелинейных функций, впоследствии он применялся к множеству различных задач, и были предложены многочисленные его модификации [25]. Он основан на моделировании группового поведения (стаи птиц или косяка рыб). *PSO* опирается на обмен информацией между индивидуумами (частицами) популяции (роя). Каждая частица регулирует свою траекторию относительно своей наилучшей предыдущей позиции (*pbest* — personal best) и наилучшей предыдущей позиции, достигнутой в ее локальном соседстве (*gbest* — global best). Действия частиц оцениваются согласно предопределенной функции пригодности. В случае оптимизации в ее роли обычно выступает целевая функция.

Начальную популяцию *PSO* можно сгенерировать случайно, также можно использовать генератор последовательности Соболя, который гарантирует равномерное распределение многомерных векторов в пространстве решений [26].

Текущее состояние частицы характеризуется координатами в пространстве решений. Если пространство решений d-мерное, i-ю частицу роя представим d-мерным вектором $X_i = (x_{i1}, ..., x_{id})$, наилучшую частицу роя (имеющую максимальное значение функции) обозначим индексом g. Наилучшее состояние i-й частицы, соответствующее наибольшему значению функции, обозначим $P_i = (p_{i1}, ..., p_{id})$, а скорость i-й частицы (изменение состояния) — $V_i = (v_{i1}, ..., v_{id})$.

PSO является итеративным процессом. Обновление векторов скоростей и позиций частиц осуществляется по следующим формулам (верхний индекс показывает итерацию):

$$V_i^{k+1} = wV_i^k + c_1 r_{i1}^k (P_i^k - X_i^k) + c_2 r_{i2}^k (P_g^k - X_i^k); \quad (11)$$

$$X_i^{k+1} = X_i^k + P_i^k,$$

где $i = 1, ..., N$ и N — размер популяции; w — коэффициент инерции; c_1 и c_2 — положительные константы, называемые когнитивным и социальным параметрами, выбираются из интервала [0, 2]; r_{i1} и r_{i2} — случайные числа, равномерно распределенные в интервале [0, 1].

Коэффициент инерции w является важным для сходимости *PSO*, он контролирует влияние предыдущей истории скоростей на текущую скорость. Эксперименты показывают, что целесообразно выбирать вначале большое значение w, стимулируя глобальное исследование пространства поиска, а затем, постепенно уменьшая, получить оптимальное решение за меньшее число итераций.

Экспериментальные результаты

Для экспериментальной проверки предложенного подхода были использованы открыто доступные мультимодальные биометрические базы данных значений соответствия NIST BSSR1, XM2VTS-Benchmark и BANCA. Краткая характеристика этих баз данных приведена в табл. 1.

NIST BSSR1 (Biometric scores set — release 1 — Биометрический набор значений соответствия — Выпуск 1) [27] состоит из трех больших наборов значений соответствия отпечатков пальцев и изображений лица, сами изображения лица и отпечатки пальцев недоступны. Одно значение соответствия отпечатка пальца было получено путем сравнения пары образцов от левого указательного пальца, а другое — путем сравнения образцов указательного пальца правой руки. Два разных изображения лица были применены для вычисления сходства между двумя фронтальными изображениями лица. Таким образом, есть четыре значения соответствия для каждого субъекта (по одному для каждой модальности).

XM2VTS-Benchmark [28] представляет собой базу данных значений соответствия, полученных из экспериментов на мультимодальной базе данных XM2VTS, которая содержит записи речи 295 субъектов. Речь каждого человека записывалась за четыре сессии, каждая сессия состояла из двух попыток, включающих три произнесения. N. Poh [28] разбивает данные из XM2VTS на группы: 200 пользова-

Таблица 1

Характеристики мультимодальных баз данных

База данных	Биометрическая характеристика	Число индивидуумов	Число значений соответствия
NIST-face (Set 1)	Лицо (2 системы)	3000	2 × 3000 × 3000
NIST-fingerprint (Set 2)	Отпечаток пальца (2 отпечатка)	6000	2 × 6000 × 6000
NIST-multimodal (Set 3)	Отпечаток пальца (2 отпечатка) Лицо (2 системы)	517	2 × 517 × 517
XM2VTS-Benchmark	Лицо (5 систем) Голос (3 системы)	295	8 × 295 × 200
BANCA	Лицо (2 системы) Голос (2 системы)	52	2 × {26 × [26 + (26 − 1)]}

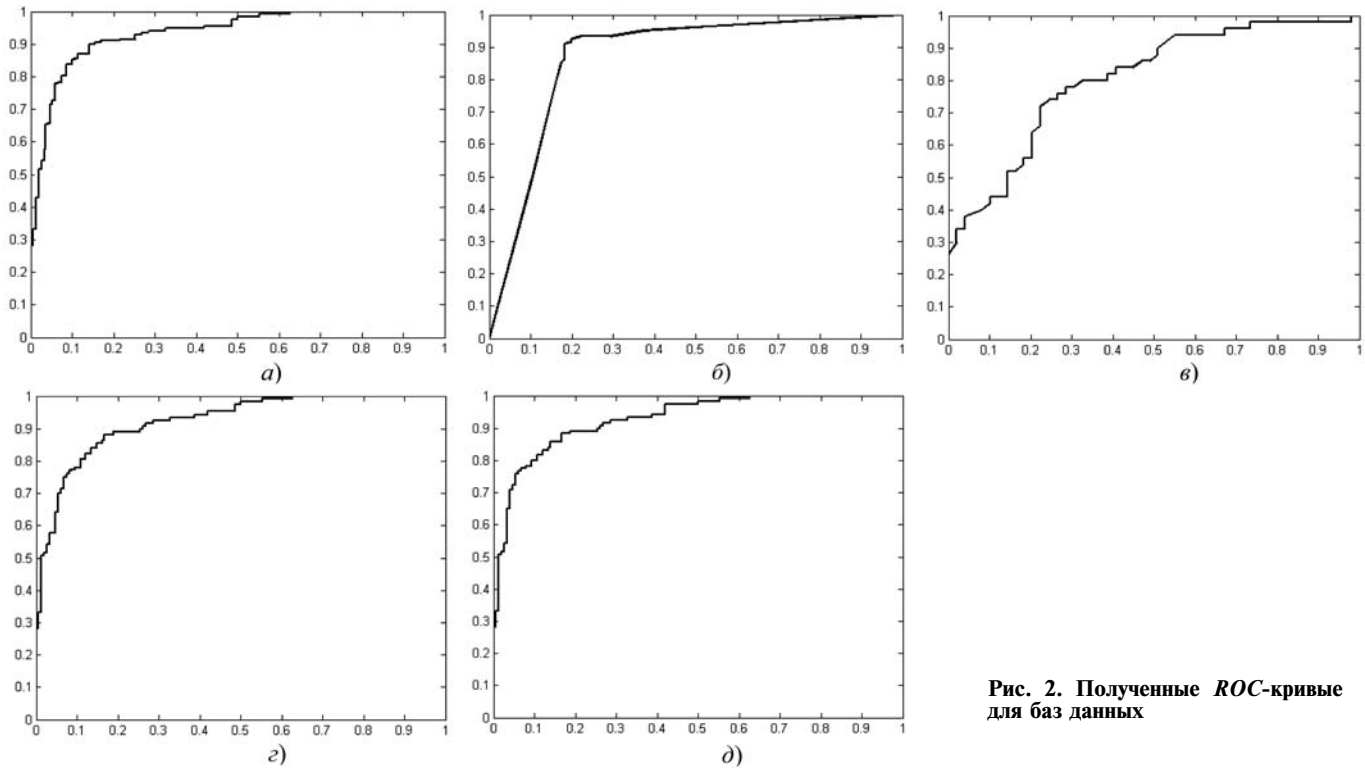


Рис. 2. Полученные *ROC*-кривые для баз данных

телей, 25 самозванцев для оценивания и 70 самозванцев для тестирования. XM2VTS-Benchmark содержит результаты пяти систем распознавания лица и трех систем распознавания голоса.

База данных значений соответствия на основе мультимодальной базы данных BANCA [29] содержит значения соответствия базовых классификаторов по лицу и голосу, значения получены от одного и того же индивидуума при контролируемых неблагоприятных и ухудшенных (разными устройствами) условиях. Данные каждого субъекта были записаны за 12 сессий.

Результаты экспериментов для баз данных NIST BSSR1, XM2VTS- Benchmark и BANCA приведены в табл. 2. Для каждого набора данных в таблице представлена мера эффективности *AUC*.

Таблица 2

Обобщение результатов слияния значений соответствия различных баз данных

База данных	*AUC*, %
NIST-multimodal	95,10
NIST-fingerprint	87,03
NIST-face	80,31
XM2VTS-Benchmark	91,75
BANCA	92,89

Как видно, предложенный метод показывает высокую эффективность.

Эксперименты проводили в среде MATLAB R2011b.

На рис. 2. приводятся *ROC*-кривые, полученные для следующих баз данных: NIST-multimodal (рис. 2, *а*); NIST-fingerprint (рис. 2, *б*); NIST-face (рис. 2, *в*); XM2VTS-Benchmark (рис. 2, *г*) и BANCA (рис. 2, *д*).

Заключение

Разработка эффективных методов слияния информации является актуальной для проектирования МС с высокими показателями распознавания личности. В этой работе для слияния значений соответствия в МС предлагается метод оптимизации показателей качества распознавания на основе построения модели непосредственной связи этих показателей с функцией слияния. В качестве целевой функции используется интегральный показатель качества классификации *AUC* (площади под *ROC*-кривой). Целевая функция представлена в виде параметрической линейной модели, которая позволяет моделировать широкий класс линейных и нелинейных функций слияния, а также разделяющих функций для классификации. На практике целевая функция аппроксимируется суммой недифференцируемых функций, и для ее оптимизации используется метод глобальной оптимизации. Предложенный подход тестируется с использованием трех открытых мультимодальных баз данных значений соответствия.

Часть данной работы выполнена Л. В. Сухостат при финансовой поддержке Фонда Развития Науки при Президенте Азербайджанской Республики — Грант № EIF-2011-1(3)-82/08/1, а вторая часть работы — Я. Н. Имамвердиевым при поддержке гранта Национального Фонда Исследований Республики Корея.

Список литературы

1. **Ross A., Nandakumar K., Jain A. K.** Handbook of Multibiometrics. 1st edition. Heidelberg: Springer, 2006. 202 p.

2. **Алгулиев Р. М., Имамвердиев Я. Н., Мусаев В. Я.** Методы обнаружения живучести в биометрических системах // Вопросы защиты информации. 2009. № 3 (86). С. 16—21.

3. **Имамвердиев Я. Н.** Метод объединения результатов ансамбля классификаторов в мультибиометрических системах // Информационные технологии. Приложение. 2011. № 9. 32 с.

4. **Ross A., Poh N.** Multibiometric Systems: Overview, Case Studies and Open Issues. In Handbook of Remote Biometrics for Surveillance and Security / M. Tistarelli, S. Z. Li and R. Chellappa (Eds.). Springer, 2009.

5. **Kittler J., Poh N.** Multibiometrics for Identity Authentication: Issues, Benefits and Challenges // Proc. of the 2nd IEEE International Conference on Biometrics: Theory, Applications and Systems — BTAS 2008. September 29 — October 1, 2008. Arlington (VA), USA. P. 1—6.

6. **Имамвердиев Я. Н.** Модель слияния информации о качестве изображений на основе теории Демпстера — Шафера для интероперабельности биометрических систем // Проблемы управления и информатики. 2010. № 2. С. 127—135.

7. **Nandakumar K., Chen Y., Pass S., Jain A.** Likelihood ratio-based biometric score fusion // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2007. P. 342—347.

8. **Jain A., Nandakumar K., Ross A.** Score normalization in multimodal biometric systems // Pattern recognition. 2005. Vol. 38, N 12. P. 2270—2285.

9. **Kittler J., Hatef M., Duin R. P. W., Matas J.** On Combining Classifiers // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1998. Vol. 20. N 3. P. 226—239.

10. **Ulery B., Hicklin A. R., Watson C., Fellner W., Hallinan P.** Studies of Biometric Fusion: NIST, Technical Report NISTIR 7346. September 2006.

11. **Ma Y., Cukic B., Singh H.** A classification approach to multi-biometric score fusion // Proc. 5th Intl Conf. Audio Video-based Biometric Person Authentication. Springer, 2005. P. 484—493.

12. **Toh K., Kirn J., Lee S.** Maximizing area under roc curve for biometric scores fusion // Pattern Recognition. 2008. Vol. 41, no. 11. P. 3373—3392.

13. **Kumar A., Kanhangad V., Zhang D.** Multimodal biometrics management using adaptive score-level combination // Proc. 19th International Conference on Pattern Recognition, 2008 (ICPR 2008). 2008. P. 1—4.

14. **Fawcett T.** An introduction to ROC analysis // Pattern Recognition Letters. 2006. V. 27, № 8. P. 861—874.

15. **Huang J., Ling C. X.** Using AUC and Accuracy in Evaluating Learning Algorithms // IEEE Transactions on Knowledge and Data Engineering. 2005. V. 17, № 3. P. 299—310.

16. **Komori O.** A boosting method for maximization of the area under the ROC curve // Annals of the Institute of Statistical Mathematics. 2011. V. 63, № 5. P. 961—979.

17. **Kennedy J., Eberhart R. C.** A New Optimizer Using Particle Swarm Theory // Proc. IEEE Int. Conf. Neural Networks. 1995. P. 1941—1948.

18. **Пантелеймонов А. В., Никитина Н. А., Решетняк Е. А., Логинова Л. П., Бугаевский А. А., Холин Ю. В.** Методики качественного анализа с бинарным откликом: метрологические

характеристики и вычислительные аспекты // Методы и объекты химического анализа. 2008. Т. 3, № 2. С. 128—146.

19. **Васильев Ф. П.** Методы оптимизации. М: Факториал пресс, 2002. 824 с.

20. **Eguchi S., Copas J.** A class of logistic-type discriminant functions // Biometrika. 2002. V. 89. P. 1—22.

21. **Ma S., Huang J.** Regularized ROC method for disease classification and biomarker selection with microarray data // Bioinformatics. 2005. V. 21. P. 4356—4362.

22. **Hastie T., Tibshirani R., Friedman J.** The elements of statistical learning: Data Mining, Inference and Prediction / 2nd ed. New York: Springer. 2009. 768 p.

23. **Леман Э. Л.** Проверка статистических гипотез / пер. с англ., 2 изд. М.: Наука, 1979.

24. **Singer I.** Duality for nonconvex approximation and optimization. Springer, 2006. 353 p.

25. **Eberhart R. C., Shi Y. H.** Swarm Intelligence. CA: Morgan Kaufmann, Jun. 2001.

26. **Press W. H., Vetterling W. T., Teukolsky S. A., Flannery B. P.** Numerical Recipes in Fortran 77. Cambridge: Cambridge University Press, 1992.

27. **National** Institute of Standards and Technology: NIST Biometric Scores Set — Release 1 (BSSR1). 2004. URL: http://www.itl.nist.gov/iad/894.03/biometricscores/.

28. **Poh N., Bengio S.** Database, Protocol and Tools for Evaluating Score-Level Fusion Algorithms in Biometric Authentication // Pattern Recognition. 2006. V. 39. № 2. P. 223—233.

29. **BANCA** scored database. URL: http://personal.ee.surrey.ac.uk/Personal/Norman.Poh/web/banca_multi

УДК 519.816

В. В. Сафронов, д-р техн. наук, проф.,
ОАО "КБ Электроприбор", г. Саратов,
e-mail: svv@kbep.ru

Упрощенный метод решения задач нечеткого многокритериального ранжирования

Поставлена задача ранжирования систем при нечетко заданных значениях критериев. Обоснован упрощенный метод решения задачи, позволяющий сделать универсальной процедуру решения детерминированных и нечетко заданных задач. Приведены численные примеры.

Ключевые слова: *функции принадлежности, нечеткое многокритериальное ранжирование, критерии*

Введение

Модели принятия решений в нечетких условиях находят широкое применение для исследования систем различного назначения [1, 3, 13, 18]. Методам решения нечетких многокритериальных задач, в том числе задач нечеткого многокритериального ранжирования, посвящено большое число работ отечественных и зарубежных ученых [9—11, 13, 16, 37]. Для решения таких задач необходима информация о значениях функций принадлежности нечеткого множества недоминируемых и доминируемых систем. Вычисления указанных функций принадлежности можно осуществить, например, методами Орловского [16] и Жуковина [9].

Метод вычисления Орловского позволяет найти следующие функции принадлежности: нечеткого отношения предпочтения; нечеткого отношения строгого предпочтения; отношения недоминирования; нечеткого множества доминируемых систем; нечеткого множества недоминируемых систем. Последние две из перечисленных функций принадлежности и используются для решения задачи нечеткого многокритериального ранжирования. Так, в работах [4—6] рассмотрены задачи ранжирования для случая, когда критерии могут быть одновременно заданы в следующих видах: в формализованном, количественном; в неопределенном, лингвистическом; в частично формализованном.

В работах [4—6] задачи решают в два этапа. На первом этапе на основе применения аппарата нечетких множеств, в частности метода Орловского, определяют функции принадлежности нечеткому множеству недоминируемых систем. На втором этапе с использованием методов теории принятия решений, в частности метода "жесткого" ранжирования [26], строят упорядоченное множество эффективных вариантов (кортеж Парето).

На первом этапе возникают следующие проблемные вопросы:

1. Как выбрать значение ширины интервала оценок по каждому критерию?

2. Каким образом унифицировать вычисления для максимизируемых и минимизируемых критериев?

3. Можно ли упростить вычисления с сохранением корректности итогового решения?

Настоящая статья посвящена обоснованию упрощенного вычисления функций принадлежности, что дает возможность исключить недостатки, присущие первому этапу. Более того, полученные результаты позволяют сделать универсальной процедуру решения детерминированных задач и задач с нечетко заданными критериями. Методы решения задач, поставленных в работах [4—6], сводятся к методу "жесткого" ранжирования.

1. Математическая постановка задачи

Для решения задач нечеткого многокритериального ранжирования в качестве критериев, как правило, используют функции принадлежности. Их формирование является непростой задачей, о чем отмечено в известной монографии В. Жуковина [9]: "...если говорить честно, то именно проблема выявления и формирования функций принадлежности для нечетких отношений предпочтения на основе исходных данных является "узким местом" на пути широкого использования нечетких задач принятия решений на практике. Несмотря на многочисленные публикации по этой проблеме, она до сих пор остается нерешенной до конца".

Рассмотрим математическую постановку задачи нечеткого многокритериального ранжирования. С этой целью введем необходимые в дальнейшем обозначения [4, 9, 16, 26, 33].

1. $S = \{S_\alpha,\ \alpha = \overline{1, n}\}$ — множество возможных вариантов систем, $S_D \subseteq S$ — множество допустимых систем, для которых, в зависимости от специфики системы, должны выполняться некоторые дисциплинирующие условия: неравенства, равенства, логические условия и т. п.

2. $F(S_\alpha) = \{f_1(S_\alpha), f_2(S_\alpha), ..., f_j(S_\alpha), ..., f_r(S_\alpha)\}$;

$K(S_\alpha) = \{K_1(S_\alpha), K_2(S_\alpha), ..., K_j(S_\alpha), ..., K_r(S_\alpha)\}$ — векторный критерий, характеризующий систему S_α до преобразования и после преобразования соответственно.

3. $f_j(S_\alpha)$, $K_j(S_\alpha)$, $(j = \overline{1, r})$ — скалярный критерий качества, характеризующий систему S_α, соответственно до преобразования (в зависимости от смысла значение критерия желательно максимизировать или минимизировать) и после преобразования (значение критерия желательно минимизировать).

4. $A = \{a_j,\ j = \overline{1, r}\}$ — множество коэффициентов важности критериев, где a_j — коэффициент важности j-го критерия, причем $\sum_{j=1}^{r} a_j = 1$.

5. $PK_j(S_k, S_l) = [(S_k, S_l); \mu K_j(S_k, S_l)]$ — нечеткое отношение предпочтения (НОП) по j-му скалярному критерию, $j = \overline{1, r}\ \ \forall\ k = \overline{1, n}$, $l = \overline{1, n}$, $k \neq l$, где (S_k, S_l) — множество упорядоченных пар систем; $\mu K_j(S_k, S_l)$ — функция принадлежности нечеткого отношения предпочтения.

6. $\mu_D K_j(S_k, S_l)$ — функция принадлежности нечеткого отношения строго предпочтения, характеризующая интенсивность доминирования системы S_k над системой S_l по j-му скалярному критерию.

7. $\mu_{ND} K_j(S_k, S_l)$ — функция принадлежности отношения недоминирования, характеризующая степень, с которой система S_k не доминируется системой S_l по j-му скалярному критерию.

8. $\mu^*_{ND} K_j(S_k)$ — функция принадлежности нечеткого множества недоминируемых систем, характеризующая степень "недоминируемости" системы S_k ни одной другой системой по j-му скалярному критерию.

9. $\mu^*_D K_j(S_k)$ — функция принадлежности нечеткого множества доминируемых систем, характеризующая степень "доминируемости" системы S_k другими системами по j-му скалярному критерию.

10. $\mu^*_{ND} K(S_k)$ — функция принадлежности нечеткого множества недоминируемых систем, характеризующая степень "недоминируемости" системы S_k ни одной другой системой по векторному критерию.

11. $\mu^*_D K(S_k)$ — функция принадлежности нечеткого множества доминируемых систем, характеризующая степень "доминируемости" системы S_k другими системами по векторному критерию.

С учетом введенных обозначений сформулируем задачу. Даны множества S, A, $F(S_\alpha)$, выбраны решающие правила [31, 32].

Требуется:

1) определить функцию принадлежности $\mu^*_D K_j(S_k)$, $\mu^*_{ND} K_j(S_k)$;

2) найти множество эффективных упорядоченных систем (кортеж Парето) $S^P \subset S_D$, для элементов которого $S^*_\alpha \in S^P$ справедливо

$$\mu^*_{ND} K(S^*_\alpha) = \max_{S_\alpha \in S_D} \mu^*_{ND} K(S_\alpha). \qquad (1)$$

Эквивалентной, по результатам решения, будет следующая задача. Найти множество эффективных упорядоченных систем (кортеж Парето) $S^P \subset S_D$, для элементов которого $S^*_\alpha \in S^P$ справедливо

$$\mu^*_D K(S^*_\alpha) = \min_{S_\alpha \in S_D} \mu^*_D K(S_\alpha). \qquad (2)$$

Для решения нечетких многокритериальных задач необходимо:

— вычислить функции принадлежности $\mu^*_{ND} K_j(S_k)$ нечеткого множества недоминируемых систем и функции принадлежности $\mu^*_D K_j(S_k)$ нечеткого множества доминируемых систем, $j = \overline{1, r}$, $\forall k = \overline{1, n}$;

— решить задачу нечеткого многокритериального ранжирования с использованием выбранного метода, построить кортеж Парето;

— сделать выводы.

Функции принадлежности нечеткого множества недоминируемых систем можно вычислить, по крайней мере, с использованием двух методов.

Первый метод основан, в частности, на работах С. А. Орловского [16], В. Г. Жуковина [9], второй метод — на работах В. Г. Жуковина [9], В. С. Михалевича, В. Л. Волковича [14].

2. Методы определения функций принадлежности

В соответствии с *первым методом* определяем нечеткое отношение предпочтения $PK_j(S_k, S_l)$ по j-му частному критерию качества для пары решений (S_k, S_l) функцией принадлежности [9, 16, 37]:

$$\mu K_j(S_k, S_l) = \begin{cases} \dfrac{f_j(S_k) - f_j(S_l)}{m_j}, \ if\ f_j(S_k) > f_j(S_l), \\ 0, \ if\ f_j(S_k) \leqslant f_j(S_l), \end{cases} \quad (3)$$

где m_j — ширина интервала оценок по j-му критерию; $f_j(S_k)$ и $f_j(S_l)$ — значения j-го критерия для систем S_k и S_l.

Назначение величины m_j является неформальной задачей и осуществляется с привлечением экспертов [4—6].

Нечеткое отношение строгого предпочтения системы S_k над системой S_l определяется функцией принадлежности $\mu_D K_j(S_k, S_l)$, характеризующей интенсивность доминирования S_k над S_l по j-му частному критерию [16]:

$$\mu_D K_j(S_k, S_l) = \begin{cases} \mu K_j(S_k, S_l) - \mu K_j(S_l, S_k), \\ if\ \mu K_j(S_k, S_l) > \mu K_j(S_l, S_k), \\ 0, \ if\ \mu K_j(S_k, S_l) \leqslant \mu K_j(S_l, S_k). \end{cases} \quad (4)$$

Отношение недоминирования системы S_k системой S_l определяется функцией принадлежности $\mu_{ND} K_j(S_k, S_l)$ как дополнение к $\mu_D K_j(S_k, S_l)$ [16]:

$$\mu_{ND} K_j(S_k, S_l) = 1 - \mu_D K_j(S_k, S_l). \quad (5)$$

Степень "недоминируемости" системы S_k ни одним другим вариантом по j-му частному критерию характеризуется функцией принадлежности нечеткому множеству недоминируемых вариантов $\mu^*_{ND} K_j(S_k)$ и показывает степень полезности варианта системы по рассматриваемому критерию:

$$\mu^*_{ND} K_j(S_k) = \min_{S_l} \mu_{ND} K_j(S_l, S_k). \quad (6)$$

Тогда степень "доминируемости" $\mu^*_D K_j(S_k)$ системы S_k другими системами по j-му скалярному критерию будет равна $\mu^*_D K_j(S_k) = 1 - \mu^*_{ND} K_j(S_k)$.

В соответствии со *вторым методом* используется преобразование, предложенное в работе Михалевича и Волковича (для исходных максимизируемых критериев) [14]:

$$K_j(S_\alpha) = \frac{f_j^0 - f_j(S_\alpha)}{f_j^0 - f_{j(\min)}^0}. \quad (7)$$

Здесь $f_j(S_\alpha)$, $j = \overline{1, r}$, $\alpha = |\overline{1, n}$ — j-й максимизируемый критерий; f_j^0 — оптимальное (наибольшее) значение j-го критерия; $f_{j(\min)}^0$ — наименьшее значение максимизируемого критерия.

Преобразование вида (7) представлено в монографии Жуковина [9].

Для исходных минимизируемых критериев применяют преобразование [14] $K_j(S_\alpha) = \dfrac{f_j(S_\alpha) - f_j^0}{f_{j(\max)}^0 - f_j^0}$. Здесь $f_j(S_\alpha)$, $j = \overline{1, r}$, $\alpha = \overline{1, n}$, — j-й минимизируемый критерий; f_j^0 — оптимальное (наименьшее) значение j-го критерия; $f_{j(\max)}^0$ — наибольшее значение минимизируемого критерия.

В дальнейшем, не теряя общности, будем использовать преобразование (7). Обозначим

$$\mu^*_D K_j(S_\alpha) = K_j(S_\alpha), j = \overline{1, r}, \alpha = \overline{1, n} \ — \quad (8)$$

степень доминируемости варианта S_α по j-му критерию. Тогда

$$\mu^*_{ND} K_j(S_\alpha) = 1 - K_j(S_\alpha), j = \overline{1, r}, \alpha = \overline{1, n} \ — \quad (9)$$

степень недоминируемости варианта S_α по j-му критерию.

Теорема 1. Результаты решения нечетких многокритериальных задач ранжирования с использованием функций принадлежности нечеткому множеству недоминируемых вариантов $\mu^*_{ND} K_j(S_k)$, полученных по методам 1, 2, совпадают.

Доказательство. Допустим, имеем следующее соотношение:

$$f_j(S_{k_1}) > f_j(S_{k_2}) > \ldots > f_j(S_{k_i}) > \ldots > f_j(S_{k_n}), \quad k_i \in \{\overline{1, n}\}. \quad (10)$$

Покажем, что значения $\mu^*_{ND} K_j(S_k)$, полученные первым методом, расположены в порядке, указанном в соотношении (10). Найдем предварительно значение $\mu^*_{ND} K_j(S_{k_1}) = \min_{l=\overline{1,n}} \mu_{ND} K_j(S_l, S_{k_1})$. С этой целью определим:

а) $\mu K_j(S_l,\ S_{k_1}) = 0$, $\forall l = \overline{1, n}$, так как выполняется условие (3);

б) $\mu_D K_j(S_l,\ S_{k_1}) = 0$, так как $\mu K_j(S_l,\ S_{k_1}) \leqslant \leqslant \mu K_j(S_{k_1},\ S_l)$;

в) $\mu_{ND} K_j(S_l,\ S_{k_1}) = 1 - \mu_D K_j(S_l,\ S_{k_1}) = 1$, $\forall l = \overline{1, n}$.

Тогда $\mu^*_{ND} K_j(S_{k_1}) = \min\limits_{l=\overline{1,n}} \mu_{ND} K_j(S_l,\ S_{k_1}) = 1$, т. е. принимает максимально возможное значение. Найдем и сравним теперь значения $\mu^*_{ND} K_j(S_{k_{i-1}})$, $\mu^*_{ND} K_j(S_{k_i})$, $i = \overline{3, n}$. С этой целью определим для $\mu^*_{ND} K_j(S_{k_{i-1}})$:

а) $\mu K_j(S_l,\ S_{k_{i-1}}) = 0$, $l = k_{i-1}, k_i, k_{i+1}, ..., k_n$, и $\mu K_j(S_l,\ S_{k_{i-1}}) = \dfrac{f_j(S_l) - f_j(S_{k_{i-1}})}{m_j} > 0$, $l = k_1, k_2, ..., k_{i-2}$;

б) $\mu_D K_j(S_l,\ S_{k_{i-1}}) = 0$, $l = k_{i-1}, k_i, k_{i+1}, ..., k_n$, и $\mu_D K_j(S_l, S_{k_{i-1}}) = \dfrac{f_j(S_l) - f_j(S_{k_{i-1}})}{m_j} > 0$, $l = k_1, k_2, ..., k_{i-2}$.

Заметим, что наибольшее значение $\mu_D K_j(S_l,\ S_{k_{i-1}})$ принимает при $l = k_1$.

в) $\mu_{ND} K_j(S_l,\ S_{k_{i-1}}) = 1$, $l = l = k_1,\ k_2,\ ...,\ k_{i-2}$ и $\mu_{ND} K_j(S_l,\ S_{k_{i-1}}) = 1 - \dfrac{f_j(S_l) - f_j(S_{k_{i-1}})}{m_j} > 0$, $l = k_1, k_2, ..., k_{i-2}$.

Тогда

$$\mu^*_{ND} K_j(S_{k_{i-1}}) = \min_{l=\overline{1,n}} \mu_{ND} K_j(S_l,\ S_{k_{i-1}}) = 1 - \frac{f_j(S_{k_1}) - f_j(S_{k_{i-1}})}{m_j}. \quad (11)$$

Аналогично, для $\mu^*_{ND} K_j(S_k)$ имеем:

а) $\mu K_j(S_l,\ S_{k_i}) = 0$, $l = k_i, k_{i+1}, ..., k_n$, и $\mu K_j(S_l,\ S_{k_i}) = \dfrac{f_j(S_l) - f_j(S_{k_i})}{m_j} > 0$, $l = k_1, k_2, ..., k_{i-1}$;

б) $\mu_D K_j(S_l,\ S_{k_i}) = 0$, $l = k_i, k_{i+1}, ..., k_n$, и $\mu_D K_j(S_l,\ S_{k_i}) = \dfrac{f_j(S_l) - f_j(S_{k_i})}{m_j} > 0$, $l = k_1, k_2, ..., k_{i-1}$.

Заметим, что наибольшее значение $\mu_D K_j(S_l,\ S_{k_i})$ принимает при $l = k_1$.

в) $\mu_{ND} K_j(S_l,\ S_{k_i}) = 1$, $l = l = k_1,\ k_2,\ ...,\ k_{i-1}$, и $\mu_{ND} K_j(S_l,\ S_{k_i}) = 1 - \dfrac{f_j(S_l) - f_j(S_{k_i})}{m_j} > 0$, $l = k_1,\ k_2,\ ...,\ k_{i-1}$.

Тогда

$$\mu^*_{ND} K_j(S_{k_i}) = \min_{l=\overline{1,n}} \mu_{ND} K_j(S_l,\ S_{k_i}) = 1 - \frac{f_j(S_{k_1}) - f_j(S_{k_i})}{m_j}. \quad (12)$$

Поскольку по условию $f_j(S_{k_{i-1}}) > f_j(S_{k_i})$, то из анализа (11), (12) следует, что выполнение условий (10) влечет за собой выполнение следующих условий:

$$\mu^*_{ND} K_j(S_{k_1}) > \mu^*_{ND} K_j(S_{k_2}) > ... > \mu^*_{ND} K_j(S_{k_i}) > ... > \mu^*_{ND} K_j(S_{k_n}),\ k_i \in \{\overline{1, n}\}. \quad (13)$$

Для *второго метода* при выполнении условий (10) на основе анализа (7) получим

$$K_j(S_{k_1}) < K_j(S_{k_2}) < ... < K_j(S_{k_i}) < ... < K_j(S_{k_n}),\ k_i \in \{\overline{1, n}\}. \quad (14)$$

Тогда с учетом (8), (9) для степеней доминируемости $\mu^*_D K_j(S_\alpha)$ вариантов $S_\alpha \in S$, $\alpha = \overline{1, n}$, по j-му скалярному критерию выполняется неравенство

$$\mu^*_D K_j(S_{k_1}) < \mu^*_D K_j(S_{k_2}) < ... < \mu^*_D K_j(S_{k_i}) < ... < \mu^*_D K_j(S_{k_n}),\ k_i \in \{\overline{1, n}\},$$

а для функций принадлежности $\mu^*_{ND} K_j(S_\alpha)$ — неравенство

$$\mu^*_{ND} K_j(S_{k_1}) > \mu^*_{ND} K_j(S_{k_2}) > ... > \mu^*_{ND} K_j(S_{k_i}) > ... > \mu^*_{ND} K_j(S_{k_n}),\ k_i \in \{\overline{1, n}\}. \quad (15)$$

Из неравенства (15) следует, что большему значению исходного критерия (меньшему значению преобразованного критерия) соответствует большее значение степени недоминируемости соответствующего варианта.

Таким образом, применение и первого, и второго методов приведет к одинаковым конечным результатам после решения задачи ранжирования.

Численный пример 1. После перехода от лингвистических значений критериев к балльным оценкам получены значения критериев для трех систем, приведенные в табл. 1.

Из табл. 1 следует:

$$f_1(S_1) < f_1(S_3) < f_1(S_2), f_2(S_3) < f_2(S_2) < f_2(S_1). \quad (16)$$

1-й метод. С использованием (3) найдем значения $\mu K_1(S_k, S_l)$, $\mu K_2(S_k, S_l)$, которые сведем в табл. 2, 3. Полагаем, что $m_1 = 6$, $m_2 = 10$.

По выражениям (4), (5) найдем значения $\mu_{ND} K_1(S_k, S_l)$, $\mu_{ND} K_2(S_k, S_l)$ (табл. 4, 5).

Таблица 1

Значения критериев

Критерии	Системы		
	S_1	S_2	S_3
$f_1(S_\alpha)$	3	6	4
$f_2(S_\alpha)$	10	8	2

Таблица 2

Значения функций принадлежности $\mu K_1(S_k, S_l)$

Системы S_k	Системы S_l		
	S_1	S_2	S_3
S_1	0	0	0
S_2	1/2	0	1/3
S_3	1/6	0	0

Таблица 3

Значения функций принадлежности $\mu K_2(S_k, S_l)$

Системы S_k	Системы S_l		
	S_1	S_2	S_3
S_1	0	1/5	4/5
S_2	0	0	3/5
S_3	0	0	0

Таблица 4

Значения функций принадлежности $\mu_{ND} K_1(S_k, S_l)$

Системы S_k	Системы S_l		
	S_1	S_2	S_3
S_1	1	1	1
S_2	1/2	1	2/3
S_3	5/6	1	1

Таблица 5

Значения функций принадлежности $\mu_{ND} K_2(S_k, S_l)$

Системы S_k	Системы S_l		
	S_1	S_2	S_3
S_1	1	4/5	1/5
S_2	1	1	2/5
S_3	1	1	1

Таблица 6

Значения функций принадлежности $\mu^*_{ND} K_1(S_\alpha)$, $\mu^*_{ND} K_2(S_\alpha)$

Функции принадлежности $\mu^*_{ND} K_j(S_\alpha)$	Системы		
	S_1	S_2	S_3
$\mu^*_{ND} K_1(S_\alpha)$	1/2	1	2/3
$\mu^*_{ND} K_2(S_\alpha)$	1	4/5	1/5

Таблица 7

Значения функций принадлежности $\mu^*_{ND} K_j(S_\alpha)$

Функции принадлежности $\mu^*_{ND} K_j(S_\alpha)$	Системы		
	S_1	S_2	S_3
$\mu^*_{ND} K_1(S_\alpha)$	1	0	2/3
$\mu^*_{ND} K_2(S_\alpha)$	0	1/4	1

Таблица 8

Значения функций принадлежности $\mu^*_D K_j(S_\alpha)$

Функции принадлежности $\mu^*_D K_j(S_\alpha)$	Системы		
	S_1	S_2	S_3
$\mu^*_D K_1(S_\alpha)$	0	1	1/3
$\mu^*_D K_2(S_\alpha)$	1	3/4	0

Определим $\mu^*_{ND} K_1(S_\alpha)$, $\mu^*_{ND} K_2(S_\alpha)$ из (6) (табл. 6).

Из табл. 6 следует:

$$\mu^*_{ND} K_1(S_1) < \mu^*_{ND} K_1(S_3) < \mu^*_{ND} K_1(S_2),$$
$$\mu^*_{ND} K_2(S_3) < \mu^*_{ND} K_2(S_2) < \mu^*_{ND} K_2(S_1). \quad (17)$$

2-й метод. Значения $\mu^*_D K_j(S_\alpha) = K_j(S_\alpha)$, $\mu^*_{ND} K_j(S_\alpha)$, $j = 1, 2$; $\alpha = \overline{1, 3}$ приведены соответственно в табл. 7, 8.

Из табл. 8 следует:

$$\mu^*_{ND} K_1(S_1) < \mu^*_{ND} K_1(S_3) < \mu^*_{ND} K_1(S_2),$$
$$\mu^*_{ND} K_2(S_3) < \mu^*_{ND} K_2(S_2) < \mu^*_{ND} K_2(S_1). \quad (18)$$

Сравнение выражений (16), (17), (18) подтверждает содержание теоремы 1.

3. Решение задачи с нечетко заданными значениями критериев методом "жесткого" ранжирования

Без потери общности изложение будем проводить для систем S_α, $\alpha = \overline{1, n}$, свойства которых задают с помощью функций принадлежности $\mu^*_D K_j(S_\alpha)$, $j = \overline{1, r}$ нечеткого множества домини-

руемых систем. В ходе решения задачи будем анализировать множество упорядоченных пар систем S_k, S_l ($k = \overline{1, n}$; $l = \overline{1, n}$; $k \neq l$), а результат анализа заносить в специальную оценочную матрицу $\|C_{kl}\|$. Сущность метода заключается в следующем [32, 33].

1. На основе попарного сравнения систем S_k,S_l ($k = \overline{1, n}$; $l = \overline{1, n}$; $k \neq l$) определяем элементы C_{kl} оценочной матрицы $\|C_{kl}\|$. Значения элементов C_{kl} подбирают таким образом, чтобы отсечь неэффективные системы.

У эквивалентных систем S_k, S_l все соответствующие значения функций принадлежности $\mu^*_D K_j(S_\alpha)$, $j = \overline{1, r}$, равны. Полагаем $C_{kl} = 1$, $C_{lk} = 1$. К числу неэффективных систем отнесем варианты, у которых:

а) все значения функций принадлежности нечеткого множества доминируемых систем l-й системы больше, чем у k-й системы, тогда полагаем $C_{kl} = N_2 \gg 1$;

б) значения m ($m < r$) функций принадлежности нечеткого множества доминируемых систем l-й системы хуже соответствующих значений функций принадлежности k-й системы при равных соответствующих значениях остальных функций принадлежности доминируемых систем; тогда полагаем $C_{kl} = N_3$, $1 \ll N_3 < N_2$.

Если же для систем k, l имеем лучшие, худшие и возможно равные значения функций принадлежности нечеткого множества доминируемых систем, то значение C_{kl} определим по методу, изложенному в работе [19].

Обозначим N^+_{kl}, N^-_{kl}, $N^=_{kl}$ — соответственно подмножества номеров лучших, худших и равных значений функций принадлежности нечеткого множества доминируемых систем для каждой пары вариантов S_k, S_l ($k = \overline{1, n}$; $l = \overline{1, n}$, $k \neq l$). Попарное сравнение систем S_k, S_l будем осуществлять на основе анализа функций принадлежности $\mu^*_D K_j(S_k)$, $\mu^*_D K_j(S_l) j = \overline{1, r}$. Значения элементов C_{kl}, C_{lk} оценочной матрицы $\|C_{kl}\|$, в зависимости от возможных значений подмножеств номеров N^+_{kl}, N^-_{kl}, $N^=_{kl}$, представлены в табл. 9.

Таблица 9

Значения элементов оценочной матрицы

N^+_{kl}	N^-_{kl}	$N^=_{kl}$	C_{kl}	C_{lk}
$\{\overline{1, r}\}$	$\varnothing$	$\varnothing$	$N_2 \gg 1$	0
$\varnothing$	$\{\overline{1, r}\}$	$\varnothing$	0	$N_2 \gg 1$
$\varnothing$	$\varnothing$	$\{\overline{1, r}\}$	1	1
$\neq \varnothing$	$\varnothing$	$\neq \varnothing$	$1 \ll N_3 < N_2$	0
$\varnothing$	$\neq \varnothing$	$\neq \varnothing$	0	$1 \ll N_3 < N_2$
$\neq \varnothing$	$\neq \varnothing$	$\varnothing$ или $\neq \varnothing$	$\sum\limits_{j \in N^+_{kl}} a_j \left(\sum\limits_{i \in N^-_{kl}} a_i \right)^{-1}$	C^{-1}_{kl}

2. Для формулировки решающих правил введем характерные числа: H_l — число элементов в l-м столбце оценочной матрицы, значения которых больше единицы; M_l — число элементов в l-м столбце той же матрицы, значения которых меньше единицы; $C_{kl\max}$ — максимальное значение элемента в l-м столбце матрицы $\|C_{kl}\|$.

3. Для реализации "жесткого" ранжирования перейдем от одношагового процесса поиска приоритетного расположения систем к многошаговому процессу [2].

Решающие правила "жесткого" ранжирования

3.1. Ранжирование необходимо проводить среди эффективных систем по шагам. Число шагов $t \leqslant (n-1)$.

3.2. На каждом шаге t ($t = 1, 2, ..., n - 1$) необходимо:

- найти числа $H^{(t)}_l$, $M^{(t)}_l$, $C^{(t)}_{kl\max}$ и определить лучшую систему S_j с минимальным значением $H^{(t)}_j$;
- номер j занести в множество P;
- исключить из оценочной матрицы j-ю строку и j-й столбец. Если системы с номерами $l_j \in \in L_{k(t)} = \{l_1, l_2, ..., l_j, ..., l_{k(t)}\}$ имеют одинаковые минимальные значения $H^{(t)}_{l_j}$, то лучшей является система S_{l_j} с максимальным значением $M^{(t)}_{l_j} = \max\limits_{l_j \in L_{k(t)}} M^{(t)}_{l_j}$.

3.3. Если системы с номерами $l_j \in L_{k(t)} = \{l_1, l_2, ..., l_j, ..., l_{k(t)}\}$ имеют соответственно одинаковые значения $H^{(t)}_{l_j}$, $M^{(t)}_{l_j}$, то лучшей является система S_{l_j} с минимальным значением $C^{(t)}_{l_j} = \min\limits_{l_j \in L_{k(t)}} C^{(t)}_{kl_j\max}$.

3.4. Если лучшие системы имеют соответственно равные значения $H^{(t)}_l$, $M^{(t)}_l$, $C^{(t)}_{kl\max}$, то такие системы считают эквивалентными.

Теорема 2. Если в l-м ($l \in \{\overline{1, n}\}$) столбце оценочной матрицы максимальный элемент равен значению N_3 или значению N_2, то l-й вариант системы не принадлежит множеству эффективных решений.

Теорема 3. Множество неэффективных систем не зависит от значений коэффициентов важности критериев.

Следствие из теоремы 3. Множество эффективных систем не зависит от значений коэффициентов важности критериев.

Впервые теорема 2 доказана в работах [25, 26], а в приведенной интерпретации изложена в работе [34]. В работе [6] ошибочно указан источник [5]. Теорема 3 сформулирована и доказана в работах [34—36].

Замечание. Созданию и становлению метода "жесткого" ранжирования способствовали работы известных отечественных и зарубежных ученых, например [2, 7, 8, 12, 14, 15, 17, 19]. В 1988 г. опубликована статья [20], в которой осуществлена постановка многокритериальной задачи и дан метод ее решения. В 1992 г. издано пособие [21], в котором фактически изложен метод жесткого ранжирования и обобщенный метод ветвей и границ. В решающем правиле учтены два числа: H_l — **введено впервые**; $C_{kl\max}$ — предложено профессором Б. Руа, Франция [19].

Дальнейшее развитие методы многокритериальной оптимизации получили в работах [22—27] (построение множества Парето, подмножества Парето заданной мощности, обобщенный метод ветвей и границ, многокритериальная задача оптимального развития систем).

В статьях [25, 26] подробно раскрыт метод "жесткого" ранжирования, для формулировки решающих правил введены два числа H_l, M_l, использовано число $C_{kl\max}$, оговорен их физический смысл, доказана теорема о непринадлежности l-го варианта множеству эффективных решений. Доказаны и другие теоремы. Введены понятия "мягкого" и "жесткого" ранжирования, "кортежа" ("подкортежа") Парето. Осуществлен переход к многошаговому процессу поиска приоритетного расположения систем.

В работе [26] доказана теорема и о правилах ветвления из вершины, еще раз более подробно раскрыт обобщенный метод ветвей и границ. На базе метода "жесткого" ранжирования, обобщенного метода ветвей и границ получен целый ряд новых результатов. В частности, разработан метод гипервекторного ранжирования [28], методы вывода сложных систем в лидеры с использованием различных решающих правил [29—31].

Определенный итог проводимым исследованиям был подведен в монографиях [32, 33]. В последнее время получены новые результаты, опубликованные в работах [34—36]. В частности, сформулирован и доказан критерий построения истинных кортежей Парето, приведена соответствующая методика. В их основе лежит метод "жесткого" ранжирования.

Численный пример 2. Для каждой из шести систем с помощью экспертов были определены значения четырех лингвистических критериев. Затем лингвистические оценки были трансформированы в балльные значения критериев $f_j(S_\alpha)$, $j = \overline{1,4}$, причем, чем значение критерия больше, тем система при прочих равных условиях лучше (табл. 10). Требуется: а) на основе применения метода 2 найти функции $\mu^*_D K_j(S_\alpha)$, $j = \overline{1,r}$; б) на основе метода "жесткого" ранжирования построить кортеж Парето; в) провести сравнение результатов решения при использовании метода Орловского.

Таблица 10

Значения скалярных критериев

Критерии	Системы						
	S_1	S_2	S_3	S_4	S_5	S_6	a_j
$f_1(S_\alpha)$	4	2	3	5	8	9	0,15
$f_2(S_\alpha)$	3	6	7	4	2	3	0,5
$f_3(S_\alpha)$	8	4	9	3	2	5	0,3
$f_4(S_\alpha)$	4	3	9	8	5	8	0,05

Таблица 11

Значения функций принадлежности $\mu^*_D K_j(S_\alpha)$

Функции принадлежности $\mu^*_D K_j(S_\alpha)$	Системы					
	S_1	S_2	S_3	S_4	S_5	S_6
$\mu^*_D K_1(S_\alpha)$	5/7	1	6/7	4/7	1/7	0
$\mu^*_D K_2(S_\alpha)$	4/5	1/5	0	3/5	1	4/5
$\mu^*_D K_3(S_\alpha)$	1/7	5/7	0	6/7	1	4/7
$\mu^*_D K_4(S_\alpha)$	5/6	1	0	1/6	2/3	1/6

Решение. Используя преобразования (7), получим значения $\mu^*_D K_j(S_\alpha)$, которые сведем в табл. 11.

Применим метод "жесткого" ранжирования. В результате получим следующий кортеж Парето: $P = < S_3, S_4, S_1, S_6 >$, т. е. предпочтение следует отдать *третьей* системе. Системы S_2, S_5 оказались неэффективными. Аналогичный результат получен и при использовании метода Орловского.

Заключение

Таким образом, поставлена и решена важная в прикладном плане задача нечеткого многокритериального ранжирования сложной системы. Доказано, что методы определения функций принадлежности, разработанные Орловским и Жуковиным с использованием преобразований Михалевича—Волковича, являются, с точки зрения получения конечного результата, эквивалентными. *Это позволяет:* на единой методической основе решать как обычные детерминированные задачи, так и задачи с нечетко заданными критериями; исключить необходимость введения и использования параметров m_j, $j = \overline{1,r}$, которые определяются экспертным путем; исключить проблему вычислительных особенностей для максимизируемых и минимизируемых критериев; упростить вычисления.

Для решения задачи нечеткого многокритериального ранжирования предлагается использовать разработанный автором метод "жесткого" ранжирования, хорошо зарекомендовавший себя для ре-

шения широкого класса задач. Особенность заключается в том, что вместо критериев рассматриваются функции принадлежности нечеткого множества доминируемых систем. Основу работ [4—6] составляет метод "жесткого" ранжирования.

На наш взгляд, предлагаемый метод может найти применение при решении прикладных задач принятия решений в экономике, социальной сфере, оценке вариантов сложных технических систем.

Список литературы

1. **Алтунин А. Е., Семухин М. В.** Модели и алгоритмы принятия решений в нечетких условиях. Тюмень: Издательство Тюменского государственного университета, 2000. 352 с.

2. **Белкин А. Р., Левин М. Ш.** Принятие решений: комбинаторные модели аппроксимации информации. М.: Наука, 1990. 160 с.

3. **Борисов А. Н., Алексеев А. Н., Крумберг О. А.** Модели принятия решений на основе лингвистической переменной. Рига: Зинатне, 1989.

4. **Ведерников Ю. В., Сафронов В. В.** Метод многокритериального ранжирования сложных систем при различных видах неопределенности исходных данных // Информационно-управляющие системы. 2008. № 3. С. 32—38.

5. **Ведерников Ю. В.** Теоретико-множественное обоснование выбора сложных систем при разнородной исходной информации. СПб.: Изд-во М-ва обороны РФ, 2008. 166 с.

6. **Ведерников Ю. В., Могиленко В. В.** Научно-методический аппарат векторного предпочтения сложных технических систем, характеризующихся показателями качества, заданными в ограниченно-неопределенном виде // Вестник Тамбовского ун-та им. В. И. Вернадского. 2011. № 1 (32). С. 81—96.

7. **Дубов Ю. А., Травкин С. И., Якимец В. Н.** Многокритериальные модели формирования и выбора вариантов систем. М.: Наука, 1986. 296 с.

8. **Емельянов С. В., Ларичев О. И.** Многокритериальные методы принятия решений. М.: Знание, 1985. 32 с.

9. **Жуковин В. Е.** Нечеткие многокритериальные модели принятия решений. Тбилиси: Мецниереба, 1988. 71 с.

10. **Заде Л. А.** Понятие лингвистической переменной и его применение к принятию приближенных решений. М.: Мир, 1976. 165 с.

11. **Заде Л. А.** Размытые множества и их применение в распознавании образов и кластер-анализе // Классификация и кластер. М.: Мир, 1980. С. 208—247.

12. **Ларичев О. И.** Наука и искусство принятия решений. М.: Наука, 1979. 200 с.

13. **Леоненков А. В.** Нечеткое моделирование в среде MATLAB и fuzzyTECH. СПб.: БХВ-Петербург, 2005. 736 с.

14. **Михалевич В. С., Волкович В. Л.** Вычислительные методы исследования и проектирования сложных систем. М.: Наука, 1982. 286 с.

15. **Моисеев Н. Н.** Математические задачи системного анализа. М.: Наука. Гл. ред. Физ.-мат. лит., 1981. 488 с.

16. **Орловский С. А.** Проблемы принятия решений при нечеткой исходной информации. М.: Наука, 1981. 203 с.

17. **Подиновский В. В., Ногин В. Д.** Парето-оптимальные решения многокритериальных задач. М.: Наука. Гл. ред. Физ.-мат. лит., 1982. 256 с.

18. **Прикладные** нечеткие системы: Пер. с яп. / К. Асаи, Д. Ватада, С. Иваи и др.; Под ред. Т. Тэрено, К. Асаи, М. Суджено. М.: Мир, 1993.

19. **Руа Б.** Проблемы и методы решений в задачах с многими целевыми функциями // Вопросы анализа и процедуры принятия решений. М.: Мир, 1976. С. 20—58.

20. **Сафронов В. В.** Многокритериальная оптимизация приборов и систем // Изв. ВУЗов СССР. Приборостроение. 1988. № 5. С. 7—10.

21. **Сафронов В. В.** Методы проектирования систем управления: учеб. пособие. Саратов: СВВКИУ, 1992. 48 с.

22. **Сафронов В. В.** Методы многокритериальной оптимизации: учебное пособие. Ч. 1, 2. Саратов: СВВКИУ, 1995. 75 с.

23. **Сафронов В. В.** Методы многокритериальной оптимизации: учеб. пособие. Ч. 3. Саратов: СВВКИУ, 1996. 24 с.

24. **Сафронов В. В.** Векторная оптимизация структур сложных систем автоматического управления // Тезисы докладов региональной научно-технической конференции "Аналитическая теория автоматического управления" / Под ред. В. А. Подчукаева (Саратов, 12—17 мая 1997 года). Саратов: Изд-во Сарат. гос. техн. ун-та, 1997. С. 64—71.

25. **Сафронов В. В.** Выбор и ранжирование эффективных вариантов в многокритериальной задаче принятия решений // Тезисы докладов научно-технического семинара "Управление в технических системах" / Под ред. В. В. Сафронова (Саратов, 3—5 декабря 1997 года). Саратов: Изд-во СВВКИУ, 1998. С. 84—91.

26. **Сафронов В. В.** Проблемы проектирования сложных технических систем и некоторые пути их решения // Доклады Академии военных наук. 1999. № 1. С. 84—95.

27. **Сафронов В. В.** Методы и алгоритмы построения оптимальных структур сложных технических систем. Саратов: Изд-во Военного артиллерийского ун-та (филиал, г. Саратов), 2000. 162 с.

28. **Сафронов В. В.** Гипервекторное ранжирование сложных систем // Информационные технологии. 2003. № 5. С. 23—26.

29. **Сафронов В. В.** Многокритериальный перевод сложной системы в число лидеров // Информационные технологии. 2002. № 4. С. 2—7.

30. **Сафронов В. В.** Многовекторный перевод сложной системы в число лидеров для различных решающих правил // Информационные технологии. 2003. № 6. С. 28—34.

31. **Сафронов В. В.** Гипервекторный перевод сложной системы в число лидеров // Информационные технологии. 2005. № 12. С. 20—25.

32. **Сафронов В. В.** Основы системного анализа: методы многокритериального ранжирования / Энгельс: Ред.-изд. центр ПКИ, 2007. 185 с.

33. **Сафронов В. В.** Основы системного анализа: методы многовекторной оптимизации и многовекторного ранжирования. Саратов: Научная книга, 2009. 329 с.

34. **Сафронов В. В.** Сравнительная оценка методов "жесткого" ранжирования и многокритериальной теории полезности в задаче гипервекторного ранжирования систем // Доклады Академии военных наук. 2010. № 5 (44). С. 101—108.

35. **Сафронов В. В.** Сравнительная оценка методов "жесткого" ранжирования, справедливого компромисса и равномерной оптимальности в задаче гипервекторного ранжирования систем // Информационно-управляющие системы. 2011. № 3. С. 2—8.

36. **Сафронов В. В.** Сравнительная оценка методов "жесткого" ранжирования и анализа иерархий в задаче гипервекторного ранжирования систем // Информационные технологии. 2011. № 7. С. 8—13.

37. **Трахтенгерц Э. А.** Эволюция компьютерных систем поддержки принятия управленческих решений // Информационные технологии. Приложение. 2006. № 1. 32 с.

УДК 004.021

А. П. Карпенко,
д-р физ.-мат. наук, проф., зав. каф.,
Е. В. Митина, студент,
А. С. Семенихин, аспирант,
МГТУ им. Н. Э. Баумана,
e-mail: apkarpenko@mail.ru

Когенетический алгоритм Парето-аппроксимации в задаче многокритериальной оптимизации

Предложен когенетический алгоритм Парето-аппроксимации. Новизна алгоритма состоит в том, что в нем генетические субалгоритмы имеют не только различные значения свободных параметров, но и различный набор эволюционных операторов. Представлены результаты исследования эффективности алгоритма на известном наборе тестовых задач многокритериальной оптимизации ZDT1 — ZDT4. Эффективность алгоритма показана также на примере двухкритеральной задачи управления спуском космического аппарата в атмосфере Земли.

Ключевые слова: *задача многокритериальной оптимизации, аппроксимация множества Парето, коэволюционный генетический алгоритм*

Введение

Известно большое число методов решения задачи многокритериальной оптимизации, из числа которых перспективными являются методы, основанные на предварительном построении аппроксимации множества Парето (а тем самым, и фронта Парето) этой задачи. Простейшим из таких методов является сеточный метод [1]. В ситуации, когда требуется высокая точность аппроксимации множеств Парето и/или когда имеет место высокая вычислительная сложность критериальных функций, сеточный метод может требовать неприемлемо высоких вычислительных ресурсов. Поэтому в настоящее время интенсивно развивают альтернативные методы, построенные на основе популяционных алгоритмов, и чаще всего, на основе генетических алгоритмов [2]. Обзор таких методов представлен, например, в работе [3].

Принципиальными в генетических методах Парето-аппроксимации являются не используемые эволюционные операторы, а правила формирования фитнесс-функции, обеспечивающие перемещение особей популяции в конечном счете в направлении множества Парето. Можно выделить следующие классы таких правил и соответствующих алгоритмов: переключающиеся критериальные функции; агрегирование критериальных функций; ранжирование особей популяции и т. д. [2]. Мы используем *алгоритм недоминируемой сортировки*, который относится к классу алгоритмов на основе ранжирования особей популяции и широко используется в популяционных методах Парето-аппроксимации.

Опыт решения сложных прикладных задач, сводящихся либо включающих в себя задачу глобальной оптимизации, показывает, что применение одного алгоритма оптимизации далеко не всегда приводит к успеху. Поэтому в последние годы большое внимание уделяется гибридизации классических и неклассических оптимизационных алгоритмов. В гибридных алгоритмах, объединяющих различные алгоритмы либо одинаковые алгоритмы, но с различными значениями свободных параметров, эффективность одного алгоритма может компенсировать слабость другого. Различные классификации гибридных алгоритмов глобальной оптимизации представлены, например, в работах [4—6].

Используем коалгоритмическую гибридизацию генетических алгоритмов, плодом которой является *генетический коэволюционный алгоритм* (*Generic Co Evolutionary algorithm*, *GCE-algorithm*), называемый далее *когенетическим алгоритмом* [7]. Основанием для такого решения является то, что из всего многообразия известных популяционных алгоритмов глобальной оптимизации в настоящее время генетические алгоритмы остаются наиболее исследованными и эффективными.

Коалгоритм можно интерпретировать как модель коэволюционирующих систем. Коэволюцией систем называют ситуацию, когда в процессе эволюции система A начинает тем или иным образом влиять на эволюцию системы B, а система B посредством механизма адаптации начинает приспосабливаться к изменениям системы A, и наоборот. В результате формируются общий темп эволюции этих систем и, по сути, новая система (метасистема), включающая в себя коэволюционирующие системы [8].

Основная идея коалгоритмов глобальной оптимизации заключается в следующем. Одновременно

в пространстве поиска эволюционируют несколько субпопуляций, каждая из которых использует, вообще говоря, свой алгоритм (субалгоритм) и решает исходную задачу оптимизации. Субпопуляции "борются" между собой за вычислительные ресурсы, которые по окончании заданного числа итераций перераспределяются в пользу более эффективной из субпопуляций.

Известно, что процесс коэволюции может многократно ускорять процесс эволюции каждой из коэволюционирующих систем. Это обстоятельство, в конечном счете, объясняет результаты исследований, показывающих, что средняя эффективность когенетического алгоритма превосходит среднюю эффективность базового генетического алгоритма.

Известные когенетические алгоритмы Парето-аппроксимации используют в качестве субалгоритмов генетические алгоритмы, имеющие один и тот же набор эволюционных операторов, но различные значения их свободных параметров. Новизна данной работы заключается в том, что субалгоритмы имеют не только различные значения этих параметров, но и разную структуру, т. е. различные наборы эволюционных операторов.

Примером коалгоритма глобальной оптимизации может служить также алгоритм эволюции разума (*Mind Evolutionary Computation*, *MEC*) [9]. С точки зрения используемых средств гибридизации основное отличие алгоритма *MEC* от когенетического алгоритма состоит в использовании более "сурового" наказания для проигрывающих субпопуляций. Если в когенетическом алгоритме только уменьшается численность проигравшей популяции, то в алгоритме *MEC* такая популяция может быть вообще выведена из конкурентной борьбы.

Одной из основных проблем эффективного использования популяционных алгоритмов вообще и генетических алгоритмов в частности является проблема обоснованного выбора значений многочисленных свободных параметров этих алгоритмов (проблема метаоптимизации) [10]. Коэволюционный подход можно считать одним из перспективных способов решения этой проблемы. Действительно, в этом случае вместо того, чтобы предварительно искать оптимальные для данной задачи оптимизации значения указанных параметров, можно сформировать достаточно представительный набор субалгоритмов, имеющих различные их значения. Коалгоритм в процессе самоадаптации выберет для данной задачи лучший из этих алгоритмов.

1. Постановка задачи Парето-аппроксимации и общая схема популяционных методов ее решения

Совокупность частных критериев оптимальности $f_i(X)$, $i \in [1{:}|F|]$ образует векторный критерий оптимальности $F(X) \in \{F\}$, где $X \in \{X\}$ — вектор варьируемых параметров; $\{X\}$, $\{F\}$ — пространства параметров и критериев соответственно. Здесь и далее запись вида $|F|$, где F — некоторый вектор или счетное множество, означает размерность этих объектов. Положим, что ставится задача минимизации каждого из частных критериев в одной и той же области допустимых значений $D_X \subset R^{|X|}$. Тогда задачу многокритериальной оптимизации *условно* записываем в виде

$$\min_{X \in D_X} F(X) = F(X^*) = F^*, \qquad (1)$$

где X^*, F^* — решения задачи. Полагаем, что частные критерии оптимальности нормализованы, так что для всех $X \in D_X$ справедливы соотношения $f_i(X) \in [0; 1]$, $i \in [1 : |F|]$.

Множество достижимости задачи (1) обозначаем D_F, а *множество Парето* и *фронт Парето* — D_X^*, D_F^* соответственно. Ставим задачу приближенного построения множества Парето (а тем самым и фронта Парето) в задаче многокритериальной оптимизации (1). Называем эту задачу *задачей Парето-аппроксимации*.

Пусть тем или иным образом уже сформировано *архивное множество* A^F, содержащее недоминируемые точки F_i^A, а также архивное множество A^X соответствующих ему точек X_i^A; $i \in [1 : |A|]$, $|A| = |A^F| = |A^X|$. Суть большинства популяционных методов Парето-аппроксимации состоит в итерационном уточнении множеств точек в архивах A^F, A^X. Если при этом на итерации t появляется новая точка F_j, доминирующая некоторые точки из архива A^F, то все доминируемые точки, а также соответствующие точки из архива A^X, удаляем. При удовлетворении некоторого критерия останова текущее содержимое архивов A^F, A^X полагаем искомой аппроксимацией фронта Парето D_F^* и множества Парето D_X^* соответственно.

В популяционных методах Парето-аппроксимации новые точки для архивов A^F, A^X "поставляет" популяция S особей s_i, текущие координаты которых в пространстве поиска $\{X\}$ равны X_i, а в пространстве $\{F\}$ — $F_i = F(X_i)$; $i \in [1 : |S|]$. Миграция особей в пространстве поиска в популяционных алгоритмах оптимизации подчинена задаче минимизации (для определенности) значений некоторой фитнесс-функции $\varphi(X)$. Основной проблемой построения популяционных методов Парето-аппроксимации является построение такой функции, обеспечивающей перемещение особей s_i, $i \in [1 : |S|]$, в направлении множества Парето D_X^*, а соответствующих точек F_i — в направлении фронта Парето D_F^*.

В силу меньшей, как правило, размерности критериального пространства $\{F\}$ по сравнению с размерностью пространства поиска $\{X\}$, ответ на вопрос о направлении и шаге перемещения особей обычно отыскивают в терминах пространства $\{F\}$, а не пространства $\{X\}$. Важно также, что относительно множества Парето, по сути, нет никой априорной информации, кроме того, что это множе-

ство точек, не связанных между собой отношением предпочтения. В то же время по отношению к фронту Парето априорной информации значительно больше [2].

Фитнесс-функцию $\varphi(X)$ строим с помощью алгоритма недоминируемой сортировки (*Non-Dominated Sorting*, *NDS*). Положим, что все частные критерии оптимальности являются одинаково важными. Ранг особи s_i, $i \in [1 : |S|]$, в его текущем положении X_i обозначаем r_i. В алгоритме *NDS* используется простейшее из правил вычисления рангов:

1) выбираем среди всех особей популяции недоминируемых, присваиваем им ранг, равный единице, и исключаем из дальнейшего рассмотрения;

2) среди оставшихся особей выбираем недоминируемых, присваиваем им ранг, равный двум, и исключаем из дальнейшего рассмотрения. И так далее до исчерпания популяции.

Приспособленность особи s_i вычисляем по формуле

$$\varphi(X_i) = \frac{1}{1 + r_i},\ i \in [1 : |S|].$$

2. Коалгоритм

Коэволюционные алгоритмы можно классифицировать по следующим признакам: используемая модель коэволюции; форма коэволюции; число коэволюционирующих субпопуляций; однородность субпопуляций; схема взаимодействия между субпопуляциями.

Выделяют два основных класса вычислительных моделей коэволюции — простые модели и композиционные модели. Чаще всего в вычислительной практике используют *простые модели коэволюции*, отличительной особенностью которых является то, что в качестве решения исходной задачи в этом случае может быть использовано решение, найденное любой из коэволюционирующих субпопуляций. В *композиционных моделях коэволюции* решение задачи представляет собой объединение его фрагментов, найденных различными субпопуляциями. Например, различные субпопуляции могут осуществлять поиск решения по различным компонентам вектора варьируемых параметров.

В зависимости от характера взаимодействия между субпопуляциями различают две основные формы коэволюции — сотрудничество и соперничество. *Коэволюция типа сотрудничество* {*cooperative coevolution*) предполагает, что каждая из субпопуляций решает одну и ту же задачу оптимизации либо ее часть. *Коэволюция типа соперничества* {*competitive coevolution*) предполагает один из следующих типов взаимодействия между субпопуляциями:

- взаимодействие по схеме "хозяин—паразит", когда перераспределение ресурсов между субпопуляциями отсутствует, но пригодность агентов данной субпопуляции определяется их сравнением с агентами другой субпопуляции;
- взаимодействие субпопуляций, имеющих различные области поиска;
- взаимодействие субпопуляций, отличающихся стратегиями поиска (алгоритмами поиска и/или значениями их свободных параметров).

С точки зрения числа субпопуляций различают *многопопуляционные коэволюции* и частный случай последних — *двухпопуляционные коэволюции*.

По принципу однородности субпопуляций выделяют однородные и неоднородные коэволюции. *Однородная коэволюция* предполагает, что каждая из субпопуляций использует один и тот же субалгоритм, хотя, быть может, с различными наборами эволюционных операторов и значениями свободных параметров. *Неоднородная коэволюция* означает, что в различных субпопуляциях используются, вообще говоря, различные субалгоритмы.

Взаимодействие между субпопуляциями может протекать по последовательной и параллельной схемам. В *последовательной схеме взаимодействия субпопуляций* обновление текущего числа особей популяций проводится последовательно, так что текущая численность данной субпопуляции зависит от ее численности на предыдущей итерации и текущей численности уже обновленных субпопуляций. В *параллельной схеме взаимодействия субпопуляций* текущая численность данной субпопуляции зависит от численности всех субпопуляций, включая данную, только на предыдущей итерации.

Используемый в работе когенетический алгоритм реализует простую однородную модель коэволюции типа соперничества. Общая схема алгоритма имеет следующий вид.

1. Задаем число и параметры субпопуляций, а также параметры коалгоритма.

2. Инициализируем субпопуляции.

3. Выполняем t_a (интервал адаптации) независимых итераций для каждой из субпопуляций.

4. Оцениваем текущие эффективности субпопуляций.

5. Проверяем выполнение условий останова.

6. Перераспределяем ресурсы и повторяем шаги 2—6 до выполнения условий останова.

Рассмотрим основные шаги приведенной схемы. Коэволюционирующие субпопуляции обозначаем S_i, $i \in [1 : |\boldsymbol{S}|]$, где $|\boldsymbol{S}|$ — их число.

Задание параметров субпопуляций и коалгоритма. По одной из рекомендаций значения параметров субпопуляций следует выбирать таким образом, чтобы в субпопуляциях были в равной степени представлены три сорта алгоритмов: алгоритмы, обладающие, в первую очередь, свойством диверсификации (широты) поиска; алгоритмы, ориентированные на интенсификацию (скорость) поиска; алгоритмы, обеспечивающие как широту, так и высокую скорость поиска.

Основными параметрам коалгоритма являются $|\boldsymbol{S}|$ — число субпопуляций; n_f — значение ресурса — максимально допустимое число испытаний (вычислений значений критериальных функций); t_a — интервал адаптации; n_p — значение штрафа, назначаемое проигравшим субпопуляциям; $|S|_{\min} = \min\limits_i S_i$, $i \in [1 : |\boldsymbol{S}|]$ — минимально допустимый размер субпопуляции.

Если на каждой итерации каждого из субалгоритмов проводится число испытаний, равное числу особей субпопуляции, то ресурс n_f определяет формула

$$n_f = \hat{t} \sum_i |S_i|,\ i \in [1 : |\boldsymbol{S}|],$$

где $\hat{t}$ — максимально допустимое число итераций. Таким образом, задание ресурса n_f эквивалентно в этом случае заданию начального размера субпопуляций $|S_i|$ и величины $\hat{t}$. Изначально ресурс распределяют обычно поровну каждой из субпопуляций, т. е. полагают $|S_i| = |S_j|$, $i, j \in [1 : |\boldsymbol{S}|]$.

Значение интервала адаптации $t_a \in (0; \hat{t})$ назначаем, исходя из следующих соображений. Если это значение "мало", то субалгоритмы не успеют продемонстрировать особенности своего поведения, т. е. все субпопуляции покажут близкие результаты и адаптация потеряет смысл. "Большие" значения величины t_a уменьшают эффективность алгоритма коэволюции, поскольку в этом случае значительное число субпопуляций может "уйти" в неперспективные подобласти области поиска.

Важным в коалгоритме является также размер штрафа $n_p \in [0; 1]$, имеющий смысл доли, на который сокращают размеры проигравших субпопуляций. Значение этой величины также не должно быть слишком малым (иначе субалгоритмы не "почувствуют" изменений) и слишком большим (поскольку генетический алгоритм с маленьким размером популяции, как известно, неэффективен).

Минимально допустимый размер субпопуляции $|S|_{\min}$ ограничивает снизу размер проигравших субпопуляций, исходя, как и в случае параметра n_p, из соображений низкой эффективности генетического алгоритма с маленьким размером популяции.

Оценку эффективности субпопуляций проводим на основе оценки текущих значений их функции пригодности $\phi_i = \phi_i(t)$, $i \in [1 : |\boldsymbol{S}|]$, $t \in [0 : \hat{t}]$. В качестве функции $\phi_i(t)$ может быть использована, например, функция

$$\phi_i(t) = \sum_{\tau = 0}^{t_a - 1} \frac{t_a - \tau}{\tau + 1} b_i(t - \tau),\ i \in [1 : |S|], \qquad (2)$$

где $b_i(t - \tau) = 1$, если субпопуляция S_i на итерации $(t - \tau)$ включает в себя лучшую среди всех субпопуляций особь, и $b_i(t - \tau) = 0$ — в противном случае. Заметим, что в сумме (2) вес величины $b_i(t - \tau)$, определяемый отношением $\frac{t_a - \tau}{\tau + 1}$, убывает с ростом τ, так что при $\tau = 0$ (текущая итерация) этот вес равен t_a, а при $\tau = t_a - 1$ (первая итерация прошедшего интервала адаптации) тот же вес равен $\frac{1}{t_a}$. Таким образом, вес величины $b_i(t - \tau)$ в функции пригодности ϕ_i быстро убывает с уменьшением номера итерации t.

Перераспределение ресурсов выполняем путем сокращения размера каждой из проигравших субпопуляций на величину n_p и увеличения размера победившей субпопуляции на число, равное сумме потерь проигравших субпопуляций, так что общий размер популяции остается неизменным. Если при этом размер проигравшей субпопуляции оказывается меньшим значения $|S|_{\min}$, то принимаем его равным этому значению. Таким образом, если $|S_i|$ — текущий размер субпопуляции S_i, проигравшей на данном интервале адаптации, то на следующем интервале размер этой субпопуляции будет равен

$$|S_i'| = \max(\lceil n_p |S_i| \rceil\, |S|_{\min}),\ i \in [1 : |\boldsymbol{S}|],$$

и в популяцию S_i' войдут $|S_i'|$ лучших индивидов популяции S_i. Здесь $\lceil \bullet \rceil$ — символ ближайшего целого большего.

3. Субалгоритмы

Субалгоритмы используют вещественное кодирование особей и различные эволюционные операторы рекомбинации, кроссовера, мутации и селекции.

Операторы рекомбинации (выбор родительской пары). Из всего многообразия возможных операторов рекомбинации субалгоритмы могут использовать операторы инбридинга и аутбридинга [13].

Инбридинг (*inbreeding*) представляет собой метод рекомбинации, в котором первую особь родительской пары выбирают из данной популяции $S = (s_i,\ i \in [1 : |S|])$ случайно, а в качестве второй особи по правилу рулетки выбирают с большей вероятностью особь той же популяции, генотип которой в некотором смысле наиболее близок к генотипу первой особи. В качестве меры $\rho(s_j, s_k)$ близости особей s_j, s_k, $j, k \in [1 : |S|]$, используем манхеттеновское расстояние между их образами F_j, F_k:

$$\rho(s_j, s_k) = \sum_{l = 1}^{|F|} abs(f_l(X_j) - f_l(X_k)) = \|s_j, s_k\|_M.$$

Таким образом, вероятность выбора особи s_k оказывается пропорциональной величине $\rho(s_j, s_k)$. Хорошо известно, что инбридинг обычно позволяет быстро найти, по крайней мере, квазиоптимальное решение, т. е. обеспечивает высокую интенсивность поиска.

Аутбридинг (*outbreeding*) предполагает выбор первой особи родительской пары из популяции S по схеме панмиксии, т. е. равномерно случайно. В качестве второй особи по правилу рулетки с большей вероятностью выбирают особь той же популяции, которая в смысле используемой меры близости генотипов наиболее далека от первой особи. В качестве меры близости особей используем рассмотренное манхеттеновское расстояние. Таким образом, особи s_j в данном случае ставим в соответствие особь s_k с вероятностью, обратно пропорциональной величине $\|s_j, s_k\|_M$. Аутбридинг предупреждает раннюю сходимость генетического алгоритма, обеспечивая исследование новых областей пространства поиска, другими словами, обеспечивая высокую диверсификацию поиска.

Оператор кроссовера. В качестве оператора кроссовера субалгоритмы могут использовать расширенный линейный кроссовер, *BLX*-кроссовер, *SBX*-кроссовер и эвристический кроссовер [13]. Полагаем, что $U_1(a; b)$ — случайное число, равномерно распределенное в интервале $(a; b)$.

Расширенный линейный кроссовер на основе особей $s_j, s_k \in S$ создает особь s' такую, что

$$x'_l = x_l^{\min} + u\delta_l,\ l \in [1 : |X|],$$

где $x_l^{\min} = \min(x_{j,l}, x_{k,l})$; $u = U_1(-0{,}25;\ 1{,}25)$; δ_l — положительный свободный параметр оператора.

BLX-кроссовер на основе особей s_j, s_k генерирует потомка s' с координатами

$$x'_l = U_1(x_l^{\min} - \alpha\Delta x_l;\ x_l^{\max} + \alpha\Delta x_l),\ l \in [1 : |X|],$$

где $x_l^{\max} = \max(x_{j,l}, x_{k,l})$; $\Delta x_l = x_l^{\max} - x_l^{\min}$; α — свободный параметр оператора, рекомендованное значение которого равно 0,5.

SBX-кроссовер на основе особей s_j, s_k порождает двух особей-потомков s'_1, s'_2 по правилу

$$x'_{1,l} = 0{,}5[(1 - u_1)x_{j,l} + (1 + u_1)x_{k,l}],$$

$$x'_{2,l} = 0{,}5[(1 - u_2)x_{k,l} + (1 + u_2)x_{j,l}],$$

где $l \in [1: |X|]$; u_1, u_2 — случайные величины, плотности вероятности которых подчинены закону

$$\xi(u) = \begin{cases} (2u)^{\frac{1}{b+1}}, & u \geqslant 0{,}5, \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{b+1}}, & u < 0{,}5. \end{cases}$$

Здесь $u = U_1(0; 1)$ — случайное число; $b \in [2{:}5]$ — натуральное число (свободный параметр кроссовера).

Эвристический кроссовер представляет собой вариант широко известного арифметического кроссовера. В данном кроссовере на основе особей s_j, s_k создается особь s', координаты которой определяет формула

$$x'_l = u(x_{j,l} - x_{k,l}) + ux_{j,l},\ l \in [1 : |X|],$$

где принято, что приспособленность особи s_j ниже приспособленности особи s_k (т. е. $r_j > r_k$); $u = U_1(0; 1)$.

Оператор мутации. Каждый из субалгоритмов может использовать только простейший *оператор случайной мутации* [13]. Пусть мутации подлежит особь $s_j \in S$. Суть оператора случайной мутации заключается в присваивании гену $x_{j,l}$, $l \in [1 : |X|]$, случайного значения из допустимого интервала $[x_l^-; x_l^+]$ с малой вероятностью ξ_m.

Оператор селекции. Субалгоритмы могут использовать турнирную и элитарную селекции [13].

Турнирная селекция предполагает случайное формирование на основе текущей популяции $S = (s_i, i \in [1 : |S|])$ некоторого числа групп из n особей в каждой, где n называется размером турнира. В каждой из групп выбираем особь с наилучшей приспособленностью (турнир) и включаем ее в промежуточную популяцию S'. Обычно группы содержат по две—три особи ($n = 2$ либо $n = 3$).

Элитарная селекция предполагает отбор в промежуточную популяцию лучших, т. е. наиболее приспособленных особей. Пусть φ_i, $i \in [1 : |S|]$, — приспособленности особей текущей популяции. Тогда элитарная селекция сводится к сортировке величин φ_i в порядке их убывания и отбору особей, соответствующих первым $|S'| < |S|$ членам полученной последовательности.

В случае как турнирной, так и элитарной селекции могут иметь место ситуации, когда критерию отбора удовлетворяет число особей, большее требуемого числа (ситуация неоднозначности выбора). В таком случае отбор проводим с помощью так называемого *критерия разреженности*.

Полагаем, что данная особь s_j, $j \in [1 : |S|]$, ранга r находится в разреженной области, если ближайшая к ней особь s_k, $k \in [1 :|S|]$, $k \neq j$, того же ранга r располагается "далеко" от особи s_j. В качестве меры расстояния между особями s_j, s_k используем норму $\|s_j, s_k\|_M$. Таким образом, разреженность $\mu_j(r)$ окрестности агента s_j будет равна

$$\mu_j(r) = \min_{k \in I_r} \|s_j, s_k\|_M,$$

где I_r — совокупность номеров особей популяции S, имеющих ранг r.

Если в ситуации неоднозначности находятся особи s_j, s_k ранга r, то в соответствии с критерием разреженности в промежуточную популяцию отбираем особь s_l, удовлетворяющую условию

$$\mu_l(r) = \max(\mu_j(r), \mu_k(r)).$$

Рассмотренная схема преодоления неоднозначностей ориентирована на обеспечение важнейшего требования к методам Парето-аппроксимации — требования равномерности покрытия множества и фронта Парето. Заметим, что аналогичную схему использует известный генетический алгоритм Парето-аппроксимации NGSA-II.

4. Исследование эффективности

Для исследования эффективности когенетического алгоритма Парето-аппроксимации выполнена его программная реализация. Разработка выполнена на языке программирования C++ в среде *Builder* 6.0 под управлением операционной системы *Windows XP*. Для экспериментов использован персональный компьютер на основе процессора *Intel Core i3* с тактовой частотой 1,33 ГГц, имеющий оперативную память объемом 4 Гбайт. Программа позволяет сформировать от двух до четырех субпопуляций, выбрать для каждого из субалгоритмов эволюционные операторы рекомбинации, кроссоверы, мутации и селекции, а также задать значения свободных параметров коалгоритма и субалгоритмов.

Исследование выполнено при следующих значениях основных параметров коалгоритма: число субпопуляций $|\boldsymbol{S}| = 4$, значение интервала адаптации $t_a = 5$, размер штрафа $n_p = 0{,}1$, минимально допустимый размер субпопуляции $|S|_{\min} = 0{,}12|S|$. В субалгоритмах во всех случаях вероятность мутации ξ_m равна 0,05.

Индикаторы эффективности. Эффективность Парето-аппроксимации оцениваем тремя индикаторами: равномерность распределения решений I_s [14]; число найденных истинно паретовских (недоминируемых) решений N_{Pareto}; общее число испытаний N_{fit}. Значения первого индикатора вычисляем по формуле

$$I_s(A^F) = \frac{\sum_{j=1}^{|A|} \left|\mu_j - \overline{\mu}\right|}{(|A| - 1)\overline{\mu}} \geqslant 0,$$

где A^F — результирующее архивное множество; μ_j — манхеттеновское расстояние от точки $F_j^* \in A^F$ до ближайшей точки того же множества; $\overline{\mu}$ — среднее всех $|A|$ указанных расстояний. Очевидно, что меньшие значения индикатора I_s свидетельствуют о лучшей равномерности распределения решений вдоль фронта Парето.

Величина N_{Pareto} представляет собой ни что иное, как мощность результирующего архивного множества A, т. е. $N_{Pareto} = |A|$. В практически значимых задачах, когда вычислительная сложность вектор-функции $F(X)$ велика, величина N_{fit} определяет вычислительную сложность алгоритма.

Тестовые задачи. Исследование эффективности выполнено с помощью двухкритериальных задач *ZDT*1 — *ZDT*4 известного набора тестовых задач многокритериальной оптимизации [11].

Задача *ZDT*1 имеет выпуклый фронт Парето (рис. 1). Трудность задачи состоит в большом числе варьируемых параметров ($|X| = 30$). Трудность задачи *ZDT*2 состоит в большом числе варьируемых параметров ($|X| = 30$) и вогнутости фронта Парето (см. рис. 3). Особенностью задачи *ZDT*3 является выпуклый, но разрывный фронт (см. рис. 5). Размерность вектора X принята в задаче также равной 30. В отличие от задач *ZDT*1—*ZDT*3, задача *ZDT*4 имеет множество локальных субоптимальных фронтов Парето, хотя эти фронты и являются выпуклыми (см. рис. 7).

Для задач *ZDT*1 — *ZDT*3 исследование выполнено при размерах субпопуляций $|S| = 20, 50, 100$ и числе поколений $\hat{t} = 10, 25, 50$. Для задачи *ZDT*4 в силу ее мультимодальности кроме того использованы $|S| = 200, 400$ и $\hat{t} = 50, 100, 1000$.

Задача *ZDT*1. Исследование выполнено для однопопуляционного генетического алгоритма (*GA*) и когенетического алгоритма (*CGA*), включающего в себя четыре субалгоритма GA_i, $i \in [1{:}4]$, состав эволюционных операторов которых приведен в табл. 1.

Полученную Парето-аппроксимацию иллюстрирует рис. 1. Здесь и далее сплошной линией показан точный фронт Парето. Результаты исследования показывают, что во всех случаях на первых

Таблица 1

Состав эволюционных операторов: задача *ZDT*1

Оператор	*GA*	*CGA*			
		GA_1	GA_2	GA_3	GA_4
Рекомбинация	Аутбридинг	Аутбридинг	Инбридинг	Аутбридинг	Инбридинг
Кроссовер	*BLX*	*BLX*	Эвристический	Эвристический	*BLX*
Селекция	Турнирная	Турнирная	Турнирная	Турнирная	Турнирная

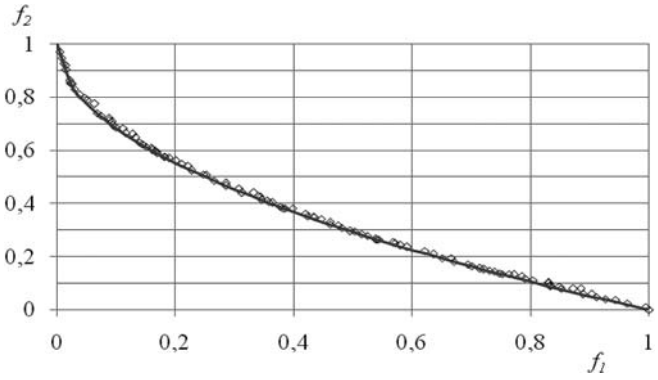


Рис. 1. Результат Парето-аппроксимации: задача *ZDT*1; $|S| = 100$; $\hat{t} = 50$

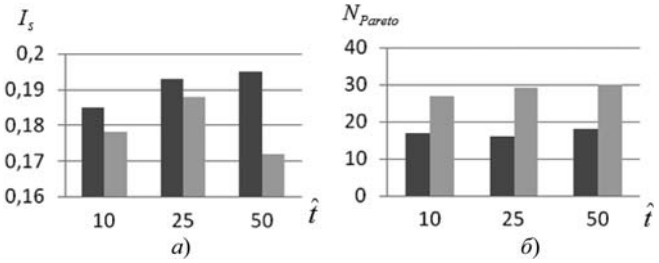


Рис. 2. Сравнение эффективности ко- и генетического алгоритмов: задача *ZDT*1; |*S*| = 20:
а — индикатор I_s; *б* — индикатор N_{Pareto}

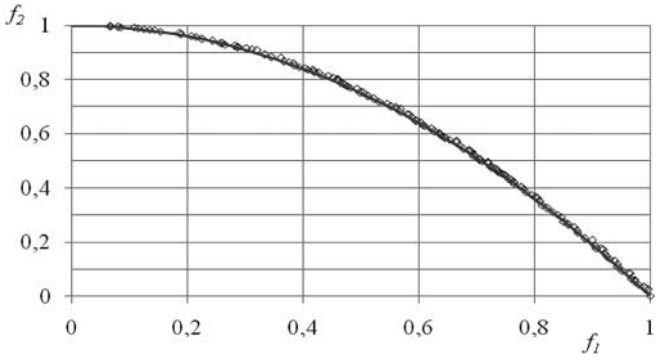


Рис. 3. Результат Парето-аппроксимации:
задача *ZDT*2; |*S*| = 100; $\hat{t}$ = 50

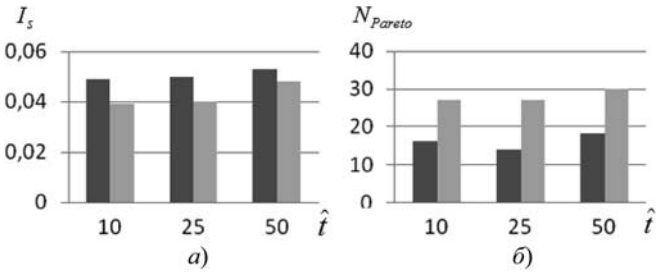


Рис. 4. Сравнение эффективности ко- и генетического алгоритмов: задача *ZDT*2; |*S*| = 20:
а — индикатор I_s; *б* — индикатор N_{Pareto}

итерациях победителем оказывался субалгоритм GA_1, а на завершающих итерациях — субалгоритм GA_4. Из этого можно сделать вывод, что для задачи *ZDT*1 более эффективным по сравнению с эвристическим кроссовером оказывается *BLX*-кроссовер. Коалгоритм демонстрирует свои самоадаптационные свойства — сначала, как этого и требует задача, обеспечивает превалирование диверсификации поиска (которую обеспечивает аутбридинг), а затем, для достижения высокой точности локализации паретовских точек, переключается на преимущественную интенсификацию поиска (обеспечиваемую инбридингом).

Результаты сравнения эффективности когенетического и генетического алгоритмов иллюстрирует рис. 2. Здесь и далее темные столбики показывают эффективность генетического алгоритма, а светлые — коалгоритма. Рис. 2 показывает, что по индикатору I_s коалгоритм до ~ 13 %, а по индикатору N_{Pareto} до ~100 % эффективнее генетического алгоритма. По индикатору N_{fit} коалгоритм в данной задаче и других рассмотренных ниже тестовых задачах почти в 4 раза проигрывает генетическому алгоритму.

Задача *ZDT*2. Состав эволюционных операторов генетического алгоритма и субалгоритмов, использованных в данном случае, представлен в табл. 2.

Полученную Парето-аппроксимацию иллюстрирует рис. 3. Наибольшее число побед одержал субалгоритм GA_4, использующий инбридинг и *SBX*-кроссовер. Сравнительную эффективность когенетического и генетического алгоритмов иллюстрирует рис. 4. Рисунок показывает незначительное, но преимущество коалгоритма по индикатору I_s; по индикатору N_{Pareto} коалгоритм до ~95 % эффективнее генетического алгоритма.

Задача *ZDT*3. Исследование выполнено для генетического алгоритма и субалгоритмов, указанных в табл. 3. Полученную Парето-аппроксимацию иллюстрирует рис. 5. В большинстве случаев здесь побеждал субалгоритм GA_1, использующий аутбридинг и *BLX*-кроссовер, т. е. данная задача, имеющая сложный фронт Парето, в большей степени требует диверсификации поиска (которую обеспечивает аутбридинг), чем его интенсификации.

Сравнительную эффективность когенетического и генетического алгоритмов иллюстрирует рис. 6,

Таблица 2

Состав эволюционных операторов: задача *ZDT*2

Оператор	*GA*	*CGA*			
		GA_1	GA_2	GA_3	GA_4
Рекомбинация	Аутбридинг	Аутбридинг	Инбридинг	Аутбридинг	Инбридинг
Кроссовер	*SBX*	*SBX*	Эвристический	Эвристический	*SBX*
Селекция	Турнирная	Турнирная	Турнирная	Турнирная	Турнирная

Таблица 3

Состав эволюционных операторов: задача *ZDT*3

Оператор	*GA*	*CGA*			
		GA_1	GA_2	GA_3	GA_4
Рекомбинация	Аутбридинг	Аутбридинг	Инбридинг	Аутбридинг	Инбридинг
Кроссовер	*BLX*	*BLX*	Эвристический	Эвристический	*BLX*
Селекция	Турнирная	Турнирная	Турнирная	Турнирная	Турнирная

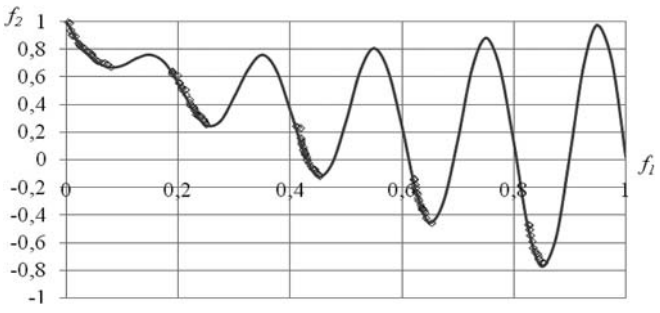


Рис. 5. Результат Парето-аппроксимации: задача ZDT3; $|S| = 100$; $\hat{t} = 50$

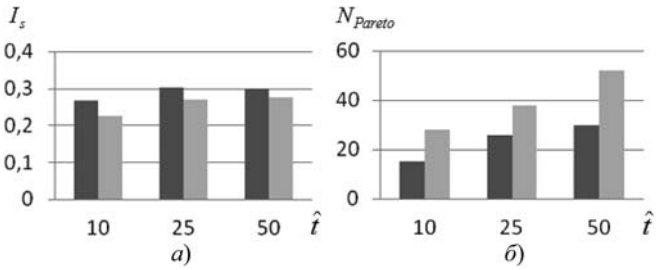


Рис. 6. Сравнение эффективности ко- и генетического алгоритмов; задача *ZDT*3; $|S| = 50$:
а — индикатор I_s; *б* — индикатор N_{Pareto}

Таблица 4

Состав эволюционных операторов: задача *ZDT*4

Оператор	*GA*	*CGA*			
		GA_1	GA_2	GA_3	GA_4
Рекомбинация	Аутбридинг	Аутбридинг	Инбридинг	Аутбридинг	Инбридинг
Кроссовер	Линейный	Линейный	*SBX*	*SBX*	Линейный
Селекция	Турнирная	Турнирная	Турнирная	Турнирная	Турнирная

показывающий, что во всех случаях коалгоритм эффективнее генетического алгоритма как по индикатору I_s, так и по индикатору N_{Pareto}.

Задача *ZDT*4. Используемые при решении данной задачи алгоритмы приведены в табл. 4.

Преодоление коалгоритмом локальных Парето-фронтов иллюстрируют рис. 7, *а*—*в*, показывающие, что с увеличением численности популяции полученные Парето-аппроксимации становятся более равномерными, преодолевают локальные фронты и приближаются к точному фронту. Генетический алгоритм с указанными в табл. 4 настройками оказался неспособным преодолеть локальные фронты и даже при очень большом числе итераций не смог приблизиться к точному фронту.

Результаты исследования показывают, что на первых интервалах адаптации всегда побеждает субалгоритм GA_4, использующий инбридинг и линейный кроссовер. Затем начинает лидировать субалгоритм GA_1 (аутбридинг и тот же кроссовер). При решении задачи *ZDT*4 оказываются ярко выраженными самоадаптационные свойства коалгоритма. При достижении локального фронта коалгоритм некоторое число итераций предпочитает инбридинг в целях более точной локализации этого фронта, но затем, чтобы преодолеть его, начинает использовать аутбридинг, расширяя тем самым область поиска. Как и для задач *ZDT*1—*ZDT*3, коалгоритм превосходит генетический алгоритм по индикаторам I_s, N_{Pareto}, но уступает по индикатору N_{fit}.

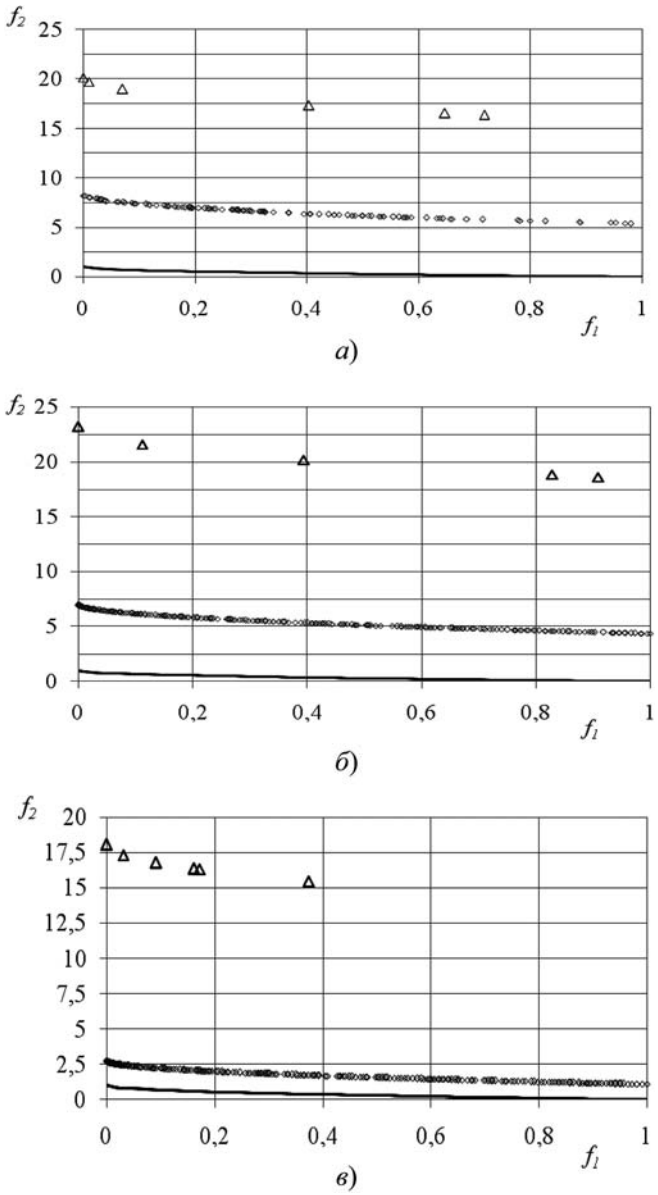


Рис. 7. Результат Парето-аппроксимации; задача *ZDT*4 (◊ — коалгоритм; △ —генетический алгоритм):
а — $|S| = 200$; $\hat{t} = 100$; *б* — $|S| = 200$; $\hat{t} = 200$; *в* — $|S| = 200$; $\hat{t} = 500$

5. Двухкритериальная задача управления спуском космического аппарата в атмосфере Земли

Эффективность Парето-аппроксимации на основе когенетического алгоритма и алгоритма недоминируемой сортировки иллюстрируем примером двухкритеральной задачи оптимального управления спуском космического аппарата в атмосфере Земли [12]. Сложность задачи как задачи оптимального управления обусловлена наличием в ней скользящих режимов. С точки зрения Парето-аппроксимации сложность задачи обусловлена высокой размерностью вектора варьируемых параметров, а также высокой вычислительной сложностью критериальных функций.

Постановка задачи. Начало системы координат $0x_1x_2x_3$, в которой рассматриваем движение аппарата, находится в центре Земли. Ось $0x_1$ направлена на исходное положение аппарата, ось $0x_3$ — в сторону его движения и перпендикулярна оси $0x_1$, ось $0x_2$ образует с указанными осями правую тройку.

Математическую модель объекта управления представляет система обыкновенных дифференциальных уравнений (ОДУ):

$$\left.\begin{aligned} \frac{dx_1}{dt} &= x_2, \\ \frac{dx_2}{dt} &= -\frac{\gamma}{r^3}x_1 - (1-u)qx_2, \\ \frac{dx_3}{dt} &= x_4, \\ \frac{dx_4}{dt} &= -\frac{\gamma}{r^3}x_3 - (1-u)qx_4 \end{aligned}\right\} \qquad (3)$$

при заданных начальных условиях $x_i(0) = x_i^0$, $i \in [1{:}4]$. Приняты следующие обозначения: x_1, x_3 — координаты центра масс космического аппарата; x_2, x_4 — компоненты его скорости; u — управление; $r = \sqrt{x_1^2 + x_3^2}$ — значение радиус-вектора аппарата; γ — гравитационная постоянная Земли; $q = c\mathbf{e}^{\frac{h-x_1}{\eta}}v$ — составляющая аэродинамической силы; $v = \sqrt{x_2^2 + x_4^2}$ — скорость аппарата; c — его аэродинамическая характеристика; h — высота атмосферы; η — согласующий коэффициент модели атмосферы.

Определены функционалы качества управления:

$$f_1(u) = \max_{\tau \in [0;\hat{\tau}]} \frac{(x_1^2(\tau) + x_3^2(\tau))\sqrt{x_2^2(\tau) + x_4^2(\tau)}}{g_0 r_3^2} \to \min_{u \in D_u}, \qquad (4)$$

$$f_2(u) = x_3(\hat{\tau}) - l \to \min_{u \in D_u}, \qquad (5)$$

имеющие смысл максимальной перегрузки и отклонения от заданной точки на поверхности Земли соответственно. Здесь $r_3 = 6371$ км — радиус Земли; $g_0 = 9{,}81$ м/с^2 — ускорение свободного падения; $\hat{\tau}$ — длительность полета; l —координата по оси $0x_3$ заданной точки на поверхности Земли; $D_u = \{u(\tau) \| u(\tau)| \leqslant u^+\}$ — множество допустимых управлений; u^+ — заданная положительная константа.

Задача состоит в определении допустимого управления $u^*(\tau) \in D_u$, удовлетворяющего на решениях системы (3) условиям (4), (5).

Задача (3)—(5) представляет собой двухкритериальную задачу оптимального управления. При всех хорошо известных недостатках метода решения данной задачи, основанного на сведении ее к задаче нелинейного программирования, используем именно этот метод [15].

Покроем интервал $[0; \hat{\tau}]$ равномерной сеткой с узлами τ_i, $i \in [0 : |U|]$, и будем искать оптимальное управление $u^*(\tau)$ в классе кусочно-постоянных функций. Обозначим $U = (u_1, u_2, ..., u_{|U|})$ — $(|U| \times 1)$-вектор, где $u_i = u(\tau_i)$. Тогда задачи (4), (5) примут вид

$$\min_{U \in D_U} f_1(U) = \min_{U \in D_U} \max_{\tau \in [0;\hat{\tau}]} \frac{(x_1^2(\tau) + x_3^2(\tau))\sqrt{x_2^2(\tau) + x_4^2(\tau)}}{g_0 r_3^2}, \qquad (6)$$

$$\min_{U \in D_U} f_2(U) = \min_{U \in D_U} (x_3(\hat{\tau}) - l), \qquad (7)$$

где $D_U = \{u_i \| u_i| \leqslant u^+,\ i \in [1 : |U|]\}$.

Таким образом, имеем двухкритериальную задачу оптимизации с критериальными функциями $f_1(U), f_2(U)$, вектором варьируемых параметров U и множеством допустимых значений вектора варьируемых параметров D_U.

Результаты. Исследование выполнено при следующих начальных условиях космического аппарата: $x_1(0) = 6471$ км, $x_3(0) = 0$, $x_2(0) = 0{,}1$ км/с, $x_4(0) = 7$ км/с. Размерность вектора варьируемых параметров $|U|$ принята равной 100. Число членов

Таблица 5

Состав эволюционных операторов: задача о спуске космического аппарата

Оператор	*CGA*			
	GA_1	GA_2	GA_3	GA_4
Рекомбинация	Аутбридинг	Инбридинг	Аутбридинг	Инбридинг
Кроссовер	*BLX*	Эвристический	Эвристический	*BLX*
Селекция	Турнирная	Турнирная	Турнирная	Турнирная

популяции $|S|$ варьируем от 100 до 200, число поколений $\hat{t}$ — от 5 до 100. Значения остальных свободных параметров коалгоритма совпадают со значениями, использованными в п. 4: $|\boldsymbol{S}| = 4$, $t_a = 5$, $n_p = 0{,}1$, $|S|_{\min} = 0{,}12 \cdot |S|$, $\xi_m = 0{,}05$. Состав эволюционных операторов представлен в табл. 5.

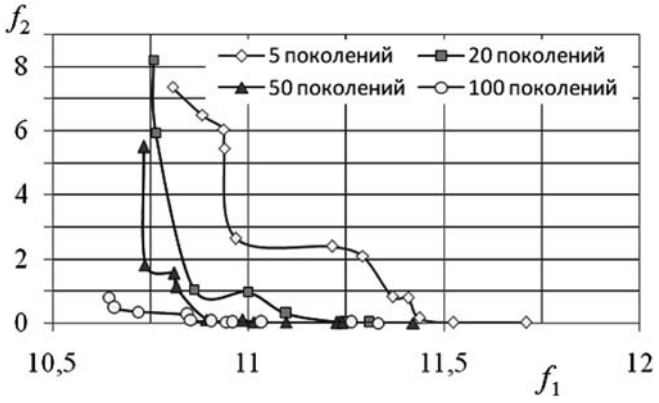


Рис. 8. Результат Парето-аппроксимации: задача о спуске космического аппарата; коалгоритм; $|S| = 100$

Результаты Парето-аппроксимации иллюстрирует рис. 8. Рисунок показывает быстрое улучшение качества Парето-аппроксимации с ростом числа поколений и небольшое число найденных паретовских точек.

В процессе функционирования коалгоритма происходит интенсивное перераспределение ресурсов между коэволюционирующими субпопуляциями. Рис. 9 иллюстрирует данный процесс. Показана численность субпопуляций $|S_i|$, $i \in [1{:}4]$ в функции номера интервала адаптации n_a. Как и для тестовых задач $ZDT1$ — $ZDT4$, из рис. 9 следует, что на первых интервалах адаптации побеждают субалгоритмы, использующие аутбридинг (диверсификация поиска). После некоторого числа интервалов адаптации начинают побеждать субалгоритмы на основе инбридинга (интенсификация поиска). Отметим также, что каждый из четырех рассматриваемых субалгоритмов выигрывал в процессе Парето-аппроксимации хотя бы один раз. Это означает, что коалгоритм не блокирует проигравшие субалгоритмы, а до конца вычислений учитывает успехи их всех.

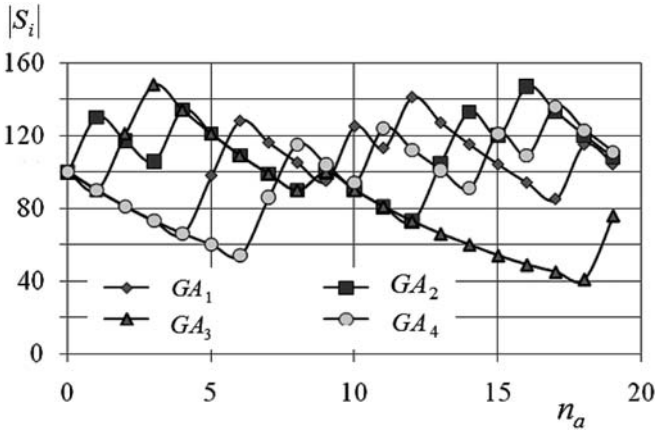


Рис. 9. Изменение численности субпопуляций в процессе итераций: задача о спуске космического аппарата; $|S| = 100$

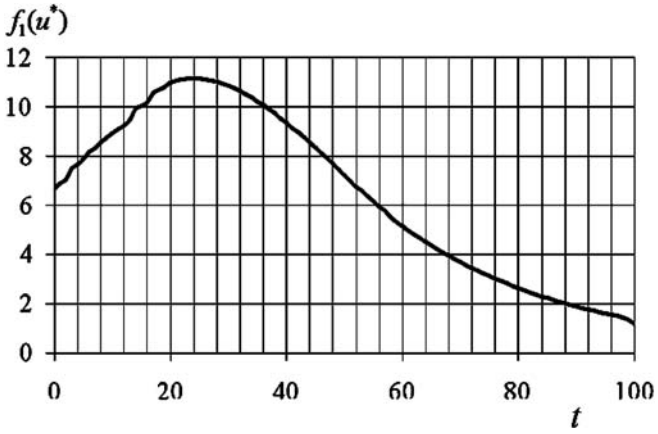


Рис. 11.Зависимость перегрузки $f_1(u^*)$ от времени

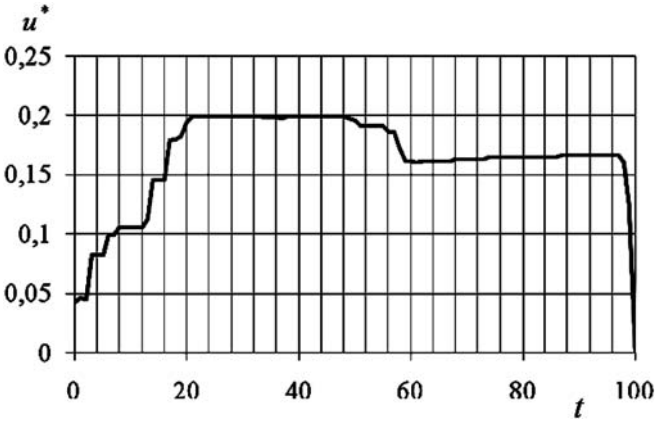


Рис. 10. Вариант оптимального управления $u^*(t)$: $f_1(u^*) = 11{,}15$; $f_2(u^*) = 0{,}34$

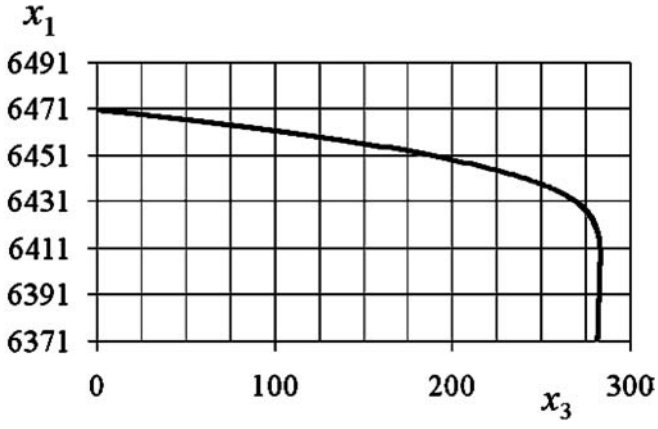


Рис. 12. Оптимальная траектория: $x_3(x_1)$

Характер одного из найденных оптимальных управлений $u^*(t)$ показан на рис. 10. Данному управлению соответствует максимальная перегрузка $f_1(u^*) = 11{,}15$ и отклонение от заданной точки на поверхности Земли $f_2(u^*) = 0{,}34$. Соответствующая зависимость перегрузки от времени приведена на рис. 11, а траектория спуска — на рис. 12.

Заключение

В работе предложен когенетический алгоритм Парето-аппроксимации, отличительной чертой которого является возможность использования субалгоритмами не только различных значений свободных параметров, но и разных наборов эволюционных операторов. Для обеспечения равномерности покрытия множества и фронта Парето в алгоритме используется отбор получаемых решений по критерию разреженности.

Выполнена программная реализация алгоритма, которая предусматривает совместное функционирование от двух до четырех субалгоритмов, использующих заданные пользователем эволюционные операторы рекомбинации, кроссовера, мутации и селекции. Реализация выполнена на языке программирования C++, среда реализации — *Builder* 6.0, операционная система — *Windows XP*.

Широкое исследование эффективности алгоритма и разработанного программного обеспечения проведено на известном наборе тестовых задач многокритериальной оптимизации $ZDT1$ — $ZDT4$. Эффективность Парето-аппроксимации оценена с помощью трех индикаторов: равномерности распределения решений; числа найденных истинно паретовских (недоминируемых) решений и общего числа испытаний. Результаты исследования показывают, что когенетический алгоритм по всем указанным индикаторам, кроме числа испытаний, превосходит соответствующий однопопуляционный генетический алгоритм, и это превосходство может достигать 100 %.

Эффективность Парето-аппроксимации на основе предложенного когенетического алгоритма продемонстрирована также на примере двухкритеральной задачи оптимального управления спуском космического аппарата в атмосфере Земли.

В целом, результаты исследования показывают перспективность дальнейшей разработки алгоритма. Предполагается, в частности, самостоятельное исследование метаоптимизационных свойств алгоритма, а также его параллельная реализация.

Список литературы

1. **Подиновский В. В., Ногин В. Д.** Парето-оптимальные решения многокритериальных задач. М.: Физматлит, 2007. 256 с.
2. **Карпенко А. П., Митина Е. В., Семенихин А. С.** Популяционные методы аппроксимации множества Парето в задаче многокритериальной оптимизации. Обзор // Наука и образование: электронное научно-техническое издание. 2012. № 4. URL: http://www.technomag.edu.ru/doc/363023.html
3. **Guliashki У., Toshev H., Korsemov Ch.** Survey of Evolutionary Algorithms Used in Multiobjective Optimization // Problems of Engineering Cybernetics and Robotics. 2009. Vol. 60. P. 42—54.
4. **Wang X.** Hybrid nature-inspired computation method for optimization / Doctoral Dissertation. Helsinki University of Technology, TKK Dissertations 161, Espoo 2009.
5. **El-Abd Kamel M.** A taxonomy of cooperative search algorithms / In: Hybrid Metaheuristics. Second International Workshop. 2005. Vol. 3636. P. 32—41.
6. **Raidl G. R.** A Unified View on Hybrid Metaheuristics // Lecture Notes in Computer Science. 2006. Vol. 4030. P. 1—12.
7. **Сергиенко Р. Б., Семенкин Е. С.** Коэволюционный генетический алгоритм решения сложных задач условной оптимизации // Вестник Сибирского государственного аэрокосмического университета имени академика М. Ф. Решетнёва. 2009. № 2 (23). С. 17—21.
8. **Rosin C., Belew R.** New Methods for Competitive Coevolution // Evolutionary Computation. 1997. № 5. P. 1—29.
9. **Jie J., Han Ch., Zeng J.** An Extended Mind Evolutionary Computation Model for Optimizations // Applied Mathematics and Computation. 2007. N 185 (2). P. 1038—1049.
10. **Karpenko A. P. Svianadze Z. O.** Meta-optimization based on self-organizing map and getetic algorithm // Optical Memory and Neural Networks (Information Optics). 2011. Vol. 20, N 4. P. 279—283.
11. **Deb K.** Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Evolutionary Computation. 1999. Vol. 7 (3). P. 205—230.
12. **Дивеев А. И., Северцев Н. А.** Метод сетевого оператора для синтеза системы управления спуском космического аппарата при неопределенных начальных условиях // Проблемы машиностроения и надежности машин. Машиноведение. 2009. № 3. С. 85—91.
13. **Семенкин Е. С.** Эволюционные методы моделирования и оптимизации сложных систем / Е. С. Семенкин и др. // Электронные публикации НГУ, URL: http://library.krasu.ru/ft/ft/_umkd/22/u_lectures.pdf
14. **Zitzler E., Thiele L., Marco Laumanns M., Fonseca C. M., da Fonseca V. G.** Performance Assessment of Multiobjective Optimizers: An Analysis and Review // IEEE Transactions of Evolutionary Computation. 2003. Vol. 7 (2). P. 117—132.
15. **Федоренко Р. П.** Приближенное решение задач оптимального управления. М.: Наука, 1978. 488 с.

УДК 004.021

Д. А. Потапов, аспирант,
Национальный исследовательский
ядерный университет "МИФИ", г. Москва,
e-mail: div-x15@yandex.ru

Оптимизация смещений и дисперсий оценок параметров математических моделей при обработке сглаженных экспериментальных данных[1]

При моделировании термодинамических свойств растворов в большинстве случаев исследователю недоступны непосредственные данные эксперимента, так как в литературе приводятся результаты сглаживания экспериментальных значений полиномами. Эти данные исследователи используют для проверки адекватности различных математических моделей, обычно являющихся нелинейными. Вследствие нелинейности уравнений моделей возникает смещенность оценок параметров. В данной работе исследуется влияние степени сглаживающего полинома на смещение и дисперсию этих оценок на примере кластерной модели растворов. Разработан алгоритм оптимизации смещений и дисперсий оценок параметров математических моделей при обработке сглаженных экспериментальных данных.

Ключевые слова: *моделирование свойств растворов, проверка адекватности модели, математическая модель, метод наименьших квадратов, смещенность оценки*

Введение

При исследовании растворов важным этапом в определении их термодинамических характеристик является оптимизация функции невязки математической модели. При этом в большинстве случаев в литературе недоступны данные, полученные непосредственно из эксперимента, а приводится результат их сглаживания полиномами некоторой степени n. Коэффициенты полинома определяют методом наименьших квадратов [1]. При адекватно выбранной степени полинома n сглаживание позволяет уменьшить дисперсию оценок параметров [2]. В качестве результата предоставляются значения найденного полинома в некоторых точках, отличных от точек, в которых были измерены экспериментальные значения. Таким образом, часть информации об эксперименте утрачивается.

Полученные данные используют другие исследователи для проверки адекватности математических моделей. На данном этапе изменить степень описывающего полинома и получить информацию о значениях аргумента, при которых проводились непосредственные измерения, обычно не представляется возможным. Вследствие погрешностей измерения экспериментальные данные представляют собой случайные величины, поэтому коэффициенты полинома также являются случайными величинами, поэтому параметры модели также случайны. Идентификацию параметров осуществляют с помощью метода наименьших квадратов, который приводит к смещенности оценок в случае нелинейности модели [3].

В настоящей работе исследуется влияние степени сглаживающего полинома на статистические характеристики искомых параметров нелинейных моделей, также предлагается методика обработки сглаженных данных, обеспечивающая нахождение более точных и устойчивых значений параметров модели по сравнению с обычным применением метода наименьших квадратов.

Влияние степени сглаживающего полинома на математическое ожидание и дисперсию параметров модели

Исследование проводилось на примере кластерной модели растворов [4], связывающей осмотический коэффициент раствора φ с моляльностью раствора m. Ниже приведены уравнения модели:

$$n_1 = 55{,}508;$$

$$A_1 = 0{,}5115\ln(10);$$

$$fe_D(B, m) = \frac{-(A_1 z_1 z_2 \sqrt{I(m)})(1 + B\sqrt{I(m)})}{(B\sqrt{I(m)})^3} -$$

$$- 2\ln(1 + B\sqrt{I(m)}) - \frac{1}{1 + B\sqrt{I(m)}};$$

$$fe_A(A_s, m) = \frac{-A_s X(m)}{1 + A_s X(m)(1 - X(m))};$$

$$fe_h(q_1, q_2, h, r, m) = \frac{(q_1 + q_2)X(m)(1 - X(m))^{r-1} h}{((1 - hX(m))(1 - X(m))^r)};$$

$$\varphi(m, h, r, A_s, B) = 1 + \frac{fe_h(q_1, q_2, h, r, m)}{(q_1 + q_2)} +$$

$$+ fe_A(A_s, m) + fe_D(B, m),$$

где n_1 — число молей воды в килограмме; A_1 — параметр уравнения Дебая—Хюккеля—Онзагера; z_1, z_2 — заряды соответственно катиона и аниона растворенного вещества; q_1, q_2 — число катионов и анионов в молекуле; m — моляльность растворенного вещества; B — коэффициент Дебая; fe_D, fe_A и

[1] Работа выполнена при финансовой поддержке ФЦП "Кадры инновационной России" (Государственный контракт № П2274).

Таблица 1

Смещения и дисперсии оценок параметров модели при различных степенях сглаживающего полинома

Параметры	Степень сглаживающего полинома n							
	2	3	4	5	6	7	8	9
$\Delta_{сгл}$	0,2162	0,0168	0,0026	0,0019	0,0023	0,0037	0,0032	0,0028
$\Delta_{шум}$	0,1961	0,0317	0,0084	0,0007	0,0017	0,0021	0,0094	0,0143
$D_{сгл}$	0,0031	0,0048	0,0053	0,0054	0,0054	0,0057	0,0060	0,0061
$D_{шум}$	0,0136	0,0183	0,0137	0,0197	0,0172	0,0185	0,0155	0,0144

fe_h — величины, характеризующие вклады соответственно процессов электростатического взаимодействия, ассоциации и гидратации в значение осмотического коэффициента раствора; A_s — коэффициент, характеризующий степень ассоциации в растворе; φ — осмотический коэффициент; I — ионная сила раствора; X — мольная доля растворенного вещества; h — средняя степень гидратации иона; r — коэффициент, характеризующий дисперсию распределения ионов в растворе по степеням гидратации.

В целях уменьшения влияния алгоритма оптимизации на нахождение минимума функции невязки, а также возможности отображения результатов на графике, было положено $A_s = 0{,}0001$, $B = 2{,}1497$, таким образом, четырехпараметрическая модель была преобразована в двухпараметрическую с параметрами h и r.

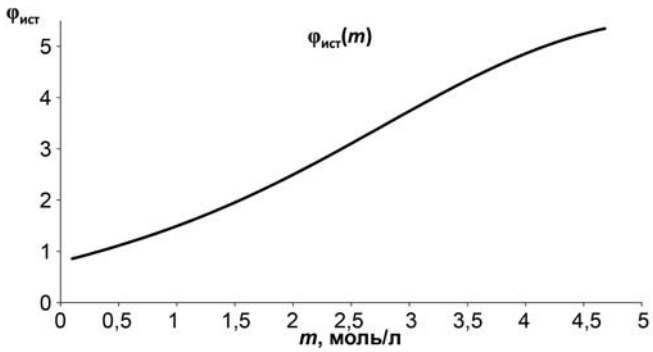


Рис. 1. Зависимость значений осмотического коэффициента, принятых за истинные для целей моделирования, от моляльности раствора

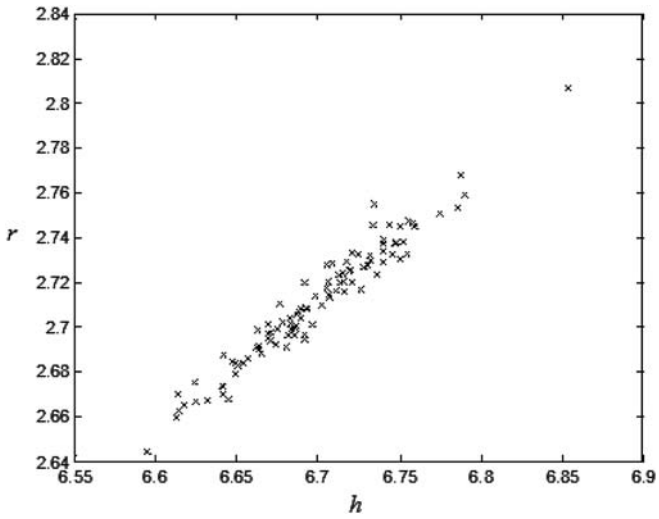


Рис. 2. Распределение оценок параметров модели при аппроксимации по сглаженным данным для $n = 4$

Для определения точности решения необходимо знать истинные значения измеряемой величины. Пусть для целей моделирования

$$\varphi_{ист}(m) = 1 + \frac{fe_h(q_1, q_2, h, r, m)}{(q_1 + q_2)} + fe_A(A_s, m) + fe_D(B, m),$$

где $q_1 = 1$, $q_2 = 3$, $h = 6{,}6918$, $r = 2{,}7068$, $A_s = 0{,}0001$, $B = 2{,}1497$.

График $\varphi_{ист}(m)$ приведен на рис. 1.

Экспериментальные данные могут быть получены из истинных с помощью следующего выражения:

$$\varphi_{эксп}(m) = \varphi_{ист}(m) + \delta(m), \qquad (1)$$

где $\delta(m)$ — погрешность измерений. Для моделирования будем считать $\delta(m)$ случайной величиной, имеющей равномерное распределение на отрезке $[-5\ \%\varphi_{ист}(m);\ 5\ \%\varphi_{ист}(m)]$.

В качестве аргументов сглаженных значений возьмем вектор $M_{сгл}$:

$$M_{сгл} = [0{,}1;\ 0{,}2;\ 0{,}3;\ 0{,}4;\ 0{,}5;\ 0{,}6;\ 0{,}7;\ 0{,}8;\ 0{,}9;\ 1;\ 1{,}2;\ 1{,}4;\ 1{,}6;\ 1{,}8;\ 2;\ 2{,}2;\ 2{,}4;\ 2{,}6;\ 2{,}8;\ 3;\ 3{,}2;\ 3{,}4;\ 3{,}6;\ 3{,}8;\ 4;\ 4{,}2;\ 4{,}4;\ 4{,}6;\ 4{,}6822].$$

В общем случае вектор $M_{сгд}$ не совпадает с вектором значений $M_{изм}$, в которых проводились непосредственные измерения. Для рассматриваемой системы вектор $M_{изм}$ содержал 79 значений, тогда как размерность вектора $M_{сгл}$ равна 29.

Возьмем полиномы степени $n = 2,3, \ldots, 9$ с неизвестными коэффициентами. Для каждой степени вычислим коэффициенты полинома с помощью метода наименьших квадратов, рассчитаем значения этого полинома в точках $M_{сгл}$, после чего с помощью того же метода рассчитаем параметры модели h и r. Данный расчет был осуществлен $N = 2000$ раз для каждой степени n, причем на каждом шаге генерировались новые значения $\varphi_{эксп}(m)$. Аналогичная процедура была проделана для сглаженных данных с добавлением случайного шума, после чего были рассчитаны дисперсии найденных параметров модели и их отклонения от истинных значений. На рис. 2

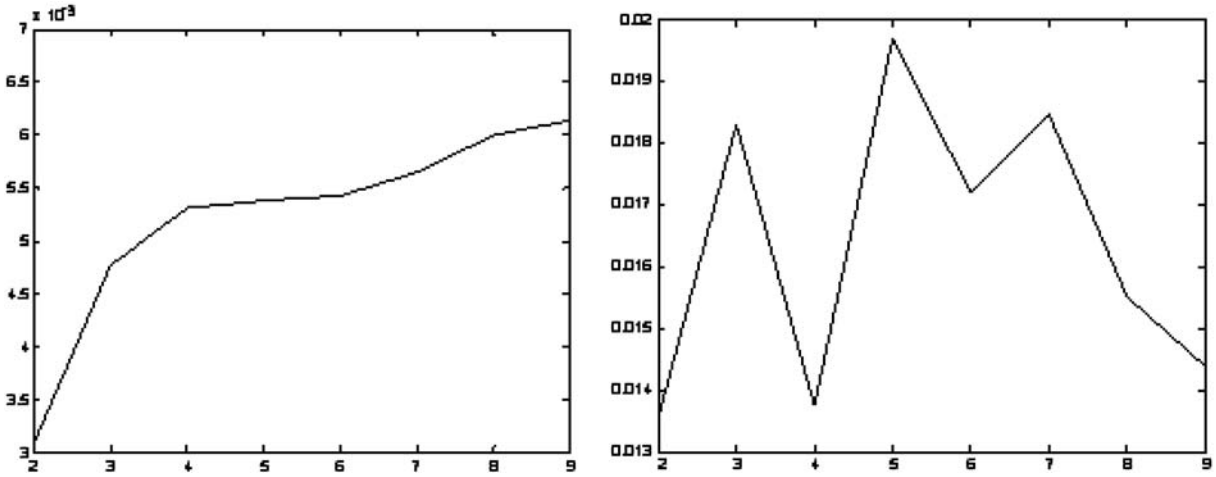


Рис. 3. Зависимости полученных дисперсий от степени *n* аппроксимирующего полинома для параметров, полученных по сглаженным и сглаженным с добавлением шума экспериментальным данным

показан пример распределения оценок параметров модели для $n = 4$.

Значения дисперсий D и отклонений математических ожиданий от истинных значений Δ для параметров, полученных по сглаженным и сглаженным с добавлением шума экспериментальным данным, приведены в табл. 1.

При моделировании непосредственно по экспериментальным данным, без использования сглаживания, результаты получаются следующими:

$$D = 0{,}0040;\ \Delta = 0{,}0057.$$

На рис. 3 приведены графические зависимости полученных дисперсий от степени n аппроксимирующего полинома.

Корректировка полученных значений параметров при работе со сглаженными данными

Из табл. 1 и рис. 2 видно, что явной закономерности между дисперсией D и n в случае зашумленных данных не наблюдается, в случае обработки сглаженных данных дисперсия растет с ростом степени сглаживающего полинома n. Минимум дисперсии для сглаженных данных соответствует $n = 2$, однако, при данном значении n возникает существенная смещенность оценок. При этом при $n \geqslant 4$ отклонения от истинного значения малы (h и r обычно рассчитываются с точностью до второго знака после запятой). Дисперсия результатов, полученных по зашумленным данным, существенно больше полученной по сглаженным данным при всех значениях n.

Если смещение по каждому из параметров известно, то при его вычитании из найденных значений можно получить параметры с минимальной дисперсией, являющиеся несмещенными. Однако прямое вычисление значений смещений невозможно в виду отсутствия у исследователя информации об истинных значениях моделируемой величины.

Приведенные ниже результаты численного моделирования показывают, что в случае сглаживания экспериментальных данных полиномом, обеспечивающим относительно малые дисперсии оценок и малые смещения (для рассматриваемой модели это $n \geqslant 4$), в целях определения значений смещений в качестве истинных значений параметров можно взять оценки, найденные по сглаженным данным методом наименьших квадратов. Численное моделирование проводилось на основании следующих соображений. При аппроксимации сглаженных с помощью полинома степени n данных уравнениями математической модели значения найденных параметров приобретают некоторое смещение и разброс относительно истинных значений. И смещение, и дисперсия могут быть определены с помощью исследования уравнения модели. Для рассматриваемой модели данные величины приведены в табл. 1.

Таблица 2

Смещения и дисперсии оценок параметров, полученные в случае принятия рассчитанных параметров модели в качестве истинных при степени сглаживающего полинома $n = 2$

Параметры	Номер эксперимента							
	1	2	3	4	5	6	7	8
$\Delta_{сгл}$	0,2168	0,2165	0,2161	0,2159	0,2163	0,2166	0,2165	0,2161
$D_{сгл}$	0,0033	0,0032	0,0035	0,0031	0,0033	0,0036	0,0034	0,0031

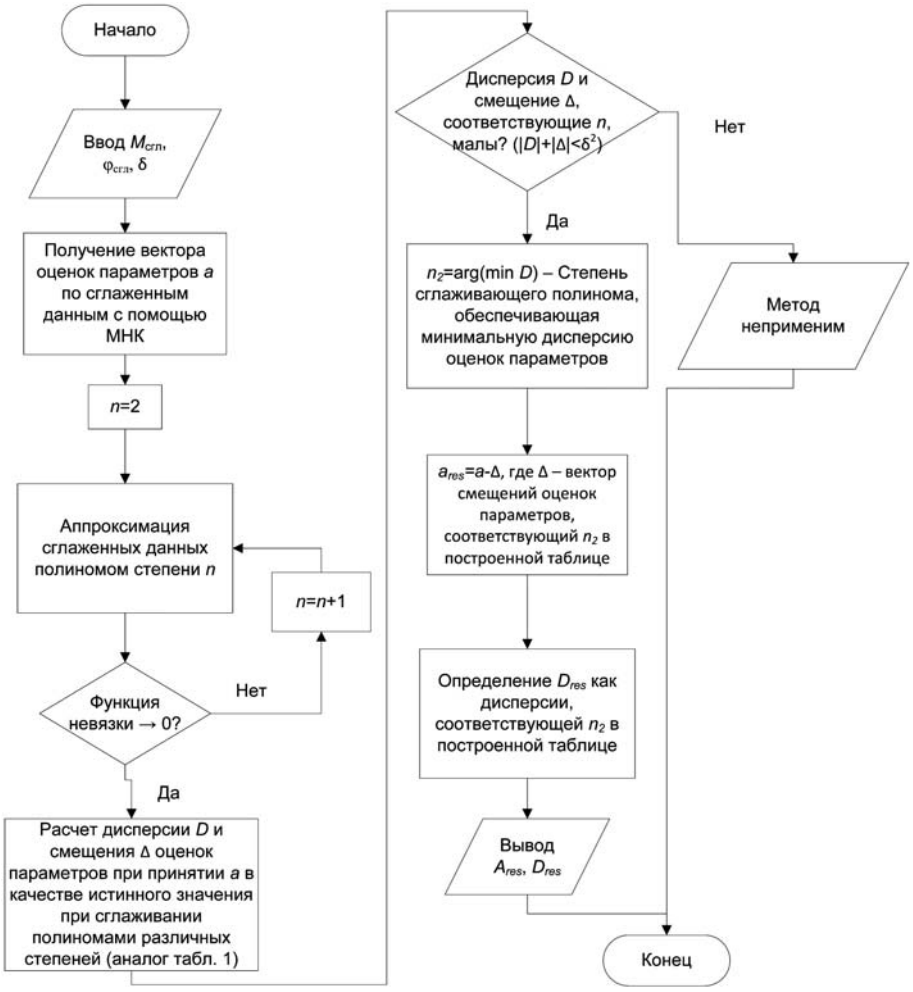


Рис. 4. Блок-схема алгоритма оптимизации смещений и дисперсий оценок параметров математических моделей

Возьмем несколько точек из области, в которую могут попадать найденные параметры, и пересчитаем табл. 1, принимая в качестве истинных найденные значения. Пересчет будем проводить только для *n*, соответствующего минимальной дисперсии. Для рассматриваемой модели $n = 2$. Результаты пересчета приведены в табл. 2. Из таблицы видно, что для $n = 2$ смещение и дисперсия, получаемые по истинным значениям ($\Delta_{сгл} = 0{,}2168$, $D_{сгл} = 0{,}0031$) близки к значениям смещения и дисперсии, полученным в случае принятия рассчитанных параметров модели в качестве истинных. Таким образом, в качестве параметров распределения оценок для истинных значений могут быть взяты данные табл. 2, которые может рассчитать исследователь, не имеющий доступа к истинным или непосредственно измеренным значениям.

В большинстве случаев непосредственная информация о степени сглаживающего полинома недоступна исследователю. Ее можно определить, последовательно аппроксимируя сглаженные данные полиномами различной степени. При соответствии степеней сумма квадратов невязок резко устремится к нулю, что свидетельствует о нахождении искомого *n*. На рис. 4 приведена блок-схема алгоритма оптимизации дисперсии и смещения оценок параметров.

Заключение

При моделировании свойств растворов исследователи обычно имеют дело с нелинейными моделями. Кроме того, данные для моделирования представляют собой измеренные данные, сглаженные полиномом неизвестной степени *n*. При этом нарушаются условия теоремы Гаусса—Маркова [3], и применяемый большинством исследователей метод наименьших квадратов не обеспечивает несмещенности и эффективности найденных оценок параметров. Это может стать причиной расхождения некоторых проверочных зависимостей, в результате чего исследователем может быть сделан ошибочный вывод о неадекватности исследуемой им математической модели. Рассмотренный в данной работе на примере кластерной модели растворов алгоритм оптимизации смещений и дисперсий оценок параметров модели позволяет преодолеть описанные ограничения при использовании метода наименьших квадратов и делать более объективные выводы об адекватности исследуемых моделей.

Список литературы

1. **Wolberg J. R.** Data Analysis Using the Method of Least Squares: Extracting the Most Information from Experiments. Berlin: Springer, 2005.

2. **Седелев Б. В.** Регрессионные модели и методы оценки параметров и структуры экономических процессов. М.: Изд-во Московского инженерно-физического института, 2009.

3. **Plackett R. L.** Some Theorems in Least Squares // Biometrika. 1950. V. 37, № 1—2. P. 149—157.

4. **Рудаков А. М., Майкова Н. С., Сергиевский В. В.** Исследование сольватации и ассоциации в бинарных растворах на основе кластерной модели // Труды конференции "Проблемы сольватации и комплексообразования в растворах". Иваново: Изд. Институт химии растворов им. Г. А. Крестова. 2011. С. 14.

УДК 004.62

М. П. Шарабайко, аспирант,
Н. Г. Марков, д-р техн. наук, проф., зав. кафедрой,
Томский политехнический университет,
e-mail: sme_box@tpu.ru

Исследование эффективности кодирования цветных изображений с помощью фракталов*

Предложены и исследованы алгоритмы фрактального сжатия цветных изображений. Исследованы структуры распределения бит в файлах фрактально сжатых изображений. На основе полученных результатов увеличена степень сжатия одного из алгоритмов. Показаны направления дальнейших исследований по данной тематике.

Ключевые слова: *фрактальное сжатие цветных изображений, структура файла фрактального кода, цветовые системы, RGB, YUV*

Введение

Задача сжатия видеоинформации является актуальной и востребованной, применяемой как для хранения домашнего фотоархива, так и для долговременного хранения космических, медицинских и иных статических изображений и видеопоследовательностей. При этом в одном случае важна высокая степень сжатия изображений при приемлемых, мало заметных для глаза потерях, а в другом — как можно меньше потерь или полное их отсутствие при хорошей степени сжатия.

В настоящее время решение такой задачи чаще всего осуществляется с помощью алгоритмов, основанных на дискретном косинусном преобразовании (ДКП). Для повседневного сжатия изображений часто используется алгоритм *JPEG* [1]. Сжатие видео строится на алгоритмах *MPEG*-2 [2], *h*.264 [3], *h*.265 [4], по сути, являющихся все тем же сжатием на основе ДКП, с добавлением предсказаний, устранением избыточности хранения информации между соседними данными.

Фрактальное сжатие статических изображений строится на абсолютно ином принципе — на самоподобии частей изображения, при этом информация о значениях яркости пикселей в сжатом потоке не хранится, а хранится лишь информация о структуре изображения, о взаимозависимостях его частей. Развитием работ по сжатию изображений являются исследования, нацеленные на создание быстродействующих алгоритмов фрактального сжатия изображений в градациях серого, результаты которых изложены в статье [5].

В данной статье рассмотрен общий подход к представлению и сжатию цветных изображений. С учетом особенностей идеи фрактального сжатия предложены и исследованы несколько алгоритмов фрактального сжатия таких изображений.

Задачи фрактального сжатия цветных изображений

Особенности фрактального сжатия изображений. Фрактальное сжатие изображений выполняют в несколько этапов. Сначала проводят разбиение изображения на множество кодируемых (сжимаемых) блоков, исторически называемых ранговыми, и на множество доменных блоков, которые будут использованы для сжатия, являясь чем-то вроде словаря кодирования.

На следующем этапе происходит кодирование каждого рангового блока. Оно основано на поиске такого доменного блока из множества сформированных, который с помощью преобразований поворота, изменения контраста и сдвига яркости будет максимально подобен текущему ранговому блоку. Более подробно процесс поиска соответствий изложен в работе [5].

Наконец, полученные коэффициенты преобразований и структура разбиения изображения сохраняются для его последующего восстановления декодером.

Представление цветных изображений. Обычно цветное изображение имеет три цветовых компоненты (плоскости, канала), вместо одной компоненты для изображений в градациях серого.

Самой распространенной цветовой моделью, используемой для цифрового представления цветных изображений и для передачи цвета в телевизорах и мониторах, является аддитивная цветовая модель *RGB*. В ней используется красный (*R*), зеленый (*G*) и синий (*B*) каналы для хранения информации о цвете (рис. 1, см. третью сторону обложки).

Система *RGB* имеет большой цветовой охват, но основной ее недостаток применительно к сжатию изображений — равнозначность каждого канала. Из рис. 1 видно, что каналы изображения имеют примерно одинаковую структуру, тем не менее,

* Работа выполнена при финансовой поддержке гранта УМНИК.

потери в любом из каналов приводят к равноценным потерям в качестве цветного изображения.

Система *YUV* [6], с учетом особенностей восприятия цвета человеческим глазом, представляет цвет в виде канала яркости (*Y*) и двух каналов цветности (*U*, *V*). При таком представлении яркостная составляющая имеет приоритет перед каналами цветности, поскольку цвет воспринимается человеческим глазом хуже яркости, и ошибки в каналах *U* и *V* менее заметны, нежели в канале *Y* (рис. 2, см. третью сторону обложки).

Этим свойством системы *YUV* пользуются при сжатии изображений, не только допуская больше потерь в плоскостях цветности, но и прореживая цветностные каналы изображения.

В системе *RGB* каждый пиксель изображения занимает 24 бита (по 8 бит на каждую компоненту). В системе *YUV* возможно представление цветного изображения теми же 24 битами на пиксель (*YV*24). Однако преобразовав изображение из системы *RGB* в систему *YUV* [7], можно уменьшить плоскости цветности в 2 раза по ширине и высоте. В результате каждые 8 бит цветности обеих плоскостей соответствуют четырем яркостным пикселям изображения (рис. 3). Такая система обозначается *YV*12 [6], поскольку каждый пиксель изображения представляется 12 битами (в среднем).

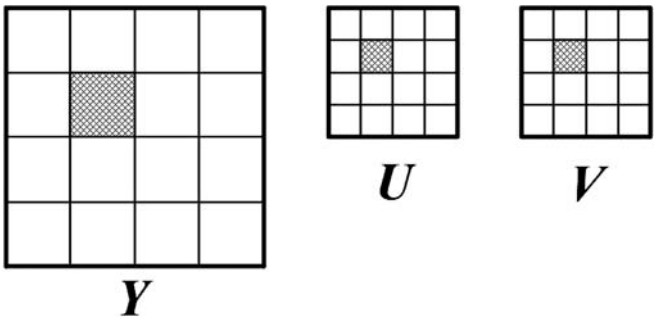


Рис. 3. Представление изображения в пространстве *YV*12

Фрактальное сжатие цветных изображений

Поскольку фрактальное сжатие изображений не хранит информацию о значениях пикселей изображения, а выявляет взаимозависимости отдельных его участков, нет необходимости использовать цветовую систему *YV*12. Незначительные потери, привносимые использованием этой системы, исказят изображения и изменят результат сжатия, никак не повлияв на степень сжатия.

При этом возможность кодирования каналов цветности с дополнительными потерями может оказаться достаточно полезной, поэтому при фрактальном сжатии цветного изображения целесообразно работать в цветовой системе *YUV*, 24 бита на пиксель (*YV*24).

Самый простой способ фрактально сжать цветное изображение — закодировать каждую цветовую плоскость отдельно, независимо от других. При таком подходе цветное изображение равнозначно трем изображениям в градациях серого. Плюсы данного подхода заключаются в простоте реализации и нахождении наилучших соответствий для ранговых блоков каждой плоскости, т. е. качество сжатия должно быть наилучшим. Отрицательной стороной подхода является игнорирование взаимозависимостей цветовых плоскостей изображения. Число блоков для перебора возрастает в 3 раза (в сравнении с независимым сжатием одной плоскости или сжатием изображения в градациях серого), и, следовательно, возрастает время сжатия.

Цветовые плоскости изображений обычно имеют похожую структуру (например, изображения на рис. 1, рис. 2, см. третью сторону обложки), т. е. имеют определенную корреляцию, которую можно более эффективно использовать при фрактальном сжатии цветных изображений. Например, найдя наиболее подходящий доменный блок для данного рангового блока одной из цветовых компонент, можно считать, что вероятнее всего наиболее подходящий доменный блок для ранговых блоков двух остальных цветовых плоскостей будет находиться в том же месте изображения.

В итоге, учитывая все изложенное, были предложены следующие алгоритмы фрактального сжатия цветных изображений (ниже описана суть этих алгоритмов).

Алгоритм А. Независимое сжатие каждого канала цветного изображения. При этом не используется никакая информация о корреляции значений яркости пикселей между каналами.

Алгоритм Б. Каждый макроблок изображения состоит из трех каналов. Для рангового блока каждого канала поиск доменного блока ведется независимо среди множества доменных блоков текущего канала. Решение о дополнительном разбиении макроблока принимается исходя из средней ошибки сопоставления блоков.

Алгоритм В. Поиск доменно-рангового сопоставления ведется для ранговых блоков яркостной составляющей. При нахождении удачного соответствия блоков далее ищутся коэффициенты преобразований для ранговых блоков цветоразностных компонент по доменным блокам, находящимся на одном участке изображения вместе с доменным блоком, выбранным для сжатия яркостного рангового блока.

Исследование эффективности алгоритмов фрактального сжатия

Для оценки эффективности предложенных алгоритмов они были реализованы на языке программирования *C++* в среде *Microsoft Visual Studio 2010 Express*.

Для оценки потерь в декодированном (восстановленном) изображении (оценка качества изо-

бражения) используются метрики *SSIM* и *PSNR*, реализованные в работе [8].

Исследования проводили для библиотеки тестовых изображений группы фрактального кодирования и анализа из университета Ватерлоо (Канада) [9]. Использовались изображения *lenna* (рис. 4, *а*, см. третью сторону обложки) *peppers* (рис. 4, *б*) и *frymire* (рис. 4, *в*).

Численные эксперименты с использованием разработанных программных реализаций алгоритмов фрактального сжатия цветных изображений проводили на ПЭВМ со следующими значимыми характеристиками:

- процессор: *Intel Core i3* 530 2,93 ГГц,
- ОЗУ: 2 Гбайт *DDR3*,
- *ОС Windows* 7 × 64.

В табл. 1 приведены результаты исследования указанных алгоритмов с ранговыми блоками размерами 4—8 пикселей в ширину и высоту, 7 бит — под коэффициенты изменения контраста, 5 бит — под коэффициенты сдвига яркости (обоснование выбора именно таких значений приведено в работе [10]). Здесь же для сравнения показаны результаты сжатия тестовых изображений с помощью алгоритма *JPEG*.

Алгоритм А позволяет сжимать цветовые плоскости изображений независимо друг от друга и является более адаптивным, поскольку удачное соответствие для разных цветовых компонент одного и того же участка изображения можно найти на разных уровнях разбиения. Это дает оптимальную степень сжатия и лучшее качество восстановленного изображения по сравнению с прочими алгоритмами.

Алгоритм Б дал самые плохие результаты. Независимый поиск блоков на одном уровне разбиения позволяет сократить лишь несколько бит за счет информации о разбиении каждого макроблока, при этом теряется возможность нахождения удачного соответствия на более высоких уровнях разбиения методом квадродерева, что дало бы лучший коэффициент сжатия. Для этого алгоритма невысокие значения имеют качество, коэффициент сжатия и время сжатия.

Алгоритм В эффективнее остальных использует взаимосвязь цветовых плоскостей. При таком подходе найти удачное соответствие блоков сложнее, но вместо трех индексов доменных блоков можно хранить один общий (домены берутся с одного участка в соответствующих цветовых плоскостях). Поскольку коэффициенты доменных индексов занимают около 30 % объема файла сжатого изображения, то алгоритм позволил снизить размер файла при несущественных потерях качества.

Все предложенные и реализованные алгоритмы по соотношению степени сжатия к качеству декодированного изображения уступают алгоритму *JPEG*.

Для исследования возможностей повышения степени фрактального сжатия цветных изображений необходимо исследовать структуру файлов фрактально сжатых изображений. В экспериментах изучалось, какую часть закодированного изображения занимает индексация доменных блоков, биты яркости и контраста, биты коэффициентов преобразований поворота, при этом принималось во внимание число ранговых блоков на каждом уровне разбиения. Результаты в виде структуры файлов фрактально сжатых изображений алгоритмами *А*, *Б*, *В* представлены на рис. 5 (см. третью сторону обложки). Из него видно, что для алгоритмов *А* и *Б* существенную часть фрактального кода занимает индексация доменов, поскольку для блоков каждой компоненты хранится отдельный индекс. Вместе с тем алгоритм *В* за счет использования одной доменной области на все компоненты каждого рангового блока порождает файл фрактального кода

Таблица 1

Результаты работы алгоритмов фрактального сжатия цветных изображений и алгоритма JPEG

Имя алгоритма	Размер исходного изображения, Кбайт	Размер сжатого изображения, Кбайт	Время сжатия, с	Метрики качества	
				SSIM	*PSNR*
Изображение *lenna* (512 × 512)					
А	768	160	13,632	0,974	33,809
Б		161	13,755	0,974	33,809
В		114	5,857	0,970	33,087
JPEG		38	< 0,5	0,986	39,191
Изображение *peppers* (512 × 512)					
А	768	120	10,683	0,962	29,151
Б		148	14,130	0,962	29,135
В		112	5,739	0,957	28,558
JPEG		42	< 0,5	0,989	40,508
Изображение *frymire* (1024 × 1024)					
А	3570	564	275,471	0,955	22,678
Б		772	333,479	0,957	23,393
В		477	195,933	0,955	23,232
JPEG		676	< 0,5	0,999	41,177

Таблица 2

Результаты работы алгоритма *В* при различном числе бит, выделенных под коэффициенты цветности

Контраст цвета, бит	Яркость цвета, бит	Исходный размер, Кбайт	Размер сжатый, Кбайт	Время сжатия, с	Метрики качества	
					SSIM	*PSNR*
Изображение *lenna* (512 × 512)						
6	4	768	106	6,398	0,966	32,542
6	3		101	7,368	0,957	31,576
5	4		101	6,498	0,966	32,480
5	3		97	7,306	0,958	31,575
Изображение *peppers* (512 × 512)						
6	4	768	105	6,529	0,952	28,567
6	3		102	7,685	0,944	28,167
5	4		101	6,513	0,953	28,451
5	3		97	7,855	0,944	28,108
Изображение *frymire* (1024 × 1024)						
6	4	3570	446	221,416	0,951	23,145
6	3		432	239,343	0,942	22,720
5	4		428	220,808	0,952	23,198
5	3		413	240,020	0,942	22,782

с меньшим процентным соотношением битов индексации доменов.

Поскольку по времени сжатия и степени сжатия алгоритм *B* превосходит два других при не столь существенных потерях качества, то его будем модифицировать и исследовать далее. Действительно, повысить степень сжатия изображения с помощью алгоритма *B* можно за счет выделения меньшего числа бит под коэффициенты яркости и контраста на каналы цветности. Результаты таких исследований приведены в табл. 2. Видно, что снижение числа бит под коэффициенты контраста и яркости для компонент цветности уменьшило размер файла, однако не столь значительно, причем с привнесением дополнительных потерь в качестве изображения. Рациональным можно считать выделение 5 бит под контраст и 4 бит под яркость, так как при примерно одинаковой степени сжатия потери в декодированных изображениях заметно меньше.

Заключение

Предложены и программно реализованы три быстродействующих алгоритма фрактального сжатия цветных изображений. В результате проведенных исследований показаны преимущества алгоритма *B* (основной поиск ведется по яркостной компоненте) по степени сжатия изображений и по потерям в декодированных изображениях в сравнении с другими предложенными алгоритмами. Для этого алгоритма изучено распределение бит в структуре фрактально сжатых файлов, на основе чего предложен подход к увеличению степени сжатия изображений с помощью алгоритма *B*.

В качестве направлений дальнейших исследований можно изучать эффективность алгоритма *B* при расширении доменного пула блоками из соседних цветовых компонент. На наш взгляд, для дальнейшего повышения степени фрактального сжатия цветных изображений следует вести исследования, нацеленные на использование методов арифметического кодирования коэффициентов преобразований.

Список литературы

1. **JPEG** File Interchange Format. Version 1.02. 1992 [Электронный ресурс]. URL: www.w3.org/Graphics/JPEG/jfif3.pdf (дата обращения 22.06.2011).
2. **ISO/IEC 13818.** Information technology. Generic coding of moving pictures and associated audio information. 1998.
3. **ITU-T H.264:** Advanced video coding for generic audiovisual services. 2010 [Электронный ресурс]. URL: http://www.itu.int/rec/T-REC-H.264-201003-I/en (дата обращения 12.07.2011).
4. **Bross B.** et al. High-Efficiency Video Coding (HEVC) text specification draft 8 // JCT-VC, Stockholm, SE, 11—20 July 2012.
5. **Шарабайко М. П., Осокин А. Н.** Быстродействующий алгоритм фрактального сжатия изображений // Известия Томского политехнического университета. 2011. Т. 318, № 5. С. 52—57.
6. **Recommended** 8-Bit YUV Formats for Video Rendering [Электронный ресурс]. URL: http://msdn.microsoft.com/en-us/library/windows/desktop/dd206750 %28v = vs.85 %29.aspx (дата обращения 07.01.2012).
7. **Converting** Between YUV and RGB. URL: http://msdn.microsoft.com/en-us/library/ms893078.aspx (дата обращения 07.01.2012).
8. **Сидоров Д. В.** Программный продукт imq оценки качества изображений / Оценка качества изображений. 2011. URL: http://imq.vt.tpu.ru/ (дата обращения 20.02.2011).
9. **Test_ image** repository / Fractal coding and analisys group. URL: http://links.uwaterloo.ca/Repository.html (дата обращения 20.02.2011).
10. **Fisher Y.** Fractal Image Compression — Theory and Application. — N. Y.: Springer-Verlag, 1994. 341 p.

УДК 004.93

В. К. Гулаков, канд. техн. наук., проф.,
С. Н. Огурцов, аспирант,
А. О. Трубаков, канд. техн. наук., доц.,
Брянский государственный
технический университет,
e-mail: Gulakov@tu_bryansk.ru

Сегментация пейзажных изображений

Рассмотрен вопрос сегментации изображений, дан анализ популярных алгоритмов и предложен модифицированный алгоритм для решения задачи сегментации пейзажных изображений. Также представлены результаты сравнительного анализа эффективности популярных алгоритмов и алгоритма, предложенного в статье.

Ключевые слова: *сегментация, алгоритмы сегментации, алгоритм водораздела, пирамидальная сегментация, контурная сегментация, CBIR*

Введение

Объем графической информации в сети Интернет очень велик и продолжает постоянно расти, поэтому обработка изображений и поиск отдельных объектов на них является актуальным направлением. Сегментация изображения является одним из основных этапов процесса обработки, и от того насколько качественно будет проделано выделение однородных объектов, зависит эффективность работы всей системы в целом. Известен ряд алгоритмов, которые отличаются по скорости выполнения и по качеству решения [1]. Однако идеального алгоритма для всех применений не существует, поэтому необходимо продолжать исследование особенностей разных типов изображений и разрабатывать новые методы и алгоритмы. В данной работе предложен новый алгоритм сегментации пейзажных изображений, имеющих свои особенности, а также проведено сравнение с наиболее распространенными

алгоритмами, являющимися представителями различных подходов к сегментации.

В первой части статьи дано общее понятие процесса сегментации и проведен анализ требований к разрабатываемому алгоритму. Также в этой части рассмотрены основные алгоритмы, существующие на сегодняшний день. Во второй части описан предлагаемый алгоритм и особенности его реализации. Приведены и обоснованы основные принципы его работы. Результаты тестирования и сравнения алгоритмов даны в третьей части статьи.

1. Сегментация изображений

Сегментация изображения — это процесс разделения изображения на множество непересекающихся областей, объединение которых образует полный набор исходных точек. Существует большое число различных алгоритмов сегментации [2]. В каждом из них к регионам, полученным после этого этапа обработки, предъявляется набор различных требований. Наиболее важными из них являются:

- полученные сегменты должны быть однородны относительно определенных характеристик;
- внутренние части сегментов по возможности должны быть простыми без большого количества внутренних пустот;
- смежные сегменты должны существенно отличаться по значениям выбранных характеристик.

Разнообразие современных алгоритмов можно разделить на ряд групп относительно принципа, на котором они основаны:

- сегментация на графах;
- пирамидальная сегментация (например на основе гауссовой пирамиды);
- контурная сегментация (на основе поиска точек перепада яркости);
- сегментация выращиванием областей (например сегментация алгоритмом водораздела).

Предлагаемый алгоритм объединяет идеи нескольких групп, поэтому для начала рассмотрим особенности и принципы работы каждой из них.

Пирамидальная сегментация [3]. В основе пирамидальной сегментации лежит кратномасштабная обработка (например построение пирамиды Гаусса). Задачей сегментации является преобразование исходного изображения в первоначальный набор кластеров небольшого размера, в котором каждый сформированный кластер характеризуется как собственными параметрами, так и параметрами связи с соседними кластерами.

Пирамидальный алгоритм обработки изображения требует задания способа вычисления уменьшенного изображения уровня $n + 1$ на основе имеющегося изображения уровня n, и применения этой процедуры рекурсивно до предельного уменьшения размера изображения. Часто для этого выбирают элементарный домен некоторой формы, позволяющей плотно покрыть всю площадь изображения, и задают способ нахождения значения элемента следующего уровня по значениям элементов домена предыдущего уровня. Рекурсивное применение данной процедуры позволяет построить дерево, в котором каждый элемент изображения некоторого уровня (кроме самого нижнего) является узлом, связанным с элементами домена предыдущего уровня, а число нисходящих связей определяется формой и размерами выбранного домена.

В случае, когда в качестве домена выбран квадрат размерами 2×2 элемента, получается квадродерево. Алгоритм пирамидальной сегментации осуществляет прямой (вверх) и обратный (вниз) проходы по квадродереву. При прямом проходе вверх по квадродереву происходит рекурсивный анализ всех уровней пирамиды, начиная с самого нижнего (исходного изображения) и заканчивая верхним уровнем, состоящим из одного узла; одновременно с этим строится само квадродерево. На каждом шаге на основе анализа четырех нижних узлов уровня n создается узел уровня $n + 1$, и в соответствующей структуре нового узла запоминается информация об узлах предыдущего уровня, соединенных с данным узлом, их средней яркости, наличии контуров. Таким образом, каждый узел является вершиной некоторого квадродерева, охватывающего расположенные под ним элементы изображения, и содержит информацию о поддеревьях предыдущего уровня.

Основной задачей пирамидального этапа сегментации является объединение соседних элементов, имеющих близкие признаки и не разделенных контуром. Эта процедура требует прослеживания контурных линий на всех уровнях пирамиды. Считается, что два вертикально или горизонтально соседствующих элемента разделены контуром в том случае, когда расстояние между их отображениями в пространстве признаков превышает некоторый заданный порог. Для второго и последующих уровней процедура обнаружения контуров также учитывает наличие контура на предыдущем уровне.

Сегментации на графах. Общая идея методов этой группы заключается в представлении изображения в виде взвешенного графа. Вершинами графа являются точки изображения, а вес ребра графа отражает сходство точек в некотором смысле (например, цветовое). Разбиение изображения моделируется разрезами графа.

Обычно в методах теории графов вводится функционал "стоимость" разреза, отражающий качество полученной сегментации. Так, задача разбиения изображения на однородные области сводится к оптимизационной задаче поиска разреза минимальной стоимости. Такой подход позволяет помимо однородности цвета и текстуры сегментов управлять также формой сегментов, их размером, сложностью границ и т. п.

Для поиска разреза минимальной стоимости применяют различные методы: жадные алгоритмы (на каждом шаге выбирается такое ребро, чтобы суммарная стоимость разреза была минимальной); методы динамического программирования (гарантируется, что, выбирая на каждом шаге оптимальное ребро, получим в итоге оптимальный путь); алгоритм Дейкстры и др.

Контурная сегментация. Методы контурной сегментации основаны на принципе поиска резких перепадов уровня яркости на изображении. Наиболее простым и эффективным методом поиска точек перепада яркости является обработка изображения скользящей маской. При этом маска составляется на основе дискретных аналогов первой или второй производной. Наиболее популярными аналогами являются операторы Робертса, Превитта и Собеля.

Вторым этапом методов контурной сегментации является выделение контуров. Для этого используются различные алгоритмы связывания точек и определения замкнутых контуров.

Методы этой группы хорошо подходят для выделения объектов с простой структурой. При этом трудности возникают тогда, когда изображение содержит множество перекрывающихся объектов, и точки перепадов образуют ветвистые структуры.

Сегментация с помощью выращивания областей. Выращивание областей — это процедура, которая группирует отдельные пиксели или подобласти в более крупные области по заранее заданным критериям. На первом шаге алгоритм рассматривает исходное множество точек, являющихся "центрами кристаллизации". По мере работы алгоритма на эти центры наращиваются области путем присоединения к каждому центру тех элементов из числа соседей, которые по своим свойствам близки к центру кристаллизации. Одним из популярных алгоритмов наращивания областей является алгоритм водораздела.

Понятие водораздела основано на представлении изображения как трехмерной поверхности (ландшафта), с двумя пространственными координатами и уровнем яркости. В такой интерпретации рассматриваются точки трех видов: 1) точки локального минимума (бассейн); 2) точки, находящиеся на склоне, т. е. с которых вода стекает в один локальный минимум; 3) точки, находящиеся на пике, с которых вода стекает более чем в один локальный минимум (водораздел). Главная цель алгоритма — нахождение линий водораздела, т. е. линий, которые разделяют разные по характеристикам области.

Алгоритм основан на абстракции поступления воды в точки локального минимума (бассейны). Когда вода в двух соседних бассейнах будет близка к тому, чтобы слиться, между ними ставится перегородка. В конечном итоге, по мере наполнения водой бассейнов останутся видны только перегородки, которые и будут соответствовать контурам объектов.

Описанные выше алгоритмы имеют как сильные, так и слабые стороны. При этом ни один из них не дает результатов, полностью совпадающих с сегментацией изображения человеком. В данной работе была сделана попытка улучшить свойства этих алгоритмов с помощью объединения ряда принципов из разных групп и внесения различных модификаций. Как показано в третьей части статьи, подобные изменения позволили несколько улучшить показатели результата сегментации.

2. Разработанный алгоритм

В данной работе рассматривается алгоритм сегментации для систем поиска изображений по содержанию в больших дизайнерских коллекциях. Большую часть подобных коллекций составляют пейзажи и фотографии природы. При сегментации пейзажных изображений требуется учитывать ряд дополнительных достаточно важных особенностей. Одной из них является то, что допускается некоторая погрешность в определении границ. Более принципиальным является правильное определение основных элементов сцены с точки зрения человека и описание их с помощью простых контуров. Иными словами алгоритм должен проводить сегментацию изображения, наиболее близкую к сегментации, проведенной человеком. Именно этот критерий ставился во главу всей разработки. Это связано с тем, что данный алгоритм сегментации планируется использовать в графических информационно-поисковых системах.

Однако сегментация изображения человеком — субъективный процесс, и добиться такого же результата в автоматическом режиме очень сложно по нескольким причинам:

- разные люди выделяют разное число значимых объектов, как следствие получается разное число сегментов;
- контуры выделенных сегментов для двух разных людей могут значительно отличаться;
- современные технологии далеки от создания настоящего искусственного интеллекта и полного понимания основных критериев, которыми руководствуется человек при анализе изображения.

За основу была взята гипотеза о том, что человек при визуальной оценке изображения, в первую очередь, учитывает области, которые сильнее всего отличаются от других по некоторой характеристике (например, яркости, цвету или текстуре). Для выделения подобных областей хорошо подходят алгоритмы слияния/деления. Однако для подобных алгоритмов необходимо решить вопрос о пороге слияния и числе выделяемых сегментов. Проведенный анализ результатов ручной сегментации пейзажных изображений показал:

- число выделяемых сегментов должно быть порядка 10 (в большинстве случаев при ручной сегментации выделяется до 10 значимых объектов);
- число сегментов может быть менее 10, если "расстояние" между сегментами меньше некоторого порогового значения.

Данные положения были взяты как дополнительные критерии при выборе параметров разрабатываемого алгоритма.

Остановимся более подробно на разрабатываемом алгоритме. В нем учтены и объединены лучшие качества уже существующих методов. Рассмотрим эти положения более подробно.

Сегментация слиянием. Основой алгоритма является процесс слияния. Сегментация слиянием очень хорошо зарекомендовала себя в решении подобных задач. Особой популярностью при этом пользуется сегментация, основанная на представлении элементов в виде узлов графа и итерационном слиянии этих узлов. Поэтому было решено использовать именно этот метод, добавив в него ряд модификаций и дополнений.

Уменьшение первоначального графа. Алгоритм сегментации на графах требует большого объема памяти и имеет большую вычислительную сложность. В классическом варианте в узел графа на первом этапе попадает одна точка изображения, и затем между этими узлами вычисляется расстояние. Однако если в качестве узлов графа выбрать изначально сформированные группы точек (области), то это позволит многократно ускорить работу алгоритма и снизить требования к расходу памяти за счет значительного уменьшения размера графа и числа итераций слияния.

Принцип начального формирования. В качестве критерия объединения пикселей изображения и начального формирования областей (вершин графа) было решено использовать критерий цвета (в одну область объединялись все соседние точки, имеющие небольшое отклонение в цвете). Даже такая простая операция, как формирование однородных с точки зрения цвета областей, позволяет значительно уменьшить размер исходного графа.

Уменьшение размера изображения. Изображения пейзажей, ландшафтов и пользовательские фотографии отличаются большим исходным размером. Это, в первую очередь, связано с выросшим в последнее время качеством цифровых фотоаппаратов. С увеличением качества изображения растет число точек на каждый объект, что существенно влияет на объем памяти и число операций при обработке, при этом мало сказывается на качестве результата сегментации.

Еще одной проблемой, связанной с размером изображения, является наличие в нем мелких, несущественных в рамках данной задачи объектов, помех и искажений регистрирующих приборов. Эти элементы негативно влияют на процесс сегментации, так как часто имеют очень большие отклонения по однородности.

Для решения двух этих проблем было решено использовать предварительную фильтрацию и кратномасштабную фильтрацию, которая используется в пирамидальной сегментации. На первом этапе для исходного изображения строят пирамиду Гаусса. В качестве изображения для сегментации на графах выбирают некоторое изображение этой пирамиды меньшего размера. Это позволяет избавиться и от помех, и от незначительных мелких объектов, выделение которых является нежелательным. В дальнейшем, на последнем шаге алгоритма, происходит обратный проход по пирамиде и уточнение границ объектов.

Фильтрация. Еще одним этапом алгоритма, нацеленным на уменьшение числа помех и мелких незначащих объектов, является фильтрация. В качестве метода фильтрации для экспериментов использовался медианный фильтр. При этом в результате применения фильтрации удается еще значительнее сократить размер первоначального графа и повысить эффективность алгоритма в целом.

Цветовая палитра. В качестве меры цветового различия в алгоритме используется зарекомендовавшая себя мера CIEDE (*International Commission on Illumination delta E*). Однако расчет этой меры для обработки полноцветного изображения сопряжен с большими вычислительными затратами. Поэтому перед выполнением всех операций изображение переводят в палитру цветов меньшего размера, между всеми цветами которой изначально рассчитаны расстояния. Анализ показал, что подобный прием не сказывается значительно на результатах работы алгоритма по нескольким причинам: во-первых, человек не выделяет при ручной сегментации близкие по оттенку цвета расположенных рядом точек, во-вторых, близкие по цвету точки в процессе слияния все равно объединяются в одну область.

Веса дуг графа. Самым важным этапом разработанного процесса сегментации является слияние вершин графа. Именно этот этап сильнее всего влияет на результат работы.

Проведенные эксперименты показали, что ориентация при слиянии на какой-либо один показатель (например ориентация на средний цвет области) не позволяет добиться приемлемого качества. Поэтому предлагаемый алгоритм основан на комплексной оценке, учитывающей следующие критерии:

- средний цвет сформированных на предыдущем шаге областей (данный критерий является основным), он позволяет предотвратить слияние сильно отличающихся областей в случае плавного перехода границы между ними;
- цвет на границе раздела областей (учет цвета на границе позволяет более точно проводить деление на сегменты и учитывать "размазанные" края и небольшие тени объектов);

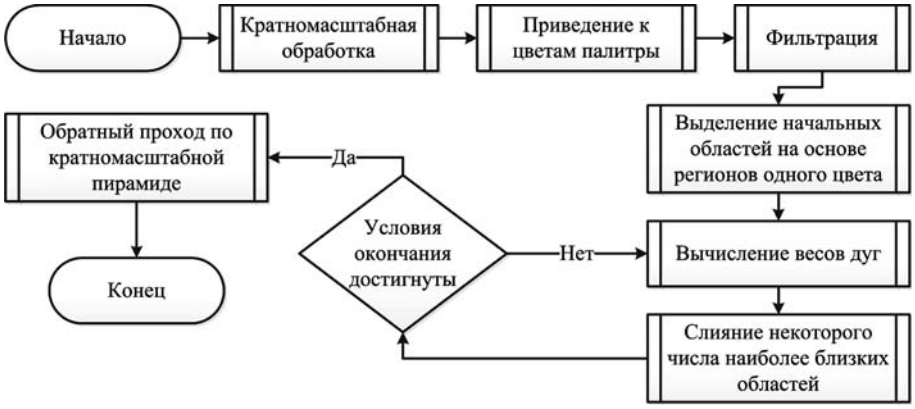


Схема алгоритма сегментации

- размер выделенной области (данный критерий позволяет избавиться от областей малого размера, чаще всего соответствующих дефектам приборов или каким-либо другим отклонениям).

Однако не только критерии слияния, но и "важность" этих критериев должны влиять на процедуру слияния. В результате проведенных экспериментов была сформирована следующая формула вычисления веса дуги графа обрабатываемого изображения:

$$V = (K_1 * ColorMed + K_2 * ColorBound) * \\ * (-\mathbf{e}^{-K_3 * MinRegionSize} + 1),$$

где V — вес дуги графа между двумя узлами; K_1 — коэффициент влияния среднего цвета области, *ColorMed* — расстояние между средними цветами областей; K_2 — коэффициент влияния цвета в точках соприкосновения на границе; *ColorBound* — расстояние между цветами точек на границе; K_3 — коэффициент влияния размера областей; *MinRegionSize* — размер минимальной из областей (в процентах) по отношению ко всему изображению.

Описанный алгоритм можно представить в виде следующей последовательности шагов (см. рисунок).

3. Оценка качества

Очень важным является вопрос выбора экспериментальной коллекции для проведения сравнения предлагаемого в статье алгоритма и уже известных методов. От выбора такой коллекции во многом зависит доверие к полученным результатам. Было решено использовать базу сегментированных изображений университета Беркли (*The Berkeley Segmentation Dataset and Benchmark*) [4]. Эта одна из лучших баз изображений, специально разработанных для тестирования алгоритмов сегментации.

База изображений университета Беркли содержит большое количество изображений, сегментация которых проведена вручную. При этом для уменьшения субъективной составляющей каждое изображение сегментировали в среднем шесть человек. Это позволило более адекватно учитывать особенности восприятия и ориентироваться на среднестатистического человека. При проведении экспериментов учитывали результаты всех ручных сегментаций, а для вычисления общей оценки рассчитывали среднее значение отклонения.

В качестве алгоритмов, с которыми сравнивали предлагаемое решение, были взяты готовые реализации наиболее популярных методов: пирамидальной сегментации, контурной сегментации (использовали сочетание алгоритмов), алгоритма водораздела (в качестве начальных приближений использовали объекты контурной сегментации). Для каждого алгоритма было разработано приложение под ОС *Windows* на языке программирования *C#* с использованием *.NET Framework 3*. Разрабатываемый алгоритм также представлен в виде аналогичного приложения.

С помощью каждого приложения была проведена сегментация всех изображений тестовой коллекции. В результате такой сегментации на выходе получилось четыре набора, которые и сравнивали с результатами ручной сегментации, полученными в университете Беркли. Для проведения сравнения также было разработано специальное приложение.

Исходя из требований к алгоритму, описанных выше, итоговое сравнение проводили по следующей методике. Для каждого сегмента $S_{j,k}^{(alg)}$, полученного после автоматической сегментации исследуемым алгоритмом изображения I_j, из коллекции изображений университета Беркли $\{I\}$ выбирается некоторый сегмент $S_{j,l}^{(h_n)}$ того же изображения, но полученный с помощью ручной сегментации n-м человеком, имеющий наибольшее пересечение с $S_{j,k}^{(alg)}$:

$$\forall S_{j,m}^{(h_n)} \in I_j : |S_{j,k}^{(alg)} \cup S_{j,l}^{(h_n)}| \geqslant |S_{j,k}^{(alg)} \cup S_{j,m}^{(h_n)}|.$$

Нормированное несовпадение двух сегментов, выбранных таким способом, принимается за коэффициент отклонения от идеальности (идеальным в данном контексте понимается сегментация человеком):

$$K_{j,k}^{(h_n)} = \frac{\left|(S_{j,k}^{(alg)} \cup S_{j,l}^{(h_n)}) \setminus (S_{j,k}^{(alg)} \cap S_{j,l}^{(h_n)})\right|}{\left|S_{j,k}^{(alg)}\right|}.$$

Общее отклонение результата автоматической и ручной сегментаций для j-го изображения рассчитывают как среднее отклонение по всем найденным сегментам для всех имеющихся в базе ручных вариантов сегментации:

$$\overline{K}_J = \frac{1}{N} \sum_{n=1}^{N} \left(\frac{1}{M} \sum_{m=1}^{M} K_{j,m}^{(h_n)} \right),$$

где M — число сегментов, выделенных алгоритмом в j-м изображении коллекции; N — число вариантов ручной сегментации из базы изображений Беркли.

Общий коэффициент точности алгоритма вычисляется, как среднее отклонение для изображений всей коллекции:

$$T = \frac{1}{J} \sum_{j=1}^{J} \overline{K}_J,$$

где J — число изображений в базе.

После проведенных сравнений были получены следующие результаты, представленные в таблице.

Результаты экспериментальной проверки

Алгоритм	Коэффициент отдаленности
Разработанный алгоритм	26,687
Алгоритм водораздела	30,224
Пирамидальная сегментация	34,880
Контурная сегментация	55,172

Заключение

Сегментация изображений является одним из важных этапов в задачах поиска, распознавания и анализа изображений. При этом качество существующих на сегодняшний день алгоритмов еще уступает ручной сегментации человеком.

В статье проведен анализ существующих алгоритмов и рассчитано качество результатов сегментирования по сравнению с сегментацией человеком. При этом предложен новый алгоритм, объединяющий сильные стороны существующих на сегодняшний день методов. Как показали исследования, этот алгоритм несколько превосходит по качеству сегментации популярные и часто используемые методы.

Список литературы

1. **Конушев В., Вежнивец В.** Методы сегментации изображений: интерактивная сегментация. // Компьютерная графика и мультимедиа. 2007. Вып. № 5 (1). URL:http://cgm.computergraphics.ru/content/view/172

2. **Поршнев С. В., Левашкина А. О.** Универсальная классификация алгоритмов сегментации изображений. Изд. Радиотехнического института Уральского Государственного Технического Университета-УПИ, 2008.

3. **Marfil R., Molina-Tanco L., Bandera A., Rodryguez J. A., Sandoval F.** Pyramid segmentation algorithms revisited. Dpto. Tecnologia Electrynica, E. T. S. I. Telecomunicaciyn, Universidad de Malaga, 2006.

4. **The Berkeley** Segmentation Dataset and Benchmark. URL: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench.

Информация

С 1 по 7 сентября 2013 г. в Калининграде состоится

37-я *конференция — школа* молодых ученых и специалистов

"ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ — 2013" (ИТиС'13)

Конференция-школа ИТиС'13 будет своеобразной Universitas, состоящей из тематических семинаров по следующим основным *научным направлениям*:

- Seminarium mathematicum, или Математика и физика сложных систем
- Seminarium explorationis datorum, или Структурные методы анализа данных и оптимизации
- Seminarium protectionis informatiae, или Теория кодирования и ее приложения;
- Seminarium technologiarum retium, или Сетевые технологии и протоколы
- Seminarium operis informationis in vivo, или Биология и биоинформатика
- Seminarium linguisticum, или Компьютерная лингвистика

Кроме seminaria, впервые на конференции-школе состоятся и учебные курсы (doctrinae) по "горячим" областям исследований.

Электронный адрес организационного комитета — itas@iitp.ru,
Сайт конференции — http://itas2013.iitp.ru/rss.xml

УДК 004.383.4

Н. С. Поливанов, студент,
e-mail: nikolay.s.polivanov@intel.com,
Г. С. Речистов, мл. науч. сотр.,
А. А. Абдухаликов, студент,
В. М. Пентковский, д-р техн. наук,
рук. лаборатории,
Московский физико-технический институт,
ЗАО "Интел А/О", Москва

Реализация инструментария для исследования сетевой производительности MPI-приложений на распределенном симуляторе

Демонстрируется использование распределенного симулятора Simics для сборки трасс вызовов процедур MPI и последующего их анализа. Описаны процесс постановки эксперимента и первые результаты анализа полученных трасс для реализации бенчмарка Linpack HPL.

Ключевые слова: *распределенная симуляция, многоядерные системы, трассировка, производительность сети, MPI, Simics, кластер, Linpack*

Введение

При разработке новых распределенных вычислительных машин возникает необходимость оценки производительности тех или иных приложений в зависимости от конфигурации этих машин (конфигурации отдельных узлов, конфигурации сети). Распределенные приложения, как правило, используют библиотеки MPI для обмена данными между узлами.

В качестве метрики производительности приложения используется величина *CPI* (англ. *cycles per instruction*), она равна среднему числу циклов, необходимых для исполнения одной инструкции процессора. Эта величина складывается из нескольких факторов [1, 2]:

$$CPI = CPI_{core} + CPI_{caches} + CPI_{memory} + CPI_{MPI},$$

где CPI_{core} — число циклов, затрачиваемых на исполнение непосредственно в ядре процессора; CPI_{caches} и CPI_{memory} — задержки при обращении в кэш-память и оперативную память соответственно; CPI_{MPI} — коммуникации с помощью MPI.

Для анализа величины CPI_{MPI} был разработан сборщик трасс для функционального симулятора Simics. Он собирает вызовы MPI-функций с метками времени, что позволяет в дальнейшем проанализировать производительность приложения в целом в зависимости от производительности и топологии сети.

Разработанный нами инструмент анализа имеет следующие возможности:

- собранные трассы позволяют проанализировать эффективность использования протокола MPI приложениями на кластере;
- используя трассы, можно проводить дальнейший анализ производительности приложения для модифицированных конфигураций сети без дополнительных запусков изучаемого приложения, таким образом, экономя время;
- возможен запуск приложения на конфигурациях оборудования с числом узлов, превышающим число имеющихся в наличии в несколько раз, позволяет изучать еще не созданные системы;
- симуляция оказывает минимальное возмущение на исполнение изучаемой программы, так как сбор и запись событий вносит минимальную и предсказуемую задержку в полное виртуальное время работы приложения — несколько инструкций на каждый вызов.

Обзор литературы

Для исследования MPI-приложений в литературе описано несколько сборщиков трасс, а также симуляторов. В [3] описывается подход к симуляции сети с возможностью настройки задержек при передаче сообщений между узлами. С помощью профилирующего интерфейса MPI и дополнительного узла-симулятора моделируется виртуальное время процесса (получаемое процессом через **MPI_Wtime**). Управление виртуальным временем позволяет минимизировать его возмущения из-за влияния симуляции. Скорость работы модели сравнима с наблюдаемой на реальном оборудовании. Однако при этом числе симулируемых узлов ограничено числом реально доступных узлов. В нашем решении размер моделируемой системы превышал физическую до 16 раз.

В работе [4] реализована специальная библиотека MPI, симулирующая процедуры передачи данных. Изучаемое MPI-приложение преобразуется в многопоточное (с общей памятью потоков) в целях экономии ресурсов системы при коммуникациях.

Это требует дополнительных усилий при его компиляции (обработка глобальных переменных как локальных, неявное порождение потоков и т. п.). Адаптация программ, написанных на языках программирования, отличных от Си, потребует дополнительной модификации компилятора. Возможны проблемы с корректностью работы приложения, если используемые сторонние библиотеки не поддерживают многопоточность. Наш подход не требует изменения сопутствующих инструментов или исходного кода приложения, за исключением стадии компоновки.

Третий способ описан в [5]. В работе проводится моделирование как сети, так и подсистемы ввода-вывода (MPI IO). Есть возможности работать с распределенной файловой системой. Кроме того, предлагается способ оценки энергопотребления по полученным заранее трассам на существующем оборудовании. Моделируется только одно пользовательское приложение, отсутствие полной симуляции узлов также ограничивает симуляцию различных конфигураций.

Использование симулятора аппаратных платформ позволяет учесть влияние операционной системы, фоновых процессов и особенностей аппаратуры моделируемой системы, тогда как модели, работающие только с одним приложением, пренебрегают этими эффектами, тем самым снижая достоверность результатов.

Изучаемое приложение и инструменты исследования

В качестве первого исследуемого приложения первоначально был взят тест (бенчмарк) High Performance Linpack [6] (сокращенно **HPL**). Он компилировался с использованием библиотекой MPICH2 [7] версии 1.4, реализующей стандарт MPI2. Данное приложение является стандартным тестом производительности вычислительных машин на операциях с плавающей запятой и представляет собой реализацию алгоритма решения системы линейных уравнений с плотной матрицей. Операции линейной алгебры осуществляются с помощью библиотеки BLAS (реализация ATLAS 3.97).

Следующее изучаемое приложение — **mdrun** из состава пакета молекулярной динамики — Gromacs [8]. В данной работе оно было собрано компилятором GCC 4.4.5 с двойной точностью вычислений.

Все приложения были собраны и запускались под управлением 64-битной операционной системой GNU/Linux Debian 6 "Wheezy".

Для построения модели был взят Wind River Simics — функциональный симулятор аппаратных платформ ЭВМ. Он поддерживает симуляцию широкого спектра систем, в том числе многомашинных конфигураций.

Особенностями Simics являются:

- детерминированность симуляции, даже в случае распределенных систем;
- возможность сохранения контрольных точек для восстановления состояния системы без полного перезапуска симуляции:
- стабильный и хорошо документированный программный интерфейс для написания новых моделей, а также обширная библиотека готовых компонентов;
- наличие интеграции со скриптовым языком Python для автоматизации процессов исследования.

Симуляция распределенных систем в Simics описано в документации [9, 10].

Кроме того, Simics позволяет создавать и привязывать собственные подпрограммы-обработчики, называемые HAP, для определенных событий в моделях, например, исключения процессора, смены его режима работы, вывод строки на экран и т. п. Среди таких событий есть исполнение "магической" инструкции (англ. *magic instruction*), которая встраивается в исследуемое приложение для отслеживания процесса исполнения.

Данный подход применим для других симуляторов; реализации его присутствуют в Qemu [11] и SoftSDV [12]. Для этого необходима возможность прерывать исполнение по "магической инструкции" и получать данные из памяти виртуальной системы. Описываемая ниже профилирующая библиотека может быть перенесена на другие симуляторы и архитектуры с минимальными изменениями, для этого достаточно выбрать подходящую "магическую" инструкцию и обеспечить перехват ее исполнения. Реализация внешнего модуля трассировщика более привязана к предоставляемому программному интерфейсу конкретного симулятора, но решение является достаточно общим и тоже может быть перенесено на другие системы.

Трассировщик

Сборщик трасс состоит из профилирующей библиотеки, подключаемой к исследуемой программе, и модуля для симулятора. Профилирующая библиотека переопределяет процедуры MPI, тем самым перехватывая обращения к ним. Модуль обрабатывает поступающие события, вызванные "магической" инструкцией, собирает и записывает трассы.

Процесс использования реализованного трассировщика выглядит следующим образом:

1. По заголовочным файлам библиотеки MPI программа на Python генерирует файл с описанием всех найденных прототипов MPI-функций и исходный код профилирующей библиотеки на языке Си, который затем преобразуется в объектный файл с помощью компилятора GCC.

2. Исследуемое приложение компонуется с полученной на предыдущем шаге профилирующей библиотекой.

3. Нужно загрузить модуль **mpi-tracker** в симулятор Simics и затем зарегистрировать с помощью команд **`g-register-mpi-tracker`** или **`g-register-mpi-tracker-delayed`**. Первая команда немедленно регистрирует и включает трассировщик во всех процессах Simics. Вторая делает это с заданной задержкой во времени, а также отключает трассировщик через заданное время. Это позволяет автоматизировать сбор данных только для интересующих нас участков процесса работы приложения. Обе команды принимают файл с описанием MPI-процедур, полученный на 2-м этапе.

4. После окончания работы следует отключить трассировщик с помощью **`g-unregister-mpi-tracker`**. Это нужно для корректного и полного сброса данных всех собранных на диск.

5. Для каждого запущенного у участвующего в симуляции процесса Simics создается бинарный файл с трассами в директории запуска модели. Для их последующего чтения написана небольшая библиотека на Python (используемая и в трассировщике). Нами созданы различные обработчики, извлекающие информацию из этих файлов: простое преобразование трасс в текстовый вид; анализ частот возникновения MPI-событий; простой симулятор для проигрывания истории взаимодействия процессов при работе MPI-программы.

Рабочий процесс сбора трасс выглядит следующим образом (рис. 1):

1. Приложение вызывает процедуру MPI, которая переопределена в профилирующей библиотеке.

2. Профилирующая библиотека вызывает настоящую процедуру MPI с теми же аргументами, получает возвращаемое значение.

3. Профилирующая библиотека исполняет "магическую" инструкцию, при ее выполнении Simics создаст событие (HAP).

4. Исполнение программы приостанавливается в месте выполнения "магической инструкции", и вызывается обработчик события, находящийся в модуле трассировщика.

5. Обработчик в модуле собирает и записывает в файл:

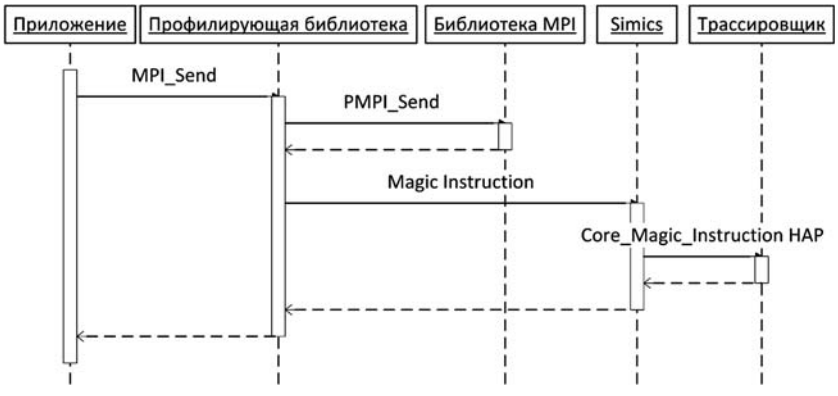


Рис. 1. Общая схема работы сборщика трасс при симуляции MPI-приложения

a. Аргументы MPI-процедуры.

b. Полученное возвращаемое значение из процедуры MPI.

c. Номер (англ. *rank*) MPI-процесса в коммуникаторе MPI_COMM_WORLD.

d. Текущее виртуальное время (число циклов текущего процессора).

e. Текущий номер узла и ядра процессора.

6. Исполнение передается обратно в профилирующую библиотеку, а затем в приложение.

Поскольку сбор и запись трассы происходит в симуляторе, а не в гостевой системе, то это время работы не учитывается в виртуальном времени системы, а значит, и в полученных трассах. Вся активность проходит незаметно для исполняемой программы, для которой она представлена выполнением одной обычной инструкции.

Профилирующая библиотека

Данная библиотека переопределяет процедуры MPI, что позволяет перехватывать управление от приложения. Программа компонуется с профилирующей библиотекой, для этого компоновщик должен поддерживать слабые ссылки. Объявления процедур MPI в библиотеке должны быть слабыми (в таком случае их можно переопределить на этапе компоновки). Чтобы вызывать процедуры MPI из профилирующей библиотеки, в стандарте MPI предусмотрен профилирующий интерфейс [13].

"Магическая" инструкция — это команда, которая не влияет на исполнение программы в симуляторе, но позволяет вызвать заранее определенный обработчик в нем. В случае Simics используется инструкция **CPUID** для архитектуры x86—64, при исполнении которой генерируется событие **Core_Magic_Instruction**. Эта инструкция позволяет передавать ограниченный объем информации в симулятор. В оригинальном макроопределении, поставляемом с Simics для x86—64, это 2 байт, в нашем исследовании оно было расширено для передачи информации объемом до 18 байт. В данном случае передается порядковый номер процедуры, MPI-rank и указатель на массив с аргументами функции, что позволяет установить возвращаемое значение, число и тип ее аргументов.

В каждой функции профилирующей библиотеки был заведен массив на стеке, в который помещаются возвращаемое значение и аргументы функции, которые потом читаются из модуля трассировщика. Для этого был модифицирован макрос магической инструкции, чтобы передавать MPI-rank и указатель на массив через регистры, задействованные инструкцией **CPUID**.

Для сохранения текущего rank-процесса после инициализации изучаемой

программы в функции **MPI_Init** дополнительно вызывается **PMPI_Comm_rank** для коммуникатора **MPI_COMM_WORLD**, и полученный номер сохраняется в глобальной переменной (номера процессов в других порожденных коммуникаторах не отслеживаются).

Профилирующая библиотека создается автоматически на основе заголовочных файлов MPI, что позволяет быстро сформировать ее заново для изучения иных реализаций MPI.

Модуль трассировщика

По событию (**Core_Magic_Instruction**), вызванному исполнением симулятором "магической инструкции", вызываются его обработчики. Каждому обработчику доступны регистры и адресное пространство процесса, что используется для считывания аргументов функций из заранее подготовленного массива (как было описано выше).

Стоит учесть, что при симуляции двух и более машин в пределах одного процесса Simics для всех регистрируется лишь один обработчик **Corc_Magic_Instruction**. Поскольку симуляция машин происходит в отдельных потоках симулятора, возникает проблема гонки при доступе к общим ресурсам. В данном случае таким ресурсом являлся объект открытого файла, куда записываются результаты. Поэтому была использована взаимная блокировка при записи в файл.

Во время тестовых запусков было замечено большое число (на 3...4 порядка большее, чем других процедур) вызовов **MPI_Iprobe** — неблокирующей проверки наличия сообщения в очереди. Запись каждого такого вызова заметно снижала производительность симуляции из-за дисковых операций; это также значительно увеличивало размер генерируемых файлов. Потому было решено реагировать только на каждый десятитысячный ее вызов.

Постановка эксперимента

Эксперименты по моделированию проводились на вычислительном кластере МФТИ, содержащем 16 узлов, каждый из которых имеет два процессора Intel Xeon E5680 с шестью ядрами; таким образом, полное число ядер равно 192. Распределение вычислительных ресурсов на нем обеспечивается программой SLURM (англ. *Simple Linux Utility for Resource Management*). Она позволяет выделять ресурсы по запросу. Так, команда **"salloc — N 1 — time = 24:00:00 ./myscript.sh"** выделит один узел на 24 ч для выполнения программы **myscript.sh.** Из-за того, что требуется собрать трассы программ для нескольких запусков различных конфигураций, а один эксперимент может выполняться несколько суток, то необходима автоматизация, минимизирующая необходимость вмешательства оператора. Для этого были выполнены следующие шаги:

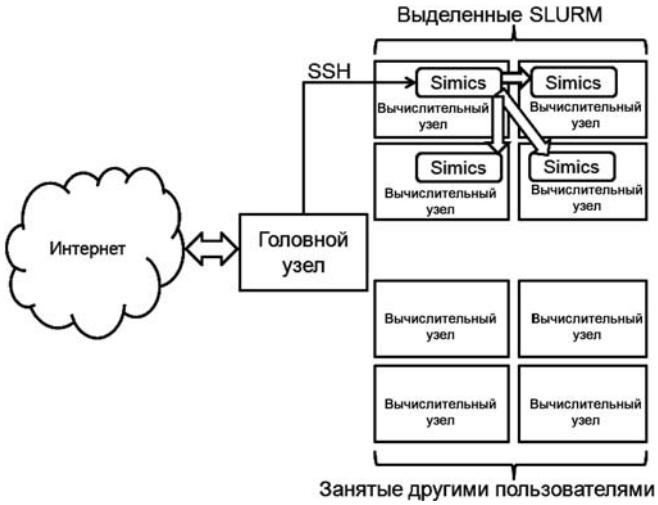


Рис. 2. Схема распределения копий Simics при старте симуляции на узлах, выделенных с помощью SLURM

1. Интеграция Simics со SLURM. Сопутствующие скрипты симулятора были изменены для того, чтобы автоматически определять число и список выделенных SLURM-узлов и распределять задачу по ним (рис. 2).

2. Для проведения серий из большого числа экспериментов с различными конфигурациями для автоматизации процесса сбора информации была создана пакетная задача, ставящая отдельные запуски в очередь SLURM и архивирующая их результаты по завершении.

Simics использует язык Python для описания модели системы, что облегчило интеграцию со SLURM, также имеющему интерфейсы к этому языку.

Для каждой запускаемой копии модели можно было установить различные значения конфигурационных элементов, таких как частота процессора, его тип (модель, число ядер), объем ОЗУ, число моделируемых узлов.

Автоматизирован ввод команд через управляющий терминал (приглашение Bash) модели. Типичный запуск моделируемого приложения требовал ввода команды вида **"mpiexec.hydra — np 256 — f hosts./a.out"** в определенный момент.

Длительность процесса сбора полезных данных о поведении MPI-приложения варьировалось в зависимости от масштаба исследуемой системы. Для небольших моделей (32 ядра на каждом из пяти узлов и 400 симулируемых секундах, в течение которых собирались трассы) она равнялась 4 ч, сама симуляция помещалась на единственной реальной машине. Для больших систем (более 14 узлов) было необходимо выделять суммарно 30...50 Гбайт физической памяти, поэтому эксперимент был распределен на 2...5 узлах. Полное время симуляции при этом достигало 12 ч, замедление модели по отношению к реальной скорости работы приложений находилось в диапазоне от 400 до 600 раз.

Результаты

Для HPL были проведены эксперименты на модели восьми двухъядерных систем, при этом они размещались на двух физических узлах (каждая ЭВМ содержала один процесс Simics, они связывались между собой по сети). Частота вызовов отдельных MPI-функций представлена на рис. 3.

Согласно этим данным, HPL использует только процедуры коммуникаций "точка—точка", среди них больше всего вызовов **MPI_Iprobe** (на 3...4 порядка больше, чем **MPI_Recv/MPI_Send**). Происходит достаточно много вызовов MPI-процедур, не связанных с коммуникациями: **MPI_Type_free**, **MPI_Type_struct**, **MPI_Type_commit**. В настоящее время проводится сбор частот вызовов MPI-функций с помощью сборщика трасс, входящего в пакет [7]. Это позволит сравнить и верифицировать полученные результаты двух различных сборщиков трасс.

После завершения наладки процесса сбора данных все полномасштабные эксперименты проводились для приложения **mdrun**, для которого изучались различные статистические характеристики поведения приложения при разном числе узлов, входящих в симуляцию. На рис. 4 приведена зависимость средней дистанции между MPI-вызовами изучаемого MPI-взаимодействия от числа параллельных потоков, участвующих в работе изучаемого приложения. Это позволит проанализировать величину *CPI*, как было сказано выше и как описано в работах [1, 2], и, таким образом, проанализировать производительность распределенной машины. Максимальный симулируемый кластер в данной серии экспериментов содержал 48 узлов, при этом число выделенных под него физических узлов равнялось пяти, т. е. "плотность" размещения составляла около 10 моделируемых ЭВМ на одну физическую.

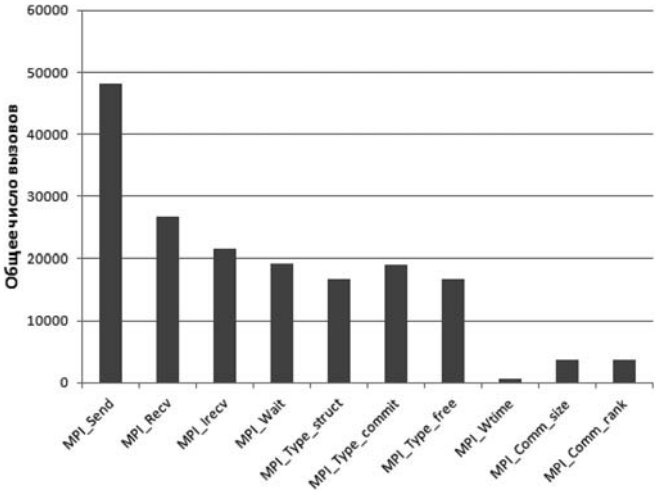


Рис. 3. Распределение частот вызовов MPI-процедур при работе HPL

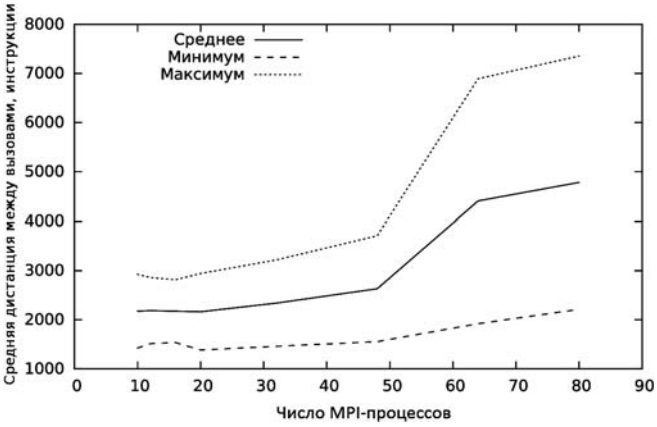


Рис. 4. Зависимость средней дистанции между двумя соседними вызовами MPI-процедур от полного числа MPI-процессов для приложения mdrun. Каждый узел содержит 16 MPI-процессов, полное число узлов варьируется от 10 до 80

Заключение

Задача анализа производительности сети распределенных машин представляет большой практический интерес. В связи с растущими потребностями в вычислительных мощностях важность задачи будет возрастать. В настоящее время способы анализа производительности сети не обладают достаточной гибкостью для анализа еще не созданных распределенных систем. Трассирование процедур с использованием симуляторов платформ позволяет анализировать сетевую производительность произвольных конфигураций распределенных систем.

Список литературы

1. **Simonson L. J., He L.** Micro-architecture Performance Estimation by Formula // Proceedings of SAMOS'05. 2005. P. 192—201.
2. **Речистов Г. С., Иванов А. А., Шишпор П. Л., Пентковский В. М.** Симуляционный подход для нахождения производительности параллельных MPI-приложений на вычислительном кластере // Тр. 54-й науч. конф. МФТИ "Проблемы фундаментальных и прикладных естественных и технических наук в современном информационном обществе". 2011. С. 82—83.
3. **Riesen R. A.** Hybrid MPI Simulator // Barcelona — 2006 IEEE International Conference on Cluster Computing. 2006.
4. **Sundeep P., Rajive L. B.** MPI-SIM: using parallel simulation to evaluate MPI programs // Los Angeles: WSC '98 Proceedings of the 30th conference on Winter simulation. 1998.
5. **Kunkel J.** HDTrace — A Tracing and Simulation Environment of Application and System Interaction. Hamburg: University of Hamburg. 2011.
6. **HPL —** A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers // Netlib Repository at UTK and ORNL. URL: http://netlib.om/benchniark/hpl/ > (26.06.2012).
7. **MPICH2:** High-performance and widely portable MPI. // Mathematics and Computer Science Division, Argonne National Laboratory. URL: < http://www.mcs.anl.gov/research/proiects/mpich2/> (26.06.2012).
8. **Van Der Spoel D.** et al. GROMACS: Fast, flexible, and free // Journal of Computational Chemistry. 2005. T. 26, N 16. P. 1701—1718.
9. **Wind** River Systems Inc. Simics Accelerator Guide. S. l.: Wind River, 2012.
10. **Wind** River Systems Inc. Understanding Simics Timing. S. l.: Wind River, 2012.
11. **Uhlig R., Fishtein R., Gershon O., Hirsh I., Wang H.** SoftSDV: A Presilicon Software Development Environment for the IA-64 Architecture // Intel Technology Journal. 1999. P. 112—126.
12. **QEMU:** Open source processor emulator. URL: < http://wiki.qemu.org/Main_Page > (26.06.2012).
13. **Dongarra J.** The MPI Profiling Interface. [Электронный ресурс] // MPI: The Complete Reference. URL: < http://www.netlib.org/utk/papers/mpi-book/node182.html > (16.04.2012).

УДК 004.7

В. Б. Механов, канд. техн. наук, проф.,
e-mail: mvb@pnzgu.ru,
С. А. Зинкин, докт. техн. наук, проф.,
e-mail: zsa49@yandex.ru,
Н. С. Карамышева, канд. техн. наук,
инж.-программист,
Пензенский государственный университет

Формализация управления вычислительными процессами в распределенных системах хранения и обработки данных и знаний

Предлагаются новые формализмы для описания моделей асинхронного логического управления процессами и ресурсами в системах и сетях хранения и обработки данных, позволяющие унифицировать представление процессов, повысить гибкость и масштабируемость управляющих программ. Основное внимание уделяется формальному описанию процессов и их свойств на основе концепции согласования взаимодействий процессов через интенсивно обновляемые базы данных и знаний.

Ключевые слова: *распределенные системы, вычислительные процессы, агенты, формальные логико-алгебраические модели, базы данных и знаний, сети абстрактных машин, асинхронные предикатные и предикатно-функциональные сети*

Введение

Иерархия программных средств систем и сетей хранения и обработки данных может рассматриваться как реализация иерархии абстрактных машин. Программные средства, реализующие абстрактную машину, выполняют вычислительные и управляющие функции. Желательно, чтобы при реализации абстрактных машин на разных уровнях иерархии использовались сходные технологии, основанные на ограниченном выборе формальных моделей.

В приложениях информатики обычно рассматривают некоторую сигнатуру, или множество Σ представлений с интерпретацией I в множестве S элементов [1]. Интерпретация I данному представлению $\sigma \in \Sigma$ ставит в соответствие некоторое абстрактное информационное содержание $I(\sigma)$, т. е. интерпретации соответствует отображение $I: \Sigma \to S$. Положим, что Σ — множество функциональных и предикатных символов различных арностей, а S — множество конкретных функций и предикатов. Предикаты и функции, реализуемые объектами некоторого информационного пространства, задают структурные связи между понятиями предметной области. Причинно-следственные связи между этими понятиями задаются модулями-процедурами, описанными логико-алгебраическими выражениями.

Формально знания о структуре, логических связях и функционировании системы предлагается представлять предикатами, функциями и формулами, которые записываются в некоторой сигнатуре. С помощью унарных предикатов и функций можно характеризовать состояния объектов, а с помощью n-арных предикатов и функций ($n > 1$) — задавать структурные связи между объектами. Логические связи между объектами задаются формулами. При подобном традиционном представлении знаний о предметной области за пределами рассмотрения остаются процедурные знания о функционировании системы. Поэтому в модель представления знаний необходимо включить и процедурную (императивную) составляющую. В качестве процедурной модели представления знаний о функционировании системы в данной работе выбраны сети абстрактных машин и модифицированные продукционные модели. Для сокращения числа продукционных правил в работе предложено включать в ядра продукций формально описанные процедуры и правила модификации функций и предикатов.

Известно [2], что исчисление предикатов (первого порядка) — это формальная теория, в которой определены следующие компоненты: основные ($\neg$, $\to$) и дополнительные (&, $\vee$) логические связки; служебные символы (“(”, “,”, “)”); кванторы всеобщности и существования, предметные константы и переменные, предметные предикаты и функторы. Каждый предикат и функтор обладает арностью, или местностью. Формулы имеют следующий синтаксис [2]:

< формула > :: = < атом > | $\neg$ < формула > |
(< формула > $\to$ < формула >) |

$\forall$ < переменная > < формула > |
$\exists$ < переменная > < формула >

< атом > :: = < предикат > (< список термов >)

< список термов > :: = < терм > |
< терм > , < список термов >

< терм > :: = < константа > | < переменная > |
< функтор > (< список термов >)

В многоосновном (многосортовом) исчислении предикатов первого порядка [2—5] каждому терму однозначно приписывается сорт данного терма (сорта определены для таких классов объектов, как агенты, сетевые узлы, наборы данных, передавае-

мые сообщения). Имеет место следующее правило вывода [5]:

$$\frac{t_1, t_2, \ldots, t_k}{f(t_1, t_2, \ldots, t_k)},$$

где $t_1, t_2, ..., t_i, ..., t_k$ — термы сортов $\pi_1, \pi_2, ..., \pi_i, ..., \pi_k$ соответственно, f — k-арный функциональный символ вида $(\pi_1, \pi_2, ..., \pi_i, ..., \pi_k \to \pi)$, $f(t_1, t_2, ..., t_i, ..., t_k)$ — терм сорта π.

В основу организации информационного пространства положена реляционная модель данных. Как известно, в реляционной модели данных предполагается, что каждая база данных представляет собой множество истинных высказываний, структурированных в отношения, каждое из которых задается областью истинности некоторого предиката. Кортежи данных отношений представляют собой наборы значений функциональных выражений, или термов, которые при подстановке в предикат превращают его в истинное высказывание.

Сети абстрактных машин (СеАМ) содержат построенные с применением определенной в работах [6, 7] алгебры абстрактных машин "модули-процедуры" (далее просто модули), которые модифицируют, или "обновляют", интерпретацию I, выполняя сгруппированные в блоки так называемые правила обновления вида $I(\sigma_i) \leftarrow s_j$, $\sigma_i \in \Sigma$, $s_j \in S$. В алгебре абстрактных машин мы включаем подобные правила обновления в систему образующих. Сеть абстрактных машин функционирует, переходя от одной интерпретации $I(\tau_i)$ к другой интерпретации $I(\tau_j)$, где τ_i и τ_j — последовательные моменты времени, $\tau_j > \tau_i$.

В модулях используются специальные операции — элементарные обновления функций и предикатов. Элементарное правило обновления функции или предиката записывается в виде правила вывода

$$\frac{t_1, t_2, \ldots, t_k, t_{k+1}}{s(t_1, t_2, \ldots, t_k) \leftarrow t_{k+1}},$$

где $t_1, t_2, ..., t_k$ — термы различных сортов, а s — функциональный или предикатный символ. В случае, если s — функциональный символ, t_{k+1} — суть терм любого сорта, а если s — предикатный символ, то t_{k+1} — булево выражение.

Определенные нами модули сетей СеАМ используют общее пространство (FS-пространство) структурированной памяти, в качестве элементов которой применяются информационные объекты, представляющие функции и предикаты. Между модулями организуются причинно-следственные связи по типу связей между процессами и объектами.

Рассматривая в качестве вычисленных значений термов $t_1, t_2, ..., t_n$ сортов 1, 2, ..., n соответственно предметные константы $x_1, x_2, ..., x_n$, приведем пример выполнения операции, или правила, обновления некоторого n-арного предиката P.

При выполнении правила обновления предиката

$$R_1 = P(x_1, x_2, ..., x_n) \leftarrow \text{false}$$

кортеж $<x_1, x_2, ..., x_n>$ исключается из множества кортежей, образующих область истинности предиката P, или одноименное отношение P. В случае, когда кортеж $<x_1, x_2, ..., x_n>$ отсутствовал в отношении P, после выполнения указанной операции данное отношение P не изменится.

При выполнении правила обновления предиката

$$R_2 = P(x_1, x_2, ..., x_n) \leftarrow \text{true}$$

кортеж $<x_1, x_2, ..., x_n>$ включается в область истинности предиката P. Если же данный кортеж уже присутствовал в отношении P, выполнение указанной операции лишь подтвердит вхождение данного кортежа в отношение.

Рассмотрим далее несколько более сложный случай. Пусть, например, в отношении P потребовалось заменить некоторые предметные константы x_i и x_j в каком-либо кортеже $<x_1, x_2, ..., x_i, ..., x_j, ..., x_n>$ константами x_i' и x_j'. Данную задачу можно выполнить с помощью двух правил обновления области истинности предиката P:

$$R_3 = P(x_1, x_2, ..., x_i, ..., x_j, ..., x_n) \leftarrow \text{false},$$

$$R_4 = P(x_1, x_2, ..., x_i', ..., x_j', ..., x_n) \leftarrow \text{true}.$$

Операции, или правила, обновления функций реализуются аналогично. Например, равенству

$$F(x_1, x_2, ..., x_n) = x_{n+1},$$

где F — n-арный функциональный символ, соответствует наличие кортежа

$$<x_1, x_2, ..., x_n, x_{n+1}>$$

в отношении P_F с сохраненной исходной функциональной зависимостью. Тогда для изменения значения данной функции со значения x_{n+1} на x'_{n+1} достаточно выполнить две операции (правила) обновления $(n + 1)$-арного предиката P_F:

$$R_5 = P_F(x_1, x_2, ..., x_n, x_{n+1}) \leftarrow \text{false},$$

$$R_6 = P_F(x_1, x_2, ..., x_n, x'_{n+1}) \leftarrow \text{true},$$

что эквивалентно выполнению правила обновления функции

$$F(x_1, x_2, ..., x_n) \leftarrow x'_{n+1}.$$

Таким образом, выше были определены операции, или правила, обновления (модификации) пре-

дикатов и функций. Понятия правил обновления предикатов и функций широко используются в работах Ю. Гуревича [8, 9] по машинам абстрактных состояний (МАС). Однако в настоящей работе принято, что эти правила реализуются как обычные операции, определенные над отношениями и кортежами реляционных баз данных.

Алгебра модулей сетей абстрактных машин

В основу языка абстрактного описания модулей положены язык многосортного исчисления предикатов первого порядка, расширенный правилами выборки и обновления кортежей информационного пространства, а также язык систем алгоритмических алгебр Глушкова [10, 11, 12].

При определении МАС используется понятие блока совместимых (непротиворечивых) обновлений с последовательным выполнением правил обновлений предикатов и функций. В формальной записи блоки ограничиваются фигурными скобками. При построении сетей СеАМ на основе узлов, или модулей, мы будем использовать обычные для параллельного и распределенного программирования бинарные операции последовательного и параллельного выполнения модулей.

Бинарные темпоральные операции ";", "," ":", "‖", "|" предписывают различные способы выполнения модулей.

Операция ";" предписывает последовательное выполнение модулей, из которых второй модуль может зависеть от первого (в продукционных правилах данная операция обозначается составным символом $\&_{\rightarrow}$).

Операция "," предписывает выполнение независимых модулей: последовательное в произвольном порядке или параллельное (в продукционных правилах данная операция обозначается составным символом $\&_{=}$).

Операция ":" предписывает непараллельное выполнение модулей в произвольном порядке (в продукционных правилах данная операция обозначается составными символами $\&_{-}$ или $\&_{\leftrightarrow}$).

Операция "‖" предписывает модулям причинно-следственную связь, по крайней мере, через единственный предикат либо функцию; при программной реализации применение данной операции требует дальнейшей детализации описания через операции ";", "," ":", .

В продукционных правилах вместо символа обновления функции или предиката "←" будет использоваться обычный символ присваивания ":=".

Операция "|^{c}" указывает на возможное конкурентное выполнение модулей, например, использующих разделяемый ресурс; непосредственно при программировании данная операция не используется — ее применение требует дальнейшей детализации описания, как и в случае операции "‖".

Алгебраические свойства реализуемых операций описаны в работах [13, 14]. Кроме того, некоторые алгебраические свойства операций очевидны — например, операции "," ":" коммутативны и ассоциативны, а операция ";" не коммутативна и ассоциативна.

В формульной записи имена модулей, сгруппированных в блоки, заключаются в фигурные скобки, а внутри блоков могут использоваться простые скобки для указания на последовательность выполнения операций, например:

$$\{m_1, m_2, m_3\}, \{m_1; m_2\}, \{(m_1; m_2), m_3\}, \{(m_1 : m_2); m_3\}.$$

В случае, когда блок содержит лишь один модуль, скобки можно опускать. В настоящей работе алгебра алгоритмов Глушкова используется в основном для записи обычных структурированных (дейкстровских) конструкций —"последовательность операторов", "ветвление", "цикл"; в принципе здесь могла бы использоваться любая другая нотация, также пригодная для записи структурированных программ. Все указанные выше операции мы также включаем в состав операций алгебры модулей, что облегчит формирование новых модулей сетей СеАМ. Из алгебры операторов мы выбираем тернарную операцию α-дизъюнкцию и бинарную операцию α-итерацию как основы для формирования модулей СеАМ. Следуя работам [8—10], напомним правила выполнения данных операций. При выполнении операции α-дизъюнкции $[\alpha](m_1 \vee m_2)$ при α = true выполняется модуль m_1, а при α = false выполняется модуль m_2. При реализации операции α-итерации $[\alpha]\{m\}$ модуль m выполняется циклически, пока α = false, а при α = true происходит выход из цикла (следует отличать итерационные фигурные скобки от блочных). Вместо имен модулей m_1, m_2 и m в указанные выражения можно подставлять подформулы с символами любых из определенных выше операторов, например, возможно построение следующих выражений для модулей:

$$m_6 = [\alpha](\{m_1, m_2, m_3\} \vee \{(m_3; m_4), m_5\}),$$

$$m_7 = [\alpha_1](([\alpha_2]\{[\alpha_3](m_1 \vee m_2)\}) \vee ([\alpha_4](m_3 \vee m_4))).$$

Элементарный модуль содержит единственное правило обновления предиката или функции. Пустое обновление R^E эквивалентно тождественному оператору E алгебры алгоритмов Глушкова, не выполняющему никаких действий по модификации информационного пространства. Неопределенное обновление R^N соответствует неопределенному оператору N. Продукционному программированию соответствует использование модулей (модулей-продукций) следующего вида:

$$m = [\alpha](L \vee R^E),$$

где L — непустая последовательность элементарных модулей. При определенных очевидных условиях данный модуль может выполняться так же, как и модуль, описываемый выражением $[\neg\alpha]\{L\}$.

Поскольку в общем случае не все модули непосредственно (без дополнительных преобразований или без введения дополнительных условий) допускают аналитическую запись в виде суперпозиций операций, при составлении сосредоточенных и распределенных программ рекомендуется использовать модули, допускающие аналитическое описание (т. е. структурированные модули), а связи между ними организовывать посредством модификации и проверки значений функций и предикатов, составляющих информационное пространство. Такие связи называются причинно-следственными, или каузальными. Подобный подход позволяет использовать при распределенном программировании модули различного вида — от простых модулей-продукций до более сложных структурированных модулей.

Алгебра модулей сетей СеАМ, подобно системам алгоритмических алгебр, имеет систему образующих — элементарные модули и элементарные логические условия. Условиями называют замкнутые (не содержащие свободных вхождений предметных переменных) логические формулы с предикатными символами в качестве логических переменных. Множество используемых предикатных символов включает символы, используемые при формировании информационного пространства, а также символы стационарных, или немодифицируемых, предикатов сравнения. Используемые при сравнениях термы строят по обычным в многоосновном исчислении предикатов первого порядка правилам (термами, или функциональными выражениями, называют слова, построенные из переменных, функциональных и специальных символов по определенным правилам). Множество используемых функциональных символов включает символы, используемые при формировании информационного пространства, а также символы стационарных, или немодифицируемых, функций, предназначенных для выполнения арифметических и логических операций.

Среди элементарных логических условий важное значение имеют условия, формируемые на основе квалифицированных операторов выборки кортежей из отношений.

Определенные сети СеАМ являются в общем случае асинхронными недетерминированными системами ввиду произвольного порядка выбора на исполнение и неопределенного времени работы модулей, модифицирующих общее информационное пространство для получения полезного результата.

Квантифицированные операторы выбора кортежей из отношений

При проектировании систем и сетей хранения и обработки данных представляет значительный интерес построение выражений для модулей СеАМ как с использованием квантифицированных операторов выбора $\tilde{\exists}!$, $\tilde{\exists}!!$, $\tilde{\forall}$ и $\tilde{\forall}!!$, так и без них. Применение квантифицированных операторов, а также дополнительных "связывающих" предикатов может привести к существенному уменьшению необходимого числа применяемых модулей СеАМ и упрощению описывающих выражений. Учитывая, что формализмы сетей абстрактных машин мы предлагаем использовать в качестве непосредственно интерпретируемых спецификаций при создании нового аппаратного, микропрограммного и программного обеспечения систем хранения и обработки данных, следует рассмотреть различные формы записи выражений в алгебре модулей СеАМ.

При выполнении оператора $\tilde{\exists}!$ из области истинности предиката, описываемого выражением справа, выбирается произвольный кортеж. При выполнении оператора $\tilde{\exists}!!$ выбирается единственный кортеж, находящийся в области истинности стоящего справа предиката. При выполнении оператора $\tilde{\forall}$ выбираются все кортежи, составляющие область истинности соответствующего предиката. Оператор $\tilde{\forall}!!$ позволяет выбрать все кортежи из области истинности предиката в случае, если его область определения совпадает с его же областью истинности. Во всех случаях подразумевается, что предикат описывается выражением, стоящим справа от символа квантифицированного оператора.

Каждому из описанных квантифицированных операторов выборки кортежей из отношений, в случае его использования в условной части выражения для модуля, ставится в соответствие элементарное логическое условие, истинное в случае успешного выполнения оператора и ложное в противном случае. Данный факт в работах [6, 7] отмечался подчеркиванием оператора снизу. Поскольку такой способ образования элементарного логического условия применяется только при формировании условного выражения, заключенного в полном выражении для модуля (узла СеАМ) в квадратные скобки, символ подчеркивания в квадратных скобках можно опускать.

Ситуационное управление в сетях

Семантику событийного и потокового асинхронного управления удобно и наглядно описывать в терминах асинхронных предикатно-функциональных (АПФС) и асинхронных предикатных (АПС) сетей. Под АПФС или АПС понимают сети, состоящие из совокупности абстрактных машин, взаимодействующих через структурированную память — пространство функций и предикатов (АПФС) или

пространство предикатов (АПС). Факты, описываемые атомарными константными формулами, представляют в моделируемой системе некоторые события. В данных сетях проверяются логические условия, представляющие собой, в свою очередь, логические функции от некоторых событий и являющиеся условиями готовности для других событий. Используемая концепция построения сети абстрактных машин базируется на согласовании асинхронных процессов через разделяемую структурированную память (базу знаний) и интеграции моделей искусственного интеллекта с моделями дискретных распределенных систем.

При формульном описании сетей АПФС и АПС частично используется нотация систем алгоритмических алгебр, введенная В. М. Глушковым. Отметим, что концепции систем алгоритмических алгебр и сетей абстрактных машин соответствуют концепциям структурного программирования. Основной же формой представления сетей АПФС и АПС являются системы продукционных правил.

К дополнительным возможностям сетей АПФС и АПС следует отнести следующее. Сетями АПФС могут быть описаны сети Петри с обычными, информационными и ингибиторными дугами. Сетями АПС могут быть описаны безопасные ингибиторные сети Петри [15], позиции в которых имеют смысл переменных высказываний. Однако сети АПФС и АПС позволяют представлять n-арные функции и отношения конечной арности, описывающие структурные связи между объектами в системе и представленные информационными объектами в информационном пространстве; их основой является, в отличие от сетей Петри, многоосновная логика предикатов первого порядка и многоосновные алгебраические системы. Продукционное представление унарных сетей АПФС и АПС может совпадать с продукционным представлением некоторых видов сетей Петри.

Далеко не для всех модификаций сетей Петри, а также сетей АПФС и АПС разработаны методы обнаружения нежелательных ситуаций в дискретных системах (помимо универсальных, но громоздких методов анализа графов достижимых состояний), например, анализа взаимных блокировок. На предварительных этапах часто оказываются полезными методы имитационного моделирования дискретных систем.

В общем случае построение имитационной модели основано на задании отношения — области истинности бинарного предиката вида

$$R\colon \mathbf{P}(S) \times \mathbf{P}(S) \to \{\text{true, false}\},$$

где S — непустое множество ситуаций, а $\mathbf{P}$ — символ булеана. Данное отношение устанавливает зависимость одних множеств ситуаций от других. На более детализированном уровне моделирования между ситуациями задается причинно-обусловленное отношение непосредственного следования; возможно задание и других временных отношений.

Сложность современных вычислительных систем и устройств, отсутствие во многих случаях близких по характеристикам и структуре прототипов приводит разработчиков к необходимости использования имитационных моделей различных уровней. Имитационные модели традиционно используют для предсказания характеристик производительности вычислительных систем. Значительно меньше внимания уделяли использованию имитационных систем как средств проектирования и дальнейшей проверки правильности функционирования вычислительных систем и устройств.

Построение имитационных, или поведенческих, моделей систем и сетей хранения и обработки данных в настоящей работе базируется на интерпретации согласованных взаимодействий объектов через общее пространство — коммуникационную среду или общее пространство информационных объектов.

Определения формализмов для описания сложных взаимодействий процессов при использовании разделяемых ресурсов

В общем случае возможно возникновение сложных ситуаций при распределении ресурсов в сетях хранения и обработки данных. Например, ресурс может состоять из многих физических узлов и иметь несколько единиц, за разделяемое или монопольное использование которых конкурируют многие процессы; узлы сети, реализующие единицы абстрактного ресурса, могут использоваться другими процессами. Возникает проблема выбора стратегий ожидания или захвата единиц ресурса, проблема разрешения тупиковых ситуаций.

Дадим пояснения к некоторым используемым далее понятиям и обозначениям. Пусть P — множество предикатов, F — множество функций, $I_{P \cup F}$ — начальная интерпретация предикатно-функциональной сигнатуры — имен предикатов и функций из множества $P \cup F$ (заданная для многоосновных, или многосортных, алгебраических систем интерпретация сигнатуры — это отображение множества имен $P \cup F$ в множество отношений и операций, определенных на некоторых декартовых произведениях основных множеств различных сортов). Под воздействием модулей из множества M, реализующих продукционные правила из множества *Prod*, текущая интерпретация $I_P \cup F$ сигнатуры модифицируется, задавая тем самым динамику предметной области. Начальная интерпретация предикатной сигнатуры обозначена как I_P. Унарные предикаты характеризуют состояния процессов, инициируемых мобильными агентами. Предметные переменные, или аргументы предикатов, пробегают по одномерным массивам предметных констант.

При определении формализмов использована (на содержательном уровне) концепция разделяе-

мого пространства кортежей и отношений, впервые предложенная профессором Д. Джелернтером в Йельском университете (США) в связи с работой над проектом Linda [16], а также модель машин абстрактных состояний Ю. Гуревича (Мичиганский университет, США) [8, 9].

Определение 1. Асинхронной предикатной сетью (АПС) *APN* называется набор

$$APN = (A, A', M, P, I_P, Prod, f_{MA}),$$

где $A = \{a_1, a_2, ..., a_n\}$ — множество агентов; $A' = A'_1 \cup \cup A'_2 \cup ... \cup A'_n$ — множество некоторых символов (меток), отмечающих отдельные стадии выполнения процессов на различных узлах распределенной вычислительной системы или сети и интерпретируемых как предметные константы, $A'_1, A'_2, ..., A'_n$ — непересекающиеся подмножества множества A', определенные соответственно для каждого из агентов $a_1, a_2, ..., a_n$; M — множество абстрактных узлов (модулей) сети *APN*, реализующих продукционные правила из множества *Prod*; P — множество модифицируемых предикатов, в том числе унарных предикатов вида p_i: $A'_i \to \{true, false\}$, $p_i \in P$, образующих модифицируемое пространство (или базу знаний) согласования взаимодействующих процессов и ресурсов; I_P — начальная интерпретация предикатной сигнатуры, задающая начальное состояние базы знаний, включающей предикаты из множества P (сеть АПС, для которой определены лишь унарные предикаты, названа унарной); *Prod* — множество продукционных правил обновления (модификации) предикатов; f_{MA}: $M \to A$ — унарная функция, ставящая в соответствие каждому модулю реализующий его агент.

Продукционные правила из множества *Prod* удобно представлять также в виде α-дизъюнкций вида $[\alpha](r_1 \vee r_2)$ или $[\alpha](r \vee E)$, т. е. в нотации систем алгоритмических алгебр Глушкова.

Определение 2. Асинхронная предикатно-функциональная сеть (АПФС) — это набор

$$APFN = (A, A', A'', M, P, F, I_{P \cup F}, Prod', f_{MA}, B, C),$$

где дополнительно заданы: A'' — множество дополнительных меток для обозначения объектов, представляемых в модели унарными функциями; F — множество модифицируемых функций, в том числе унарных функций вида f: $A'' \to z$ (z — множество целых чисел; множество $P \cup F$ образует модифицируемое пространство, или базу знаний, согласования взаимодействующих процессов и ресурсов); $I_{P \cup F}$ — начальная интерпретация предикатно-функциональной сигнатуры, задающая начальное состояние базы знаний, включающей предикаты и функции из множества $P \cup F$; *Prod'* — множество продукционных правил обновления (модификации) предикатов и функций из множества $P \cup F$; $B = \{p_{ge}, p_{le}, p_{gt}, p_{lt}, p_{eq}, p_{ne}\}$ — множество бинарных предикатов сравнения, $B \cap P = \varnothing$; $C = \{f_{\text{plus}}, f_{\text{minus}}\}$ — множество, содержащее две бинарные арифметические функции сложения и вычитания, определенные для целых чисел, $C \cap F = \varnothing$. Остальные элементы кортежа соответствуют **определению 1**. Сеть АПФС, для которой определены только унарные предикаты и функции, назовем унарной.

Полученная АПФС-модель распределенной системы логического управления также, как и предыдущая АПС-модель, относится к классу непосредственно исполняемых формальных спецификаций и может быть использована для моделирования и непосредственно при написании сетевых управляющих программ. В распределенной системе логического управления происходит интенсивное обновление фактов в распределенной базе знаний о функционировании системы (рис. 1).

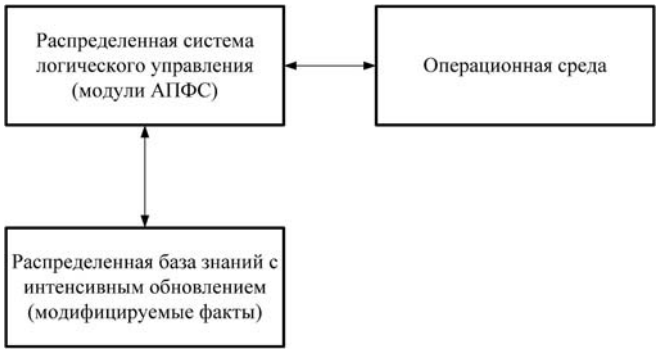


Рис. 1. Организация распределенной системы логического управления

Определение 3. Асинхронная предикатная сеть АПС $APN_{I/O}$ с входом и выходом определяется набором

$$APN_{I/O} = (A, A', M, P, I_P, Prod, f_{MA}, X, Y, p_X, p_Y),$$

где X, Y — множества входных и выходных символов (меток), отмечающих соответственно виды входных и выходных воздействий при взаимодействиях с операционной средой (например, при занятии и освобождении ресурсов) и интерпретируемых как предметные константы, p_X: $X \to \{true, false\}$ — входной унарный предикат, $p_X \in P$, p_Y: $Y \to \{true, false\}$ — выходной унарный предикат, $p_Y \in P$. Остальные элементы кортежа соответствуют **определению 1**.

Определение 4. Асинхронная предикатно-функциональная сеть АПФС $APFN_{I/O}$ с входом и выходом определяется набором

$$APFN_{I/O} = (A, A', A'', M, P, F, I_{P \cup F}, Prod', f_{MA}, B, C, X, Y, p_X, p_Y, f_X, f_Y),$$

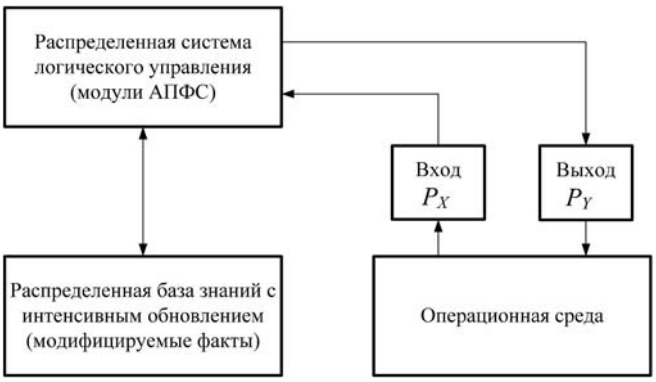


Рис. 2. Организация распределенной системы логического управления с входом и выходом

где f_X: $X \to z$, f_Y: $Y \to z$ — входная и выходная унарные функции, $f_X \in F$, $f_Y \in F$, z —множество целых чисел. Остальные элементы кортежа соответствуют **определениям 1, 2 и 3**.

Входными и выходными предикатами и функциями в сетях АПС и АПФС задаются входные и выходные воздействия. Общую структуру распределенной системы логического управления конкурирующими распределенными процессами через распределенную базу знаний иллюстрирует рис. 2.

Об эквивалентных преобразованиях сетевых моделей

Переход от сетей АПФС к сетям АПС можно осуществить при учете того известного факта [1], что n-арная функция $f(x_1, x_2, ..., x_n)$ представима в виде $(n + 1)$-арного предиката $p_f(x_1, x_2, ..., x_n, y)$ при условии, что $f(x_1, x_2, ..., x_n) = y$ тогда и только тогда, когда предикат $p_f(x_1, x_2, ..., x_n, y)$ истинен. Здесь $x_1, x_2, ..., x_n, y$ — предметные переменные различных сортов. В отношении, представленном областью истинности данного предиката, должна сохраняться функциональная зависимость y от $x_1, x_2, ..., x_n$. Поэтому "чистое" (без функций) исчисление предикатов эквивалентно по принципиальной выразительности многосортному исчислению предикатов [1] и, следовательно, сети АПФС могут быть представлены эквивалентными по принципиальной выразительности сетями АПС путем представления всех функций, определенных для АПФС, соответствующими предикатами.

При программной реализации сетей АПФС и АПС используется одинаковое табличное представление не полностью определенных функций и предикатов. Приведем пример модификации функции $f(x_1, x_2, ..., x_n)$, представленной в виде $(n + 1)$-арного предиката $p_f(x_1, x_2, ..., x_n, y)$, при которой старое значение функции y' заменяется на новое значение y''. В сетях АПФС при модификации функции выполняется операция $f(t_1, t_2, ..., t_n) \leftarrow y''$, где $t_1, t_2, ..., t_n$ — термы соответствующих сортов, а при модификации предиката $p_f(x_1, x_2, ..., x_n, y)$ сначала требуется найти кортеж $<t_1, t_2, ..., t_n, y'>$ в его области истинности, предварительно вычислив значения термов, удалить его, а затем вставить новый кортеж $<t_1, t_2, ..., t_n, y''>$.

Опишем выполнение последней операции в терминах сетей абстрактных машин. Следующее выражение описывает модификацию предиката $p_f(x_1, x_2, ..., x_n, y)$:

$$[\tilde{\exists}!(t_1, t_2, ..., t_n, y)p_f(x_1, x_2, ..., x_n, y)] \times$$
$$\times (\{p_f(x_1, x_2, ..., x_n, y) \leftarrow \text{false};$$
$$p_f(x_1, x_2, ..., x_n, y'') \leftarrow \text{true}\} \vee$$
$$\vee\, p_f(x_1, x_2, ..., x_n, y'') \leftarrow \text{true}).$$

Пусть в качестве термов используются предметные константы $x'_1, x'_2, ..., x'_n$, тогда последнее выражение можно переписать в следующем виде:

$$[\tilde{\exists}!(x'_1, x'_2, ..., x'_n, y)p_f(x_1, x_2, ..., x_n, y)] \times$$
$$\times (\{p_f(x_1, x_2, ..., x_n, y) \leftarrow \text{false};$$
$$p_f(x_1, x_2, ..., x_n, y'') \leftarrow \text{true}\} \vee$$
$$\vee\, p_f(x_1, x_2, ..., x_n, y'') \leftarrow \text{true}).$$

Данное выражение реализуется модулем (узлом) сети абстрактных машин. Выражению в квадратных скобках соответствует логическое условие α, истинное в случае, когда значение y определено для значений $x'_1, x'_2, ..., x'_n$ предметных переменных $x_1, x_2, ..., x_n$ соответственно, т. е. в области истинности предиката $p_f(x_1, x_2, ..., x_n, y)$ найден кортеж $<x'_1, x'_2, ..., x'_n, y'>$ (при $y = y'$). Модуль далее выполняет последовательно два правила: $\{p_f(x_1, x_2, ..., x_n, y) \leftarrow \text{false}$; $p_f(x_1, x_2, ..., x_n, y'') \leftarrow \text{true}\}$, сгруппированных в блок. При выполнении первого правила найденный кортеж $<x'_1, x'_2, ..., x'_n, y'>$ удаляется из области истинности предиката $p_f(x_1, x_2, ..., x_n, y)$, и далее при выполнении второго правила в эту область добавляется новый кортеж $<x_1, x_2, ..., x_n, y''>$.

В случае, когда кортеж $<x'_1, x'_2, ..., x'_n, y'>$ в области истинности предиката не найден, значение y не было определено (в этом случае полагается, что $y = \text{undef}$), значение логического условия α становится ложным и модуль выполняет лишь одно правило модификации предиката $p_f(x_1, x_2, ..., x_n, y'') \leftarrow \text{true}$, что соответствует помещению кортежа $<x'_1, x'_2, ..., x'_n, y''>$ в область истинности данного предиката. В результате выполнения модуля, таким образом, осуществляется модификация (обновление) функции $f(x_1, x_2, ..., x_n)$, представленной в виде $(n + 1)$-арного предиката $p_f(x_1, x_2, ..., x_n, y)$, при этой модификации старое значение функции y' заменяется на новое значение y''.

Выражения, интерпретируемые модулями сетей абстрактных машин как сетей АПС, так и сетей

АПФС, могут быть представлены в продукционной форме:

$$[\widetilde{\exists}!(x'_1, x'_2, ..., x'_n, y)p_f(x_1, x_2, ..., x_n, y)] \Rightarrow$$
$$\Rightarrow (p_f(x_1, x_2, ..., x_n, y) := \text{false}) \&_{\rightarrow}$$
$$(p_f(x_1, x_2, ..., x_n, y'') := \text{true});$$

$$\neg[\widetilde{\exists}!(x'_1, x'_2, ..., x'_n, y)p_f(x_1, x_2, ..., x_n, y)] \Rightarrow$$
$$\Rightarrow p_f(x_1, x_2, ..., x_n, y'') := \text{true}).$$

На практике механизмы работы с предикатами, представляющими функции, могут быть скрыты от пользователя-программиста, который использует обычные функции и связанные с ними операции. Значение переменной y труднее получить, если в сети АПС n-арная функция представлена $(n + 1)$-арным предикатом:

$[\widetilde{\exists}!(x'_1, x'_2, ..., x'_n, y)p_f(x_1, x_2, ..., x_n, y)]$({выражение, в котором используется значение функции $y = y'$} $\vee$ $\vee$ {выражение, учитывающее неопределенное значение функции $y = \text{undef}$}),

поэтому при наличии значительного числа функций рекомендуется использовать формализм сетей АПФС.

Заключение

Предложены формализмы для описания моделей асинхронного логического управления процессами и ресурсами в системах и сетях хранения и обработки данных, основанные на реализации согласованных взаимодействий в коллективе агентов, разделяющих общую распределенную базу знаний о функционировании системы, что позволяет унифицировать представление процессов, повысить гибкость и масштабируемость управляющих программ.

Многоосновная логика предикатов первого порядка, положенная в основу сетей АПС и АПФС, является логическим формализмом, который задает формальную семантику и операционную поддержку в виде механизмов логического вывода. Формализмы АПС и АПФС дополнительно позволяют задавать операционную семантику процессов в заданной предметной области в виде сочетания декларативной и императивной (процедурной) парадигм, применяемых в программировании. Особым удобством формализмов АПС и АПФС является также возможность использования в их составе реляционной модели данных как формализма, основанного на языке многоосновного исчисления предикатов. Результаты настоящей работы являются развитием идей и методов, описанных ранее в работах [17—19].

Список литературы

1. **Плесневич Г. С.** Логические модели // Искусственный интеллект. В 3-х кн. Кн. 2. Модели и методы: Справочник / под ред. Д. А. Поспелова. М.: Радио и связь, 1990. С. 14—28.

2. **Новиков Ф. А.** Дискретная математика для программистов. СПб: Питер, 2001. 304 с.

3. **Общая** алгебра / В. А. Артамонов, В. Н. Салий, Л. А. Скорняков, Л. Н. Шеврин, Е. Г. Шульгейфер. М.: Наука, 1991. 480 с.

4. **Плоткин Б. И.** Универсальная алгебра, алгебраическая логика и базы данных. М.: Наука, 1991. 448 с.

5. **Колмогоров А. Н., Драгалин А. Г.** Математическая логика. М.: Изд-во УРСС, МГУ. 2005. 240 с.

6. **Зинкин С. А.** Сети абстрактных машин высших порядков в проектировании систем и сетей хранения и обработки данных (базовый формализм и его расширения) // Известия высших учебных заведений. Поволжский регион. Технические науки. 2007. № 3. С. 13—22.

7. **Зинкин С. А.** Сети абстрактных машин высших порядков в проектировании систем и сетей хранения и обработки данных (механизмы интерпретации и варианты использования) // Известия высших учебных заведений. Поволжский регион. Технические науки. 2007. № 4. С. 37—51.

8. **Gurevich Y.** Evolving algebras — a tutorial introduction // Bulletin of the EATS. 1991. N 43. P. 264—284.

9. **Dexter S., Doyle P., Gurevich Y.** Gurevich abstract state machines and Shonhage storage modification machines // Journal of Universal Comp. Science. 1997. Vol. 3, № 4. P. 279—303.

10. **Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л.** Методы символьной мультиобработки. Киев: Наукова думка, 1980. 252 с.

11. **Многоуровневое** структурное проектирование программ. Теоретические основы, инструментарий / Е. Л. Ющенко, Г. Е. Цейтлин, В. П. Грицай, Т. К. Терзян. М.: Финансы и статистика, 1989. 208 с.

12. **Капитонова Ю. В., Летичевский А. А.** Математическая теория проектирования вычислительных систем. М.: Наука, 1988. 296 с.

13. **Зинкин С. А.** Самомодифицируемые сценарные модели функционирования систем и сетей хранения и обработки данных (базовый формализм и темпоральные операции // Известия высших учебных заведений. Поволжский регион. Технические науки. 2007. № 1. С. 3—12.

14. **Зинкин С. А.** Самомодифицируемые сценарные модели функционирования систем и сетей хранения и обработки данных (реализация и свойства сценарных моделей) // Известия высших учебных заведений. Поволжский регион. Технические науки. 2007. № 2. С. 13—21.

15. **Котов В. Е.** Сети Петри. М.: Наука, 1984. 160 с.

16. **Gelernter P., Zuck L. D.** On what Linda is: formal description of Linda as a reactive system // Lecture Notes in Computer Science. Proc. of the Second International Conference on Coordination Languages and Models. 1997. Vol. 1282. P. 187—204.

17. **Зинкина Н. С.** Методы и модели логического управления дискретными процессами в распределенных вычислительных системах на основе концепции согласования // Известия высших учебных заведений. Поволжский регион. Технические науки. 2011. № 1. С. 35—47.

18. **Зинкин С. А.** Элементы новой объектно-ориентированной технологии для моделирования и реализации систем и сетей хранения и обработки данных // Информационные технологии. 2008. № 10. С. 20—27.

19. **Зинкин С. А.** Реализация барьерной синхронизации и управление процессами в виртуальном сетевом дисковом массиве // Информационные технологии. 2008. № 12. С. 22—29.

ЖУРНАЛ В ЖУРНАЛЕ

НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ

№ 1
ЯНВАРЬ
2013

УДК: 004.032.26, 543.424.2

С. А. Доленко[1],
канд. физ.-мат. наук, ст. науч. сотр.,
С. А. Буриков[2],
канд. физ.-мат. наук, ст. науч. сотр.,
Т. А. Доленко[2],
канд. физ.-мат. наук, ст. науч. сотр.,
И. Г. Персианцев[1],
д-р физ.-мат. наук, вед. науч. сотр.,
А. Р. Сабиров[2], мл. науч. сотр.,
В. В. Фадеев[2], д-р физ.-мат. наук, проф.
[1] МГУ имени М. В. Ломоносова,
НИИ ядерной физики имени Д. В. Скобельцына
[2] МГУ имени М. В. Ломоносова,
Физический факультет

Нейросетевое решение обратной задачи лазерной спектроскопии по дистанционному определению температуры и солености природных вод с учетом влияния растворенного органического вещества

Данная работа посвящена развитию метода нейросетевого решения задачи одновременного определения температуры и солености морской воды по спектрам комбинационного рассеяния света. В работе продолжено развитие метода на основе использования отбора существенных входных признаков. Представлены сравнение разных способов отбора существенных входных признаков и анализ их влияния на погрешность определения температуры и солености.

Ключевые слова: *нейронные сети, обратные задачи, отбор существенных входных признаков, спектроскопия комбинационного рассеяния света*

Введение

Информация о таких параметрах морской воды как температура (T) и соленость (S) очень важна, так как помогает понять динамику климатических изменений, изучить энергообмен между водной поверхностью и атмосферой. Необходимость глобального мониторинга T и S возникает вследствие наблюдаемого в течение последних нескольких лет таяния льдов в полярных широтах в результате глобального потепления, что может стимулировать перестроение системы океанских течений и стать причиной серьезных климатических изменений не только в полярных областях, но и в планетарном масштабе.

Очевидно, что для экологического мониторинга природных вод — определения ключевых параметров T и S — необходимы экспрессные неконтактные методы диагностики воды, которые можно применять в режиме реального времени. Такими свойствами обладает широко распространенный в океанологии неконтактный радиометрический метод определения солености или температуры поверхностного слоя морской воды [1, 2]. Метод измерения солености морской воды с помощью радиометров, основанный на зависимости поглощающей способности водной поверхности от концентрации солей, позволяет определять S с точностью не лучше, чем десятые доли практических единиц солености — PSU (*practical salinity units*) [3]. Точность определения температуры водной поверхности с помощью радиометрического метода в настоящее время составляет 1 °C [4, 5].

Ошибки в определении T и S радиометрическим методом обусловлены необходимостью выделения малых изменений теплового излучения за счет изменения T и S на фоне интенсивного изменения сигнала, связанного с неоднородностью поверхности. Также следует учитывать влияние погодных условий на поглощение излучения приповерхностным слоем воды. Методы лазерной спектроскопии дают более удобный инструмент для определения параметров природной воды. Использование флуоресцентной спектроскопии и спектроскопии комбинационного рассеяния (КР) позволяет получить информацию о параметрах воды дистанционно (с помощью лидара или световода) в режиме реального времени.

Влияние температуры и солености на спектр КР воды было обнаружено Дж. Волрафеном [6—8]. Метод определения этих характеристик морской воды, основанный на зависимости формы и положения валентной полосы КР воды от температуры и солености, был разработан авторами [9—11]. Использование зависимости соотношения интенсивностей высоко- и низкочастотной областей валентной полосы КР воды от температуры позволило авторам этих работ получить точность определения температуры 0,5 °C в лабораторных и 2 °C в полевых условиях.

Колебательная спектроскопия (в частности спектроскопия КР) может быть использована для определения параметров воды благодаря чувствительности спектров КР к типу и концентрации растворенных солей и температуре воды [12—14]. Влияние температуры и растворенных солей на валентную полосу КР воды проявляется в изменении ее формы и положения ([12—14], рис. 1—2 из работы [15]). При повышении температуры воды и/или концентрации растворенных в ней солей возрастает интенсивность высокочастотной области полосы, интенсивность низкочастотной области уменьша-

ется. При этом полоса сужается и сдвигается в сторону высоких частот.

В работе [16] для определения температуры воды проводилось разложение валентной полосы КР воды на контуры формы Гаусса или Фойгта. Использовалась линейная область зависимости интенсивности двух компонент — высоко- и низкочастотной от температуры. Точность определения температуры составляла 1 °С [16]. Аналогичным образом определялась соленость морской воды авторами работ [17, 18].

Авторы данной работы показали, что по валентной полосе КР воды температура и соленость могут быть измерены одновременно [15, 19]. Точность определения T и S с помощью трехчастотного метода составляла 0,7 °С и 1,0 PSU — практических единиц солености (в лаборатории) и 1,1 °С и 1,4 PSU в полевых условиях.

Использование искусственных нейронных сетей (ИНС) позволило уменьшить погрешность определения температуры и солености до 0,5 °С и 0,7 PSU в лабораторных условиях [15].

В данной работе методом лазерной спектроскопии решалась новая задача: определение T и S воды осуществлялось в присутствии в ней растворенного органического вещества (РОВ) в широком диапазоне концентраций. РОВ всегда присутствует в природных водах, при этом его концентрация меняется в зависимости от района измерений (в устьях рек концентрация РОВ выше), времени года и т. д. [20]. Сложность задачи заключалась в том, что полоса флуоресценции РОВ перекрывается с валентной полосой КР воды: это является дополнительным источником ошибок.

Эксперимент

Решение поставленной задачи (определение T и S с учетом флуоресценции РОВ) с применением искусственных нейронных сетей (ИНС) было проведено с помощью подхода "от эксперимента" [21]. Это означает, что для тренировки ИНС были использованы только экспериментально полученные спектры. В этом случае не нужна предварительно созданная модель, и все специфические особенности объекта учитываются автоматически.

Для решения поставленной задачи был получен массив экспериментальных спектров водных растворов с изменяющимися параметрами (температура, соленость, концентрация РОВ). Растворы были приготовлены из дважды дистиллированной воды, речного гумуса и морской соли. Соленость изменялась в диапазоне от 0 до 45 PSU (с шагом 5 PSU), концентрация гумуса — от 0 до 350 мг/л, температура — от 0 до 35 °С (с шагом 5 °С).

Схема КР-спектрометра представлена на рис. 1. Спектры были измерены в области 800...4000 $см^{-1}$ с практическим разрешением 2 $см^{-1}$. Мощность аргонового лазера, используемого для возбуждения сигнала КР, составляла 500 мВт на длине волны 488 нм. Для регистрации спектров использовалась CCD-камера. Система термостабилизации позволяла измерять и поддерживать температуру образцов с точностью 0,1 °С. Спектры были нормированы на мощность лазера и время накопления сигнала. Время накопления сигнала составляло 5 с для валентных полос и 10 с для низкочастотной области спектра КР.

Регистрация спектров КР водных растворов проводилась таким образом, чтобы к массиву валентных полос (в области волновых чисел 2220...3870 $см^{-1}$, 1024 признака, рис. 2) получить дополнительный набор идентификационных признаков — спектров КР в низкочастотной области (800...1800 $см^{-1}$, 1024 признака, рис. 3). Низкочастотные колебательные полосы воды тоже зависят от температуры, солености и РОВ и могут включать собствен-

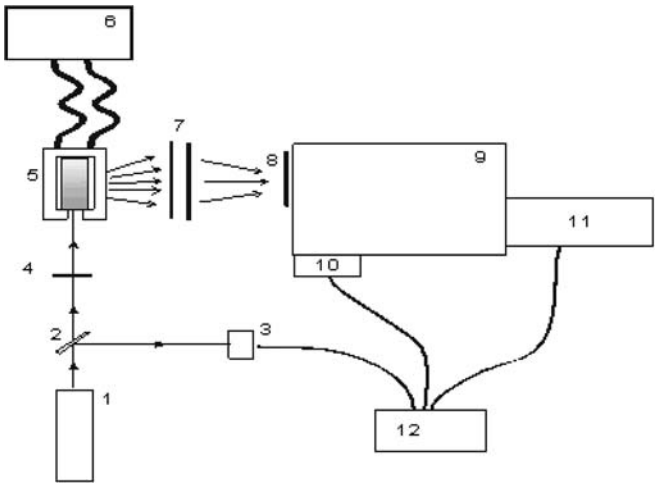


Рис. 1. Схема экспериментальной установки:
1 — аргоновый лазер (488 нм); *2* — светоделительная пластинка; *3* — измеритель мощности лазерного излучения; *4* — фокусирующая линза; *5* — термостабилизированная кювета; *6* — система термостабилизации; *7* — система светосбора; *8* — фильтр; *9* — монохроматор; *10* — ФЭУ; *11* — ССР-камера; *12* — компьютер

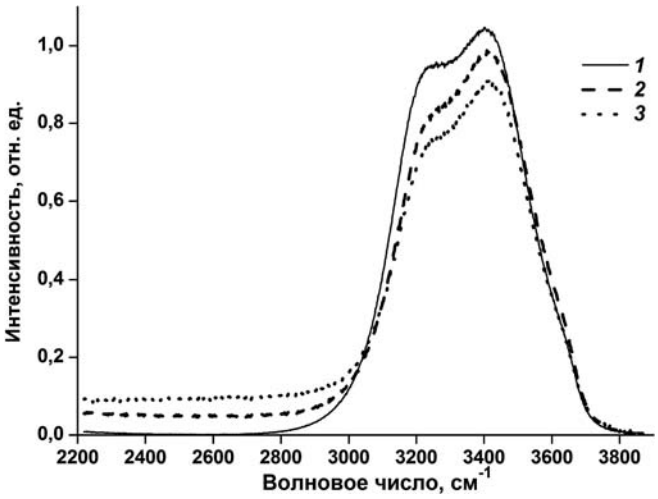


Рис. 2. Валентные полосы КР воды:
1 — 0 °С, 25 PSU, 0 мг/л; *2* — 25 °С, 15 PSU, 175 мг/л; *3* — 15 °С, 45 PSU, 350 мг/л

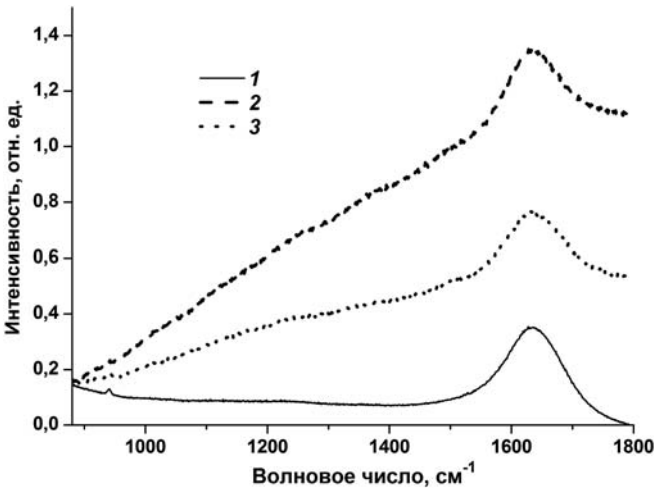


Рис. 3. Низкочастотные полосы спектров КР водных растворов: *1* — 15 °С, 40 PSU, 0 мг/л; *2* — 25 °С, 15 PSU, 175 мг/л; *3* — 21 °С, 15 PSU, 58,3 мг/л

ные полосы КР таких анионов, как NO_3^-, SO_4^{2-}, PO_4^{3-}, HCO_3^- [22]. Предполагалось, что дополнительные идентификационные признаки обеспечат более точное определение солености. Флуоресценция РОВ при решении поставленной задачи рассматривалась как шум.

Методы

В работе [22] по таким же наборам экспериментальных данных определяли T и S воды в присутствии РОВ с помощью прямого применения ИНС. Наилучшие результаты были получены с помощью персептронов с тремя скрытыми слоями. При использовании только валентной полосы КР воды наилучшая точность (средняя абсолютная ошибка (САО)) определения температуры и солености составляла 1,2 °С и 1,5 PSU (практических единиц солености), соответственно.

Использование как валентной полосы, так и низкочастотной области спектра позволило уменьшить погрешность определения температуры до 0,8 °С, а солености — до 1,1 PSU. Напомним, что максимальные значения САО, при которых метод представляет интерес для практических приложений, составляют 1 °С и 1 PSU.

Целью настоящей работы было достижение такого же (или лучшего) результата при использовании только валентной полосы. Это является важным на практике, так как регистрация низкочастотных полос спектра КР требует более сложного и, следовательно, более дорогого экспериментального оборудования.

Достигнуть указанной цели планировалось с помощью уменьшения исходной размерности входных данных (1024 признака — спектральных канала). Довольно очевидно, что реальная размерность задачи должна быть намного ниже. Для компрессии входных данных использовали различные методы отбора существенных входных признаков.

Во всех экспериментах с ИНС в данной работе использовали фиксированную архитектуру НС. Это был персептрон с одним скрытым слоем из 64 нейронов, логистической передаточной функцией в скрытом слое и линейной передаточной функцией в выходном слое. Скорость обучения составляла $r = 0{,}01$, момент $m = 0{,}5$. Обучение прекращалось через 1000 эпох после достижения минимальной ошибки на тестовом наборе. Результаты оценивали по экзаменационному набору, состоящему из примеров, не входящих в тестовый и тренировочный наборы. Чтобы учесть возможное влияние случайных факторов при инициализации весов, в каждом эксперименте были натренированы пять НС с разными исходными весами.

1. Кросс-корреляция. Были рассчитаны абсолютные значения коэффициентов кросс-корреляции (КК) между всеми входными признаками, с одной стороны, и выходными признаками, с другой стороны. В дальнейшем при решении задачи использовали только те входные признаки, абсолютные значения КК которых превышали установленный порог (0,3). Основной недостаток данного метода заключается в том, что линейная корреляция учитывает только линейные соотношения между переменными, пропуская существенные входные признаки, которые нелинейным образом влияют на выходные переменные. На рис. 4 представлена полученная зависимость КК от волнового числа, соответствующего каждому признаку.

2. Кросс-энтропия. Были рассчитаны коэффициенты кросс-энтропии (КЭ) каждого из входных признаков с выходными. В дальнейшем использовали только те признаки, значения коэффициентов кросс-энтропии которых превышали заданный порог (0,2). Хотя КЭ может учитывать нелинейные соотношения, точность расчета невелика для небольшого числа примеров, получаемых в эксперименте. Полученная спектральная зависимость коэффициентов КЭ представлена на рис. 4.

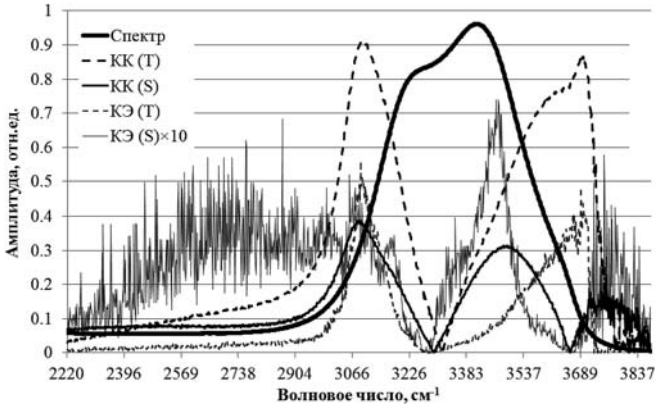


Рис. 4. Спектральные зависимости коэффициентов кросс-корреляции (абсолютные значения) и кросс-энтропии для температуры и солености

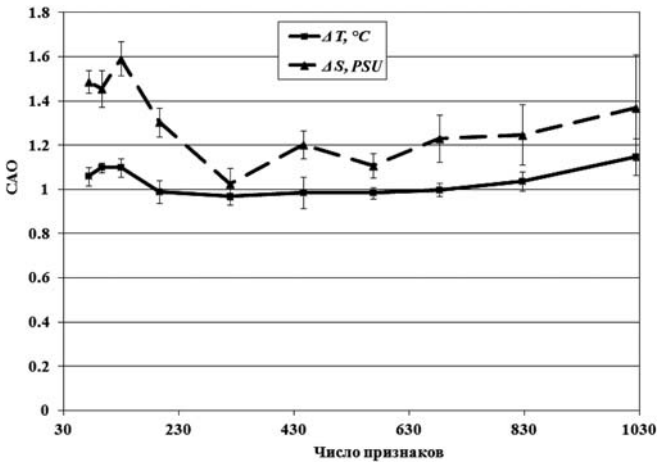


Рис. 5. Зависимость средней абсолютной ошибки определения *T* и *S* от числа признаков, отобранных с помощью НСОР-ГА

3. НС с общей регрессией (НСОР) [23] с поправочными коэффициентами для показателя сглаживания для всех входных признаков (как было реализовано в программном пакете NeuroShell 2 [24]). Использовались только те входные признаки, поправочные коэффициенты для которых превышали установленный порог (0,5). Поскольку существует очевидная взаимосвязь между входными признаками, а поправочные коэффициенты определяются с помощью генетических алгоритмов (ГА), набор коэффициентов, определяемых при однократном запуске алгоритма, подвержен сильному влиянию случайных факторов. По этой причине процедура повторялась рекуррентно несколько раз, при этом каждый новый ее запуск позволял получить более узкий набор существенных признаков. Каждый из последовательно полученных наборов использовался при решении задачи. Зависимость САО при определении *T* и *S* от числа выбранных таким образом признаков представлена на рис. 5.

Результаты

Наилучшие результаты, полученные в настоящей работе для разных методов отбора существенных входных признаков, сведены в таблицу. Представленные значения — это средние абсолютные ошибки, полученные на экзаменационном наборе (состоящем из примеров, не входящих в тестовый и тренировочный наборы).

Средние абсолютные ошибки определения *T* и *S* на экзаменационном наборе для разных методов отбора существенных входных признаков

Метод отбора признаков	Число входных признаков	ΔT, °C	ΔS, PSU
Нет	1024	1,15 ± 0,08	1,37 ± 0,24
Кросс-корреляция	375	0,92 ± 0,06	1,18 ± 0,09
Кросс-энтропия	694	0,91 ± 0,05	1,15 ± 0,07
НСОР-ГА	319	0,97 ± 0,04	1,02 ± 0,07

Заключение

Данная работа посвящена сравнению разных методов отбора существенных входных признаков при нейросетевом решении обратной задачи определения температуры и солености морской воды по валентной полосе КР воды с учетом флуоресценции растворенного органического вещества в широком диапазоне концентраций.

Наилучшие результаты были получены при использовании НСОР-ГА, при 319 входных признаках. Значения средней абсолютной ошибки, полученные на экзаменационном наборе, составили 0,97 ± 0,04 °C и 1,02 ± 0,07 PSU. Полученные низкие значения погрешности определения параметров *T* и *S* уже представляют интерес при решении практических задач в океанологии.

Данная работа была выполнена при поддержке грантов РФФИ № 11-05-01160-а и 12-01-00958-а. Все вычисления с использованием НС были выполнены с помощью программного пакета NeuroShell 2 [24].

Список литературы

1. **Font J., Camps A., Borges A., Martin-Neira M., Boutin J., Reul N., Kerr Y., Hahne A., Mecklenburg S.** SMOS: The challenging measurement of sea surface salinity from space // Proc. IEEE. 2010. V. 98 (5). P. 649—665.

2. **Turiel A., Nieves V., Garcia-Ladona E., Font J., Rio M. H., Larnicol G.** The multifractal structure of satellite sea surface temperature maps can be used to obtain global maps of streamlines // Ocean Sci. 2009. V. 5. P. 447—460.

3. **Boutin J., Waldteufel P., Martin N., Caudal G., Dinnat E.** Surface salinity retrieved from SMOS measurements over the global ocean: Imprecisions due to sea surface roughness and temperature uncertainties // J. Atmos. Ocean. Technol. 2004. V. 21. P. 1432—1447.

4. **Eugenio F., Marcello J., Hernandez-Guerra A., Rovaris E.** Methodology to obtain accurate sea surface temperature from locally received NOAA-14 data in the Canary-Azores-Gibraltar area // Scientia Marina. 2001. V. 65 (1). P. 127—137.

5. **Garcia-Santos V., Valor E., Caselles V.** Determination of temperature by remote sensing // J. of Mediterranean Meteorology & Climatology, 2010. V. 7. P. 67—74.

6. **Walrafen G. E.** Raman Spectral Studies of Water Structure // J. Chem. Phys. 1964. V. 40. P. 3249—3256.

7. **Walrafen G. E.** Raman Spectral Studies of the Effects of Temperature on Water and Electrolyte Solutions // J. Chem. Phys. 1966. V. 44. P. 1546—1558.

8. **Walrafen G. E.** Raman Spectral Studies of the Effects of Temperature on Water Structure // J. Chem. Phys. 1967. V. 47. P. 114—126.

9. **Chang C. H., Young L. A.** Seawater Temperature Measurement from Raman Spectra. Avco Everett Research Laboratory. Interim technical report, 1972.

10. **Leonard D., Chang C., Yang L.** Remote measurement of fluid temperature by Raman scattered radiation, 1974, U. S. Patent 3.986.775, Class 356-75.

11. **Leonard D., Caputo B., Hoge F.** Remote sensing of subsurface water temperature by Raman scattering // Applied Optics. 1979. V. 18 (11). P. 1732—1745.

12. **Terpstra P., Combes D., Zwick A.** Effect of salts on dynamics of water: A Raman spectroscopy study // J. Chem. Phys. 1990. V. 92 (1). P. 65—70.

13. **Dolenko T. A., Churina I. V., Fadeev V. V., Glushkov S. M.** Valence band of liquid water Raman scattering: some peculiarities and applications in the diagnostics of water media // J. of Raman Spectroscopy. 2000. V. 31 (8—9). P. 863—870.

14. **Sherer J., Go M., Kint S.** Raman spectra and structure of water from −10 to 90 °C // J. Phys. Chem. 1974. V. 78 (13). P. 1304—1313.

15. **Burikov S. A., Churina I. V., Dolenko S. A., Dolenko T. A., Fadeev V. V.** New approaches to determination of temperature and salinity of seawater by laser Raman spectroscopy // 3nd EARSeL Workshop on Remote Sensing of the Coastal Zone, 2003. P. 298—305.

16. **Karl J., Ottmann M., Hein D.** Measuring water temperatures by means of linear Raman spectroscopy // Proc. of the 9th International Symposium on Application of Laser Techniques to Fluid Mechanics. 1998. V. II. P. 23.2.1—23.2.8.

17. **Becucci M., Cavalieri S., Eramo R., Fini L., Materazzi M.** Raman spectroscopy for water temperature sensing // Laser Physics. 1999. V. 9 (1). P. 422—425.

18. **Furic K., Ciglenecki I., Cosovic B.** Raman spectroscopic study of sodium chloride water solutions // J. Mol. Str. 2000. N 550—551. P. 225— 234.

19. **Беккиев А. Ю., Гоголинская Т. А., Фадеев В. В.** Одновременное определение температуры и солености морской воды методом лазерной КР спектроскопии // Докл. АН СССР. 1983. Т. 271, № 4. С. 849—853.

20. **Горшкова О. М., Пацаева С. В., Федосеева Е. В., Шубина Д. М., Южаков В. И.** Флуоресценция растворенного органического вещества природной воды // Вода: химия и экология. 2009. № 11. С. 31— 39.

21. **Gerdova I. V., Churina I. V., Dolenko S. A., Dolenko T. A., Fadeev V. V., Persiantsev I. G.** New Opportunities in Solution of Inverse Problems in Laser Spectroscopy Due to Application of Artificial Neural Networks // Proc. SPIE. 2002. V. 4749. P. 157—166.

22. **Dolenko T. A., Burikov S. A., Sabirov A. R., Fadeev V. V.** Remote determination of temperature and salinity in consideration of dissolved organic matter in natural waters using laser spectroscopy // In: 5th EARSeL Workshop on Coastal Zones. 2011. V. 10 (2). P. 159—165.

23. **Specht D.** A General Regression Neural Network // IEEE Trans. on Neural Networks. 1991. V. 2 (6). P. 568—576.

24. **NeuroShell 2.** URL:http://www.wardsystems.com/neuroshell2.asp

УДК 621.67.001.57

А. В. Кретинин, д-р техн. наук, проф.,
e-mail: avk-vrn@mail.ru,
А. А. Бураков, аспирант,
М. И. Кирпичев, канд. техн. наук, доц.,
Воронежский государственный
технический университет

Профилирование лопасти центробежного насоса с использованием нейросетевого алгоритма решения уравнений гидродинамики

В качестве универсального алгоритма численного моделирования гидродинамических процессов в проточной части центробежного насоса предлагается метод взвешенных невязок на базе нейросетевых пробных функций, который является модифицированным интегрально-сопряженным методом решения уравнений математической физики. Разработанный нейросетевой алгоритм использован для профилирования лопасти двойной кривизны.

Ключевые слова: *центробежный насос, лопасть двойной кривизны, нейросетевое профилирование*

Использование искусственных нейронных сетей (ИНС) для исследования гидродинамических процессов может быть представлено двумя принципиально разными подходами. Первый заключается в нейросетевой обработке результатов натурного либо вычислительного эксперимента для формирования факторных моделей на основе нейросетевой вычислительной архитектуры. В этом случае нейронная сеть выполняет роль нелинейного аппроксиматора, т. е. используется в своем самом многочисленном приложении. Второй заключается в использовании численного нейросетевого метода взвешенных невязок (НМВН) на основе нейросетевых пробных функций для непосредственного решения дифференциальных уравнений гидродинамики. Суть метода состоит в модификации выражения для функционала качества таким образом, что ошибка работы сети оценивается как суммарная невязка решаемых уравнений в произвольных точках расчетной области, число и координаты которых меняются на каждой итерации. Данный подход также известен и описан в работах [1—4]. Модификация данного метода для задач проектирования заключается в использовании рассчитанных координат линий тока как образующих граничных поверхностей проточной части изделия. Для примера рассмотрим процесс профилирования пространственной лопасти магистрального центробежного нефтяного насоса МНН 7500.249 [5].

Рассматривается задача в двумерной постановке. Используется декартова система координат. Для моделирования течения в идеализированном центробежном колесе может использоваться модель, изображенная на рис. 1. Подвод жидкости происходит по внутреннему кругу вращающегося кольца нормально к границе. Совместно с вращающейся областью рассматривается прилегающая неподвижная область произвольного размера. Уравнения математической модели записаны в тензорном виде: уравнение неразрывности

$$\frac{\partial u_j}{\partial x_j} = 0, \qquad (1)$$

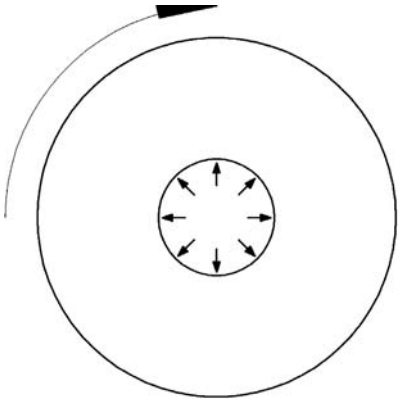


Рис. 1. Вращающаяся зона

где u_j — компонента скорости; x_j — пространственная координата, и уравнения количества движения

$$\frac{\partial}{\partial x_j}(\overline{u_i u_j}) + \frac{\partial}{\partial x_j}(\overline{u_i' u_j'}) =$$

$$= -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)\right] + f_i, \qquad (2)$$

где p — давление; μ — динамический коэффициент вязкости; f_j — компонента равнодействующей центробежных и кориолисовых сил; $\overline{u}_i$ — осредненное значение i-й компоненты скорости; u_i' — значение пульсационной составляющей i-й компоненты скорости.

Течение во вращающейся зоне рассматривается в относительной системе отсчета [6]. В векторном виде

$$f_i = -\rho(2\omega \times u + \omega \times (\omega \times r)),$$

где ρ — плотность; ω — частота вращения; u — вектор скорости; r — радиус-вектор.

Для моделирования турбулентности используется k—ε-модель турбулентности. Под решением уравнений модели будем понимать нейросетевые функции $\overline{u}_i$, p, k, $\varepsilon = f_{NN}(w, x, y)$, где w — вектор всех весов и порогов сети, доставляющие минимумы суммарных квадратических невязок по каждому решаемому уравнению в произвольной совокупности расчетных узлов, координаты которых на каждой итерации обучения нейросетевого решения генерируются с использованием датчика случайных чисел [7]. Результаты решения представлены на рис. 2 и 3.

Используем полученные линии тока в качестве образующих для профилей лопаток центробежного колеса. Длина лопатки, углы установки на входе и выходе будут зависеть от частоты вращения кольца модели. Предположим, что профиль лопатки определяется зависимостью декартовых координат точек профиля x, $y = f_{NN}(\omega, D/D_1)$, где D/D_1 — отношение диаметра, соответствующего расположению точки лопатки с координатами (x, y), к диаметру входа в модель, причем $x = f_{NN}(\omega, 1) = 0$, а $y = f_{NN}(\omega, 1) = D_1/2$.

Можно провести серию вычислительных экспериментов для определения нейросетевой зависимости профиля лопатки от частоты вращения, значение которой будет соответствовать различным углам установки лопасти, следовательно, различным углам атаки. В самом деле, пусть ω_* — номинальная частота вращения, тогда угол установки лопатки на входе в случае нулевого угла атаки $i = 0$ ("безнапорная" лопатка) будет

$$\beta_1 = \operatorname{arctg}\left(\frac{2u_1}{\omega_* D_1}\right),$$

где $u_1 = \dfrac{Q}{\pi D_1 b}$ — скорость на входе в расчетную область; Q — объемный расход; b — ширина отвода на выходе из колеса.

Далее в зависимости от принимаемого угла атаки рассчитывается соответствующая частота вращения кольца модели $\omega = \dfrac{2}{D_1}\dfrac{u_1}{\operatorname{tg}(\beta_1 + i)}$, проводится расчет линий тока с использованием уравнений неразрывности и импульса методом НМВН, и данные линии тока используются как образующие лопаток,

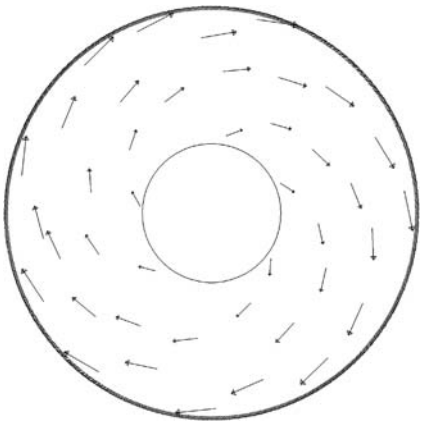


Рис. 2. Векторное распределение скорости на одной из итераций НМВН

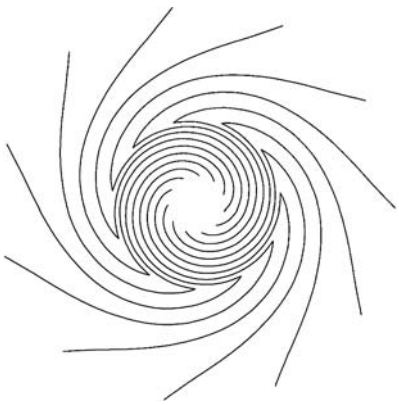


Рис. 3. Линии тока

координаты которых заносятся в статистическую базу данных для построения нейросетевой зависимости x, $y = f_{NN}(\omega, D/D_1)$. На рис. 4 приведены по одной линии тока, построенных для номинальной частоты вращения ω_* (*1*) и $\omega = 0{,}5\omega_*$ (*2*).

Изложенный метод легко обобщить на трехмерный случай. При этом можно использовать геометрическую модель, изображенную на рис. 5, т. е. вместо вращающегося кольца для 2D-модели используется вращающийся "жидкий" цилиндр с определенной частотой вращения, которую можно варьировать. Подвод жидкости осуществляется симметрично с противоположных торцев цилиндра. Таким образом моделируется центробежный насос с двумя входами.

Полученное решение уравнений гидродинамики в виде нейросетевой функции (т. е., по сути, в аналитическом виде) позволяет использовать поверхности тока в качестве образующих пространственных лопастей центробежного колеса. В частности, мы можем получить вид для поверхности тока такой, что выходная кромка данной поверхности будет параллельна оси вращения. На рис. 6 изображено семь таких лопастей, спрофилированных для рабочего колеса магистрального насоса МНН 7500.249.

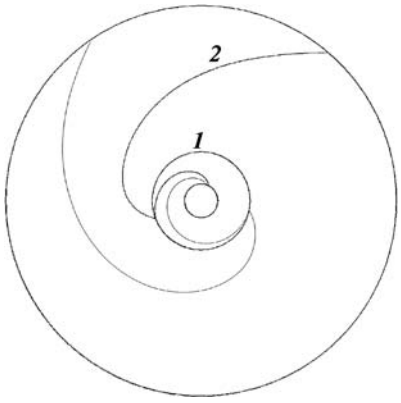


Рис. 4. Линии тока, соответствующие различным частотам вращения

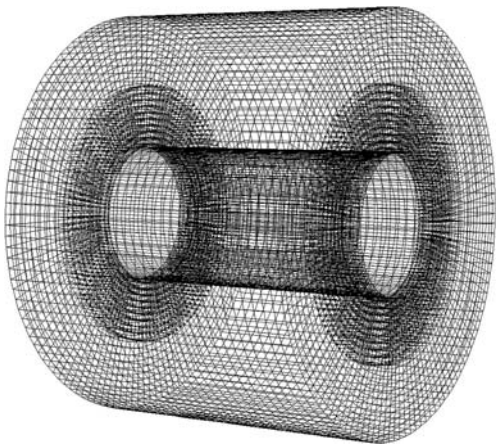


Рис. 5. Геометрическая модель вращающегося "жидкого" цилиндра

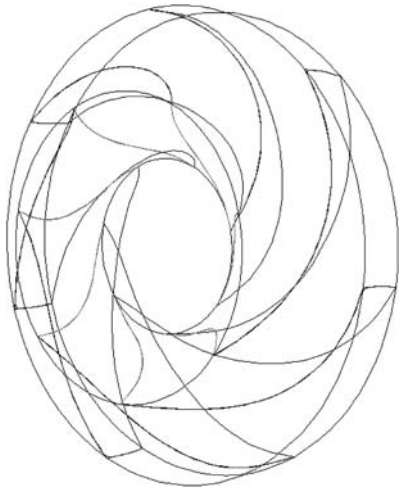


Рис. 6. Рабочее колесо центробежного насоса с лопатками, построенными по линиям тока

Применение теоретических исследований гидродинамических процессов в проточной части турбомашин на практике определяется возможностью создания качественных расчетных методик. Гидродинамические процессы описываются законами в виде дифференциальных уравнений различной сложности, как правило, не имеющих аналитического решения. Традиционные источники возникновения погрешностей числовых результатов, полученных с использованием широко распространенных конечно-разностных либо конечно-элементных методов моделирования, такие как погрешности дискретизации, недостаточная аппроксимационная мощность функций решения, низкая точность расчета особенностей и границ, отчасти сдерживают использование математического моделирования рабочих процессов на микроуровне. Несмотря на обширное использование в настоящее время в научных исследованиях современных "тяжелых" конечно-элементных пакетов применение полученных с их помощью результатов на практике и внедрение их в процессы проектирования наталкиваются на определенное сопротивление инженеров в связи с отсутствием доверия к численным алгоритмам и невозможностью контроля получения результатов.

В качестве универсального численного алгоритма моделирования рабочих процессов трубопроводного транспорта предлагается модифицированный интегрально-сопряженный численный метод решения уравнений математической физики методом взвешенных невязок на базе нейросетевых пробных решений. Сущность метода заключается в подборе параметров глобального нейросетевого пробного решения для минимизации суммарной невязки решаемых уравнений в произвольно расположенных расчетных точках. Применение НМВН позволяет устранить погрешности решения дифференциальных уравнений, вызванные дискретизацией производных и низкой точностью представления границ, что повышает адекватность моделирования.

НМВН служит для настройки параметров нейросетевых моделей, когда имеющейся экспериментальной информации об исследуемых явлениях недостаточно, но известны физические законы, описываемые соответствующими уравнениями, что является основой создания информационных баз данных физических процессов со встроенным нейросетевым алгоритмом поиска решений методом НМВН, открытых для уточнения и постоянной идентификации на основе появляющихся новых экспериментальных знаний.

Применение НМВН позволяет моделировать произвольные физические процессы в едином нейросетевом логическом базисе, при этом численные алгоритмы отличаются логической простотой, ясностью и прозрачностью получения числовых результатов. Эволюционные методы моделирования с момента их появления рассматриваются как некая альтернатива традиционным фундаментальным подходам научных исследований, и использование их при проектировании можно назвать в большой степени новым и нетрадиционным научным подходом. Органичное сочетание его с современной технологией оптимизации позволит получить принципиально новые важные практические результаты.

Список литературы

1. **Кретинин А. В.** Метод взвешенных невязок на базе нейросетевых пробных функций для моделирования задач гидродинамики // Сиб. журн. вычисл. математики. РАН. Сиб. отдние. 2006. Т. 9. № 1. С. 23—35.

2. **Kretinin A. V., Bulygin Yu. A. & Kirpichev M. I.** Method of Weighted Residuals on the Base of Neuronet's Approximations for Computer Simulation of Hydrodynamics Problems // Proceedings of IEEE 6th International Conference on Computational Cybernetics ICCC 2008. Stara Lesná, Slovakia, 2008. P. 237—240.

3. **Kretinin A. V., Bulygin Yu. A., Kirpichev M. I. & Darneva T. I.** Neuronet's method of weighted residuals for computer simulation of hydrodynamics problems // Proceedings of 2010 International Joint Conference on Neural Networks. Barcelona, Spain, 2010.

4. **Valyuhov S., Kretinin A. and Burakov A.** Neural Network Modeling of Hydrodynamics Processes, Hydrodynamics — Optimizing Methods and Tools / H. E. Schulz (ed.). 2011. URL: http://www.intechopen.com/articles/show/title/neural-network-modeling-of-hydrodynamics-processes

5. **Валюхов С. Г., Кретинин А. В.** Математическое моделирование гидродинамических процессов в проточной части центробежного насоса с использованием нейросетевых алгоритмов // Насосы. Турбины. Системы. 2011. № 1. С. 53—60.

6. **Флетчер К.** Вычислительные методы в динамике жидкостей. М.: Мир, 1991.

7. **Кретинин А. В., Булыгин Ю. А., Волгин В. А.** Использование динамических расчетных сеток в нейросетевом алгоритме взвешенных невязок для моделирования гидродинамических задач // Нейрокомпьютеры: разработка, применение. 2007. № 9. С. 33—39.

УДК 621.391

С. Н. Данилин[1], канд. техн. наук, доц.,
С. В. Пантелеев[2], канд. техн. наук, зав. каф.
[1] Муромский институт ВлГУ
им. А. Г. и Н. Г. Столетовых,
e-mail:dsn-55@mail.ru
[2] Выксунский филиал НИТУ "МИСиС",
e-mail: s@w52.ru

Алгоритм контроля отказоустойчивости нейронных сетей

Сформулирован общий подход к разработке методов количественного определения уровня отказоустойчивости нейронных сетей произвольной структуры и назначения. Предложено определение термина "отказоустойчивость" технических объектов. На основе общего подхода разработан вариант количественного критерия уровня отказоустойчивости нейронных сетей. Исследована отказоустойчивость пяти нейронных сетей прямого распространения. Изучена зависимость результата определения уровня отказоустойчивости от выбранного показателя качества работы нейронной сети.

Ключевые слова: *нейронные сети, отказоустойчивость, критерий отказоустойчивости, качество работы нейронной сети*

К нерешенным задачам в теории нейронных сетей относится количественное определение отказоустойчивости, как одного из важнейших свойств технических объектов [1]. Аналитический обзор отечественных и зарубежных научно-технических публикаций по проблемам отказоустойчивости устройств с нейросетевой архитектурой или работающих в нейросетевом логическом базисе (искусственных нейронных сетей) позволяет сделать вывод, что до настоящего времени не разработаны теоретические или экспериментальные методы определения отказоустойчивости нейронных сетей произвольной архитектуры. Известные методы имеют много недостатков, применимы в основном для анализа частных типов нейронных сетей, трудно сопоставимы между собой и практически не согласуются с действующими как российскими, так и международными стандартами в области проектирования технических объектов. В известных работах не сообщается о количественной оценке отказоустойчивости технических объектов или алгоритмов обработки информации.

Понятие отказоустойчивости в настоящее время трактуется неоднозначно. Согласно одной точки зрения, отказоустойчивость — это свойство объекта, позволяющее ему продолжать работу в случае возникновения отказов какой-либо из его частей. Согласно другой точки зрения, отказоустойчивость

является свойством объекта, характеризующим его надежность.

Авторами предложено уточненное определение отказоустойчивости: "свойство технического объекта сохранять требуемое качество (точность) работы в пределах заданных допусков при любых вариациях параметров элементов или структур под воздействием внутренних или внешних факторов".

Сформулированная точка зрения позволяет предложить общий подход к разработке методов определения количественных критериев отказоустойчивости искусственных нейронных сетей произвольной архитектуры и назначения, предполагающий:

а) получение информации об изменениях любого из показателей качества их функционирования путем математического моделирования вариаций параметров нейронов (или его элементов);

б) расчет количественных значений уровня отказоустойчивости нейронной сети по каждому ее элементу;

в) расчет количественных значений уровня отказоустойчивости нейронной сети по обобщающим показателям;

г) визуализации процесса определения отказоустойчивости нейронных сетей в процессе их анализа или синтеза.

На основе общего подхода разработан вариант критерия U_i определения количественного уровня отказоустойчивости нейронных сетей произвольной структуры и назначения по каждому i-му ее нейрону или элементу (устойчивости к отказу каждого i-го ее нейрона или элемента):

$$U_i = 1 - (x_i - x_{\text{дос}})/(x_{\text{доп}} - x_{\text{дос}}), \quad \text{при } x_{\text{доп}} > x_{\text{дос}}; \qquad (1)$$

$$U_i = 1 - (x_{\text{дос}} - x_i)/(x_{\text{дос}} - x_{\text{доп}}), \quad \text{при } x_{\text{доп}} < x_{\text{дос}}; \qquad (2)$$

где $x_{\text{доп}}$ — допустимое значение (допуск) показателя качества работы нейронной сети; $x_{\text{дос}}$ — значение показателя качества работы нейронной сети, достигнутое при обучении; x_i — значение показателя качества работы нейронной сети при вариации параметра i-го нейрона или элемента сети.

Для количественной оценки общего уровня отказоустойчивости нейронной сети вычисляется средний уровень отказоустойчивости $U_{\text{ср}}$, определяемый выражением

$$U_{\text{ср}} = \frac{1}{N} \sum_{i=1}^{N} U_i, \qquad (3)$$

где N — число нейронов (структурных элементов) в нейронной сети, и относительный уровень отказоустойчивости $U_{\text{отн}}$, определяемый выражением

$$U_{\text{отн}} = N_{\text{ОУ}}/N_{\text{общ}}, \qquad (4)$$

Таблица 1

Отказоустойчивость нейронных сетей, реализующих аппроксимацию функций $y = e^{3x}$ и $y = \text{tg}(x)$ для отказов нейронов типа "0"

Аппроксимация функции $y = e^{3x}$			Аппроксимация функции $y = \text{tg}(x)$		
Номер отказавшего нейрона	Значение MSE	Значение U_i	Номер отказавшего нейрона	Значение MSE	Значение U_i
1	$3{,}73 \cdot 10^{-10}$	0,7597	1	$1{,}75 \cdot 10^{-7}$	−4540,3
2	0,19165	$-6{,}37 \cdot 10^{+8}$	2	0,00056	$-1{,}45 \cdot 10^{+7}$
3	8,0568	$-2{,}68 \cdot 10^{+10}$	3	0,0121	$-3{,}15 \cdot 10^{+8}$
4	55,212	$-1{,}83 \cdot 10^{+11}$	4	0,3336	$-8{,}67 \cdot 10^{+9}$
5	55,361	$-1{,}84 \cdot 10^{+11}$	5	0,5575	$-1{,}45 \cdot 10^{+10}$
$U_{\text{ср}}$	$-7{,}9009 \cdot 10^{+10}$		$U_{\text{ср}}$	$-4{,}6965 \cdot 10^{+9}$	
$U_{\text{отн}}$	0,2		$U_{\text{отн}}$	0	
Номер отказавшего нейрона	Значение MAE	Значение U_i	Номер отказавшего нейрона	Значение MAE	Значение U_i
1	$1{,}52 \cdot 10^{-5}$	0,93199	1	0,0003	−56,308
2	0,21268	−14888	2	0,0153	−2971,9
3	1,7733	$-1024 \cdot 10^{+5}$	3	0,0638	−12363
4	5,3531	$-3{,}75 \cdot 10^{+5}$	4	0,455	−88119
5	5,3623	$-3{,}75 \cdot 10^{+5}$	5	0,6157	$-1{,}19 \cdot 10^{+5}$
$U_{\text{ср}}$	$-1{,}7785 \cdot 10^{+5}$		$U_{\text{ср}}$	$-4{,}4547 \cdot 10^{+4}$	
$U_{\text{отн}}$	0,2		$U_{\text{отн}}$	0	
Номер отказавшего нейрона	Значение SSE	Значение U_i	Номер отказавшего нейрона	Значение SSE	Значение U_i
1	$3{,}73 \cdot 10^{-6}$	0,7597	1	0,0017	−4540,3
2	1916,7	$-6{,}37 \cdot 10^{+8}$	2	5,5902	$-1{,}45 \cdot 10^{+7}$
3	80576	$-2{,}68 \cdot 10^{+10}$	3	121,48	$-3{,}15 \cdot 10^{+8}$
4	$5{,}52 \cdot 10^{+5}$	$-1{,}83 \cdot 10^{+11}$	4	3336,5	$-8{,}67 \cdot 10^{+9}$
5	$5{,}53 \cdot 10^{+5}$	$-1{,}84 \cdot 10^{+11}$	5	5575,5	$-1{,}45 \cdot 10^{+10}$
$U_{\text{ср}}$	$-7{,}9009 \cdot 10^{+10}$		$U_{\text{ср}}$	$-4{,}6965 \cdot 10^{+9}$	
$U_{\text{отн}}$	0,2		$U_{\text{отн}}$	0	
Номер отказавшего нейрона	Значение $\varepsilon_{\text{абс.max}}$	Значение U_i	Номер отказавшего нейрона	Значение $\varepsilon_{\text{абс.max}}$	Значение U_i
1	0,00012	0,93141	1	0,0007	−16,247
2	1,7807	−15445	2	0,0594	−1520
3	8,7395	−75814	3	0,3668	−9401,8
4	19,068	$-1{,}65 \cdot 10^{+5}$	4	1,3022	−33387
5	19,086	$-1{,}66 \cdot 10^{+5}$	5	1,5574	−39930
$U_{\text{ср}}$	$-8{,}4447 \cdot 10^{+4}$		$U_{\text{ср}}$	$-1{,}6851 \cdot 10^{+4}$	
$U_{\text{отн}}$	0,2		$U_{\text{отн}}$	0	

где $N_{ОУ}$ — число отказоустойчивых нейронов (структурных элементов) в нейронной сети; $N_{общ}$ — общее число нейронов (структурных элементов) в нейронной сети.

Критерий U_i [2] изменяется в диапазоне от $-\infty$ до 1. Если значение показателя качества работы нейронной сети x_i при каких-либо изменениях параметров ее составных элементов становится ниже допустимого уровня $x_{доп}$ (происходит отказ), то нейронная сеть не является отказоустойчивой. В этом случае коэффициент U_i становится отрицательным и его абсолютное значение, увеличенное на единицу, показывает во сколько раз изменение значения показателя качества работы нейронной сети при наступлении отказа соответствующего элемента (нейрона) превышает допустимое изменение. Чем ближе значение коэффициента к 1, тем более отказоустойчива нейронная сеть.

Для повышения отказоустойчивости нейронной сети необходимо либо увеличивать разницу между допустимым $x_{доп}$ и достигнутым $x_{дос}$ при обучении значениями показателя качества работы, либо снижать изменение показателя качества работы нейронной сети при отказе каждого структурного элемента (его физической или информационной составляющей) за счет уменьшения значения соответствующего коэффициента влияния.

Таблица 2

Отказоустойчивость нейронных сетей, реализующих аппроксимацию функций $y = \mathbf{e}^{3x}$ и $y = \mathrm{tg}(x)$ для отказов нейронов типа "1"

Аппроксимация функции $y = \mathbf{e}^{3x}$			Аппроксимация функции $y = \mathrm{tg}(x)$		
Номер отказавшего нейрона	Значение MSE	Значение U_i	Номер отказавшего нейрона	Значение MSE	Значение U_i
1	$3{,}12 \cdot 10^{-10}$	0,96283	1	$1{,}75 \cdot 10^{-7}$	−4536,2
2	0,19176	$-6{,}37 \cdot 10^{+8}$	2	0,00056	$-1{,}46 \cdot 10^{+7}$
3	7,117	$-2{,}37 \cdot 10^{+10}$	3	0,01188	$-3{,}09 \cdot 10^{+8}$
4	60,236	$-2 \cdot 10^{+11}$	4	0,54433	$-1{,}41 \cdot 10^{+10}$
5	59,18	$-1{,}97 \cdot 10^{+11}$	5	0,52329	$-1{,}36 \cdot 10^{+10}$
$U_{ср}$	$-8{,}4264 \cdot 10^{+10}$		$U_{ср}$	$-5{,}6123 \cdot 10^{+9}$	
$U_{отн}$	0,2		$U_{отн}$	0	
Номер отказавшего нейрона	Значение MAE	Значение U_i	Номер отказавшего нейрона	Значение MAE	Значение U_i
1	$1{,}47 \cdot 10^{-5}$	0,994	1	0,0003	−56,28
2	0,21287	−14901	2	0,0155	−2995,3
3	1,6345	$-1{,}14 \cdot 10^{+5}$	3	0,0629	−12175
4	5,7353	$-4{,}01 \cdot 10^{+5}$	4	0,6087	$-1{,}18 \cdot 10^{+5}$
5	5,6645	$-3{,}96 \cdot 10^{+5}$	5	0,5929	$-1{,}15 \cdot 10^{+5}$
$U_{ср}$	$-1{,}8549 \cdot 10^{+5}$		$U_{ср}$	$-4{,}9585 \cdot 10^{+4}$	
$U_{отн}$	0,2		$U_{отн}$	0	
Номер отказавшего нейрона	Значение SSE	Значение U_i	Номер отказавшего нейрона	Значение SSE	Значение U_i
1	$3{,}12 \cdot 10^{-6}$	0,96283	1	0,0017	−4536,2
2	1917,8	$-6{,}37 \cdot 10^{+8}$	2	5,6418	$-1{,}46 \cdot 10^{+7}$
3	71177	$-2{,}37 \cdot 10^{+10}$	3	118,81	$-3{,}09 \cdot 10^{+8}$
4	$6{,}02 \cdot 10^{+5}$	$-2 \cdot 10^{+11}$	4	5443,9	$-1{,}41 \cdot 10^{+10}$
5	$5{,}92 \cdot 10^{+5}$	$-1{,}97 \cdot 10^{+11}$	5	5233,5	$-1{,}36 \cdot 10^{+10}$
$U_{ср}$	$-8{,}4264 \cdot 10^{+10}$		$U_{ср}$	$-5{,}6123 \cdot 10^{+9}$	
$U_{отн}$	0,2		$U_{отн}$	0	
Номер отказавшего нейрона	Значение $\varepsilon_{абс.max}$	Значение U_i	Номер отказавшего нейрона	Значение $\varepsilon_{абс.max}$	Значение U_i
1	0,0001163	0,99072	1	0,0007	−16,235
2	1,781	−15448	2	0,0595	−1524,7
3	8,3378	−72329	3	0,3638	−9324,9
4	19,465	$-1{,}69 \cdot 10^{+5}$	4	1,5395	−39470
5	19,361	$-1{,}68 \cdot 10^{+5}$	5	1,5238	−39068
$U_{ср}$	$-8{,}4917 \cdot 10^{+4}$		$U_{ср}$	$-1{,}7881 \cdot 10^{+4}$	
$U_{отн}$	0,2		$U_{отн}$	0	

Исследуем отказоустойчивость нейронных сетей, реализующих аппроксимацию функций (преобразование информации): $y = \mathrm{tg}(x)$ и $y = \mathbf{e}^{3x}$. Для этого создадим двухслойные нейронные сети прямого распространения с пятью нейронами в первом слое (тангенциальная функция активации) и одним нейроном во втором (линейная функция активации).

Определение отказоустойчивости проводим по следующей методике:

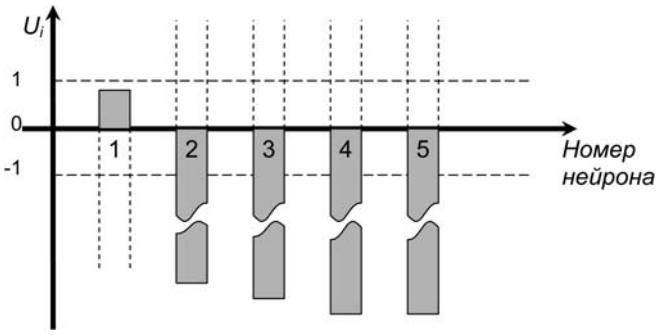
а) обучаем нейронную сеть до достижения наилучшего результата по критерию суммы квадратов ошибок (SSE);

б) фиксируем полученные значения критериев: среднюю квадратическую ошибку (MSE), среднюю абсолютную ошибку (MAE), максимальную абсолютную $\varepsilon_{абс.max}$ или относительную $\varepsilon_{отн.max}$ ошибку (погрешность) работы сети или решения задачи [3];

в) моделируем критические отказы нейронов ("0" или "1");

г) фиксируем полученные при моделировании значения критериев SSE, MSE, MAE, $\varepsilon_{абс.max}$;

д) по выражениям (1)—(3) при соотношении $x_{доп}/x_{дос} = 2$ рассчитываем значения уровня отказоустойчивости (ОУ) нейронных сетей U_i, $U_{ср}$, $U_{отн}$, которые сводим в табл. 1, 2 и пред-



Уровень отказоустойчивости U_i нейронной сети, реализующей аппроксимацию функции $y = \mathbf{e}^{3x}$ для отказов нейронов типа "0" по значению MSE

ставляем графически в виде диаграмм (пример на рисунке).

Уровни отказоустойчивости U_i нейронной сети, аппроксимирующей функцию $y = \mathbf{e}^{3x}$, рассчитанные по значениям всех показателей качества работы, различаются между собой, но однозначно определяют нейронную сеть как устойчивую к отказам первого нейрона, $U_{\text{отн}} = 0{,}2$ (см. рисунок).

Уровни отказоустойчивости U_i нейронной сети, аппроксимирующей функцию $y = \text{tg}(x)$, рассчитанные по значениям всех показателей качества обучения, различаются между собой, но однозначно определяют нейронную сеть как не устойчивую к отказам всех нейронов $U_{\text{отн}} = 0$.

Авторы ранее исследовали по критерию (1) отказоустойчивость нейронных сетей одинаковой архитектуры (семь нейронов в первом слое) преобразования информации (аппроксимации функций) $y = \sqrt{x}$, $y = \mathbf{e}^x$, $y = 1/x$ при различных показателях качества работы и получили следующие результаты [4].

Уровень отказоустойчивости U_i нейронных сетей, аппроксимирующих функции $y = 1/x$, $y = \sqrt{x}$, рассчитанный по значениям абсолютной погрешности $\varepsilon_{\text{абс.max}}$, определяет ее как устойчивую к отказам четырех нейронов, а рассчитанный по значениям других критериев — SSE, MSE, MAE — к отказам только одного.

Уровень отказоустойчивости U_i нейронной сети, аппроксимирующей функцию $y = \mathbf{e}^x$, рассчитанный по значениям всех выбранных критериев, определяет ее как устойчивую к отказам только одного нейрона.

Таким образом, уровень отказоустойчивости U_i имеет при отказах одних и тех же нейронов каждой нейронной сети различные значения в зависимости от выбранного критерия (показателя) качества (точности) работы нейронной сети.

Количественный критерий отказоустойчивости U позволяет получить дополнительную информацию о свойствах нейронных сетей и может быть рекомендован для применения как в теоретических исследованиях, так и в инженерной практике.

Работа выполнена при поддержке гранта РФФИ № 11-08-97551.

Список литературы

1. **Галушкин А. И.** Теория нейронных сетей. М.: ИПРЖР, 2000. 416 с.

2. **Данилин С. Н., Пантелеев С. В.** Контроль отказоустойчивости нейронных сетей. Методы и устройства передачи и обработки информации. Вып. 7 / Под ред. В. В. Ромашова, В. В. Булкина. С.-Петербург: Гидрометеоиздат, 2006. С. 177—181.

3. **Медведев В. С., Потемкин В. Г.** Нейронные сети. MATLAB 6 / Под общ. ред. В. Г. Потемкина. М.: ДИАЛОГ-МИФИ, 2002. 496 с.

4. **Данилин С. Н., Пантелеев С. В.** Исследование отказоустойчивости нейронных сетей // Методы и устройства передачи и обработки информации. Вып. 8. С.-Петербург: Гидрометеоиздат, 2007. С. 167—173.

CONTENTS

We present an approach to use the distributed full platform simulator Simics for collecting traces of MPI functions for applications for the offline analysis. The experimental setup and results of analysis of traces for the Linpack benchmark implementation HPL are given.

Keywords: distributed simulation, Simics, multi core systems, network performance, tracing, cluster, MPI, Linpack

New formalisms for the description of models of asynchronous logic control processes and resources in systems and storage networks and data are proposed. Formalisms aims is to unify the representation of processes, to increase flexibility and scalability of distributed control programs. The focus is on the formal description of the processes and their properties based on the concept of coordinations processes through intensive interactions updated databases and knowledge bases.

Keywords: distributed systems, distributed computing, program agents, formal logic-algebraic models, databases and knowledge bases, networks of abstract machines, asynchronous predicate and predicate-functional networks

This article is devoted to development of the method of neural network solution of the problem of simultaneous determination of temperature and salinity of seawater by Raman spectra. In this article, the development of the method was continued by using significant input feature selection. Comparison of different ways of significant input feature selection and their influence on the error of determination of temperature and salinity are presented.

Keywords: neural networks, inverse problems, significant input feature selection, Raman spectroscopy

A modified integral-conjugate numerical method of the mathematical physics equations solution by the weighted residual method based on the learning neuronet functions is proposed as a universal numerical modeling algorithm of the hydrodynamic processes in the centrifugal pump flow part. The designing neuronet algorithm of the bicurvature special blade is formed.

Keywords: centrifugal pump, bicurvature blade, neuronet designing

A general approach to the development of methods for the quantitative determination of the level of neural networks fault tolerance of arbitrary structure and function. A definition of the term "fault tolerance" of technical objects. Based on the general approach developed version of a quantitative criterion for the fault tolerance level of neural networks. Studied fault tolerance of five neurons neural network of back propagation. The dependence of the result of determining the level of fault tolerance on the selected quality index of the neural network.

Keywords: neural networks, fault-tolerance, criterion of fault tolerance, work quality of the neural network

Адрес редакции:

107076, Москва, Стромынский пер., 4

Телефон редакции журнала **(499) 269-5510**

E-mail: it@novtex.ru

Дизайнер *Т.Н. Погорелова*. Технический редактор *Е. В. Конова*.

Корректор *Т.В. Пчелкина*.

Сдано в набор 07.11.2012. Подписано в печать 17.12.2012. Формат 60×88 1/8. Бумага офсетная.

Усл. печ. л. 8,86. Заказ IT113. Цена договорная.

Журнал зарегистрирован в Министерстве Российской Федерации по делам печати, телерадиовещания и средств массовых коммуникаций.

Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".

105120, г. Москва, ул. Нижняя Сыромятническая, д. 5/7, стр. 2, офис 2.