ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Том 21
2015
№ 2

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.

INFORMATION TECHNOLOGIES
INFORMACIONNYE TEHNOLOGII

Vol. 21
2015
No. 2

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

Published since November 1995

ISSN 1684-6400

CONTENTS

Complete Internet version of the journal at site: **http://novtex.ru/IT.**
According to the decision of the Higher Certifying Commission of the Ministry of Education of Russian Federation, the journal is inscribed in "The List of the Leading Scientific Journals and Editions wherein Main Scientific Results of Theses for Doctor's or Candidate's Degrees Should Be Published"

УДК 004.414.23

В. П. Кулагин, д-р техн. наук, проф., зам. директора, e-mail: vkulagin@hse.ru
Московский институт электроники и математики Национального исследовательского университета "Высшая школа экономики"

Тензорные методы исследования структур сетей Петри

Описан тензорный подход к исследованию сложных систем, представленных в терминах сетей Петри (СП). Введены понятия различных систем, которые представляют исходную СП-структуру и ее производные в различных системах координат. Показано, что использование тензорных методов существенно упрощает процедуру построения возможных структур исследуемой сложной системы в системе координат примитивной системы, а также делает рутинной процедуру преобразования новых структур сложных систем в исходную систему координат. Описанный подход дает новые возможности для построения методов синтеза структур сложных систем.

Ключевые слова: *тензорные методы, сети Петри, структуры сложных систем, системы координат*

Введение

В последние годы наблюдается растущий интерес к использованию аппарата сетей Петри (СП) [3, 4] при решении задач, связанных с исследованием сложных систем. Еще в работе [5] был дан достаточно подробный обзор методов анализа и синтеза сложных систем и показана роль СП в формализованном описании распределенных структур и их последующем исследовании. За это время круг задач, решаемых с использованием аппарата сетей Петри, только расширился. К актуальным направлениям, развитие которых сопровождается активным использованием СП, можно отнести следующие.

- *Интеллектуальный анализ данных.* Используется для обнаружения в имеющихся данных ранее неизвестных интерпретаций, позволяющих получить новые знания, необходимые для принятия решений. Современные подходы в этой области сосредоточились, с одной стороны, на анализе программного обеспечения [6, 7], а с другой стороны на исследовании таких сложных продуктов как веб-службы [8], агент-ориентированные программные структуры [9, 10], моделирование сложных систем с использованием нейроподобных структур и СП [11].
- *Моделирование сложных динамических систем.* Модульное построение сложных систем существенно упростило соответствующие процедуры описания и последующего анализа. Новое развитие, в этой связи, получили иерархические сети Петри [12, 13], позволяющие использовать более высокий уровень спецификаций, что, в свою очередь, дает возможность проводить более эффективный анализ сложных динамических систем.
- *Проектирование и верификация сложных систем.* Во многих случаях пространство состояний сложных систем может быть напрямую связано с особенностями моделей, которые строятся с использованием различных семантических инструментов (например, сети Петри, алгебра процессов, диаграммы состояний, UML-описания [7, 21], YAWL [22], BPEL [23] и др.). Процесс исследования в этом случае состоит в том, чтобы для некоторой исследуемой структуры, представленной в терминах того или иного формализма, построить наиболее точную модель. Построение подобных моделей в терминах СП и последующий ее анализ являются сегодня достаточно актуальной и востребованной задачей [24—26].
- *Построение сетевых моделей для сенсорных сетей.* Построение и дальнейшее сопровождение сенсорных сетей является сложной задачей, при решении которой необходимо учитывать такие особенности, как: срок службы сенсорных сетей; параллелизм и асинхронность протекающих процессов (что при некорректном управлении может привести к таким явлениям, как тупиковые ситуации и "гонки"); возможная неоднородность узлов сенсорной сети; ограниченность ресурсов и др. В результате возникает проблема, связанная с надежностью обработки критически

важной информации и требующая особых подходов к верификации и валидации сенсорных сетей. Существенная роль в решении данных вопросов отводится СП [27—29].

Исследование, расчет и проектирование сложных систем давно являются одной из основных проблем современной науки. Многообразие объектов, которые называют сложными системами, столь велико, что трудно найти между ними что-то общее, кроме их сложности. Накопленный опыт разработки таких систем показывает, что установилось мнение об индивидуальности каждой сложной системы, необходимости разрабатывать специально для нее теорию, методы расчета и проектирования. Это снижает эффективность разработок, не позволяет использовать уже полученные результаты при разработке новых систем.

Для того, чтобы получить единый подход к исследованию сложных систем любой структуры и различной природы, предназначена тензорная методология. Роль тензорных методов сформулирована в книге академика В. Г. Афанасьева "Общество: системность, познание, управление" [1]: "...Тензорный анализ позволяет отделить субъективное в изучении явления, связанное с позицией ученого, с выбором той или иной системы координат, от объективного, объективной реальности, которая не зависит от точки зрения, от системы координат".

Функции, которые при изменении системы координат преобразуются по линейному закону, Эйнштейн в 1916 году назвал тензорами. Особенности тензорного подхода заключаются в следующем. Во-первых, если обобщенную систему координат рассматривать как тензор, а конкретные системы этого класса — как ее проекции в частных системах координат, то можно использовать главное для тензорного анализа, а именно: надо выбрать одну систему и использовать ее в качестве "эталонной" системы координат, а описания других систем приводить к описанию в терминах этого эталона, т. е. моделирование будет рассматриваться как описание исследуемой модели в заданной эталонной системе координат. Во-вторых, в качестве эталона желательно использовать такие системы, для которых разработаны тензорные методы расчета.

Тензорное исчисление возникло в математике для решения проблем, методологически аналогичных тем, которые решаются при анализе сложных систем. Понятие тензора связано с преобразованием систем координат и является закономерным развитием представлений о пространстве. В данной статье рассматриваются вопросы применения тензорных методов при исследовании сложных систем, представляемых структурами сетей Петри (СП-структурами) [2].

1. Геометрическая интерпретация СП-структур

1.1. Типы геометрий

Известно, что каждая геометрия определяется группой преобразований, оставляющих инвариантными те или иные свойства геометрических фигур. Оказалось, что тензорному анализу сетей соответствует геометрия нового типа, не сводимая к известным геометриям [14, 15]. Например, *евклидова* геометрия допускает преобразования, не меняющие форму фигур. При этом можно переносить, вращать, изменять масштаб фигур. *Афинная* геометрия в дополнение к преобразованиям геометрии Евклида допускает такие деформации фигур, при которых сохраняются условия прямолинейности и параллельности линий. Например, параллелограмм и квадрат в данной геометрии представляют одну и ту же геометрическую фигуру. В *проективной* геометрии в результате преобразований должно сохраняться условие прямолинейности. В *топологии* допустимы почти любые преобразования, но при этом должно сохраняться свойство принадлежности (точки по-прежнему должны принадлежать линиям, линии — поверхностям и т. д.).

Однако при преобразовании структур дискретного пространства, к которым относятся СП-структуры, где важную роль играют операции разрывания и соединения, свойство принадлежности уже не является инвариантом такой группы преобразования. Возникает необходимость рассмотреть новый тип геометрии. В связи с этим возникают два вопроса. 1. Что является инвариантом вновь рассматриваемой группы преобразований, т. е. что остается постоянным при операциях разделения и соединения структур дискретного пространства? 2. В чем заключаются особенности пространства структур по сравнению с уже известными пространствами? Попробуем ответить на эти вопросы применительно к пространству СП-структур.

1.2. Геометрия тензорного анализа СП-структур

Для того чтобы определить геометрию тензорного анализа СП-структур, необходимо определить такие понятия, как *пространство, размерность, система координат, преобразование системы координат*.

Понятия пространства СП-структур и его размерности введены в работе [2]. Рассмотрим понятия системы координат и преобразования системы координат. Для этого напомним некоторые определения и введем новые.

Обозначим через $pre(t)$ и $post(t)$ множества входных и выходных позиций перехода t, а через $pre(p)$ и $post(p)$ — множества входных и выходных переходов позиции p в сети Петри. СП, в которой множество позиций $\mathbf{P} = \{p', p''\}$, множество переходов

$\mathbf{T} = \{t\}$, $p' \in pre(t)$ и $p'' \in post(t)$, назовем *элементарной сетью.*

Примитивной системой $\mathbf{N}_{PR}$ назовем множество элементарных сетей заданной мощности.

СП-структуру, полученную из примитивной системы путем объединения наперед заданного множества вершин, будем называть *эталонной моделью* **N**.

Пусть система координат СП-структур определяется способом соединения элементарных сетей примитивной системы $\mathbf{N}_{PR}$. При этом множество элементарных сетей, составляющих СП-структуру, можно рассматривать как некоторую обобщенную сеть, а любое их соединение в произвольную структуру — как проекцию обобщенной сети в частной системе координат.

Под *преобразованием системы координат* будем понимать переход от одного способа соединения элементарных сетей к другому. В процессе преобразования системы координат выполняются операции разрыва и соединения связей между переходами и позициями СП-структуры [16—18]. При этом количество элементарных сетей, составляющих СП-структуры, представленных в разных системах координат, должно быть одинаковым (постоянство размерности пространства).

Что целесообразно взять в качестве инварианта при тензорном анализе СП? Для этого необходимо определить, какая величина, характеризующая СП-структуру некоторой сложной системы, остается неизменной при переходе от одной системы координат к другой. Будем считать, что способ соединения структурных частей системы не изменяет их исходного состояния и состояния всей системы в целом. Так как исходное состояние системы моделируется в СП-структуре начальной разметкой, то сделаем предположение об инвариантности начальной разметки к способу соединения фрагментов СП-структуры.

Рассмотрим этапы использования тензорных методов при исследовании СП-структур.

1. Рассмотреть физическую, геометрическую и структурную стороны процесса функционирования сложной системы. Получить уравнения, описывающие основные процессы, протекающие в ней.

2. Построить СП-структуру, моделирующую работу системы.

3. Привести полученные уравнения к тензорному виду. Для этого необходимо дать геометрическую трактовку задачи: ввести пространство, размерность, определить системы координат и способы их преобразования.

4. Выбрать эталонную СП-структуру для моделирования сложной системы.

5. Построить модель для тензорной формы уравнений рассматриваемой сложной системы.

6. Провести анализ исследуемой сложной системы.

Анализ систем можно проводить двумя способами, которые не исключают друг друга. Если сложность СП-структуры не позволяет рассчитывать ее как единое целое, то можно применить аппарат диакоптики [19] — структурировать СП-структуру путем ввода иерархических переходов [3], исследовать полученные компоненты отдельно, а затем объединить результаты анализа. Полученный результат интерпретируется в термины исходной системы.

1.3. Постулаты обобщений

В работах [19, 20] было показано, что сложные системы не только описываются тем же количеством символов, что и простые системы, но и весь метод рассуждения, используемый в анализе их поведения, соответствует этапам анализа простейших систем. Другими словами, прежде чем исследовать любую сложную систему со многими переменными, необходимо сначала выполнить анализ простой системы с минимальным числом степеней свободы. После этого можно перенести все этапы этого анализа на сложную систему, заменяя каждую величину соответствующей ей *n*-матрицей. Этот прием называется *постулатом первого обобщения* и выражается следующим образом [14]: метод анализа и окончательные уравнения, описывающие поведение сложной физической системы (с *n* степенями свободы), могут быть найдены последовательно при анализе простейшего, но наиболее общего элемента системы при условии, что каждая величина заменяется соответствующей *n*-матрицей. Простейший элемент системы может иметь одну или несколько степеней свободы.

Очевидно, что простое увеличение числа элементов не вводит никаких новых физических явлений, которые не наблюдались бы в простейшем элементе, а следовательно, новые символы, представляющие новые физические понятия, не могут возникнуть при объединении нескольких элементов. Оказывается, что организация множества величин в одну *n*-матрицу и представление последней с помощью одной базовой буквы является чем-то несравненно бóльшим, чем эффективный рабочий прием. Организация множества величин в *n*-матрицу и одновременное введение понятий "преобразование", "инвариантность", "группа" создают условия для получения принципиально новой математической сущности, обладающей такими свойствами, которые отсутствуют в образующих ее строительных блоках. Созидание посредством "организации" новых сущностей из простого набора *n*-матриц и наделение этих новых сущностей новыми свойствами и составляют основную цель тензорного анализа [15].

Постулат второго обобщения является фундаментальным предположением тензорного анализа и состоит в следующем: 1) новая система описывается тем же числом *n*-матриц и того же типа, что и старая система, но отличается от нее численным значением компонент *n*-матриц; 2) уравнение новой

системы, записанное в n-матрицах, имеет тот же вид, что и уравнение старой системы; 3) n-матрицы новой системы могут быть найдены из n-матриц старой системы с помощью рутинного преобразования.

Таким образом, переход от одного способа соединения блоков некоторой системы к другому не требует введения новых n-матриц и изменения формы уравнений. Отличие состоит в том, что n-матрицы, принадлежащие разным уравнениям, имеют разные компоненты.

Операцию перехода от одного способа соединения к другому мы назвали преобразованием системы координат. При решении различных физических проблем часто возникают многочисленные типы систем координат, которые радикально отличаются друг от друга, а физические явления при рассмотрении их в различных типах систем координат описываются уравнениями различного вида. В данном случае понятия *геометрический объект* недостаточно для объединения уравнений различных типов систем, так как в уравнениях движения, описывающих данные системы, появляются производные и дифференциалы, которые не преобразуются как тензоры, поскольку они зависят от типа движения частей, образующих систему. Для того чтобы объединить уравнение движения электродинамических систем независимо от относительного движения материальных или электрических составляющих, все производные и дифференциалы (не являющиеся тензорами) должны быть заменены тензорами так, чтобы каждый символ, каждый геометрический объект в уравнениях стал тензором. Это преобразование проводится в соответствии с *постулатом третьего обобщения*.

2. Тензорная методология исследования СП-структур

2.1. Построение тензорных уравнений

Пусть $\mathbf{d}^-$ и $\mathbf{d}^+$ — векторы, описывающие множество входных и множество выходных позиций перехода t, т. е. $d_i^- > 0$, если $p_i \in pre(t)$ и $d_j^+ > 0$, если $p_j \in post(t)$. Элементы векторов $\mathbf{d}^-$ и $\mathbf{d}^+$ принимают положительные целочисленные значения. Построим вектор $\mathbf{d} = \mathbf{d}^+ - \mathbf{d}^-$. Данный вектор описывает связи перехода t с позициями СП-структуры: $\mathbf{d}_i > 0$, если $p_i \in post(t)$; $\mathbf{d}_i < 0$, если $p_i \in pre(t)$; $\mathbf{d}_i = 0$, если $\mathbf{p}_i \notin ((post(t) \cup pre(t))$, где $i = 1, 2, ..., n$; $n = |\mathbf{P}|$; $\mathbf{P}$ — множество позиций СП-структуры.

Уравнение изменения разметки СП-структуры при срабатывании перехода t можно записать следующим образом:

$$\boldsymbol{\mu}' = \boldsymbol{\mu} + d, \qquad (1)$$

где $\boldsymbol{\mu}$ — старый вектор разметки СП-структуры; $\boldsymbol{\mu}'$ — вектор разметки СП-структуры после срабатывания перехода t.

В соответствии с постулатом первого обобщения уравнение (1) можно представить следующим образом:

$$\boldsymbol{\mu}' = \boldsymbol{\mu} + \mathbf{D} \times \mathbf{f}(\boldsymbol{\sigma}),$$

где $\mathbf{f}(\boldsymbol{\sigma})$ — степень последовательности срабатывания, представляющая собой вектор, каждый элемент которого указывает на число срабатываний соответствующего перехода СП-структуры; $\mathbf{D}$ — матрица инцидентности, составленная из векторов $\mathbf{d}$, построенных для каждого перехода, т. е. $\mathbf{D} = \mathbf{O} - \mathbf{I}$, где $\mathbf{I}$:$\mathbf{P} \times \mathbf{T} \to \{0, 1, 2, ...\}$ и $\mathbf{O}$:$\mathbf{T} \times \mathbf{P} \to \{0, 1, 2, ...\}$, $\mathbf{O}$, $\mathbf{I}$ — матрицы инцидентности.

Если учесть, что СП-структура начинает функционировать от начальной разметки $\boldsymbol{\mu}_0$, то

$$\boldsymbol{\mu} = \boldsymbol{\mu}_0 + \mathbf{D} \times \mathbf{f}(\boldsymbol{\sigma}). \qquad (2)$$

Уравнение (2) в индексной форме имеет следующий вид:

$$\boldsymbol{\mu}^\gamma = \boldsymbol{\mu}_0^\gamma + \mathbf{D}_\beta^\gamma \times \mathbf{f}(\boldsymbol{\sigma})^\beta,$$

где индексы γ и β изменяются от 1 до n и от 1 до m соответственно; $n = |\mathbf{P}|$, $m = |\mathbf{T}|$.

Рассмотрим проекции $\mathbf{N}$ и $\mathbf{N}'$ некоторой СП-структуры в различных системах координат, описываемых матрицами инцидентности $\mathbf{D}_\beta^\gamma$, и $\mathbf{D}_{\beta'}^{\gamma'}$. Пусть переход от одной системы координат к другой определяется тензорами преобразования $\mathbf{C}_{\beta'\gamma}^{\beta\gamma'}$ и $\mathbf{E}_{\beta\gamma'}^{\beta'\gamma}$, т. е.

$$\mathbf{D}_{\beta'}^{\gamma'} = \mathbf{C}_{\beta'\gamma}^{\beta\gamma'} \times \mathbf{D}_\beta^\gamma; \qquad (3)$$

$$\mathbf{D}_\beta^\gamma = \mathbf{E}_{\beta\gamma'}^{\beta'\gamma} \times \mathbf{D}_{\beta'}^{\gamma'}. \qquad (4)$$

На основе второго постулата обобщения можно записать, что

$$\boldsymbol{\mu}_\alpha^\gamma = \boldsymbol{\mu}_{0\alpha}^\gamma + \mathbf{D}_\beta^\gamma \times \mathbf{f}(\boldsymbol{\sigma})_\alpha^\beta; \qquad (5)$$

$$\boldsymbol{\mu}_{\alpha'}^{\gamma'} = \boldsymbol{\mu}_{0\alpha'}^{\gamma'} + \mathbf{D}_{\beta'}^{\gamma'} \times \mathbf{f}(\boldsymbol{\sigma})_{\alpha'}^{\beta'}, \qquad (6)$$

где α, α' — мощности множеств степеней последовательностей срабатывания $\mathbf{f}(\boldsymbol{\sigma})_\alpha^\beta$, $\mathbf{f}(\boldsymbol{\sigma})_{\alpha'}^{\beta'}$ и достижимых разметок $\boldsymbol{\mu}_\alpha^\gamma$ ($\boldsymbol{\mu}_{\alpha'}^{\gamma'}$) в СП-структуре $\mathbf{N}$($\mathbf{N}'$).

Попытаемся получить зависимости между множествами достижимых разметок СП-структур $\mathbf{N}$ и $\mathbf{N}'$. Для этого умножим правую и левую части уравнения (5) на тензор $\mathbf{C}_{\beta'\gamma}^{\beta\gamma'}$:

$$\mathbf{C}_{\beta'\gamma}^{\beta\gamma'} \times \boldsymbol{\mu}_\alpha^\gamma = \mathbf{C}_{\beta'\gamma}^{\beta\gamma'} \times \boldsymbol{\mu}_{0\alpha}^\gamma + \mathbf{C}_{\beta'\gamma}^{\beta\gamma'} \times \mathbf{D}_\beta^\gamma \times \mathbf{f}(\boldsymbol{\sigma})_\alpha^\beta.$$

Учитывая соотношение (3), свойства инварианта $\mathbf{C}^{\beta\gamma'}_{\beta'\gamma} \times \boldsymbol{\mu}^{\gamma}_{0\alpha} = \boldsymbol{\mu}^{\gamma}_{0\alpha} = \boldsymbol{\mu}^{\gamma'}_{0\alpha'}$ и то, что $\mathbf{f}(\boldsymbol{\sigma})^{\beta}_{\alpha} = \mathbf{f}(\boldsymbol{\sigma})^{\beta'}_{\alpha'}$ (эквивалентность последовательностей воздействия на СП-структуры **N** и **N′**), получаем

$$\mathbf{C}^{\beta\gamma'}_{\beta'\gamma} \times \boldsymbol{\mu}^{\gamma}_{\alpha} = \boldsymbol{\mu}^{\gamma'}_{0\alpha'} + \mathbf{D}^{\gamma'}_{\beta'} \times \mathbf{f}(\boldsymbol{\sigma})^{\beta'}_{\alpha'}. \quad (7)$$

Сравнивая выражения (6) и (7), можно получить, что

$$\boldsymbol{\mu}^{\gamma'}_{\alpha'} = \mathbf{C}^{\beta\gamma'}_{\beta'\gamma} \times \boldsymbol{\mu}^{\gamma}_{\alpha}. \quad (8)$$

Аналогичным образом можно получить, что

$$\mathbf{E}^{\beta'\gamma}_{\beta\gamma'} \times \boldsymbol{\mu}^{\gamma'}_{\alpha'} = \boldsymbol{\mu}^{\gamma}_{0\alpha} + \mathbf{D}^{\gamma}_{\beta} \times \mathbf{f}(\boldsymbol{\sigma})^{\beta}_{\alpha} \quad (9)$$

и

$$\boldsymbol{\mu}^{\gamma}_{\alpha} = \mathbf{E}^{\beta'\gamma}_{\beta\gamma'} \times \boldsymbol{\mu}^{\gamma'}_{\alpha'}. \quad (10)$$

2.2. Выбор примитивной системы

Постулат второго обобщения утверждает, что если компоненты геометрических объектов известны для одной частной системы координат, то для любой другой системы координат того же типа (число которых в общем случае бесконечно) компоненты этих геометрических объектов могут быть определены рутинной процедурой с помощью тензоров преобразования. Уравнения поведения, записанные в терминах геометрических объектов, имеют одинаковый вид для всех систем данного типа.

Следует заметить, что совершенно несущественно, какова та частная система (и связанная с ней система координат), в которой заданы геометрические объекты и инвариантное уравнение. Вместе с тем, до того как постулат может быть использован, необходимо, чтобы уравнения поведения в терминах геометрических объектов были уже известны, по крайней мере для одной частной системы, и в этой же системе известны все компоненты всех геометрических объектов. Тогда эта система может быть использована как стартовая точка при анализе других систем того же класса.

Далее, когда нужно анализировать уравнения функционирования большого числа систем, представляется логичным из всего многообразия систем выделить одну частную систему (и связанную с ней систему координат), для которой сравнительно легко можно:

1) определить компоненты различных геометрических объектов, которые необходимы для решения конкретной задачи в численном виде;

2) установить тензоры преобразования, показывающие, чем эта частная система и связанная с ней система координат отличаются от любой другой системы;

3) вычислить новые компоненты различных геометрических объектов, которые должны быть проанализированы во всех других системах.

Эту частную систему и связанную с ней систему координат, выделенную из большого числа подобных систем и используемую как стартовую точку, будем называть в дальнейшем *примитивной системой*.

В общем случае имеется столько примитивных систем, сколько существует различных типов фундаментальных систем, подлежащих анализу, и используемых точек зрения. Кроме того, чем больше деталей системы должно быть рассмотрено, тем более сложную форму принимает примитивная система. Выбор той или иной системы из множества возможных систем в качестве примитивной достаточно произволен и определяется способностью выбранной системы удовлетворять заданным условиям.

Если система используется в качестве стартовой точки для установления характеристик одной или нескольких новых систем, то вычисления носят один и тот же рутинный характер и требуют применения одних и тех же процедур для каждой новой системы, а именно:

- построения тензоров преобразования;
- вычисления новых компонент геометрических объектов с помощью формул преобразования;
- отображения полученных результатов на исходную систему.

С точки зрения простоты исследования сложных СП-структур с помощью более наглядных и удобных проекций в других системах координат заслуживают внимания два вида примитивных систем: система, представляющая собой совокупность линейно-циклических фрагментов ($\mathbf{N}_{\mathrm{L}}$), которые получаются в результате операций деления [17] позиций и переходов исследуемой СП-структуры, и система, представляющая собой множество элементарных переходов ($\mathbf{N}_{\mathrm{PR}}$).

Сравнивая данные примитивные системы, можно отметить следующее: 1) обе системы являются достаточно простыми для исследования; 2) преимуществом системы $\mathbf{N}_{\mathrm{L}}$ по отношению к системе $\mathbf{N}_{\mathrm{PR}}$ является то, что она структурно описывает число параллельно взаимодействующих процессов, а также взаимосвязь последовательных операций в каждом процессе; 3) преимуществом системы $\mathbf{N}_{\mathrm{PR}}$ по отношению к системе $\mathbf{N}_{\mathrm{L}}$ является возможность генерации более мощного множества СП-структур (путем объединения позиций и переходов).

2.3. Декомпозиция СП-структур

Общие положения. Алгоритмы. Часто случается, что некоторую систему строят из нескольких частей. Причем компоненты каждой из отдельных частей уже бывают вычислены в предыдущих исследованиях. В таких случаях можно снова разбить систему на уже встречавшиеся ранее и незнакомые части,

а затем объединить их снова с помощью тензоров преобразования. Таким образом, многие полученные ранее при анализе отдельных систем результаты можно снова использовать уже при анализе результирующей системы.

Подобный метод рассуждения можно использовать не только, когда система состоит из известных частей, но и тогда, когда необходимо анализировать уже существующую сложную систему, которую трудно рассматривать как одно целое. Обычно сложную систему можно разбить на несколько составляющих подсистем (подвергнуть декомпозиции), которые в отдельности легче исследовать и геометрические объекты которых легче получить. После того как каждая составляющая подсистема исследована в отдельности, объединить их в одно целое несравненно легче, чем анализировать исходную систему. В итоге, сущность декомпозиционных методов заключается в разбиении сложной системы на более простые, в исследовании этих простых систем и объединении результатов в целях получения некоторого целостного и единого решения. Идея применения методов декомпозиции в задачах параллельной обработки информации состоит в том, что структурированная тем или иным образом задача представляет собой определенную систему, структура которой является графом общего вида.

Рассмотрим с общих позиций основные положения исчисления систем и декомпозиции. Под *исчислением систем* понимается формализованная теория систем, которая включает в себя формальный логический язык $\mathbf{\Lambda}$, определенную системную структуру $\mathbf{\Theta}$ и набор аксиом $\mathbf{Z}$ данной системной структуры, который дает возможность с помощью логического языка получать теоремы теории систем. Таким образом, исчисление систем— это тройка следующего вида:

$$\mathbf{\Phi} = <\mathbf{\Lambda}, \mathbf{\Theta}, \mathbf{Z}>.$$

Как правило, языком исчисления систем является один из формализованных логических языков, а системной структурой $\mathbf{\Theta}$ с аксиомами $\mathbf{Z}$ — математическая структура определенного рода с некоторой типовой характеристикой. При этом род структуры определяется ее аксиомами, а типовая характеристика — шкалой множеств.

Одним из шагов при построении исчисления систем является установление аксиоматики таких систем, которые могут соединяться между собой. Для этого вводятся входные и выходные терминалы, с помощью которых реализуются соединения систем, индуцирующие разнообразные структурные решения. При этом устанавливаются наиболее общие условия, которым должны удовлетворять соединения, чтобы их можно было отнести к классу операций. Аналоги теоретико-множественных операций (объединение, пересечение) на системах могут быть определены на подсистемах данной системы. При этом подсистемы определяются так, чтобы они имели все характерные свойства системы (наличие входных и выходных терминалов, наличие путей, ведущих от входа к выходу, и др.). Все операции соединения можно считать частным случаем операции Δ-соединения, которая определяется следующим образом. Пусть S_1, ..., S_n — системы, а $b(S_i)$, ..., $b(S_n)$ — множества их терминалов. Определим на терминалах отношение соединения $\Delta \in b(S_i) \times ... \times b(S_n)$, которое описывает соединение терминалов систем S_i, ..., S_n. Обозначим n-арную операцию соединения систем S_i, ..., S_n с помощью отношения соединения Δ через $\Delta[S_i, ..., S_n]$. В том случае, когда $n = 2$, операцию Δ-соединения будем записывать в виде $\Delta[S_1, S_2]$.

Рассмотрим частные случаи бинарной операции Δ-соединения. Операция прямой суммы систем получается в том случае, когда $\Delta = \varnothing$. Это означает, что S_1 и S_2 не имеют общих терминалов. Операция самосоединения определяется в случае, когда $S_1 = S_2 = S$ и $\Delta \in b(S) \times b(S)$. Это означает, что соединяются терминалы, принадлежащие одной системе.

Таким образом, в качестве основной модели исчисления терминальных систем можно выбрать алгебру соединений терминальных систем, т. е. систем, способных к соединению с помощью терминалов. Алгебра соединений дается в виде

$$\mathbf{\Psi} = <\mathbf{W}, \mathbf{V}, \Delta, \mathbf{\Omega}>,$$

где $\mathbf{W}$ — множество всех терминальных систем; $\mathbf{V}$ — множество отношений соединения систем из $\mathbf{W}$; $\mathbf{\Omega}$ — сигнатура операций на отношениях из $\mathbf{V}$.

Исчисление терминальных систем может стать основой разработки различных декомпозиционных методов, поскольку оно дает возможность корректно определить Δ-декомпозицию систем. Будем говорить, что система S поддается Δ-декомпозиции на системы S_1 и S_2, если для $S \in \mathbf{W}$ найдутся такие системы $S_1 \in \mathbf{W}$ и $S_2 \in \mathbf{W}$, и такое отношение $\Delta \in \mathbf{V}$, что будет иметь место равенство $\Delta[S_1, S_2] = S$. Очевидно, что можно рассматривать декомпозицию в более общем плане, а именно Δ-декомпозицию системы S на системы S_1, ..., S_n, которая имеет место в том случае, когда для S найдутся такие системы S_1, ..., S_n и такое отношение Δ, что будет выполняться равенство $S = \Delta[S_1, ..., S_n]$.

Отобразим вышесказанное на аппарат СП-структур. Под системной структурой $\mathbf{\Theta}$ с аксиомами $\mathbf{Z}$ будем понимать СП-структуры и правила их построения. Под множеством входных и выходных терминалов будем понимать множество входных и выходных позиций СП-структур. Модель исчисления терминальных систем $\mathbf{\Psi}$ можно интерпретировать следующим образом: $\mathbf{W}$ — примитивная система либо система линейных базовых фрагментов размерностью m; $\mathbf{V}$ — множество отношений соединения позиций и переходов; Δ-соединение — операция соединения подмножеств позиций и пе-

переходов, которая выполняется на множестве СП-структур.

Рассмотрим процедуру анализа исходной СП-структуры и построения эквивалентной СП-структуры с использованием приемов декомпозиции и тензоров преобразования. Для этого дадим некоторые определения и обозначения:

исходная СП-структура ($\mathbf{N}_\mathrm{I}$) — некоторая структура из заданного класса СП-структур;

линейная СП-структура ($\mathbf{N}_\mathrm{L}$) — структура, полученная в результате декомпозиции исходной СП-структуры ($\mathbf{N}_\mathrm{I}$) и представленная набором линейных и линейно-циклических базовых фрагментов (ЛБФ);

примитивная система ($\mathbf{N}_\mathrm{PR}$) — множество элементарный сетей, мощность которого определяется числом переходов линейной СП-структуры ($\mathbf{N}_\mathrm{L}$);

эталонная модель ($\mathbf{N}$) — СП-структура, полученная из примитивной системы путем объединения заданного множества вершин;

эквивалентная СП-структура ($\mathbf{N}'$) — структура, полученная из эталонной модели с помощью тензора преобразования $\mathbf{C}_{\beta'\gamma}^{\beta\gamma'}$.

Процедуру построения эквивалентной СП-структуры можно представить этапами, приведенными ниже.

Этап 1. Построение системы ЛБФ. На данном этапе исходная СП-структура $\mathbf{N}_\mathrm{I}$ преобразуется в систему ЛБФ $\mathbf{N}_\mathrm{L}$. В процессе данного преобразования необходимо решить две задачи: 1) выделить в $\mathbf{N}_\mathrm{I}$ все имеющиеся линейно-циклические фрагменты; 2) разделить оставшуюся после выделения циклов СП-структуру на линейные фрагменты.

Для поиска линейно-циклических фрагментов можно использовать понятие достижимости из теории графов, на основе чего алгоритм поиска циклов будет иметь следующий вид.

Алгоритм 1. Поиск и выделение циклов в СП-структурах.

Исходные данные: матрицы инцидентности $\mathbf{I}(p, t)$ и $\mathbf{O}(p, t)$, число переходов m и число позиций n в СП-структуре $\mathbf{N}_\mathrm{I}$.

Выходные данные: последовательность вершин $\mathbf{v}$ СП-структуры $\mathbf{N}_\mathrm{I}$, образующая цикл.

А. Поиск циклов.

1. Построить обобщенную матрицу $\mathbf{A} = (\mathbf{P} \cup \mathbf{T}) \times (\mathbf{P} \cup \mathbf{T})$ размерностью $((n + m) \times (n + m))$ СП-структуры $\mathbf{N}_\mathrm{I}$ и вектор вершин $\mathbf{B} = \mathbf{P} \cup \mathbf{T}$ размерностью $(n + m)$.

2. Построить матрицу достижимости $\mathbf{AD} = \mathbf{A}$ путями длиной в одну дугу.

3. Пусть $i = 2$.

4. Построить матрицу достижимости путями длиной в i дуг: $\mathbf{AD} = \mathbf{AD} \times \mathbf{A}$.

5. Если на главной диагонали матрицы **AD** стоят только нули, т. е. $\mathbf{AD}(i, i) = 0$, для $i = \overline{1, n + m}$, то переход к п. 6; иначе — переход к п. 8.

6. Пусть $i = i + 1$.

7. Если $i <= n + m$, то переход к п. 4; иначе: *СП-структура N_I не содержит циклов*, СТОП.

Б. Выделение циклов.

8. Пусть $\mathbf{AD}(k, k) \neq 0$.

9. Пусть $j = 1$, $h = k$, $\mathbf{X}(0) = \{\mathrm{B}(k)\}$, $Y(i) = \{\mathrm{B}(k)\}$.

10. Построить множество вершин $X(j)$ и $Y(h)$: $X(j) = post(X(j - 1))$, $Y(h) = pre(Y(h + 1))$.

11. Пусть $j = j + 1$, $h = h - 1$.

12. Если $j <= i$, то переход к п. 10; иначе — переход к п. 13.

13. Если $(\mathbf{B}(k) \in X(\mathrm{i})) \,\&\, (\mathbf{B}(k) \in Y(0))$, то переход к п. 14; иначе: "ошибка", СТОП.

14. Пусть $j = 0$.

15. Выполнить: $Z(j) = X(j) \cap Y(j)$.

16. Пусть $j = j + 1$.

17. Если $j <= i$, то переход к п. 15; иначе — переход к п. 18.

В. Построение последовательности вершин.

18. Пусть $\mathbf{v} = \varnothing$, $d = Z(0)$.

19. Пусть $j = 1$, $\mathbf{v} = \mathbf{v}|Z(0)$, $d = Z(0)$.

20. Объединить $\mathbf{v}$ с любым элементом множества $post(d) \cap Z(j)$. Пусть данным элементом будет $z_q \in post(d) \cap Z(j)$, тогда: $\mathbf{v} = \mathbf{v}\|z_q$.

21. Пусть $j = j + 1$, $d = z_q$.

22. Если $j <= i - 1$, то переход к п. 20 ; иначе — переход к п. 23.

23. Конец алгоритма.

Разделение элементов СП-структуры и отображение разметки позиций будем проводить с использованием операций, описанных в работе [17]. Процесс разделения СП-структуры на ЛБФ сводится к последовательной проверке условий (11) и (12) над вершинами СП-структуры и выполнения при необходимости операции разделения. Данный процесс продолжается до тех пор, пока указанные условия выполняются хотя бы для одной вершины:

$$|pre(t)| + |post(t)| > 2; \qquad (11)$$

$$|pre(p)| > 1 \text{ или } |post(p)| > 1. \qquad (12)$$

В итоге последовательность действий на первом этапе можно описать следующим образом.

Алгоритм 2. Выделение линейных и линейно-циклических фрагментов.

1. Поиск циклов в СП-структуре $\mathbf{N}_\mathrm{I}$.

2. Если цикл найден, выделить его из СП-структуры $\mathbf{N}_\mathrm{I}$, образовать ЛБФ и перейти к п. 1; если в СП-структуре циклов нет или выделены все циклы — перейти к п. 3.

3. Разделить полученную после выделения циклов СП-структуру $\mathbf{N}_\mathrm{I}$ на линейные фрагменты.

4. Получить систему ЛБФ $\mathbf{N}_\mathrm{L}$.

Пример. Рассмотрим пример разложения исходной СП-структуры $\mathbf{N}_\mathrm{I}$ (рис. 1) на систему ЛБФ.

1. Обобщенная матрица A для СП-структуры $\mathbf{N}_\mathrm{I}$ имеет вид, представленный на рис. 2 (содержимое пустых клеток соответствует "0"). В более компактной форме данную матрицу можно представить следующим образом:

$$\mathbf{A} = \begin{pmatrix} \varnothing & \mathbf{O} \\ \mathbf{I} & \varnothing \end{pmatrix},$$

где **O** и **I** — матрицы инцидентности.

2. При $i = 6$ (т. е. $\mathbf{AD} = \mathbf{A}^6$) получим значения элементов $AD(17, 17)$, $AD(7, 7)$, $AD(18, 18)$, $AD(8, 8)$, $AD(19, 19)$, $AD(9, 9)$, расположенных на главной диагонали, отличные от нуля.

3. Пусть $k = 17$. Тогда

$$X(0) = \{p8\},\ Y(6) = \{p8\},\ X(1) = \{t6\},\ Y(5) = \{t8\},$$

$$X(2) = \{p2\},\ Y(4) = \{p9\},\ X(3) = \{t5, t7\},\ Y(3) = \{t7\},$$

$$X(4) = \{p3, p9\},\ Y(2) = \{p2\},\ X(5) = \{t11, t8\},\ Y(1) = \{t6\},$$

$$X(6) = \{p10, p8\},\ Y(0) = \{p8\}.$$

Строим множества $\mathbf{Z}(j)$, $j = \overline{0, 5}$, которые принимают следующий вид:

$$Z(0) = \{p8\},\ Z(1) = \{t6\},$$

$$Z(2) = \{p2\},\ Z(3) = \{t7\},\ Z(4) = \{p9\},\ Z(5) = \{t8\}.$$

4. Строим последовательность вершин, образующих цикл: $\mathbf{v}_\mathrm{I} = (p8, t6, p2, t7, p9, t8)$.

5. После выделения последовательности $\mathbf{v}_\mathrm{I}$ СП-структура $\mathbf{N}_\mathrm{I}$ примет вид $\mathbf{N}_\mathrm{I}'$, представленный на рис. 3, *а*. Выделенный при этом линейно-циклический фрагмент (фр. 1), представлен на рис. 4.

6. Строим обобщенную матрицу $\mathbf{A}'$ для СП-структуры $\mathbf{N}_\mathrm{I}'$ и возводим ее в степень. При $i = 8$ ($\mathbf{AD} = (\mathbf{A}')^8$) получим значения элементов, расположенных на главной диагонали **AD**, отличные от нуля.

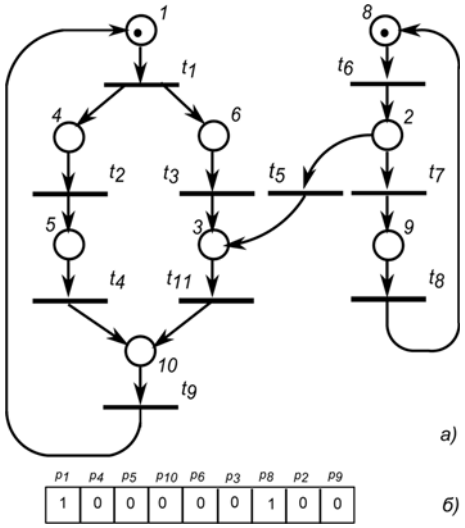


Рис. 1. Исходная СП-структура $\mathbf{N}_\mathrm{I}$ (*а*) и вектор начальной разметки μ_0 (*б*)

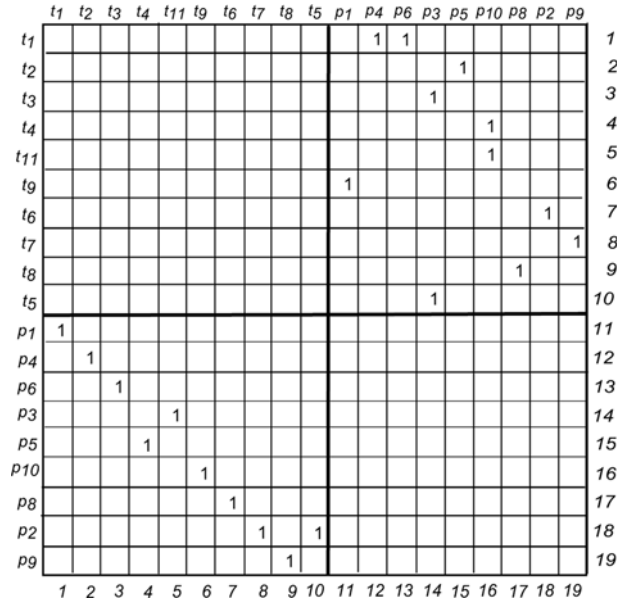


Рис. 2. Обобщенная матрица A исходной СП-структуры $\mathbf{N}_\mathrm{I}$

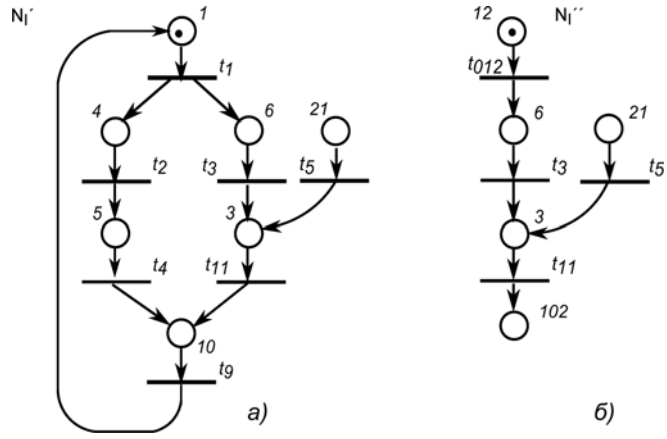


Рис. 3. Промежуточные состояния исходной СП-структуры $\mathbf{N}_\mathrm{I}$

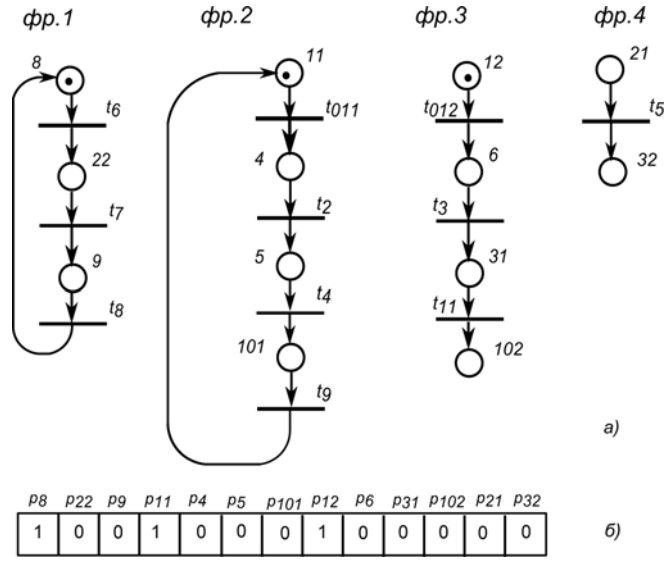


Рис. 4. Результаты анализа исходной СП-структуры $\mathbf{N}_\mathrm{I}$:
а — система ЛБФ $\mathbf{N}_\mathrm{L}$; *б* — вектор начальной разметки $\mu_{0\mathrm{L}}$

7. Пусть $k = 11$. Тогда

$$X(0) = \{p1\} \text{ и } Y(8) = \{p1\},$$
$$X(1) = \{t1\} \text{ и } Y(7) = \{t9\},$$
$$X(2) = \{p4, p6\} \text{ и } Y(6) = \{p10\},$$
$$X(3) = \{t2, t3\} \text{ и } Y(5) = \{t4, t11\},$$
$$X(4) = \{p3, p5\} \text{ и } Y(4) = \{p3, p5\},$$
$$X(5) = \{t4, t11\} \text{ и } Y(3) = \{t2, t3, t5\},$$
$$X(6) = \{p10\} \text{ и } Y(2) = \{p4, p6, p21\},$$
$$X(7) = \{t9\} \text{ и } Y(1) = \{t1\}, X(8) = \{p1\} \text{ и } Y(0) = \{p1\}.$$

Строим множества $\mathbf{Z}(j)$, при $j = \overline{0, 7}$:

$$Z(0) = \{p1\}, Z(4) = \{p3, p5\}, Z(1) = \{t1\},$$
$$Z(5) = \{t4, t11\}, Z(2) = \{p4, p6\}, Z(6) = \{p10\},$$
$$Z(3) = \{t2, t3\}, Z(7) = \{t9\}.$$

8. Строим последовательность вершин $\mathbf{v}_2$, образующих цикл. Для определенности в качестве элемента z_q будем всегда брать первый элемент множества $post(d) \cap \mathbf{Z}(j)$. Тогда: $\mathbf{v}_2 = (p1, t1, p4, t2, p5, t4, p10, t9)$.

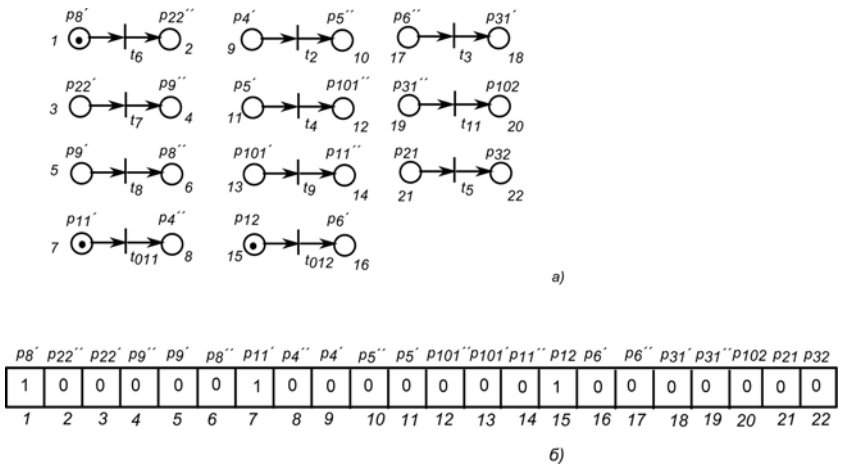


Рис. 5. Примитивная система $\mathbf{N}_{PR}$ (*а*), соответствующая исходной СП-структуре $\mathbf{N}_I$, и вектор начальной разметки μ_{0PR} (*б*)

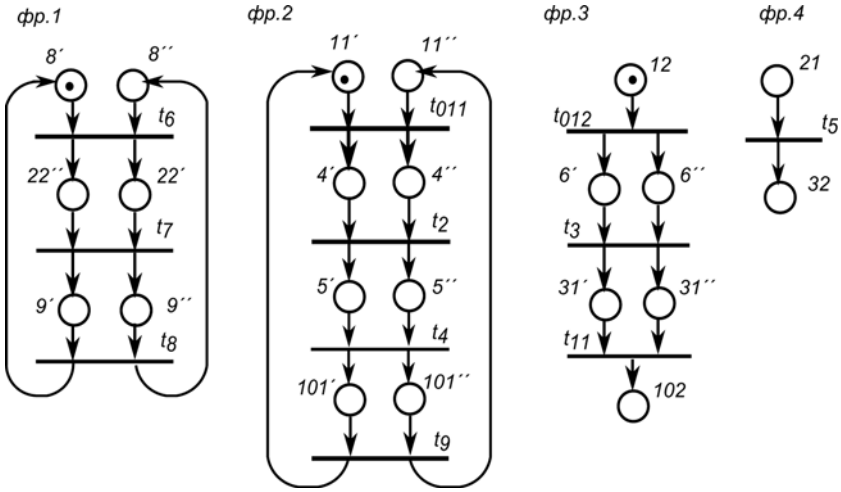


Рис. 6. Эквивалентная СП-структура $\mathbf{N}_{LE}$

9. После выделения последовательности $\mathbf{v}_2$ СП-структура $\mathbf{N}_I'$ примет вид $\mathbf{N}_I''$ изображенный на рис. 3, *б*. Выделенный линейно-циклический фрагмент (фр. 2) представлен на рис. 4.

10. Дальнейший поиск показывает, что в СП-структуре $\mathbf{N}_I'$ циклов нет. Поэтому выполняем деление вершин СП-структуры до тех пор, пока выполняются условия (11) и (12). Результаты деления (фрагменты 3 и 4) представлены на рис. 4.

Этап 2. Построение примитивной системы. На данном этапе на основе системы ЛБФ строится примитивная система ($\mathbf{N}_{PR}$). Данная система включает в себя множество элементарных переходов, которые входят во фрагменты системы ЛБФ. На рис. 5 изображена примитивная система, которая получена на основе системы ЛБФ, представленной на рис. 4. Чтобы сопоставить примитивную СП-структуру $\mathbf{N}_{PR}$ и систему ЛБФ $\mathbf{N}_L$, построим для $\mathbf{N}_L$ эквивалентную СП-структуру ($\mathbf{N}_{LE}$), которая будет включать все множество позиций и переходов, входящих в СП-структуру $\mathbf{N}_{PR}$. Данная СП-структура изображена на рис. 6.

Можно отметить, что действия, соответствующие этапу 1, представляют собой декомпозицию СП-структуры, а совокупность этапов 1 и 2 соответствует анализу СП-структуры, интерпретирующей сложную ВС.

Этап 3. Построение эталонной модели. Эталонная модель строится путем отображения некоторого множества связей вершин на примитивную систему (путем выполнения программы объединения). При этом следует учитывать, что примитивная система представляет собой систему ЛБФ, но выраженную в другой системе координат, т. е. примитивная система в иной системе координат учитывает связи, задаваемые ЛБФ в исходной системе координат. Следовательно, для построения эталонной модели, которая описывала бы, например, связи исходной СП-структуры $\mathbf{N}_I$, но в системе координат примитивной системы, необходимо между вершинами $\mathbf{N}_{PR}$ восстановить связи, которые были разорваны на этапе декомпозиции (при построении множества ЛБФ). Поэтому для получения исходной СП-структуры $\mathbf{N}_I$ (см. рис. 1) преобразование примитивной системы будет определяться программой 1.

Программа 1: $t_1 = t_{011} + t_{012}$, $p_3 = p_{31} + p_{32}, p_1 = p_{11} + p_{12}, p_{10} = p_{101} + p_{102}$, $p_2 = p_{21} + p_{22}$.

Учитывая обозначения примитивной системы, программу 1 можно представить в следующем виде.

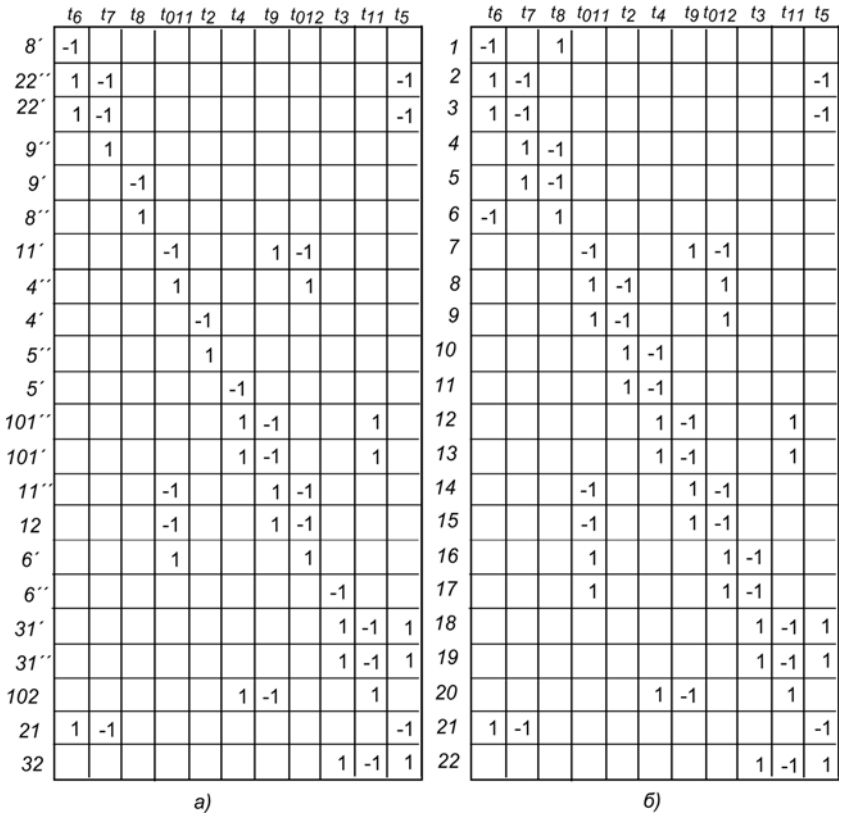


Рис. 7. Матрицы инцидентности эталонной (*а*) и эквивалентной (*б*) СП-структур

Программа 2: $t_1 = t_{011} + t_{012}$, $p_3 = p'_{31} + p''_{31} + p_{32}$, $p_1 = p'_{11} + p''_{11} + p_{12}$, $p_{10} = p'_{101} + p''_{101} + p_{102}$, $p_2 = p_{21} + p'_{22} + p''_{22}$.

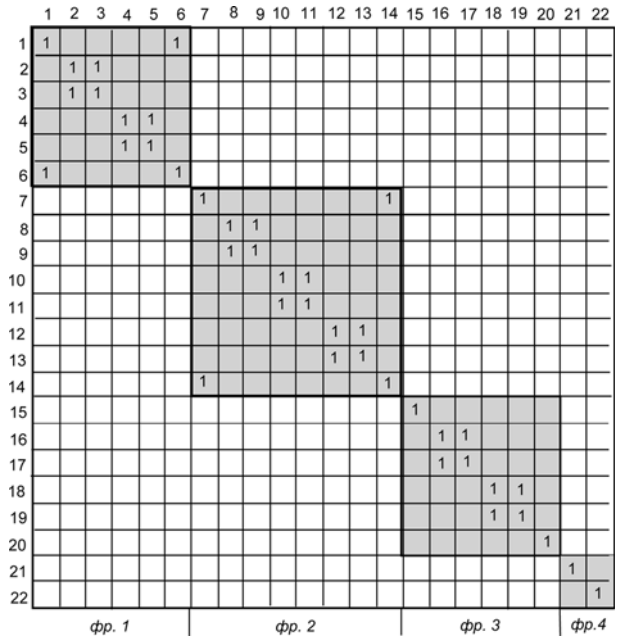


Рис. 8. Тензор преобразования $\mathbf{C}^{\beta\gamma'}_{\beta'\gamma}$

Матрица инцидентности эталонной модели, полученной в результате выполнения программы 2, изображена на рис. 7, *а* (содержимое пустых клеток соответствует "0").

Этап 4. Построение эквивалентной СП-структуры. Эквивалентная СП-структура получается в результате преобразования эталонной модели в соответствии с выражением (3). Предположим, что СП-структуры $\mathbf{N}_{\mathrm{L}}$ и $\mathbf{N}_{\mathrm{PR}}$ — это выражения одной и той же СП-структуры, но представленные в различных системах координат. Поэтому должен существовать тензор преобразования, переводящий представление СП-структуры из одной системы координат в другую. При этом, как инвариант, должна сохраняться начальная разметка СП-структуры. Пусть матрицы инцидентности $\mathbf{D}^{\gamma'}_{\mathrm{L}\beta'}$ и $\mathbf{D}^{\gamma}_{\mathrm{PR}\beta}$ описывают СП-структуры $\mathbf{N}_{\mathrm{L}}$ и $\mathbf{N}_{\mathrm{PR}}$ соответственно. Тогда имеет место следующая система уравнений:

$$\begin{cases} \mathbf{D}^{\gamma'}_{\mathrm{L}\beta'} = \mathbf{C}^{\beta\gamma'}_{\beta'\gamma} \times \mathbf{D}^{\gamma}_{\mathrm{PR}\beta}; \\ \boldsymbol{\mu}^{\gamma'}_{0\mathrm{PR}} = \mathbf{C}^{\beta\gamma'}_{\beta'\gamma} \times \boldsymbol{\mu}^{\gamma}_{0\mathrm{L}}; \\ \boldsymbol{\mu}^{\gamma}_{0\mathrm{L}} = \boldsymbol{\mu}^{\gamma'}_{0\mathrm{PR}}. \end{cases}$$

Из данной системы можно определить искомый тензор преобразования $\mathbf{C}^{\beta\gamma'}_{\beta'\gamma}$.

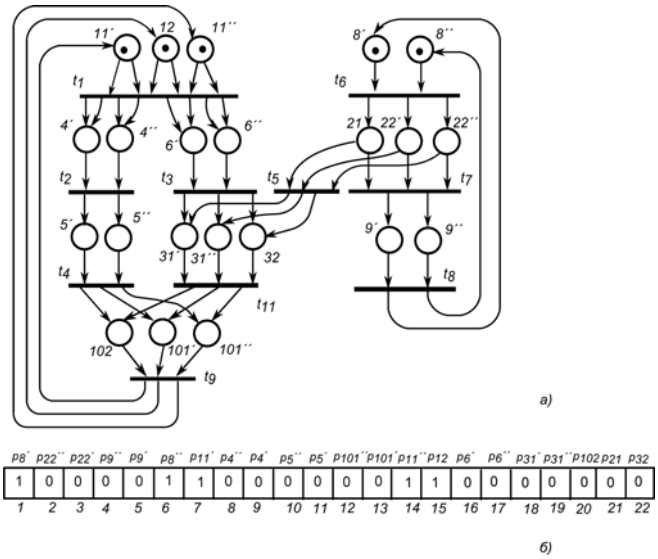


Рис. 9. Эквивалентная СП-структура N′, соответствующая исходной СП-структуре $\mathrm{N_I}$ (*а*), и вектор начальной разметки μ'_0 (*б*)

Очевидно, что действия, выполняемые на этапах 3 и 4, соответствуют процедуре синтеза СП-структур, исходными данными для которой является наперед заданное множество связей вершин примитивной системы.

Возвращаясь к нашему примеру, можно найти один из возможных вариантов тензора преобразования $\mathbf{C}_{\beta'\gamma}^{\beta\gamma'}$, который представлен на рис. 8. Матрица инцидентности, полученная из уравнения (3), представлена на рис. 7, *б*, а соответствующая эквивалентная СП-структура $\mathbf{N}'$ и вектор начальной разметки $\boldsymbol{\mu}_0'$ изображены на рис. 9. В результате минимизации СП-структуры $\mathbf{N}'$ путем сокращения эквивалентных вершин можно прийти к исходной СП-структуре $\mathbf{N}_\mathrm{I}$.

Заключение

В данной работе описан тензорный подход к исследованию сложных систем, представленных в терминах сетей Петри. При этом введены понятия различных систем, которые представляют исходную СП-структуру и ее производные в различных системах координат. Следует отметить, что использование тензорных методов существенно упрощает процедуру построения возможных структур исследуемой сложной системы в системе координат примитивной системы $\mathbf{N}_{PR}$, а также делает рутинной процедуру преобразования новых структур сложных систем в исходную систему координат. Особо следует выделить понятие эквивалентных СП-структур, которые соответствуют новым возможным структурам анализируемой сложной системы, отвечающим тем же функциональным свойствам, но имеющим другие характеристики, например, с точки зрения параллельности или последовательности выполнения операций. Описанный подход дает новые возможности для построения методов синтеза новых структур сложных систем.

Список литературы

1. **Афанасьев В. Г.** Общество: системность, познание, управление. М.: Политиздат, 1981. 432 с.
2. **Кулагин В. П.** Структуры сетей Петри // Информационные технологии. 1997. № 4. С. 17—22.
3. **Котов В. Е.** Сети Петри. М.: Наука, 1984. 160 с.
4. **Питерсон Дж.** Теория сетей Петри и моделирование систем. М.: Мир, 1984. 264 с.
5. **Кулагин В. П.** Проблемы анализа и синтеза структур параллельных вычислительных систем // Информационные технологии. 1997. № 1. С. 2—8.
6. **Rubin V., Gunther C. W., van der Aalst W. M. P., Kindler E., van Dongen B. F., Schafer W.** Process mining framework for software processes. BPM Center Report BPM-07-01.2007 WWW, BPMcenter.org.
7. **Воевода А. А., Марков А. В.** Методика автоматизированного проектирования программного обеспечения функционирования сложных систем на основе совместного использования UML-диаграмм и сетей Петри // Современные технологии. Системный анализ. Моделирование. 2014. № 2 (42). С. 110—115.
8. **Dustdar S., Gombotz R.** Discovering web service workflows using web services interaction mining // International Journal of Business Process Integration and Management (IJBPIM). 2006.
9. **Cabac L., Knaak N., Moldt D., Rölke H.** Analysis of multi-agent interactions with process mining techniques. Fischer K., Timm I. J., Andr'e E., Zhong N. (eds.). Proc. of MATES. LNCS. 2006. Heidelberg: Springer, 2006. Vol. 4196. P. 12—23.
10. **Van Dongen B., van Luin J., Verbeek E.** Process mining in a multi-agent auctioning system // Proc. of the 4th International Workshop on Modelling of Objects, Components, and Agents / Ed. D. Moldt. Turku, June 2006. P. 145—160.
11. **Поляков В. С., Поляков С. В.** Моделирование сложных систем с использованием нейроподобных структур // Известия Волгоградского государственного технического университета. 2012. Т. 13, № 100. С. 119—122.
12. **Storrle H.** Semantics of Uml 2.0 Activities with Data-Flow // Proc. of Int. Symp. of Visual Languages/Human Computer Centered Systems, Rome, Italy. 2004.
13. **Яковлев А. В., Дидрих В. Е., Шамкин В. Н., Краснянский М. Н.** Моделирование распределенных систем с модульной архитектурой на основе сетей Петри // Приборы и системы. Управление, контроль, диагностика. 2012. № 3. С. 34—37.
14. **Крон Г.** Тензорный анализ сетей. М.: Сов. радио, 1978. 720 с.
15. **Петров А. Е.** Тензорная методология в теории систем. М.: Радио и связь, 1985. 152 с.
16. **Кулагин В. П.** Тензорные методы проектирования структур вычислительных систем // АВТ. 1989. № 2. С. 64—71.
17. **Кулагин В. П.** Методы анализа сетевых моделей вычислительных систем // Автоматизация и современные технологии. 1993. № 1. С. 31—34.
18. **Кулагин В. П.** Исследование моделей вычислительных систем методом преобразования координат // Вычислительная техника в автоматизированных системах контроля и управления: Межвуз. сб. научн. тр. Вып. 20 / Пенза: Пенз. политехи, ин-т. 1990. С. 4—7.
19. **Крон Г.** Исследование сложных систем по частям (диакоптика). М.: Наука, 1972. 544 с.
20. **Анкундинов Г. И.** Синтез структуры сложных объектов. Логико-комбинаторный подход. Л.: ЛГУ, 1986. 258 с.
21. **Harel D., Thiagarajan P. S.** Message sequence charts. // UML for Real: Design of Embedded Real-Time Systems, Norwell, MA, USA. Dordrecht: Kluwer Academic Publishers, 2003. P. 77—105.
22. **Van der Aalst W. M. P., ter Hofstede A. H. M.** YAWL: yet another workflow language // Information Systems. 2005. N 30 (4). P. 245—275.
23. **Alves A., Arkin A., Askary S., Barreto C.** et al. Web Services Business Process Execution Language Version 2.0 (OASIS Standard). WS-BPEL TC OASIS. 2007. URL: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
24. **Марков А. В., Воевода А. А.** Проверка достижимости маркировки сетей Петри при помощи инвертирования деревьев состояний для протокола передачи данных // Доклады Томского государственного университета систем управления и радиоэлектроники. 2014. № 1 (31). С. 143—148.
25. **Lorenz R., Juhas G.** Towards Synthesis of Petri Nets from Scenariose / Eds. Donatelli S., Thiagarajan P. S. // CATPN 2006. LNCS. Vol. 4024. Heideiberg: Springer, 2006. P. 302—321.
26. **Амбарцумян А. А.** Моделирование и синтез супервизорного управления на сетях Петри для рассредоточенных объектов. I. Механизм взаимодействия и базовый метод // Автоматика и телемеханика. 2011. № 8. С. 151—160.
27. **Rubin V., Gunther C. W., van der Aalst W. M. P., Kindler E., van Dongen B. F., Schafer W.** Process mining framework for software processes. / Eds. Wang Q., Pfahl D., Raffo D. M. ICSP 2007. LNCS. Vol. 4470. Heideiberg: Springer, 2007. LNCS. Vol. 4470. P. 169—181.
28. **BenMaïssa Y., Kordon F., Mouline S., Thierry-Mieg Y.** Modeling and Analyzing Wireless Sensor Networks with VeriSensor // Petri Net and Software Engineering (PNSE 2012). Hamburg: CEUR, 2012. Vol. 851. P. 60—76.
29. **Gnawali O., Welsh M.** Sensor networks architectures and protocols // Emerging Wireless Technologies and the Future Mobile Internet. Cambridge University Press. 2011. p. 125—153.

V. P. Kulagin, Deputy Director in Innovations, Professor, e-mail: vkulagin@hse.ru,
National Research University Higher School of Economics (HSE)

Tensor Methods of Research Structures of Petri Nets

The article describes the tensor approach to the study of complex systems in terms of Petri nets. Introduced the concept of different systems, which represent the original SP-structure and its derivatives in different coordinate systems. It is shown that using tensor methods, greatly simplifies the procedure of construction of possible structures of the studied complex systems in the coordinate system of the primitive system, and makes routine transformation of new structures of complex systems in the original coordinate system. The described approach provides new opportunities to build methods for synthesis of structures of complex systems.

Keywords: *tensor calculus, Petri nets, patterns of complex systems, coordinate system*

References

1. **Afanas'ev V. G.** *Obshchestvo: sistemnost, poznanie, upravlenie*. M.: Politizdat, 1981. 432 p.
2. **Kulagin V. P.** Struktury setei Petri. *Informatcionnye tekhnologii*. 1997. N 4. P. 17—22.
3. **Kotov V. E.** *Seti Petri*. M.: Nauka, 1984. 160 p.
4. **Peterson Dzh.** *Teoriia setei Petri i modelirovanie sistem*. M.: Mir, 1984. 264 p.
5. **Kulagin V. P.** Problemy analiza i sinteza struktur parallelnykh vychislitelnyjkh sistem // Informatcionnye tekhnologii. 1997. N 1. P. 2—8.
6. **Rubin V., Gunther C. W., van der Aalst W. M. P., Kindler E., van Dongen B. F., Schafer W.** Process mining framework for software processes. *BPM Center Report BPM-07-01*. 2007. URL: www, BPMcenter.org
7. **Voevoda A. A., Markov A. V.** Metodika avtomatizirovannogo proektirovaniya programmnogo obespechenija funktcionirovanija slozhnykh sistem na osnove sovmestnogo ispolzovaniya UML-diagramm i setei Petri. *Sovremennye tekhnologii. Sistemnyi analiz. Modelirovanie*. 2014. N. 2 (42). P. 110—115.
8. **Dustdar S., Gombotz R.** Discovering web service workflows using web services interaction mining. *International Journal of Business Process Integration and Management, IJBPIM*. 2006.
9. **Cabac L., Knaak N., Moldt D., Rölke H.** Analysis of multi-agent interactions with process mining techniques. Fischer K., Timm I. J., Andr'e, E., Zhong, N. (eds.) *MATES 2006. LNCS*. Vol. 4196. P. 12—23. Springer, Heidelberg (2006).
10. **Van Dongen B., van Luin J., Verbeek E.** Process mining in a multi-agent auctioning system. Ed. Moldt D. *Proc. of the 4th International Workshop on Modelling of Objects, Components, and Agents*, Turku, June 2006. P. 145—160.
11. **Poliakov V. S., Poliakov S. V.** Modelirovanie slozhnykh sistem s ispolzovaniem neiropodobnykh struktur. *Izvestiia Volgogradskogo gosudarstvennogo tekhnicheskogo universiteta*. 2012. T. 13. N. 100. P. 119—122.
12. **Storrle H.** Semantics of Uml 2.0 Activities with Data-Flow. *Proc. of Int. Symp. of Visual Languages/Human Computer Centered Systems*. Rome, Italy, 2004.
13. **Iakovlev A. V., Didrikh V. E., Shamkin V. N., Krasnianski M. N.** Modelirovanie raspredelenny'kh sistem s modul'noi arhitekturoi na osnove setei Petri. *Pribory i sistemy. Upravlenie, kontrol, diagnostika*. 2012. N. 3. P. 34—37.
14. **Kron G.** Tenzornyi analiz setei. M.: Sov. radio, 1978. 720 p.
15. **Petrov A. E.** Tenzornaia metodologiia v teorii sistem. M.: Radio i sviaz, 1985. 152 p.
16. **Kulagin V. P.** Tenzornye metody proektirovaniia struktur vychislitelnykh sistem // AVT. 1989. N. 2. P. 64—71.
17. **Kulagin V. P.** Metody analiza setevykh modelei vychislitelnykh sistem. *Avtomatizatciia i sovremennye tekhnologii*. 1993. № 1. P. 31— 34.
18. **Kulagin V. P.** Issledovanie modelei vychislitelnykh sistem metodom preobrazovaniya koordinat. *Vychislitel'naia tekhnika v avtomatizirovarmykh sistemakh kontrolya i upravleniya*: Mezhvuz. sb. nauchn. tr. Vyp. 20. Penza: Penz. politekhn. in-t, 1990. P. 4—7.
19. **Kron G.** Issledovanie slozhnykh sistem po chastiam (diakoptika). M.: Nauka, 1972. 544 p.
20. **Ankundinov G. I.** Sintez struktury slozhnykh ob'ektov. Logiko-kombinatornyi' podhod. L.: LGU, 1986. 258 p.
21. **Harel D., Thiagarajan P. S.** Message sequence charts. *UML for Real: Design of Embedded Real-Time Systems*. Norwell, MA, USA. 2003. P. 77—105. Kluwer Academic Publishers, Dordrecht.
22. **Van der Aalst W. M. P., ter Hofstede A. H. M.** YAWL: yet another workflow language. *Information Systems*. 2005. N. 30 (4). P. 245—275.
23. **Alves A., Arkin A., Askary S., Barreto C.** at al. *Web Services Business Process Execution Language Version 2.0 (OASIS Standard)*. WS-BPEL TC OASIS (2007). URL: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
24. **Markov A. V., Voevoda A. A.** Proverka dostizhimosti markirovki setei Petri pri pomoshchi invertirovaniia dereviev sostoianii dlya protokola peredachi dannykh. *Doclady Tomskogo gosudarstvennogo universiteta sistem upravleniia i radioelektroniki*. 2014. № 1 (31). P. 143—148.
25. **Lorenz R., Juhas G.** Towards Synthesis of Petri Nets from Scenariose. Donatelli S., Thiagarajan P. S. (eds.) *ICATPN 2006. LNCS*. Vol. 4024. P. 302—321. Heidlberg: Springer, 2006.
26. **Ambartcumian A. A.** Modelirovanie i sintez supervizornogo upravleniia na setiakh Petri dlia rassredotochennykh obektov. I. Mehanizm vzaimodeistviia i bazovyi metod. *Avtomatika i telemehanika*. 2011. N. 8. P. 151—160.
27. **Rubin V., Gunther C. W., van der Aalst W. M. P., Kindler E., van Dongen B. F., Schafer W.** Process mining framework for software processes. Wang Q., Pfahl D., Raffo D. M. (eds.) *ICSP 2007. LNCS*. Vol. 4470. P. 169—181. Heidelberg: Springer, 2007.
28. **BenMaïssa Y., Kordon F., Mouline S., Thierry-Mieg Y.** Modeling and Analyzing Wireless Sensor Networks with VeriSensor. In: *Petri Net and Software Engineering (PNSE 2012)*. Hamburg: CEUR, 2012. Vol. 851. P. 60—76.
29. **Gnawali O., Welsh M.** Sensor networks architectures and protocols. In: *Emerging Wireless Technologies and the Future Mobile Internet*. Cambridge University Press, 2011. P. 125—153.

УДК 621.391:396

А. А. Сирота, д-р техн. наук, проф., зав. каф., e-mail: sir@cs.vsu.ru.,
А. В. Цуриков, аспирант, e-mail: andrew.tsurikov@gmail.com,
Воронежский государственный университет

Модели и алгоритмы классификации фрагментов текста и их применение для создания контентно-зависимых цифровых водяных знаков

Рассматриваются модели и алгоритмы классификации многомерных данных с использованием различных подходов к построению классификатора применительно к задаче создания контентно-зависимых цифровых водяных знаков. Проводится сравнение исследуемых алгоритмов, синтезированных на основе различных методов машинного обучения (нейронные сети, машины опорных векторов, потенциальные функции). Исследуется вероятность ошибки классификации многомерных данных в зависимости от размерности признакового пространства.

Ключевые слова: *цифровой водяной знак, классификация данных, радиально-базисные функции, машина опорных векторов, метод потенциальных функций, нейронные сети*

Введение и постановка задачи

В настоящее время повышенное внимание уделяется задачам автоматического распознавания текстовых документов как с точки зрения понимания его содержания, так и с точки зрения защиты авторских прав на произведения, представленные в текстовом формате, проверки наличия заимствований и т. п. Одной из перспективных информационных технологий, направленных на поддержку организационно-технических мероприятий по защите интеллектуальной собственности, является технология цифровых водяных знаков (ЦВЗ) [1, 2], реализующая в том или ином виде алгоритмы стеганографического скрытия информации (ССИ). При реализации технологий создания ЦВЗ применительно к текстовым документам основной задачей является обеспечение высокой достоверности восстановления привязанных к тексту ЦВЗ и их устойчивости к переформатированию текстовых данных. Известные способы [3], такие как метод выравнивания пробелами, метод чередования маркеров конца строки, двоичных нулей, знаков одинакового начертания и др., этому требованию не удовлетворяют. В силу этих особенностей эффективным способом обеспечения заданных требований, по нашему мнению, является использование ЦВЗ, которые не зависят от конкретной формы представления текста. Такие ЦВЗ являются контентно-зависимыми, так как фактически при их использовании реализуется процедура "узнавания" кодированных определенным образом фрагментов текста для извлечения последовательностей элементов ЦВЗ.

В общем виде модель стеганографического скрытия информации для создания ЦВЗ может быть представлена следующим образом. Пусть Z, D, K есть множества возможных контейнеров, скрываемых сообщений (ЦВЗ) и ключей, тогда процедура встраивания и извлечения сообщения может быть представлена в виде отображений вида

$$F_*\colon Z \times D \times K \to Z,\ \bar{z} = F_*(z, d, k),$$

$$z \in Z,\ \bar{z} \in Z,\ d \in D,\ k \in K,\ \|z - \bar{z}\| \to \min,$$

$$F_{**}\colon Z \times K \to D,\ \tilde{d} = F_{**}(\bar{z}, k),\ \tilde{d} \in D,\ \|d - \tilde{d}\| \to \min,$$

где z, $\bar{z}$ — исходный и заполненный контейнер; d, $\tilde{d}$ — исходное и восстановленное сообщение. В задаче создания контентно-зависимых ЦВЗ встраивание и восстановление данных может быть реализовано одним оператором, и соответствующее отображение нужно искать в виде

$$\tilde{d} = F(d, z),\ \tilde{d} \in D,\ \|\tilde{d} - d\| \to \min,$$

где оператор F реализует одновременно встраивание и извлечение информации, не искажая контейнер. Далее без ограничения общности в качестве сообщения или ЦВЗ будем рассматривать двоичную последовательность $d^{(p)}$, $p = \overline{1, P}$, где $d^{(p)} \in \{-1, +1\}$ — скалярная величина, несущая бит информации.

Пусть $z = (z_1, ..., z_n)^{\mathrm{T}}$ — случайный вектор, представляющий фрагмент текстового файла контейнера. Совокупность реализаций вектора $z \in R^n$: $z^{(p)}$, $p = \overline{1, P}$, представляет текст в целом и сопоставляется с элементами последовательности ЦВЗ. При формализации задачи необходимо представить каждый фрагмент контейнера в виде некоторого многомерного вектора данных $z \in R^n$, однозначно характеризующего этот фрагмент.

Компоненты вектора $z \in R^n$, а также его размерность непосредственно зависят от способа его фор-

мирования на основе исходного текста. К ним можно отнести:

- способы, основанные на кодировании символов текста;
- способы, основанные на подсчете длины каждого слова в тексте, подсчете числа слов в предложении;
- способы, основанные на числовом представлении основных структурных элементов организации текста и т. п.

Каждый из этих способов дает возможность получения некоторых структурных характеристик текста. Таким образом, преобразование текста в последовательность $B = \{z^{(p)}, p = \overline{1,P}\}$ можно осуществить различными способами, которые образуют конечное множество S. Тогда любой фрагмент текстового контейнера может быть представлен в виде вектора $z = z(s)$, где $s \in S$, $P = P(s)$. Используя введенные обозначения, можно свести исходную задачу к задаче нахождения отображения вида

$$\tilde{d}^{(p)} = F(d^{(p)}, z^{(p)}(s)),\ p = \overline{1,P}(s),$$
$$s \in S,\ E_d = \|d - \tilde{d}\| \to \min.$$

Это означает, что задача всегда сводится к задаче классификации произвольного множества точек B в многомерном пространстве на два класса, первый из которых соответствует значениям $d^{(p)} = 1$, а второй — значениям $d^{(p)} = -1$. В итоге задача создания ЦВЗ для текстового контейнера сводится к построению алгоритма классификации многомерных данных, представляющих текст. Такие ЦВЗ, очевидно, являются контентно-зависимыми, так как фактически при их создании реализуется процедура "узнавания" кодированных фрагментов текста для извлечения последовательности элементов ЦВЗ. Как показано в работах [4, 5], данная задача может быть достаточно эффективно решена путем построения классификатора, основанного на использовании различных методов машинного обучения. К ним можно отнести нейронные сети на основе многослойных персептронов (MLP) и радиально-базисных функций (RBF), метод потенциальных функций (PF), а также метод машин опорных векторов (SVM).

Целью настоящей работы является сравнительный анализ различных алгоритмов классификации фрагментов текста на основе эталонной статистической модели и реальных текстовых данных.

Эталонная модель классификации текстовых данных

В работе [5] предложена эталонная статистическая модель классификации текста в интересах создания *контентно-зависимых ЦВЗ*, т. е. меток, привязанных к конкретному тексту. Использование данной модели позволяет провести оценку потенциальной эффективности классификации фрагментов текста при их привязке к элементам двоичной последовательности ЦВЗ с использованием различных методов и алгоритмов классификации. При ее обосновании учитывались следующие соображения. Так как общая задача классификации текста сводится к определению принадлежности каждого его фрагмента одному из двух классов (1 и -1), в модели должно генерироваться два класса случайных векторов, образующих точки в многомерном пространстве произвольной размерности. Учитывая, что в процессе преобразования текст разбивается на фрагменты, каждый из которых содержит одинаковое число структурных элементов (в зависимости от типа разбиения это может быть набор букв, слов, абзацев), размерность пространства можно интерпретировать как число элементов в каждом фрагменте. Эта размерность напрямую влияет на потенциальную разделимость данных при классификации, так как ее увеличение означает увеличение числа признаков, по которым классифицируется каждый фрагмент. Таким образом, для исследования потенциальных возможностей создания контентно-зависимых ЦВЗ для контейнеров текстового типа предлагается следующая модель. В единичном гиперкубе I размерности n в соответствии с равномерным законом распределения случайным образом размещается P точек двух классов:

$$B_+ = \{z^{(p)}, p = \overline{1, P_1}\},\ B_- = \{z^{(p)}, p = \overline{1, P_2}\},$$
$$B = \{z^{(p)}, p = \overline{1,P}\} = B_+ \cup B_-,\ P_1 + P_2 = P;$$

$$D_+ = \{d^{(p)} = 1, p = \overline{1, P_1}\},\ D_- = \{d^{(p)} = 0, p = \overline{1, P_2}\},$$
$$D = \{d^{(p)}, p = \overline{1,P}\} = D_+ \cup D_-.$$

Вероятность появления каждой точки первого класса обозначим Q_1, вероятность появления точек второго класса — $Q_2 = 1 - Q_1$. Для генерации случайного числа точек с заданными вероятностями можно использовать алгоритм генерации биномиального распределения.

Основной задачей являются разработка обучаемого классификатора для совокупности векторов $B = \{z^{(p)}, p = \overline{1,P}\}$ и оценка вероятности ошибки классификации точек двух типов, равномерно распределенных в гиперкубе I. Следует также учитывать, что после создания классификатора предъявляемые для классификации данные могут быть некоторым образом искажены, что также отражается в модели. Для этого вводится вероятность искажения каждого фрагмента P_r, при этом искажение вносится в одну из компонент соответствующего вектора, которая модифицируется по равновероятному закону распределения. Синтезируемый классификатор должен быть устойчивым по отношению к данным искажениям.

Как уже упоминалось выше, в ходе исследования проводился сравнительный анализ возможностей применения различных методов классификации данных. С учетом специфики поставленной задачи при задании структуры и параметров классификаторов использовались определенные приемы, направленные на повышение качества получаемых алгоритмов. Рассмотрим реализованные варианты построения классификаторов подробно.

Нейронные сети с RBF и MLP

При решении задачи с использованием нейронных сетей на основе радиально-базисных функций (сети с RBF) в множестве B выделяются группы точек B_+ и B_-, соответствующие точкам первого и второго класса, после чего строится классификатор вида

$$\tilde{d} = F(d, z) = \operatorname{sign}\Phi(z) = \begin{cases} 1, \Phi(z) \geqslant 0, \\ -1, \Phi(z) < 0, \end{cases}$$

$$\Phi(z) = \sum_{i=1}^{K} w_i \varphi_i(z), \qquad (1)$$

$$\varphi_i(z) = \exp\left[-\frac{\|z - u_i\|^2}{2\sigma_i^2}\right],$$

где φ_i — радиально-базисные функции; u_i — центр i-й радиально-базисной функции; σ_i — параметр влияния i-й радиально-базисной функции; w_i — соответствующий весовой коэффициент этой функции; K — число используемых функций.

При фиксированном K классификатор зависит от трех групп параметров: центров радиальных функций $U = (u_1, ..., u_K)^{\mathrm{T}}$, параметров влияния $\Sigma = \{\sigma_1, ..., \sigma_K)^{\mathrm{T}}$, а также весовых коэффициентов $W = (w_1, ..., w_K)^{\mathrm{T}}$. При обосновании алгоритма настройки этих параметров в процессе обучения классификатора с учетом специфики решаемой задачи учитывались следующие соображения.

Для упрощения структуры и повышения эффективности нейронной сети число радиально-базисных функций должно быть меньше общего числа точек ($K < P$). При уменьшении числа радиально-базисных функций начинает расти ошибка классификации, поэтому для ее минимизации к каждому из множеств B_+, B_- независимо применяется алгоритм кластеризации точек k-средних, который выделяет заданное число кластеров, основываясь на определении центров масс векторов (центроидов) и поиске наиболее близких к каждому центроиду элементов по критерию $\sum_{z \in B_{+,-}^{(l)}} (z^{(p,\,l)} - u_l)^2 \to \min$.

Эти центроиды назначаются в качестве центров радиальных функций:

$$u_l^{(+)} = \sum_{z \in B_+^{(l)}} z^{(p,\,l)},\ l = \overline{1, P_1},$$

$$u_l^{(-)} = \sum_{z \in B_-^{(l)}} z^{(p,\,l)},\ l = \overline{1, P_2}.$$

В итоге формируется разбиение общего гиперкуба, соответственно, на K_1 и K_2 ячеек; при этом в пространстве данных образуется многомерная "решетка", получаемая при разбиении на кластеры точек первого класса. Поверх нее независимо накладывается решетка, получаемая при разбиении точек второго класса. Так как каждая ячейка будет классифицироваться своей функцией RBF, то очевидно, что наложение ячеек с точками разных классов будет приводить к ошибкам классификации.

Помимо центров радиальных функций необходимо назначить параметры влияния для RBF-функций. В предлагаемом алгоритме рассматриваются два способа задания этого параметра. Первый способ предполагает задание общего значения σ для всех функций путем последовательного перебора в заданном диапазоне от 0 до 1. Второй способ основан на определении радиуса каждого полученного кластера (расстояния от центра до наиболее удаленной точки) и задании параметра влияния как

$$\sigma_l^{(+)} = \rho R^{(p,l)},\ R^{(p,l)} = \max_{z \in B_+^{(l)}} (\|z^{(p,\,l)} - u_l\|),\ l = \overline{1, P_1},$$

$$\sigma_l^{(-)} = \rho R^{(p,l)},\ R^{(p,l)} = \max_{z \in B_-^{(l)}} (\|z^{(p,\,l)} - u_l\|),\ l = P_2,$$

где ρ — эмпирически подбираемый коэффициент, обычно равный 0,3...0,5.

На заключительном этапе построения классификатора необходимо вычисление весовых коэффициентов на основе решения системы линейных алгебраических уравнений (СЛАУ) [6]. В данном случае ($K < P$) СЛАУ для нахождения весовых коэффициентов является переопределенной. Для ее решения могут быть использованы различные методы, в том числе метод псевдоинверсии Мура — Пенроуза (нормальное псевдорешение), метод, основанный на разложении SVD (Singular Value Decomposition), метод регуляризации по А. Н. Тихонову [6]. Как показали исследования, последний вариант позволяет сформировать устойчивые решения в виде

$$GW = d,\ w = w^{(a)} + (G^{\mathrm{T}}G + \alpha I)^{-1} G^{\mathrm{T}}(d - Gw^{(a)}),$$

$$G = \|g_{p,\,i}\|,\ g_{p,\,i} = \|\varphi_i(z^{(p)})\|,\ p = \overline{1, P},\ i = \overline{1, K},\ K < P,$$

где G — матрица Грина [6], являющаяся в данном случае квадратной; $d = (d^{(1)}, ..., d^{(p)})^{\mathrm{T}}$ — целевой вектор, определяемый исходя из исходного множества тре-

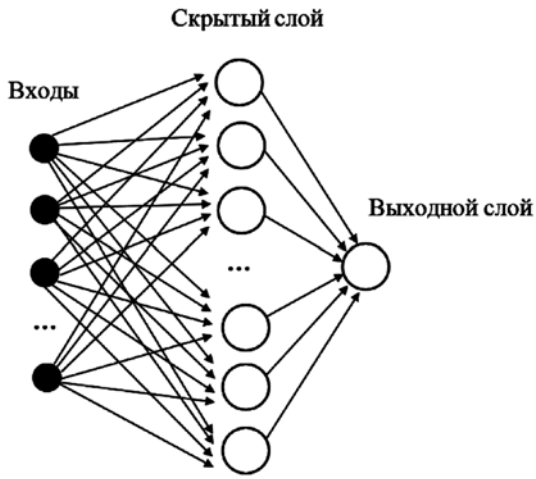


Рис. 1. Архитектура нейронной сети типа MLP

буемой классификации данных $D = \{d^{(p)}, p = \overline{1, N}\} = D_+ \cup D_-$; $w^{(a)}$ — априорное решение, которое в данном случае целесообразно выбрать следующим образом: $w_l^{(a)} = 1$, $l = \overline{1, K_1}$, $w_l^{(a)} = -1$, $l = \overline{K_1 + 1, K_2}$; α — параметр регуляризации, выбираемый одним из стандартных методов.

При построении классификатора на основе многослойных нейронных сетей персептронного типа (сети типа MLP) так же, как и в случае с сетями на основе радиально-базисных функций, возникает проблема определения оптимальной конфигурации сети, усложняющаяся в силу того, что MLP, в отличие от RBF, могут содержать более одного скрытого слоя. В работе [7] рассматривается *теорема об универсальной аппроксимации* для нелинейного отображения "вход—выход", которая определяет математические обоснования возможности аппроксимации любой непрерывной функции. Теорема утверждает, что многослойного персептрона с одним скрытым слоем достаточно для построения равномерной аппроксимации с точностью ε для любого обучающего множества, представленного набором входов $z^{(1)}, z^{(2)}, ..., z^{(P)}$ и желаемых откликов $f(z^{(1)}, ..., z^{(P)})$ [3]. При проведении исследования далее рассматривается сеть с одним скрытым слоем, содержащим K нелинейных нейронов, и выходным слоем, содержащим один линейный нейрон. Общий вид такой сети представлен на рис. 1.

Метод потенциальных функций (PF)

Подход к построению классификатора, основанный на методе потенциальных функций [7], также позволяет достаточно эффективно проводить разделение произвольной выборки данных на классы. Общая идея заключается в представлении элементов выборки в виде точек пространства, в которые помещается электрический заряд $+q$, если точка принадлежит элементу 1-го класса, и $-q_i$, если точка принадлежит элементу 2-го класса. Учитывая, что в рамках метода потенциальных функций можно рассматривать каждый элемент $z = (z_1, ..., z_n)^T$, $z \in R^n$, как единичный заряд, конечный вид классификатора, представляющего собой кумулятивный потенциал, создаваемый P зарядами в точке z, будет определяться следующим выражением:

$$g(z) = \sum_{i=1}^{P} q_i \Phi(z, z_i) = \Phi_N(z), \qquad (2)$$

где в качестве функции Φ выступает функция $\Phi(z, z_k) = \exp[-\|z - z_k\|^2 / 2\sigma^2]$, а коэффициенты q_i определяются соотношением

$$q_i = \begin{cases} 0, & z_i \in B_+, \ \Phi_{i-1}(z_i) > 0; \\ 0, & z_i \in B_-, \ \Phi_{i-1}(z_i) \leqslant 0; \\ 1, & z_i \in B_+, \ \Phi_{i-1}(z_i) \leqslant 0; \\ -1, & z_i \in B_-, \ \Phi_{i-1}(z_i) > 0. \end{cases}$$

Стоит отметить, что число ненулевых коэффициентов q_i определяет общее число нелинейных классифицирующих элементов K, составляющих классификатор, которое в данном случае зависит от обучающей выборки.

Машина опорных векторов (SVM)

Суть работы алгоритма, основанного на машине опорных векторов, состоит в построении гиперплоскости, максимально разделяющей элементы разных классов в многомерном пространстве [9—11]. В общем виде линейный классификатор для задачи классификации данных на два непересекающихся класса может быть представлен в виде

$$a(z) = \operatorname{sign}\left(\sum_{j=1}^{n} w_j z_j - w_0 \right), \qquad (3)$$

где $z = (z_1, ..., z_n)^T$ — фрагмент текста, вектор $w = (w_1, ..., w_n) \in R^n$ и скалярный порог $w_0 \in R$ являются параметрами алгоритма. Задача нахождения этих параметров сводится к эквивалентной двойственной задаче поиска седловой точки функции Лагранжа.

При построении нелинейного классификатора данных используются функции, обеспечивающие нелинейное преобразование исходного пространства, при этом классификатор ищется в виде [9]

$$a(z) = \operatorname{sign}\left(\sum_{i=1}^{l} \lambda_i d_i \Phi(z_i, z) - w_0 \right),$$

$$\Phi(z_i, z) = \exp[-(\|z_i - z\|)^2 / 2\sigma_i^2], \qquad (4)$$

где λ_i — компоненты вектора двойственных переменных; d_i — компоненты целевого вектора; σ_i — параметр влияния i-й функции.

Стоит отметить, что при решении задачи классификации фрагментов текста на основе машины опорных векторов классификатор можно считать

эффективным, если число используемых опорных векторов $l < P$ существенно меньше, чем число классифицируемых точек, т. е. на один опорный вектор приходится несколько классифицируемых точек (фрагментов текста).

Тогда настройка параметров классификатора (4) сводится к решению задачи квадратичного программирования вида

$$\max_{\lambda} W(\lambda) = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} d_i d_j \varphi(z_i, z_j) \lambda_i \lambda_j;$$

$$\sum_{i=1}^{l} d_i \lambda_i = 0, \ \forall i, \ 0 \leqslant \lambda_i \leqslant C.$$

Существует несколько различных алгоритмов решения данной задачи, среди которых непосредственное решение задачи квадратичного программирования (QP), использование алгоритма последовательной минимальной оптимизации (SMO) [9] и метод "бюджетированых" SVM, позволяющий выдержать заданный "бюджет" числа опорных векторов [10, 11]. Последний также предоставляет интерес для решения поставленной задачи.

В отличие от непосредственного решения задачи квадратичного программирования алгоритм SMO выбирает решение наименьшей возможной проблемы оптимизации на каждом шаге, что позволяет существенно ускорить и оптимизировать вычисления. Метод "бюджетированных" SVM основан на применении стохастического градиентного спуска при решении задачи построения классификатора, но при этом реализуется задача сохранения заданного "бюджета" опорных векторов. Для реализации подобного подхода на каждом шаге обучения алгоритм проверяет, не превышает ли число найденных опорных векторов (ОВ) заданное ограничение ($|I_{t+1}| < T$). В случае превышения число ОВ уменьшается на 1. При использовании данного метода эффективность классификации данных может снижаться по сравнению с обычными градиентными методами. В работе [10] показывается, что ошибка классификации, накладываемая введением в алгоритм стратегии поддержания опорных векторов на заданном уровне, ограничена сверху значением средней ошибки градиента

$$E = \sum_{t=1}^{P} \|E_t\|/P,$$

при этом $E_t = \Delta_t/\eta_t$, где Δ_t — разница между значениями параметров классификатора до начала процедуры сохранения ограничений и после; η_t — скорость обучения.

С учетом того, что влияние процедуры сохранения бюджета на ошибку классификации должно быть минимальным, алгоритм старается уменьшить величину E путем минимизации $\|E_t\|$ на каждом шаге, что сводится к минимизации $\|\Delta_t\|^2$ [10].

При реализации ограничений могут быть использованы различные стратегии сохранения числа опорных векторов: удаление избыточных опорных векторов, проецирование новых векторов на уже найденные и слияние векторов. В первых двух случаях вычислительная сложность алгоритмов достаточно высока, в связи с этим наиболее оптимальным представляется последний подход.

Необходимо отметить, что одним из главных условий эффективности всех описанных выше способов построения классификаторов для решения рассматриваемой задачи является минимизация общего числа нелинейных классифицирующих элементов, составляющих каждый классификатор (для RBF — это число радиально оазисных функций, для MLP — число персептронов в скрытом слое, для PF — число функций Ф, для SVM — число опорных векторов). Поэтому наряду с вероятностью общей ошибки классификации предлагается рассматривать универсальную величину K, которая является *"бюджетом" нелинейных классифицирующих элементов*, в качестве критерия эффективности для произвольного классификатора. Кроме того, для корректности сравнения классифицирующих способностей при различных подходах можно также рассматривать обратную величину *"бюджетного" отношения* $C = P/K$, которое показывает число классифицируемых элементов, приходящихся на единицу бюджета классифицирующих элементов. Для классификаторов на основе SVM, RBF и MLP существует возможность управлять бюджетом, в то время как для метода потенциальных функций ограничения такой возможности нет. Поэтому в данном случае проводилось усреднение бюджета, получаемого для нескольких реализаций процесса обучения классификатора.

Исследование алгоритмов на основе эталонной статистической модели

Представляется важным исследование потенциальных возможностей различных подходов к созданию классификатора фрагментов текста на основе эталонной статистической модели данных. В процессе моделирования в среде MATLAB проводилось сравнение вероятности ошибки классификации многомерных данных для классификаторов на основе RBF, нейронной сети MLP, машин опорных векторов, а также для классификатора, основанного на методе потенциальных функций. Для SVM использовалось три разных метода настройки параметров: SMO, непосредственное решение задачи квадратичного программирования (QP) и алгоритм "бюджетированного" стохастического градиентного спуска [10], причем в качестве эталонной была выбрана стандартная реализация алгоритма SMO [8] в MATLAB.

На рис. 2 показана зависимость вероятности ошибки классификации для классификаторов на

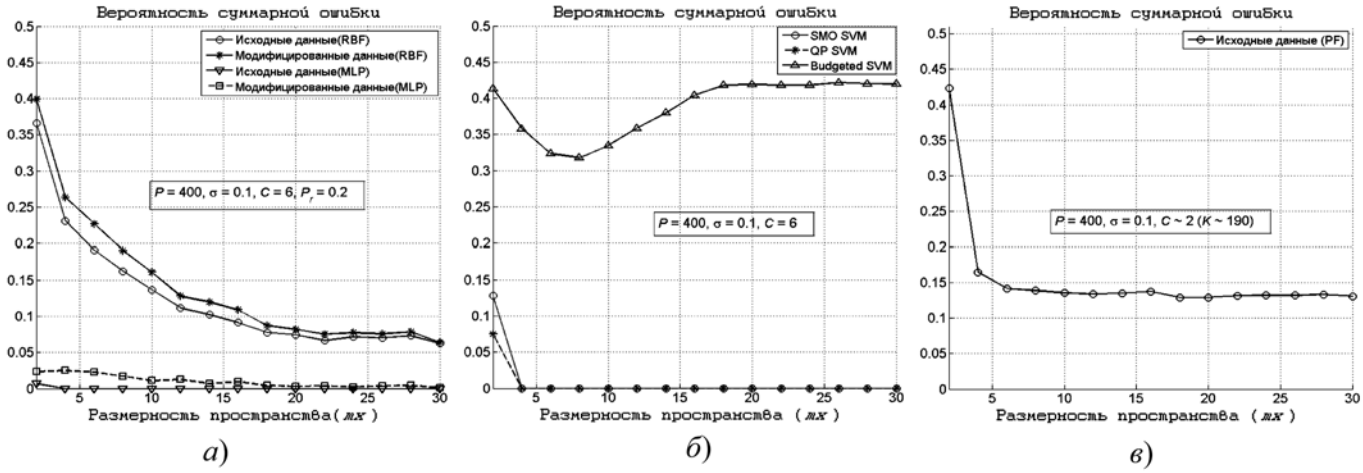


Рис. 2. Зависимость вероятности ошибки классификации для классификаторов на основе RBF, MLP, SVM и PF

основе RBF, MLP, SVM и PF от размерности *mx* признакового пространства при фиксированном параметре влияния RBF σ = 0,1 и "бюджетном" отношении $C = 6$. Так как для метода потенциальных функций невозможно установление ограничения на величину K, то на рисунке приведено фактическое значение параметра, получившееся в результате работы алгоритма. Для обеспечения приемлемой точности данные брались как усредненное значение результатов 100 реализаций процесса обучения на каждом шаге. Кроме того, график на рис. 2, *а* содержит результаты тестирования классификаторов RBF и MLP на модифицированных данных (каждая компонента всех элементов модифицируется с вероятностью $P_r = 0{,}2$).

Из графиков следует, что в целом наиболее эффективным классификатором является MLP, а наиболее эффективным алгоритмом настройки параметров SVM при малых значениях σ является SMO, который сохраняет показатели при увеличении размерности пространства. В то же время можно сказать, что несмотря на самую высокую эффективность классификации данных в целом, классификатор MLP является менее устойчивым к модификации исходных данных, чем RBF. Также стоит отметить, что классификатор, реализованный на основе метода потенциальных функций, является сопоставимым по величине ошибки классификации с другими алгоритмами, но гораздо менее эффективным с точки зрения величины бюджета нелинейных классифицирующих элементов.

При изменении значения параметра влияния σ эффективность алгоритма SMO остается на прежнем высоком уровне, в то время как метод, основанный на сохранении "бюджета" опорных векторов, показывает снижение ошибки классификации только при увеличении значения параметра σ и очень больших размерностях пространства.

График на рис. 3 показывает вероятность суммарной ошибки метода "бюджетированного" стохастического градиентного спуска для SVM при изменении ограничений на "бюджетное" соотношение C.

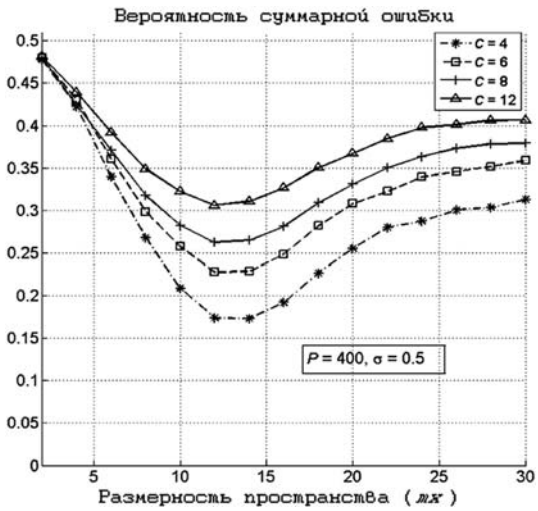


Рис. 3. Вероятность ошибки классификации для классификатора SVM при различных ограничениях на "бюджет" векторов

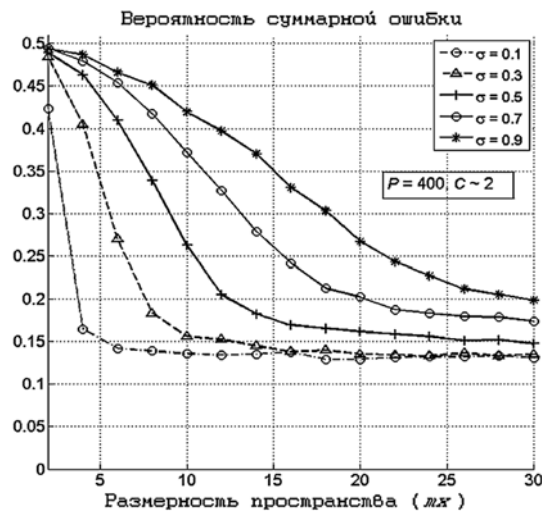


Рис. 4. Вероятность ошибки классификации для метода потенциальных функций при различных значениях параметра влияния

Учитывая, что для машины опорных векторов данная величина показывает отношение числа классифицируемых элементов к числу опорных векторов, зависимости ожидаемо показывают, что при уменьшении числа опорных векторов (соответственно, при увеличении числа элементов, приходящихся на один опорный вектор) вероятность ошибки классификации существенно возрастает, причем увеличение размерности пространства позволяет снизить эту величину лишь до некоторого предела. В частности, как можно видеть на графиках, минимальное значение вероятности ошибки наблюдается при выборе размерности пространства в диапазоне 12...14.

Рис. 4 показывает зависимость вероятности ошибки классификации от размерности пространства для классификатора на основе метода потенциальных функций при использовании разных значений параметра влияния σ. Очевидно, что увеличение параметра σ приводит к увеличению ошибки классификации, так как области влияния каждого нелинейного классифицирующего элемента начинают пересекаться все сильнее. В случае применения метода потенциальных функций оптимальным представляется использование наименьшей величины σ.

Необходимо отметить, что данные результаты были получены при наиболее плохом для алгоритма "бюджетированных" SVM случае распределения данных — равномерном. В случае даже незначительной неравномерности распределения данных показатели "бюджетированных" SVM улучшаются.

Тестирование алгоритмов на реальных данных

Для проверки работоспособности алгоритма при использовании реальных данных был выбран текст объемом около 3000 слов, который был закодирован несколькими способами. В частности, в качестве кодирования рассчитывались длины слов и брались коды символов. В отличие от эталонной модели, где число распределяемых в гиперкубе элементов было строго фиксированным и задаваемым в начале моделирования, при моделировании работы алгоритма на реальных данных число элементов, закодированных и нормализованных выбранным способом, будет меняться в силу того, что исходный текст представляется в виде последовательности $B = \{z^{(p)},\ p = \overline{1, P}\}$, где величина P, характеризующая число векторов, образующих последовательность B, зависит от размерности гиперкуба (фактически — от выбранной размерности векторов $z^{(p)} = (z_1, ..., z_n)^{\mathrm{T}}$). Кроме того, реальные данные имеют гораздо большую неравномерность распределения в силу естественных ограничений, накладываемых на способы кодирования. На рис. 5 приведен пример распределения нормализованных данных в двумерном гиперкубе для эталонной модели и реального текста соответ-

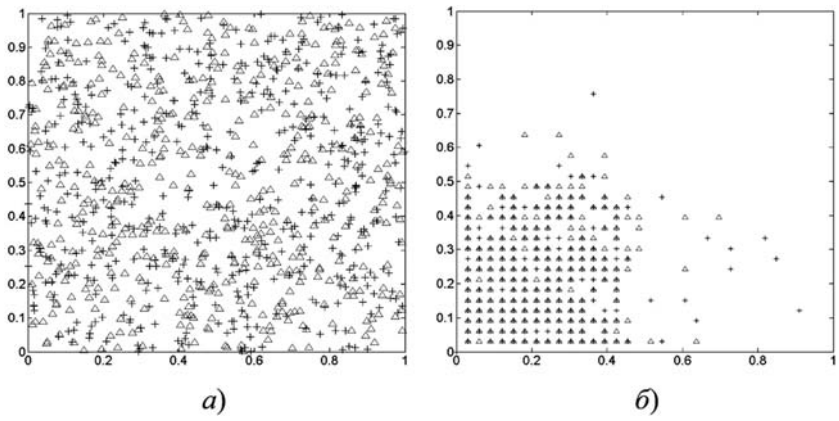


Рис. 5. Распределение элементов в двумерном гиперкубе для данных, полученных на основе эталонной модели, и для реального текста

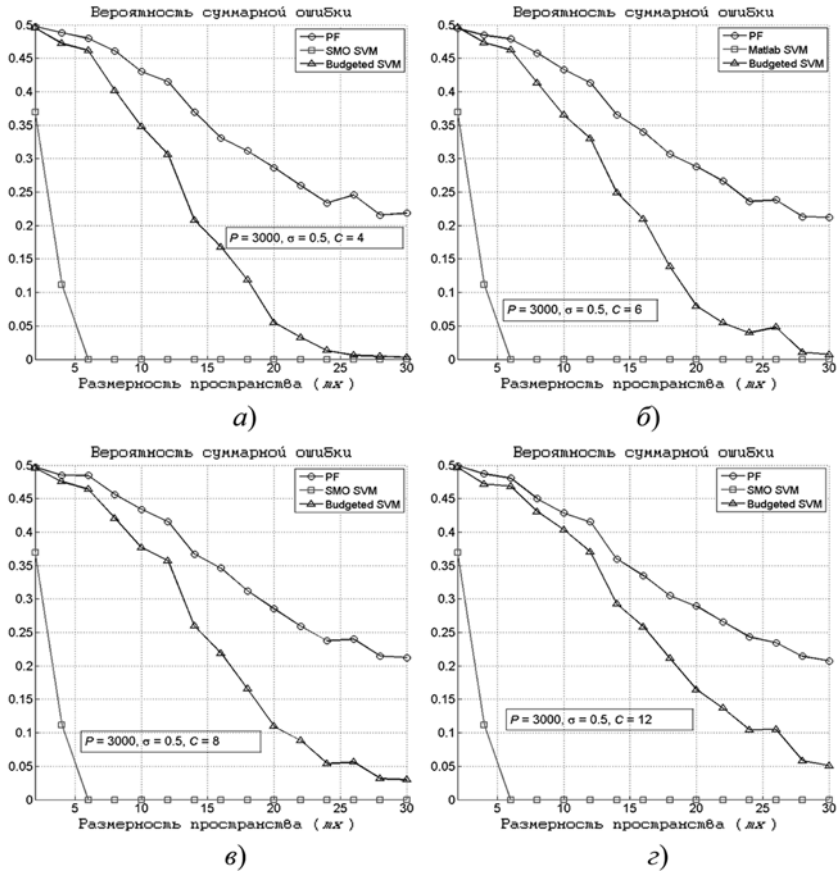


Рис. 6. Вероятность ошибки классификации для классификатора SVM при различных ограничениях на бюджет векторов (реальный текст)

ственно в случае, когда в качестве ЦВЗ использовалась псевдослучайная двоичная последовательность, а кодирование текста проводилось на основе подсчета длин слов.

Наличие сильной неравномерности распределения реальных данных существенно меняет результаты классификации. Как видно на рис. 6, наибольшую эффективность классификации по-прежнему сохраняет алгоритм SMO, но теперь алгоритм "бюджетированных" SVM позволяет достичь практически таких же результатов для размерности классифицируемых векторов больше 20.

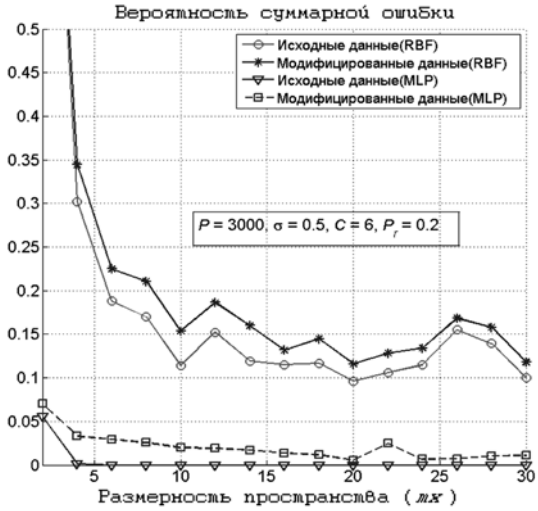


Рис. 7. Вероятность ошибки классификации для классификатора на основе RBF и MLP (реальный текст)

Таблица 1

Вероятности ошибки классификации в зависимости от размерности признакового пространства для эталонной модели

Размерность признакового пространства (*mx*)	SVM		RBF	MLP	PF
	SMO	Budgeted SVM			
2	0,3930	0,5028	0,2965	0,175	0,4885
4	0,1198	0,4810	0,2288	0,08	0,4638
8	0	0,4138	0,1624	0,0025	0,3303
16	0	0,3023	0,0960	0	0,1758
20	0	0,2665	0,0790	0	0,1540
24	0	0,2808	0,0679	0	0,1623
30	0	0,3163	0,0629	0	0,1475

Таблица 2

Вероятности ошибки классификации в зависимости от размерности признакового пространства для реального текста

Размерность признакового пространства (*mx*)	SVM		RBF	MLP	PF
	SMO	Budgeted SVM			
2	0,3691	0,4907	1	0,2913	0,4949
4	0,1117	0,4769	0,3667	0,1193	0,4913
8	0	0,4170	0,1538	0	0,4659
16	0	0,1280	0,1220	0	0,3227
20	0	0,0274	0,1151	0	0,3142
24	0	0,0102	0,0920	0	0,2364
30	0	0,0014	0,0861	0	0,2097

Для сравнения на рис. 7 показана вероятность ошибки классификации данных для реального текста при использовании нейронной сети на основе RBF и MLP при бюджетном отношении $C = 6$. На данном графике также представлен результат распознавания искаженных данных. При подобном искажении отдельные компоненты векторов $z^{(p)}$ модифицируются случайным образом с некоторой вероятностью P_r.

Можно заметить, что результаты работы алгоритма как с использованием нейронных сетей, так и с использованием машины опорных векторов, показывают достаточную для успешного разделения данных на два класса эффективность. В табл. 1 и 2 представлены общие результаты работы алгоритмов с использованием различных классификаторов для эталонной статистической модели и реального текста, соответственно.

Заключение

Результаты показывают, что использование машины опорных векторов позволяет достичь сопоставимых с классификатором на основе RBF показателей. Кроме того, стоит отметить, что кодирование реального текста накладывает естественные ограничения, непосредственно связанные с лингвистикой, которые сказываются на конечном распределении закодированных элементов. Наиболее эффективным алгоритмом с точки зрения вероятности ошибки классификации является нейронная сеть MLP, хотя данная вероятность падает при искажении текста. Сопоставимым с ней является алгоритм SMO SVM, который фактически обучается классифицировать уникально каждый элемент текста по аналогии с сетью на основе RBF с числом радиально-базисных функций, равным размеру обучающей выборки. Вместе с тем, в отличие от эталонной статистической модели, где метод бюджетированных SVM не показывал достаточно высокую ошибку классификации, при использовании реальных данных он становится более эффективным, чем классификатор на основе RBF, и успешно приближается в плане эффективности классификации к SMO, сохраняя при этом возможности по управлению ограничениями на число опорных векторов. Таким образом, можно сделать вывод, что классификатор на основе машины опорных векторов является альтернативой нейронным сетям и обладает сопоставимым качеством классификации.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 13-01-97507 р_центр_а.

Список литературы

1. **Мельников Ю. П., Теренин А. В., Погуляев В. Г.** Цифровые водяные знаки — новые методы защиты информации // Компьютерная неделя. 2007. № 48 (606). URL: http://www.pcweek.ru/security/article/detail.php?ID=105054 (дата обращения 08.06.2014).
2. **Барсуков В. С., Шувалов А. В.** Еще раз о стенографии — самой современной из древнейших наук // Специальная техника. 2004. № 2. URL: http://www.ess.ru/sites/default/files/files/articles/2004/02/2004 02 04.pdf (дата обращения 08.06.2014).
3. **Текин В.** Текстовая стеганография // МирПК. URL: http://www.osp.ru/pcworld/2004/11/169154/ (дата обращения 17.04.2014).
4. **Сирота А. А., Цуриков А. В.** Модели и алгоритмы классификации многомерных данных на основе нейронных сетей с радиально-базисными функциями // Вестник ВГУ. Сер. "Системный анализ и информационные технологии". Воронеж, 2013. № 1. С. 154—161.
5. **Сирота А. А., Цуриков А. В., Дрюченко М. А.** Модели и алгоритмы классификации фрагментов текста на основе нейронных сетей с радиально-базисными функциями // Нейрокомпьютеры: разработка, применение. 2013. № 5. С. 26—37.
6. **Хайкин С.** Нейронные сети: полный курс. 2-е изд., испр. М.: ООО "И. Д. Вильямс", 2006. 1104 с.
7. **Осовский С.** Нейронные сети для обработки информации / Пер с польского И. Д. Рудинского. М.: Финансы и статистика, 2002. 344 с.
8. **Вапник В. Н., Червоненкис А. Я.** Теория распознавания образов (статистические проблемы обучения). М.: Наука, 1974. 416 с.
9. **Platt J.** Fast Training of Support Vector Machines Using Sequential Minimal Optimization // Advances in Kernel Methods. — Support Vector Learning, 1998. P. 185—208.
10. **Wang Z., Crammer K., Vucetic S.** Breaking the Curse of Kernelization: Budgeted Stochastic Gradient Descent for Large-Scale SVM Training // Journal of Machine Learning Research. 2012. N. 3. P. 3103—3131.
11. **Dekel O., Singer Y.** Support Vector Machines on a Budget // Advances in Neural Information Processing Systems. 2006. N. 19. P. 345—352.

A. A. Sirota, Professor, **A. V. Tsurikov,** Postgraduate Student, e-mail: andrew.tsurikov@gmail.com,
Voronezh State University

Creating Content-Dependent Digital Watermarks Using their Structural Patterns: Models and Algorithms of Text Fragments Classification

The paper introduces approaches to creating content-dependent digital watermarks for the text data containers. It proposes ways of encoding text for further implementation of the matching procedure between the encoded text fragments and binary encoded digital watermark elements. The paper shows that this matching procedure is fully equivalent to the classification of the high-dimensional data. It also outlines approaches to creating high-dimensional data classifiers using various techniques for developing content-dependent digital watermarks, and compares them with other classification algorithms (neural networks, support vector machines and potential functions). Along with common algorithms of machine learning, it considers the ways of making them more efficient by limiting the number of the elements that take part in the classification. It also examines the dependency between the data classification errors and the space dimension size for different classifiers using simulated and real text data.

Keywords: *content-dependent digital watermark, data classification, radial-basis function, support vector machine, potential functions method, neural networks*

References

1. **Mel'nikov Ju. P., Terenin A. V., Poguljaev V. G.** Cifrovye vodjanye znaki — novye metody zashhity informacii. *Komp'juternaja nedelja*. 2007. N. 48 (606). URL: http://www.pcweek.ru/security/article/detail.php?ID=105054
2. **Barsukov V. S., Shuvalov A. V.** Eshhe raz o stenografii — samoj sovremennoj iz drevnejshih nauk. *Special'naja tehnika*. 2004. N. 2. URL: http://www.ess.ru/sites/default/files/files/articles/2004/02/2004 02 04.pdf
3. **Tekin V.** Tekstovaja steganografija. *MirPK*. URL: http://www.osp.ru/pcworld/2004/l 1/169154/
4. **Sirota A. A., Curikov A. V.** Modeli i algoritmy klassifikacii mnogomernyh dannyh na osnove nejronnyh setej s radial'no-bazisnymi funkcijami. *Vestnik VGU, Ser.: "Sistemnyj analiz i informacionnye tehnologii"*. Voronezh, 2013. N. 1. P. 154—161.
5. **Sirota A. A., Curikov A. V., Drjuchenko M. A.** Modeli i algoritmy klassifikacii fragmentov teksta na osnove nejronnyh setej s radial'no-bazisnymi funkcijami. *Neirokomp'jutery: razrabotka, primenenie*. 2013. N. 5. P. 26—37.
6. **Hajkin S.** Nejronnye seti: polnyj kurs. 2-e izd., ispr. M.: OOO "I. D. Vil'jams", 2006. 1104 p.
7. **Osovskij S.** *Nejronnye seti dlja obrabotki informacii* / Per s pol'skogo I. D. Rudinskogo. M.: Finansy i statistika, 2002. 344 p.
8. **Vapnik V. N., Chervonenkis A. Ja.** *Teorija raspoznavanija obrazov (statisticheskie problemy obuchenija)*. M.: Nauka, 1974. 416 p.
9. **Platt J.** Fast Training of Support Vector Machines Using Sequential Minimal Optimization. *Advances in Kernel Methods*. — Support Vector Learning, 1998. P. 185—208.
10. **Wang Z., Crammer K., Vucetic S.** Breaking the Curse of Kernelization: Budgeted Stochastic Gradient Descent for Large-Scale SVM Training. *Journal of Machine Learning Research*. 2012. N. 13. P. 3103—3131.
11. **Dekel O., Singer Y.** Support Vector Machines on a Budget. *Advances in Neural Information Processing Systems*. 2006. N. 19. P. 345—352.

УДК 519.233

Б. Г. Кухаренко, канд. физ.-мат. наук, ст. науч. сотр., вед. науч. сотр.,
Институт машиноведения имени А. А. Благонравова РАН, г. Москва, e-mail: kukharenko@imash.ru,
М. О. Солнцева, аспирант,
Московский физико-технический институт (ГУ), e-mail: solnceva.chalei@gmail.com

Анализ результатов кластеризации многомерных траекторий посредством моделей линейных динамических систем

Модели линейных динамических систем используются для анализа результатов кластеризации траекторий объектов по методу полиномиальных регрессий. Преимуществом этих моделей является уменьшение размерности анализируемого пространства. Для проекций траекторий кластера по каждому измерению выделяется наиболее информативная составляющая (полиномиальная регрессия) и проявляется тонкая структура кластера. Эффективность моделей линейных динамических систем демонстрируется на примере анализа результатов кластеризации для траекторий движения самолетов в воздушном пространстве аэропорта.

Ключевые слова: *анализ данных, многомерные траектории, кластеризация, полиномиальная регрессия, фильтр Калмана, сглаживатель Рауха, линейные динамические системы, алгоритм ожидания и максимизации правдоподобия*

Введение

Задача кластеризации траекторий движения обусловлена необходимостью предсказания движения для организации взаимодействия управляемых объектов на основе выделяемых паттернов в траекториях движения. Одновременная кластеризация и выравнивание траекторий, т. е. векторов переменной длины, представляющих их координатные временные зависимости, по методу полиномиальной регрессии представлена в работах [1, 2]. В работе [1] каждый вектор $\mathbf{z}_i \in \mathbb{R}^{N_1 \times 1}$, $i = \overline{1, q}$, (одномерный временной ряд) состоит из последовательности измерений координатной зависимости $z_i = z_i(t)$ в моменты времени $t_i \in \mathbb{R}^{N_1 \times 1}$. Вектор $\mathbf{z}_i$ моделируется полиномиальной регрессией:

$$\mathbf{z}_i = \mathbf{T}_i\boldsymbol{\beta} + \boldsymbol{\varepsilon}_i, \qquad (1)$$

где $\boldsymbol{\beta}$ — вектор коэффициентов регрессии размерности $(m + 1) \times 1$, $\boldsymbol{\varepsilon}_i$ — Гауссов шум с нулевым средним, а $\mathbf{T}_i$ — регрессионная матрица, которая зависит от типа используемой регрессионной модели. Для полиномиальной регрессии $\mathbf{T}_i$ имеет вид стандартной матрицы Вандермонта:

$$\mathbf{T}_i = \begin{bmatrix} 1 & t_i[1] & (t_i[1])^2 & \dots & (t_i[1])^m \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 1 & t_i[N_i] & (t_i[N_i])^2 & \dots & (t_i[N_i])^m \end{bmatrix}. \qquad (2)$$

В основе модели одновременной кластеризации и выравнивания лежит модель смеси регрессий, в которую вводятся четыре независимых параметра преобразований выравнивания и масштабирования во времени и пространстве $\{\Phi_i\} = \{a_i, b_i, c_i, d_i\}$ (параметры a_i и b_i описывают масштабирование и сдвиг во времени, а параметры c_i и d_i — масштабирование и смещение в пространстве измерений) [3]. Полиномиальная регрессия для одномерного случая имеет вид

$$\mathbf{z}_i = c_i\boldsymbol{\Upsilon}_i\boldsymbol{\beta}_k + d_i + \boldsymbol{\varepsilon}_i, \qquad (3)$$

где матрица $\boldsymbol{\Upsilon}_i$ получается из $\mathbf{T}_i$ (2) подстановкой $\mathbf{t}_i \to a_i\mathbf{t}_i - b_i$; параметры $\boldsymbol{\beta}_k$ определяют полиномиальную регрессию для траекторий из k-го кластера $(k = \overline{1, K})$; $\boldsymbol{\varepsilon}_i$ — Гауссов шум с нулевым средним и дисперсией $\sigma_k^2\mathbf{I}$. Поэтому распределение плотности условной вероятности имеет вид

$$p_k(\mathbf{z}_i|a_i, b_i, c_i, d_i) = \mathcal{N}(\mathbf{z}_i|c_i\boldsymbol{\Upsilon}_i\boldsymbol{\beta}_k + d_i, \sigma_k^2\mathbf{I}).$$

Плотность вероятности для кривой $\mathbf{z}_i$ однозначно задается соответствующим множеством параметров $\{\Phi_i\}$, которые подлежат определению. Задача кластеризации кривых решается как стандартная задача оценки значений скрытых переменных. Каждый из параметров преобразования в выражении (3) рассматривается как характерная для $\mathbf{z}_i$ случайная переменная с заранее известным распределением вероятности для кластера. Параметры преобразования в (3) и параметры $\boldsymbol{\beta}_k$, $k = \overline{1, K}$ полиномиальной регрессии оцениваются одновременно посредством

алгоритма ожидания и максимизации правдоподобия (*Expectation Maximization*) (EM-алгоритма) [4]. Однако число определяемых кластеров K не может быть большим, и на практике $K \leqslant 10$. Поэтому для набора достаточно неоднородных траекторий, определяемые кластеры траекторий также будут неоднородными. Для неоднородного кластера полиномиальная регрессия является весьма общим представлением. Чтобы продемонстрировать неоднородность кластеров (а следовательно, недоклассификацию траекторий) требуются методы сокращения размерности. Поэтому для анализа проекций траекторий кластера по каждой из пространственных координат могут быть использованы модели линейных динамических систем ограниченной размерности. При этом для удобства описания вместо матрицы траекторий (измерений)

$$\mathbf{Z} = [\mathbf{z}_1, ..., \mathbf{z}_q]$$

с векторами-столбцами $\mathbf{z}_i$, $i = \overline{1, q}$, (1) и (3) используется транспонированная матрица, т. е.

$$\mathbf{Z} = \mathbf{Z}^{\mathrm{T}} \in \mathbb{R}^{q \times N}.$$

1. Модель линейной динамической системы

Пусть $\mathbf{X} = \mathbf{X}[\overline{1, p}; \overline{1, N}]$ — многомерный временной ряд для вектора состояния, имеющего размерность $p \times N$, а $\mathbf{Z} = \mathbf{Z}[\overline{1, q}; \overline{1, N}]$ — многомерный наблюдаемый временной ряд размерности $q \times N$, причем в общем случае $p < q$ (размерность линейной динамической системы меньше размерности измерений (траекторий)). Как шум состояния $\mathbf{U} = \mathbf{U}[\overline{1, p}; \overline{1, N}]$, так и шум измерений $\mathbf{W} = \mathbf{W}[\overline{1, q}; \overline{1, N}]$ — многомерные временные ряды для Гауссовых случайных переменных с нулевым средним и ковариационными матрицами $\mathbf{Q}$ и $\mathbf{R}$, соответственно. Линейные инвариантные относительно времени динамические системы, называемые также линейными Гауссовыми моделями в пространстве состояний, описываются двумя уравнениями для векторов-столбцов $\mathbf{x}[k] = \mathbf{X}[\overline{1, p}; k]$, $\mathbf{z}[k] = \mathbf{Z}[\overline{1, q}; k]$, $\mathbf{u}[k] = \mathbf{U}[\overline{1, p}; k]$ и $\mathbf{w}[k] = \mathbf{W}[\overline{1, q}; k]$ (индекс k представляет дискретное время):

$$\mathbf{x}[k + 1] = \mathbf{F}\mathbf{x}[k] + \mathbf{u}[k]; \quad (4)$$

$$\mathbf{z}[k] = \mathbf{G}\mathbf{x}[k] + \mathbf{w}[k], \quad (5)$$

где $\mathbf{F}$ — матрица переходов; $\mathbf{G}$ — матрица наблюдений. В теории линейной фильтрации временной ряд $\mathbf{Z}$ для вектора наблюдений рассматривается, как зашумленный детерминированный временной ряд $\mathbf{X}$ для вектора состояния. В теории Байесовской фильтрации отличающиеся одним временным шагом векторы-столбцы (значения переменных состояния в последовательные дискретные моменты времени), на основе уравнения (4) объединяются в Гауссову случайную переменную со статистическими характеристиками шума состояния $\mathbf{u}$ [5]. Для Гауссовой случайной переменной $\mathbf{w}$ аналогичная комбинация одновременных векторов-столбцов переменной состояния системы и переменной наблюдения формируется на основе уравнения (5) [5]. В результате условные распределения вероятности для векторов наблюдения и состояния линейной динамической системы имеют вид

$$P(\mathbf{x}[k]\mathbf{x}[k-1]) = (2\pi)^{-p/2}|\mathbf{Q}|^{-1/2}\exp\Big(-\frac{1}{2}(\mathbf{x}[k] - \mathbf{F}\mathbf{x}[k-1])^{\mathrm{T}}\mathbf{Q}^{-1}(\mathbf{x}[k] - \mathbf{F}\mathbf{x}[k-1])\Big); \quad (6)$$

$$P(\mathbf{z}[k]|\mathbf{x}[k]) = (2\pi)^{-q/2}|\mathbf{R}|^{-1/2}\exp\Big(-\frac{1}{2}(\mathbf{z}[k] - \mathbf{G}\mathbf{x}[k])^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{z}[k] - \mathbf{G}\mathbf{x}[k])\Big), \quad (7)$$

где $|...|$ обозначает определитель матрицы. Как и распределения (6) и (7), исходное распределение вероятности состояний в момент времени $k = 1$ также предполагается Гауссовым со средним $\pi[1]$ и вариацией $\mathbf{V}[1]$:

$$P(\mathbf{x}[1]) = (2\pi)^{-p/2}[\mathbf{V}[1]]^{-1/2}\exp\Big(-\frac{1}{2}(\mathbf{x}[1] - \boldsymbol{\pi}[1])^{\mathrm{T}}\mathbf{V}[1]^{-1}(\mathbf{x}[1] - \boldsymbol{\pi}[1])\Big). \quad (8)$$

В уравнении (4) состояние системы (линейно) зависит только от предыдущего состояния, отстоящего на один временной шаг (марковский процесс). Поэтому с учетом свойства марковости условных вероятностей формула для совместной вероятности $P(\mathbf{X}, \mathbf{Z})$ имеет вид

$$P(\mathbf{X}, \mathbf{Z}) = P(\mathbf{x}[1])\Big(\prod_{k=2}^{N} P(\mathbf{x}[k]|\mathbf{x}[k-1])\Big)\Big(\prod_{k=1}^{N} P(\mathbf{z}[k]|\mathbf{x}[k])\Big). \quad (9)$$

Из выражений (6)—(8) следует, что логарифм $F(\mathbf{X}, \mathbf{Z})$ (9) является квадратичной формой (постоянный член опущен):

$$\log(P(\mathbf{X}, \mathbf{Z})) = -\frac{1}{2}\sum_{k=2}^{N}(\mathbf{x}[k] - \mathbf{F}\mathbf{x}[k-1])^{\mathrm{T}}\mathbf{Q}^{-1}(\mathbf{x}[k] - \mathbf{F}\mathbf{x}[k-1]) - \frac{N-1}{2}\log(|\mathbf{Q}|) - \frac{1}{2}\sum_{k=1}^{N}(\mathbf{z}[k] - \mathbf{G}\mathbf{x}[k])^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{z}[k] - \mathbf{G}\mathbf{x}[k]) - \frac{N}{2}\log(|\mathbf{R}|) - \frac{1}{2}(\mathbf{x}[1] - \boldsymbol{\pi}[1])^{\mathrm{T}}\mathbf{V}[1]^{-1}(\mathbf{x}[1] - \boldsymbol{\pi}[1]) - \frac{1}{2}\log(|\mathbf{V}[1]|). \quad (10)$$

EM-алгоритм. В работах [7—10] описано использование алгоритма ожидания и максимизации правдоподобия (*EM*-алгоритма) для оценки параметров линейной динамической системы (4)—(5) на основе многомерного наблюдаемого временного ряда $\mathbf{Z}$. *E*-шаг описываемого *EM*-алгоритма состоит в вычислении условного среднего логарифма правдоподобия (10):

$$LL = E[\log(P(\mathbf{X}, \mathbf{Z}))|\mathbf{Z}]. \quad (11)$$

Из выражения (10) следует, что LL (11) зависит от трех типов условных средних (ожиданий), для которых используются следующие обозначения:

$$\langle \mathbf{x}[k] \rangle = E[\mathbf{x}[k]|\mathbf{z}]; \quad (12)$$

$$\mathbf{P}[k] = E[\mathbf{x}[k]\mathbf{x}[k]^{\mathrm{T}}[\mathbf{z}]; \quad (13)$$

$$\mathbf{P}[k;\ k-1] = E[\mathbf{x}[k]\mathbf{x}[k-1]^{\mathrm{T}}|\mathbf{z}]. \quad (14)$$

В момент времени k оценка состояния $\langle \mathbf{x}[k] \rangle$ (12) зависит от прошлых $\mathbf{z}[\overline{1,\ k-1}]$ и будущих $\mathbf{z}[\overline{k+1,\ N}]$ наблюдений [6]. Следовательно, она отличается от оценки, вычисленной посредством фильтра Калмана, который оценивает состояние только на основе прошлых наблюдений [5]. Вычисление условных средних (ожиданий) (12)—(14) на E-шаге EM-алгоритма приводится после описания M-шага, т. е. оценки параметров линейной динамической системы (4)—(5).

M-шаг. Параметры линейной динамической системы (4) и (5) — это матрицы $\mathbf{F}$ и $\mathbf{G}$, а также статистические характеристики $\mathbf{Q}$ и $\mathbf{R}$ распределений (6) и (7), соответственно, и характеристики $\boldsymbol{\pi}[1]$ и $\mathbf{V}[1]$ распределения (8). На основе оценок (12)—(14) и наблюдения $\mathbf{z}[\overline{1,\ N}]$ каждый из параметров линейной динамической системы оценивается из условия равенства нулю соответствующей частной производной условного среднего логарифма правдоподобия LL (11). В результате

$$\mathbf{F} = \left(\sum_{k=1}^{N} \mathbf{P}[k;\ k-1] \right) \left(\sum_{k=2}^{N} \mathbf{P}[k-1] \right)^{-1}; \quad (15)$$

$$\mathbf{Q} = \frac{1}{N-1} \left(\sum_{k=2}^{N} \mathbf{P}[k] - \mathbf{F} \sum_{k=2}^{N} \mathbf{P}[k;\ k-1] \right); \quad (16)$$

$$\mathbf{G} = \left(\sum_{k=1}^{N} \mathbf{z}[k]\langle \mathbf{x}[k] \rangle^{\mathrm{T}} \right) \left(\sum_{k=1}^{N} \mathbf{P}[k] \right)^{-1}; \quad (17)$$

$$\mathbf{R} = \frac{1}{N-1} \sum_{k=1}^{N} (\mathbf{z}[k]\mathbf{z}[k]^{\mathrm{T}} - \mathbf{G}\langle \mathbf{x}[k] \rangle \mathbf{z}[k]^{\mathrm{T}}); \quad (18)$$

$$\boldsymbol{\pi}[1] = \langle \mathbf{x}[1] \rangle; \quad (19)$$

$$\mathbf{V}[1] = E[(\mathbf{x}[1] - \langle \mathbf{x}[1] \rangle)(\mathbf{x}[1] - \langle \mathbf{x}[1] \rangle)^{\mathrm{T}}|z] = \mathbf{P}[1] - \langle \mathbf{x}[1] \rangle \langle \mathbf{x}[1] \rangle^{\mathrm{T}}. \quad (20)$$

E-шаг. Используются следующие обозначения: $\langle \mathbf{x}[k;\ l] \rangle = E[\mathbf{x}[k]|\mathbf{z}[\overline{1,\ l}]]$ (таким образом, для условного среднего (12) имеем $\langle \mathbf{x}[k] \rangle \equiv \langle \mathbf{x}[k;\ N] \rangle$) и $\mathbf{V}[k;\ l] = E[(\mathbf{x}[k] - \langle \mathbf{x}[k] \rangle)(\mathbf{x}[k] - \langle \mathbf{x}[k] \rangle)^{\mathrm{T}}|\mathbf{z}[\overline{1,\ l}]]$. Используя оценки (15)—(20), сначала приводятся рекурсии вперед для линейного фильтра Калмана

$$\langle \mathbf{x}[k;\ k-1] \rangle = \mathbf{F}\langle \mathbf{x}[k-1;\ k-1] \rangle; \quad (21)$$

$$\mathbf{V}[k;\ k-1] = \mathbf{F}\mathbf{V}[k-1;\ k-1]\mathbf{F}^{\mathrm{T}} + \mathbf{Q} \quad (22)$$

и вычисляются матрицы усиления Калмана

$$\mathbf{K}[k] = \mathbf{V}[k;\ k-1]\mathbf{G}^{\mathrm{T}}(\mathbf{G}\mathbf{V}[k;\ k-1]\mathbf{G}^{\mathrm{T}} + \mathbf{R})^{-1}; \quad (23)$$

$$\langle \mathbf{x}[k;\ k] \rangle = \langle \mathbf{x}[k;\ k-1] \rangle + \mathbf{K}[k](\langle \mathbf{y}[k] \rangle - \mathbf{G}\langle \mathbf{x}[k;\ k-1] \rangle), \quad (24)$$

$$\mathbf{V}[k;\ k] = \mathbf{V}[k;\ k-1] - \mathbf{K}[k]\mathbf{G}\mathbf{V}[k;\ k-1], \quad (25)$$

где $\langle \mathbf{x}[1] \rangle = \boldsymbol{\pi}[1]$ из (19) и $\mathbf{V}[1]$ (20). Вывод рекурсии (21)—(25) основан на Байесовском подходе [10].

На основании работ [7—10] для вычисления оценки $\langle \mathbf{x}[k] \rangle \equiv \langle \mathbf{x}[k;\ N] \rangle$ (12) и условного среднего $\mathbf{P}[k] \equiv \mathbf{V}[k;\ N] + \langle \mathbf{x}[k;\ N] \rangle \langle \mathbf{x}[k;\ N] \rangle^{\mathrm{T}}$ (13) выполняется рекурсия назад (здесь $\mathbf{J}[k]$ — вспомогательная матрица):

$$\mathbf{J}[k-1] = \mathbf{V}[k-1;\ k-1]\mathbf{F}^{\mathrm{T}}(\mathbf{V}[k;\ k-1])^{-1}; \quad (26)$$

$$\langle \mathbf{x}[k-1;\ N] \rangle = \langle \mathbf{x}[k-1;\ k-1] \rangle + \mathbf{J}[k-1](\langle \mathbf{x}[k;\ N] \rangle - \mathbf{F}\langle \mathbf{x}[k-1;\ k-1] \rangle); \quad (27)$$

$$\mathbf{V}[k-1;\ N] = \mathbf{V}[k-1;\ k-1] + \mathbf{J}[k-1](\mathbf{V}[k;\ N] - \mathbf{V}[k;\ k-1])\mathbf{J}[k-1]^{\mathrm{T}}. \quad (28)$$

Условное среднее (14) имеет вид

$$\mathbf{P}[k;\ k-1] \equiv \mathbf{V}[k;\ k-1;\ N] + \langle \mathbf{x}[k;\ N] \rangle \langle \mathbf{x}[k-1;\ N] \rangle^{\mathrm{T}},$$

где

$$\mathbf{V}[k;\ k-1;\ N] = E[(\mathbf{x}[k] - \langle \mathbf{x}[k] \rangle)(\mathbf{x}[k-1] - \langle \mathbf{x}[k-1] \rangle)^{\mathrm{T}}|\mathbf{z}[\overline{1,\ N}]].$$

Значения $\mathbf{V}[k;\ k-1;\ N]$ также вычисляются рекурсией назад

$$\mathbf{V}[k-1;\ k-2;\ N] = \mathbf{V}[k-1;\ k-1]\mathbf{J}[k-2]^{\mathrm{T}} + \mathbf{J}[k-1](\mathbf{V}[k;\ k-1;\ N] - \mathbf{F}\mathbf{V}[k-1;\ k-1])\mathbf{J}[k-2]^{\mathrm{T}}, \quad (29)$$

которая инициализируется как

$$\mathbf{V}[N;\ N-1;\ N] = (\mathbf{I} - \mathbf{K}[N]\mathbf{G})\mathbf{F}\mathbf{V}[N-1;\ N-1].$$

Итерации в виде последовательного чередования M- и E-шагов с оценкой изменения ожидаемого условного логарифма правдоподобия LL (11) обеспечивают оценку $\langle \mathbf{x}[\overline{1,\ N}] \rangle$ (12) временного ряда состояния размерности $p \times N$.

Эффективность описанного EM-алгоритма демонстрируется на примере анализа результатов кластеризации траекторий по данным радаров международного аэропорта г. Сан-Франциско, находящихся в свободном доступе на сайте https://c3.nasa.gov/daslilink/resources/132/.

2. Численный эксперимент

В настоящей работе выполняется анализ неоднородности кластеров траекторий движения самолетов, полученных с помощью метода полиномиальных регрессий [1, 2]. Этот метод, несмотря на то, что является одним из надежных, так как учитывает реальную форму траекторий в пространстве, тем не менее, может приводить к довольно абстрактной линии тренда в неоднородных кластерах. Последующее применение моделей линейных динамических систем в случае неоднородных кластеров проявляет их тонкую структуру. Модели линейных динамических систем позволяют значительно сократить размерность многомерного временного ряда. Неоднородная структура кластера выявляется при рассмотрении проекций траекторий, соответствующих этому кластеру, на координатные оси

трехмерного пространства. Эффективность такого подхода демонстрируется на примере выборки траекторий первых 110 самолетов, идущих на посадку в международном аэропорту и зарегистрированных радаром TRACON 1 января 2006 года (https:// c3.nasa.gov/dashMinK/resources/132/). Начало координат (0,0,0) совпадает с положением радара, интервал времени между точками регистрации составляет около 5 с. В работе рассматриваются только 160 последних точек каждой траектории, чтобы исключить случайные маневры самолетов до начала захода на посадку.

На рис. 1 (см. вторую сторону обложки) показано трехмерное представление всех анализируемых траекторий самолетов.

Методом полиномиальной регрессии [1, 2] анализируемые траектории самолетов (см. рис. 1) разделяют на пять кластеров, как показано на рис. 2 (см. вторую сторону обложки): розовый — с 16 траекториями; зеленый — с 13 траекториями; синий — с 3 траекториями; черный — с 37 траекториями; красный — с 38 траекториями. Каждый кластер соответствует "посадочному" паттерну, который задается существующим маршрутом посадки.

Кроме того, в анализируемой выборке траекторий (см. рис. 1) выделяются три посторонние траектории (рис. 3), которые не могут быть отнесены ни к одному из выявленных кластеров.

На рис. 4 (см. вторую сторону обложки) показано изменение координат самолетов отдельно по каждой оси x, y, z в соответствии с последовательностью моментов времени k регистрации радаром. Линия тренда (полиномиальная регрессия), выделенная жирной линией, представляет обобщенную форму траекторий в кластере. Несмотря на то что снижение самолетов происходит неодновременно, сходство их траекторий обусловлено существованием типичных маршрутов посадки, например, когда самолеты выстраиваются в "караван", ожидая своей очереди приземления на посадочную полосу.

Для траекторий пяти выявленных кластеров, выделенных на рис. 2 и 4 (см. вторую сторону обложки) различными цветами, на рис. 5—9 показаны их соответствующие проекции на оси x и y вместе с компонентами вектора состояний линейной динамической системы размерности $p = 2$. Линия *1* на рис. 5—9 соответствует единственной компоненте вектора состояний при $p = 1$ и представляет полиномиальную регрессию на рис. 4. Линия *2* на рис. 5—9 соответствует второй компоненте вектора состояний при $p = 2$ и описывает тонкую структуру кластера. Проекции на оси x и y траекторий в кластерах

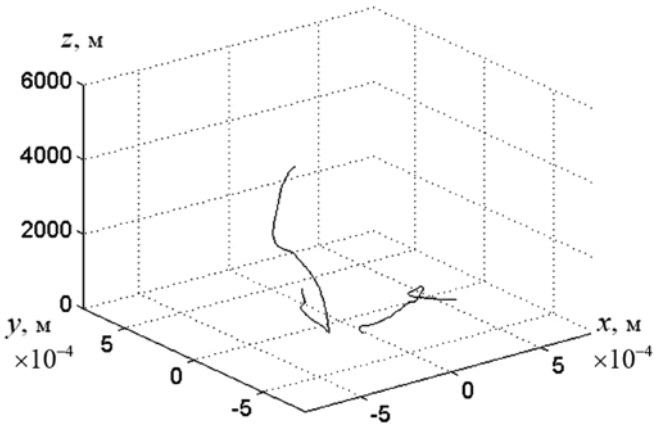


Рис. 3. Три посторонние траектории

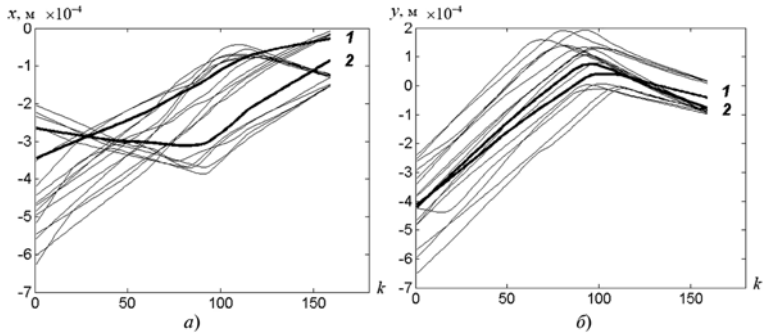


Рис. 5. Проекции на оси *x* и *y* траекторий розового кластера (см. рис. 2 и 4) показаны совместно с компонентами вектора состояний (линии *1* и *2*):
а — линейная модель проекций на ось *x*; *б* — линейная модель проекций на ось *y*

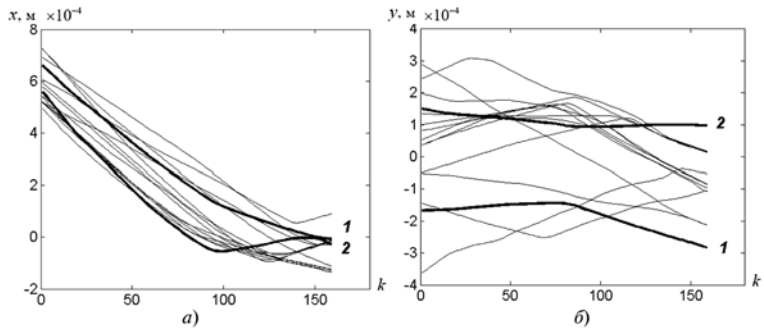


Рис. 6. Зеленый кластер на рис. 2 и 4:
а — линейная модель проекций на ось *x*; *б* — линейная модель проекций на ось *y*

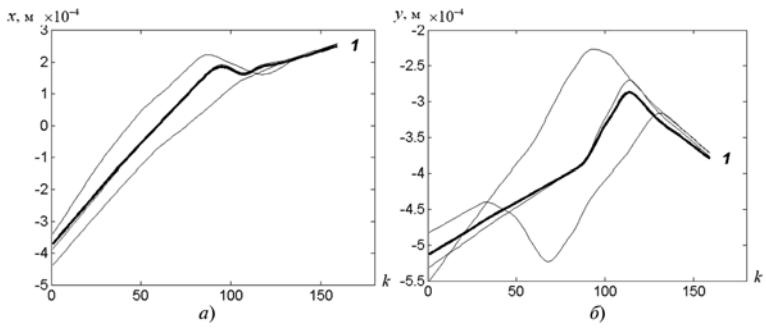


Рис. 7. Синий кластер на рис. 2 и 4:
а — линейная модель проекций на ось *x*; *б* — линейная модель проекций на ось *y* (определена одна компонента)

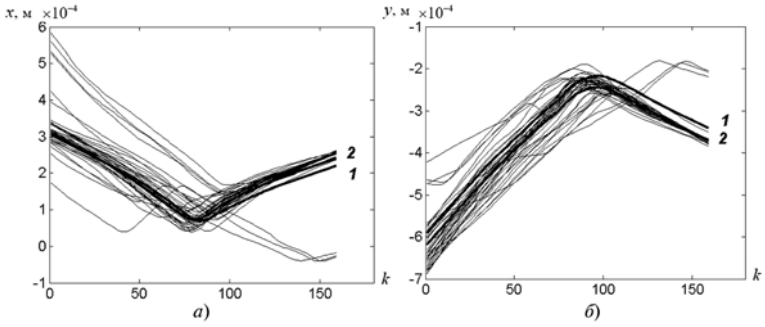


Рис. 8. Черный кластер на рис. 2 и 4:
а — линейная модель проекций на ось *x*; *б* — линейная модель проекций на ось *y*

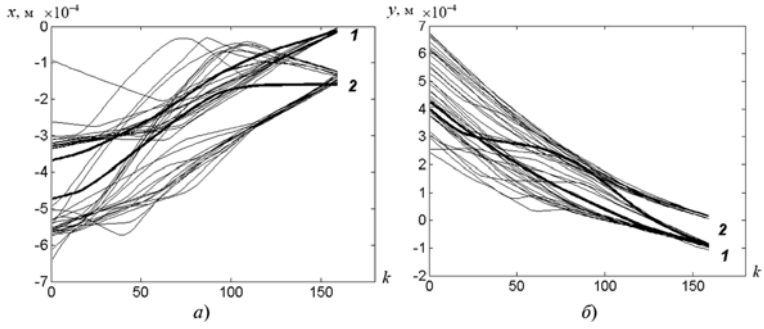


Рис. 9. Красный кластер на рис. 2 и 4:
а — линейная модель проекций на ось *x*; *б* — линейная модель проекций на ось *y*

на рис. 4 представляют переходные зависимости от времени и поэтому описываются линейной моделью. Проекции на ось *z* траекторий в кластерах (см. рис. 2 и 4) не позволяют дифференцировать эти кластеры, поэтому далее они не анализируются.

На рис. 5—9 видно, как в случае неоднородных кластеров модели линейных динамических систем выявляют тонкую структуру кластера, разделяя исходные кластеры на два подкластера.

Заключение

В настоящей работе метод линейных динамических систем демонстрируется на примере снижения размерности для результатов кластеризации траекторий самолетов, идущих на посадку в зоне крупного международного аэропорта. Эффективность метода линейных динамических систем достигается благодаря обучению параметров посредством EM-алгоритма.

Список литературы

1. **Gaffney S., Smyth P.** Joint probabilistic curve clustering and alignment / Saul L., Weiss Y., Bottou L., eds // Proc. of Neural Information Processing Systems (NIPS 2004). December 13—18, 2004. Vancouver, British Columbia, Canada. Advances in Neural Information Processing Systems. V. 17. Cambridge, MA: MIT Press, 2005. P. 473—480.
2. **Кухаренко Б. Г., Солнцева М. О.** Кластеризация управляемых объектов на основе сходства их многомерных траекторий // Информационные технологии. 2014. № 5. С. 3—7.
3. **Gaffney S., Smyth P.** Trajectory clustering with mixtures of regression models / Chaudhuri S., Madigan D., eds. // Proceedings of Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August 15—18 1999. New York, NY: ACM Press, 1999. P. 63—72.
4. **Dempster A. P., Laird N. M., Rubin D. B.** Maximum likelihood from incomplete data via the EM algorithm // Proc. of the Royal Statistical Society. 1976. P. 1—38.
5. **Simon D.** Optimal State Estimation: Kalman, H_∞ and Nonlinear Approaches. Hoboken, New Jersey: John Wiley & Sons, Inc., 2006.
6. **Rauch H. E.** Solutions to the linear smoothing problem // IEEE Transactions on Automatic Control. 1963. V. 8. P. 371—372.
7. **Shumway R. H., Stoffer D. S.** Time Series Analysis and Its Applications. New York: Springer, 2011.
8. **Shumway R. H., Stoffer D. S.** An approach to time series smoothing and forecasting using the EM algorithm // Journal of Time Series Analysis. 1982. V. 3, N. 4. P. 253—264.
9. **Shumway R., Stoffer D.** Dynamic linear models with switching // Journal of the American Statistical Association. 1992. V. 86. P. 763—769.
10. **Roweis S., Ghaliramani Z.** A unifying review of linear Gaussian models // Neural Computation. 1999. V. 11, N. 2. P. 305—345.

B. G. Kukharenko, Leading research scientist, Blagonravov Institute of Engineering Science of RAS,
M. O. Solntseva, Post-graduate student, Moscow Institute of Physics and Technology (SU)

Analysis of Multi Dimensional Trajectory Clustering by Models of Linear Dynamical System

For clustering multidimensional trajectories polynomial regression method with parameter leaning by the Expectation-Maximization algorithm is in use. The method based on polynomial regression is characterized by the joint clustering and continuous alignment of curve sets in time and space. Nevertheless, a number of defined clusters can't be large enough. Thus, for a set of sufficiently heterogeneous trajectories, the defined clusters are heterogeneous also. For the inhomogeneous cluster, its polynomial regression is a too strong abstraction. To demonstrate the cluster heterogeneity (and, thus, non full clustering trajectories) a method of dimension reducing is in need. So, linear dynamical system models are applied to object multi dimensional trajectory clustering by polynomial regression method. An advantage of linear dynamical systems is reducing clustering result dimension. For trajectory coordinate projections in a cluster most informative component (polynomial regression) is extracted and the cluster fine structure is appeared. An efficiency of linear dynamical systems is demonstrated by example of clustering results of airplane flight tracks in an airport space.

Keywords: *data mining, multi-dimensional trajectories, clustering, polynomial regression, Kalman filter, Rauch smoother, linear dynamical systems, Expectation-Maximization algorithm*

References

1. **Gaffney S., Smyth P.** Joint probabilistic curve clustering and alignment / Saul L., Weiss Y., Bottou L. (eds.). *Proc. of Neural Information Processing Systems (NIPS 2004)*. December 13—18, 2004, Vancouver, British Columbia, Canada. Advances in Neural Information Processing Systems. V. 17. Cambridge, MA: MIT Press. 2005. P. 473—480.
2. **Kukharenko B. G., Solntseva M. O.** Klasterizacia upravlyaemych objektov na osnove shodstva ih mnogomernyh trajektoryi. *Informacionnye tehnologii*. 2014, N. 5. P. 3—7.
3. **Gaffney S., Smyth P.** Trajectory clustering with mixtures of regression models / Chaudhuri S., Madigan D., eds. *Proceedings of Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. August 15—18, 1999. New York, NY: ACM Press, 1999. P. 63—72.
4. **Dempster A. P., Laird N. M., Rubin D. B.** Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*. 1976. P. 1—38.
5. **Simon D.** Optimal State Estimation: Kalman, H_∞ and Nonlinear Approaches. Hoboken, New Jersey: John Wiley & Sons, Inc., 2006.
6. **Rauch H. E.** Solutions to the linear smoothing problem. *IEEE Transactions on Automatic Control*. 1963. V. 8. P. 371—372.
7. **Shumway R. H., Stoffer D. S.** Time Series Analysis and Its Applications. New York: Springer, 2011.
8. **Shumway R. H., Stoffer D. S.** An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*. 1982. V. 3, N. 4. P. 253—264.
9. **Shumway R., Stoffer D.** Dynamic linear models with switching. *Journal of the American Statistical Association*. 1992. V. 86. P. 763—769.
10. **Roweis S., Ghahramani Z.** A unifying review of linear Gaussian models. *Neural Computation*. 1999. V. 11, N. 2. P. 305—345.

УДК 004.89 + 004.021

П. В. Казаков, канд. техн. наук, доц., e-mail: pvk_mail@list.ru
Брянский государственный технический университет

Использование дифференциальной эволюции при определении множества Парето генетическими алгоритмами многокритериальной оптимизации

Рассматривается новый способ повышения эффективности работы генетических алгоритмов при определении множества Парето в задачах многокритериальной оптимизации. Он основан на использовании принципов дифференциальной эволюции при развитии популяции и формировании ее новых индивидов. Приводится сравнительный анализ эффективности использования дифференциальной эволюции при решении задач многокритериальной оптимизации разной сложности.

Ключевые слова: *многокритериальная оптимизация, множество и граница Парето, многокритериальные генетические алгоритмы, дифференциальная эволюция*

Введение

Необходимость решения задач многокритериальной оптимизации (МО) возникает в самых разных теоретических и прикладных областях. Качество решения задач МО зависит от точности найденного множества Парето [1, 2]. В то же время, его определение остается сложным и плохо автоматизированным процессом ввиду теоретически неограниченного числа недоминируемых решений, образующих множество Парето, а также вычислительно трудоемкой процедуры его построения. Применяемые для этих целей традиционные численные методы МО во многом ориентированы на преобразование задачи со множеством критериев к однокритериальной ее постановке. Подобные упрощения не только снижают точность найденных решений, но и существенно ограничивают возможность нахождения всего множества Парето. Поэтому разработка и совершенствование алгоритмов МО остаются актуальным направлением исследований.

В настоящее время одним из перспективных подходов к определению множества Парето в задачах МО является использование эволюционных методов, а именно специальных версий генетических алгоритмов для многокритериальной оптимизации (МГА) [3—5]. Они относятся к многоточечным методам оптимизации, где каждое решение ассоциируется с так называемыми индивидами [6]. Вместе они образуют популяцию, итерационное изменение которой в процессе оптимизации позволяет эффективно исследовать пространства поиска (переменных и критериев). В то же время возможное многообразие постановок задачи МО, сложность анализа многомерных пространств переменных и критериев, вычислительная трудоемкость определения множества Парето не позволяют выбрать универсальный МГА. Так, исследование возможностей наиболее известных МГА показало снижение качества их результата при увеличении числа критериев в решаемых задачах МО [7]. Вместе с тем потенциал применения МГА для определения множества Парето остается достаточно большим, поэтому актуальным является разработка новых и совершенствование существующих МГА. Достигается это прежде всего интеграцией в МГА новых способов исследования пространства поиска, например, с использованием методов параллельных вычислений [8], коэволюционных технологий [9, 10] и др. В настоящей работе предлагается подход к развитию

возможностей МГА посредством использования в них принципов дифференциальной эволюции. В дополнение к достаточно развитым возможностям современных МГА исследовать пространство критериев она позволяет повысить разнообразие популяции, а также эффективность анализа решений в пространстве переменных.

1. Основные сведения о дифференциальной эволюции в генетических алгоритмах

Идея дифференциальной эволюции (*Differential Evolution* — DE) была предложена Прайсом и Сторном (K. Price, R. Storn) [11] как расширение генетических алгоритмов для повышения их точности и скорости схождения при оптимизации с непрерывными значениями переменных. Главная особенность DE состоит в том, что каждое новое поколение формируется с участием индивидов текущей и промежуточной (пробной) популяций. Ее члены образуются посредством определения различий между выбранными по определенным правилам индивидами текущей популяции. Конкретная реализация этого зависит от так называемой схемы DE, параметры которой записываются в виде следующей последовательности: DE/α/β/γ. Здесь α указывает на изменяемый родительский индивид; β — число различных индивидов, участвующих в изменении α; γ определяет тип используемого кроссинговера (exp — экспоненциальный, bin — бинарный). В случае экспоненциального кроссинговера значения изменяемых генов потомка берутся только у первого индивида-родителя. При бинарном кроссинговере каждое значение гена потомка формируется из двух соответствующих генов индивидов-родителей. В качестве родительского может выступать как лучший индивид текущей популяции (α = best), так и случайно выбранный индивид (α = rand). Алгоритм DE с классической схемой выполнения состоит в следующем.

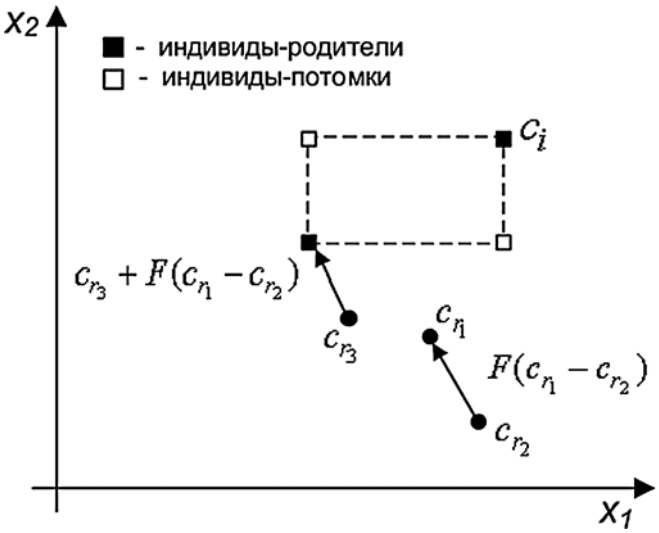


Рис. 1. Формирование новых индивидов по схеме DE/rand/1/bin

Алгоритм. Differential Evolution — DE/rand/1/bin (CR, F, n_P, n_G), где $CR \in [0, 1]$ — параметр управления интенсивностью кроссинговера; $F > 0$ — параметр, масштабирующий степень изменения индивидов; n_P — размер популяции; n_G — число поколений.

Шаг 1. Создать начальную популяцию размером n_P и вычислить значения пригодности ее индивидов $f(c_i)$, $i = 1, ..., n_P$, $t = 0$.

Шаг 2. Для каждого индивида популяции c_i, $i = 1, ..., n_P$, выполнить следующие действия.

2.1. Выбрать три разных случайных числа (индекса индивидов) $r_1, r_2, r_3 \in 1, ..., n_P | r_1 \neq r_2 \neq r_3$.

2.2. Выбрать случайное число $k \in 1, ..., n$, где n — число оптимизируемых переменных в задаче МО.

2.3. На основе всех генов $c_{i,j}$, $j = 1, ..., n$, индивида c_i, сформировать промежуточный (пробный) индивид c'_i:

$$c'_{i,j}(t+1) = \begin{cases} c_{r_3,j}(t) + F(c_{r_1,j}(t) - c_{r_2,j}(t)), \\ \text{если } (\text{rand}(0, 1) < CR) \vee (j = k); \\ c_{i,j}(t), \text{ иначе.} \end{cases}$$

2.4. Выбрать индивид для следующего поколения:

$$c_i(t+1) = \begin{cases} c'_i(t+1), \text{ если } f(c'_i(t+1)) \leqslant f(c_i(t)); \\ c_i(t), \text{ иначе.} \end{cases}$$

Шаг 3. $t = t + 1$.

Шаг 4. Если $t > n_G$, то закончить, иначе перейти к шагу 2.

Как следует из приведенного алгоритма, при дифференциальной эволюции новое поколение формируется на основе предыдущего с использованием так называемых пробных индивидов c'_i, $i = 1, ..., n_P$. Они играют роль промежуточных индивидов-кандидатов в следующее поколение, их общее число совпадает с размером популяции. При этом для каждого основного индивида создается только один пробный. Для этого из текущей популяции случайно выбираются три разных индивида. Среди них индивид с индексом r_3 выполняет роль второго родителя, значения генов которого присваиваются пробному индивиду. Но вначале эти гены подвергаются мутации, уровень которой зависит от различия между индивидами с индексами r_1, r_2. В пространстве поиска они задают координаты вектора мутации, изменяющего положение второго индивида-родителя и влияющего на область размещения индивидов-потомков (рис. 1). В зависимости от состава текущей популяции вектор мутации динамически меняется, формируя траекторию поиска. При этом считается, что вектор мутации самоадаптируется к рельефу поверхности оптимизируемой функции в зависимости от положения популяции в поисковом пространстве [12]. На процесс дифференциальной эволюции влияют два управляющих параметра — CR и F, значения которых не меняются в процессе работы генетического алгоритма. Параметр CR (*Crossover Rate*) отвечает за кроссинговер и

определяет вероятность изменения каждого гена промежуточного индивида. Параметр F позволяет усилить или ослабить такие изменения, определяя тем самым силу мутации генов. Выбор значений этих параметров зависит от оптимизируемой функции, ее ограничений, размера популяции и, как правило, проводится экспериментально. Выполнение условия ($j = k$) гарантирует, что пробный и родительский индивиды будут отличаться хотя бы в одном гене. Из этапа 2.4 алгоритма видно, что в следующее поколение переходят индивиды не хуже (предполагается решение задачи минимизации) своих аналогов из текущего поколения. Было выявлено [11], что в итоге средняя пригодность всей популяции не будет ухудшаться.

Для управления процессом генерации и исследования новых решений можно использовать не только различные значения параметров CR, F, но и различные варианты схем DE. Друг от друга они отличаются реализацией кроссинговера, а также числом и правилами выбора индивидов для определения векторов в траектории направления поиска [13]. Анализ применения различных схем DE при решении задач однокритериальной оптимизации подтвердил эффективность механизмов дифференциальной эволюции при исследовании поисковых пространств большой размерности.

2. Использование дифференциальной эволюции в генетических алгоритмах многокритериальной оптимизации

Возможности дифференциальной эволюции впоследствии были применены и при решении задач многокритериальной оптимизации [13]. При этом особенности задачи МО потребовали корректировки существующих и создания новых схем DE. Прежде всего, это формулировка правил замены в новой популяции индивида-предка соответствующим потомком. Так, например, в одной из первых работ (автор H. Abbass) [14], посвященных применению DE при многокритериальной оптимизации, для этого было предложено выбирать в следующее поколение только недоминируемые индивиды. Существующие другие подходы к переносу принципов DE в область многокритериальной оптимизации во многом объединяются предложениями более широкого (на разных этапах DE) использования принципов Парето при формировании новой популяции. Так, в работе [15] была предложена общая концепция дифференциальной эволюции для многокритериальной оптимизации — MODE (*Multi-Objective Differential Evolution*). В ней классическая схема DE дополнена механизмами управления популяцией, использующимися в основных генетических алгоритмах МО [5]. Прежде всего, это оценка пригодности индивидов на основе принципов Парето, а также сохранение недоминируемых индивидов в Парето-архиве. В виде алгоритма MODE представляется следующим образом.

Алгоритм MODE (n_P, n_G, n_A, CR, F), где n_A — размер Парето-архива.

Шаг 1. Создать начальную популяцию размером n_P. Сформировать пустые Парето-архив $A = \varnothing$ размером n_A и временную популяцию $Q = \varnothing$; $t = 0$.

Шаг 2. Вычислить пригодность каждого индивида популяции. Все недоминируемые индивиды сохранить в Парето-архиве A.

Шаг 3. Выполнить над популяцией основные действия дифференциальной эволюции (операции 2.1—2.3 алгоритма "Differential Evolution — DE/rand/1/bin").

Шаг 4. Вычислить пригодность новых индивидов. Обновить текущую популяцию по следующему правилу:

$$c_i(t) = \begin{cases} c_i'(t+1), \text{ если } (c_i'(t+1) \succeq c_i(t)); \\ c_i(t), \text{ если } (c_i(t) \succeq c_i'(t+1)); \\ c_i(t);\ c_i'(t+1) \to Q, \text{ если } (c_i(t) \prec\succ c_i'(t+1)), \end{cases}$$

где $i = 1, \ldots, n_P$.

Шаг 5. Скопировать все индивиды текущей популяции во временную популяцию Q:

$$Q = Q + \{c_i(t)\},\ i = 1, \ldots, n_P.$$

Шаг 6. Сформировать из Q новую популяцию размером n_P посредством последовательных сортировок по рангам Парето, а также значениям параметра *crowding distance* (протяженность скучивания точек вдоль границы Парето) [16] всех индивидов популяции Q.

Шаг 7. Все недоминируемые индивиды новой популяции скопировать в Парето-архив. Если размер Парето-архива превысил значение n_A, то сократить его, оставив решения, наименее плотно размещенные на границе Парето (с наименьшими значениями параметра *crowding distance*).

Шаг 8. $t = t + 1$; $Q = \varnothing$. Если $t > n_G$, то закончить. Иначе перейти к шагу 3.

Основой алгоритма MODE для формирования новых индивидов является набор операций классической дифференциальной эволюции (схема DE/rand/1/bin). Однако при выборе в следующее поколение между промежуточным и текущим индивидами используется отношение "лучше" $c_i'(t+1) \succeq c_i(t)$, $i = 1, \ldots, n_P$. Его интерпретация означает, что отбору подлежит индивид, который лучше (в терминах принципов Парето) своего конкурента, либо его пригодность выше (по результатам вычисления *fitness*-функции), но возможны и другие интерпретации [12]. Также для увеличения числа и точности недоминируемых решений в названном алгоритме задействуются идеи двух наиболее известных алгоритмов МО — NSGA-II [16] (управления плотностью решений на границе Парето) и SPEA2 [17] (использование Парето-архива). При исследовании возможностей алгоритма MODE были получены результаты, превосходящие по некоторым индикаторам эффективности SPEA2 и NSGA-II [15].

Последующее развитие DE для задач многокритериальной оптимизации связано с разработкой для SPEA2 и NSGA-II модификаций, непосредственно использующих принципы дифференциальной эволюции. Такими алгоритмами являются NSDE [18], DEMO^{NS-II} [19] и DEMO^{SP2} [19]. Первая пара алгоритмов построена на базе NSGA-II, а последний алгоритм соответственно — на SPEA2, Несмотря на появление алгоритмов NSDE и DEMO^{NS-II} независимо друг от друга, они имеют одинаковые принципы работы, которые, так же как и в DEMO^{SP2}, заключаются в замене операторов кроссинговера и мутации базовых алгоритмов операциями дифференциальной эволюции. Сравнительное исследование этих алгоритмов на 16 различных тестовых задачах МО показали, что во многих случаях полученные DEMO^{NS-II} и DEMO^{SP2} результаты оказались лучше значений соответствующих показателей эффективности алгоритмов NSGA-II и SPEA2 [20]. Однако следует заметить, что решаемые тестовые задачи МО имели не более четырех критериев оптимизации. При этом с максимальным числом критериев наблюдалась слишком быстрая сходимость алгоритмов, использующих DE, к локально оптимальным границам Парето. Подбор и изменение значений управляющих параметров CR и F не позволили достичь лучших результатов в сравнении с МГА, не использующими DE. В связи с этим можно предположить, что простое использование операции рекомбинации, заимствованной из DE для однокритериальной оптимизации и учитывающей только особенности размещения решений в пространстве переменных, оказывается недостаточным при исследовании пространства критериев. Очевидно, что вследствие независимости этих пространств друг от друга, траектории поиска в них будут разными. Поэтому важно в схемах DE при рекомбинации непосредственно учитывать возможные траектории поиска в обоих пространствах (решений и критериев). В частности, это может быть реализовано в виде приведенной ниже предлагаемой схемы дифференциальной эволюции.

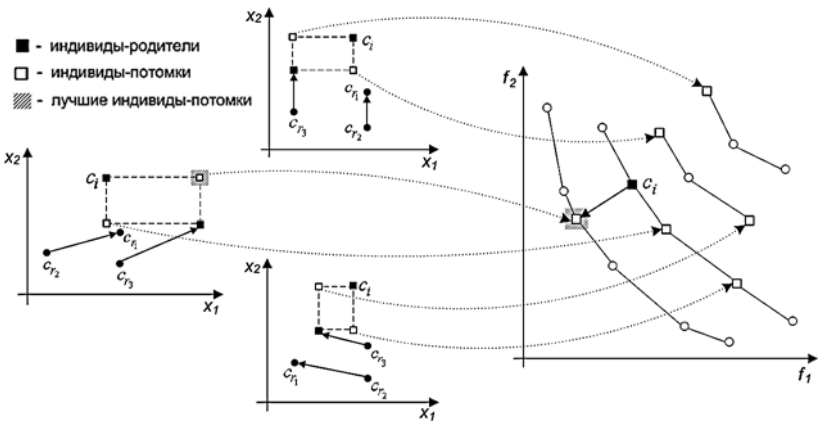


Рис. 2. Влияние выбора разных родительских индивидов в DE при поиске Парето-оптимальных решений

3. Схема дифференциальной эволюции DE/rand/1X/bin

Анализ различных схем дифференциальной эволюции [13] позволяет сделать вывод, что очень часто второй родитель, а также остальные (два и более) индивидов, необходимые для создания потомка, выбираются случайно. При этом не гарантируется, что полученный в итоге новый индивид окажется лучше своего родителя. В общем случае это приводит к снижению разнообразия популяции, а в условиях МО — к сохранению в ней неизменяемой в процессе поиска небольшой доли недоминируемых решений. Увеличение интенсивности операции рекомбинации для DE часто приводит к росту случайной составляющей в процессе оптимизации, потере найденных оптимальных решений. В то же время для увеличения вероятности создания индивидов, превосходящих своих родителей по оптимизируемым критериям, могут быть исследованы различные варианты векторов их изменений, образуемые множеством сочетаний выбранных для рекомбинации индивидов. Для классической схемы DE/rand/1/bin это означает, что каждый из трех индивидов с индексами r_1, r_2, r_3 может быть выбран в качестве второго родителя (рис. 2).

В случае, когда не одно из сочетаний индивидов в операторе рекомбинации не позволило получить индивида-потомка лучше своего индивида-родителя, последний переходит в следующее поколение. В целом, такая схема дифференциальной эволюции DE/rand/1X/bin должна обеспечить более активное замещение доминируемых индивидов популяции. Для управления найденными недоминируемыми решениями, их размещением на границе Парето предложенный вариант дифференциальной эволюции должен быть интегрирован в другие МГА. Алгоритм предлагаемой схемы дифференциальной эволюции для генетических алгоритмов многокритериальной оптимизации имеет следующий вид.

Алгоритм DE/rand/1X/bin (n_P, n_G, CR, F).

Шаг 1. Создать начальную популяцию размером n_P и вычислить значения пригодности ее индивидов, $t = 0$.

Шаг 2. Для каждого индивида популяции c_i, $i = 1, ..., n_P$ выполнить следующие действия.

2.1. Выбрать три разных случайных числа (индекса индивидов) q, r, $s \in 1, ..., n_P | q \neq r \neq s$.

2.2. Множество индивидов, выбранных кандидатами в родители, сделать пустым $R = \varnothing$.

2.3. Среди множества индексов индивидов $\{q, r, s\}$ выбрать любой, не принадлежащий R. Присвоить его переменной r_3 и считать индексом индивида-родителя. $R = R + \{r_3\} \mid r_3 \notin R$.

2.4. Переменным r_1, r_2 произвольно присвоить остальные выбранные индексы. $r_1 = q \vee r \vee s \mid r_1 \neq r_3$, $r_2 = q \vee r \vee s \mid r_2 \neq r_1 \neq r_3$.

2.5. Выбрать случайное число $k \in 1, ..., n$.

2.6. На основе всех генов $c_{i,j}, j = 1, ..., n$, индивида c_i, сформировать промежуточный индивид c'_i:

$$c'_{i,j}(t+1) = \begin{cases} c_{r_3,j}(t) + F(c_{r_1,j}(t) - c_{r_2,j}(t)), \\ \text{если } (\text{rand}(0,1) < CR) \vee (j = k); \\ c_{i,j}(t), \text{ иначе.} \end{cases}$$

2.7. Выбрать индивид для следующего поколения, либо создать новый пробный индивид в соответствии с одним из следующих правил:

а) если $c'_i(t+1) \succeq c_i(t)$, то $c_i(t+1) = c'_i(t+1)$;

б) если $c_i(t) \succeq c'_i(t+1)$ и $\{q, r, s\} \notin R$, то перейти к шагу 2.3;

в) если $c_i(t) \succeq c'_i(t+1)$ и $\{q, r, s\} \in R$, то $c_i(t+1) = c_i(t)$;

г) если $c_i(t) \prec\succ c'_i(t+1)$, то $c_i(t+1) = c_i(t)$ или $c_i(t+1) = c'_i(t+1)$.

Шаг 3. $t = t + 1$.

Шаг 4. Если $t > n_G$, то закончить, иначе перейти к шагу 2.

Таким образом, предлагаемый вариант DE отличается возможностью формирования пробного индивида с привлечением различных пар родительских индивидов. В них первый индивид остается неизменным, а второй может многократно выбираться из множества случайно отобранных в текущей популяции индивидов перед процедурой рекомбинации. В обычном случае максимальное число таких пар не превышает трех, однако, при использовании альтернативных правил рекомбинации в схеме DE [13], таких пар может быть больше. Число проверок разных комбинаций родительских индивидов для формирования одного потомка зависит от состава текущей популяции.

Практические исследования рассматриваемой схемы дифференциальной эволюции показали, что чаще всего несколько "попыток" при создании пробного индивида используются, когда в популяции много копий одного индивида либо она заполнена большим числом недоминируемых индивидов. В таких случаях генерация разных вариантов одного промежуточного индивида позволяет поддерживать разнообразие популяции, снижать вероятность схождения алгоритма к локально оптимальному множеству Парето. В отличие от обычной в предлагаемой версии дифференциальной эволюции, траектория поиска в пространстве переменных формируется в зависимости от расположения индивидов и, прежде всего, получаемого потомка в пространстве критериев. В итоге, исследуются те участки пространства переменных, где содержатся решения, являющиеся недоминируемыми относительно уже исследованных решений. Следует отметить, что при сравнении пробного и текущего индивидов оба могут оказаться недоминируемыми (операция 2.7 правило (г)) и, следовательно, могут претендовать на переход в следующее поколение. В этом случае процедура выбора индивида (индивидов) должна определяться МГА, использующим данную схему дифференциальной эволюции.

Как следует из приведенного алгоритма DE/rand/1X/bin, он содержит только средства изменения популяции, направленные на поиск и исследование недоминируемых решений. Для увеличения их численности, управления распределением на границе Парето необходимы дополнительные процедуры, реализованные в ряде известных МГА, например в таких, как NSGA-II и SPEA2. Интеграция в них предлагаемой схемы DE (подобно алгоритмам NSDE, DEMO$^{NS\text{-}II}$ и DEMO^{SP2}) позволит получить новые версии этих алгоритмов, расширяющие свои функциональные возможности при исследовании пространств переменных и критериев.

4. Исследование эффективности использования схемы DE\rand\1X\bin

Для оценки эффективности использования предложенной схемы дифференциальной эволюции был проведен ряд экспериментов. Они заключались в решении набора тестовых задач многокритериальной оптимизации (DTLZ1, DTLZ2, DTLZ3, DTLZ6) [21] с различным числом критериев. Это позволило оценить влияние схемы DE/rand/1X/bin на сохранение эффективности использующих ее алгоритмов NSGA-II и SPEA2 при росте сложности решаемых задач. Сравнивали результаты, полученные оригинальными версиями этих алгоритмов, их модификациями с классической схемой дифференциальной эволюции, а также алгоритмом MODE.

Результаты работы всех алгоритмов оценивали по нескольким показателям качества. Для их вычисления использовали набор индикаторов [6], позволяющий оценить различные свойства найденного МГА множества Парето в пространстве решений или критериев (табл. 1).

Для каждого МГА экспериментально были подобраны значения управляющих параметров. Ряд значений (размеры популяции и Парето-архива, число поколений) для всех алгоритмов совпадали и

Таблица 1

Набор индикаторов для оценки эффективности МГА

№	Название	Назначение
1	I_{ONVG} (Overall Nondominated Vector Generation)	Определяет мощность найденного множества Парето
2	I_S (Spacing)	Используется для оценки равномерности распределения решений вдоль границы Парето
3	I_{DE} (Dimensions Extent)	Позволяет оценить максимальную протяженность границы Парето по каждой из размерностей
4	I_{GD} (Generational Distance)	Позволяет оценить степень близости между полученной и заданной эталонной границами Парето
5	I_{OT} (Overall Time Computing)	Предназначен для оценки времени работы МГА при определении множества Парето

Таблица 2

Результаты исследования эффективности схемы DE/rand/1X/bin (DE*)

Задача	Индикатор	SPEA2	NSGA-II	MODE	SPEA2 + DE*	NSGA-II + DE*
DTLZ1 (m = 2)	I_{ONVG}	28	35	28	28 (28)	**37** (35)
	I_{GD}	0,071	0,067	0,064	**0,058** (0,072)	0,061 (0,063)
	I_S	0,186	0,223	0,231	**0,185** (0,192)	0,243 (0,237)
	I_{DE}	0,971	0,964	0,968	**0,975** (0,958)	0,971 (0,966)
	I_{OT} (сек.)	6,3	4,8	7,3	6,5 (6,4)	**4,7** (5,3)
DTLZ2 (m = 4)	I_{ONVG}	113	**138**	113	113 (113)	136 (135)
	I_{GD}	5,732	6,124	5,693	**5,641** (5,692)	5,972 (5,969)
	I_S	0,133	**0,124**	0,139	0,141 (0,145)	0,137 (0,141)
	I_{DE}	1,714	1,847	1,793	1,729 (1,748)	**1,867** (1,845)
	I_{OT} (сек.)	191,4	**31,6**	217,4	213,8 (204,6)	34,5 (32,1)
DTLZ3 (m = 6)	I_{ONVG}	188	207	178	193 (188)	**214** (201)
	I_{GD}	226,953	310,603	297,762	**223,714** (243,493)	308,547 (314,603)
	I_S	0,328	**0,287**	0,317	0,343 (0,339)	0,312 (0,315)
	I_{DE}	2,137	2,312	2,197	2,261 (2,178)	**2,324** (2,297)
	I_{OT} (сек.)	6217,3	**698,7**	6943,7	6327,3 (6252,4)	714,5 (702,7)
DTLZ6 (m = 8)	I_{ONVG}	319	**371**	312	324 (318)	367 (354)
	I_{GD}	16,623	11,237	11,549	16,121 (15,176)	**11,073** (12,472)
	I_S	0,251	**0,201**	0,237	0,249 (0,243)	0,221 (0,227)
	I_{DE}	1,862	1,913	1,937	1,974 (1,952)	**2,016** (1,931)
	I_{OT} (сек.)	76784,3	**2843,8**	98212,1	83113,4 (78036,7)	3224,6 (2914,7)

зависели от решаемой задачи [21]. При определении вероятностей операторов кроссинговера и мутации в алгоритмах SPEA2, NSGA-II учитывалась точность решения ими (индикатор I_{GD}) всех задач с двумя критериями. Для алгоритмов, использующих дифференциальную эволюцию, исследовали влияние на результат различных значений вероятности кроссинговера $CR = \{0{,}3,\ 0{,}6,\ 0{,}9\}$, при этом значение другого управляющего параметра F не изменяли и принимали равным 0,5 [19]. Было выявлено, что с увеличением вероятности кроссинговера $CR = \{0{,}6,\ 0{,}9\}$ в задачах DTLZ3, DTLZ6 существенно ухудшались показатели индикатора I_{GD}, что может свидетельствовать о не нахождении глобальной границы Парето. Поэтому во всех последующих испытаниях параметр CR устанавливали равным 0,3. Итоговые результаты решения всех задач приведены в табл. 2. Значения всех индикаторов усреднены по выполненному числу запусков МГА. Лучшие значения показателей выделены полужирным шрифтом. Столбцы SPEA2 + DE*, NSGA-II + DE* в скобках содержат значения, полученные соответствующими МГА при использовании обычной схемы DE (DE/rand/1/bin).

Анализ результатов в таблице позволяет сформулировать следующие выводы.

1. Применение дифференциальной эволюции во всех случаях позволило улучшить сходимость алгоритмов к глобальной границе Парето. В то же время, в трех задачах качество найденных решений по индикатору I_{GD} оказалось наилучшим при использовании предложенной схемы DE/rand/lX/bin.

2. Равномерность распределения решений вдоль границы Парето (индикатор I_S) в большинстве случаев оказалась лучше у алгоритмов, не использующих DE. При анализе этой причины было выявлено, что у большинства популяций, формируемых с использованием дифференциальной эволюции, новые индивиды часто оказываются очень плотно расположенными друг другу в пространстве критериев.

3. При решении задач DTLZ1, DTLZ2 и DTLZ6 алгоритмы, использующие схему DE/rand/1X/bin, по большинству показателей (в задаче DTLZ1 по всем) превзошли оригинальные версии SPEA2, NSGA-II.

4. При решении задач с $m = \{6, 8\}$ результаты SPEA2 + DE*, NSGA-II + DE* почти по всем индикаторам оказались лучше алгоритмов, использующих обычную схему DE.

5. В большинстве задач использование дифференциальной эволюции позволило определить границу Парето с большей протяженностью, чем у оригинальных SPEA2, NSGA-II, при этом были достигнуты лучшие значения индикатора I_{GD}.

6. Во всех случаях общее время вычислений при использовании схемы DE/rand/1X/bin незначительно превышает значения соответствующего индикатора остальных анализируемых алгоритмов.

7. Результаты алгоритма MODE, использующего как дифференциальную эволюцию, так и ключевые особенности SPEA2, NSGA-II, по всем показателям оказались достаточно близки к наилучшим их значениям (по некоторым индикаторам MODE превзошел SPEA2, NSGA-II).

Заключение

Дифференциальная эволюция позволяет повысить эффективность исследования пространства решений. Ее применение при решении задач многокритериальной оптимизации оказывает существенное влияние на возможность нахождения глобального множества Парето. В то же время, для достижения лучших результатов и по другим показателям качества границы Парето требуется интеграция DE в различные генетические алгоритмы МО. Исследования таких алгоритмов показали, что с ростом числа критериев эффективность от использования обычной схемы DE, как правило, не увеличивается. Возможная причина этого видится в необходимости учитывать действия дифференциальной эволюции и в пространстве критериев. Один из вариантов реализации этого предложен в виде ее модификации — схемы DE/rand/1X/bin. Она также может быть

использована с любым МГА, не требует подбора значений дополнительных управляющих параметров, незначительно увеличивает временную сложность базового алгоритма. Тестирование этой модификации подтвердило ее эффективность как относительно классических МГА, так и традиционной схемы DE при решении большинства рассмотренных тестовых задач с пространством критериев различной размерности.

Список литературы

1. **Соболь И. М., Статников Р. Б.** Выбор оптимальных параметров в задачах со многими критериями. 2-е изд., перераб. и доп. М.: Дрофа, 2006. 175 с.

2. **Ногин В. Д.** Принятие решений в многокритериальной среде: количественный подход. — 2-е изд., испр. и доп. М.: ФИЗМАТЛИТ, 2004. 176 с.

3. **Гладков Л. А., Курейчик В. В., Курейчик В. М.** Генетические алгоритмы. М.: ФИЗМАТЛИТ, 2006. 320 с.

4. **Deb K.** Multi-Objective Optimization-Using Evolutionary Algorithms. Hoboken: Wiley, 2009. 536 p.

5. **Казаков П. В.** Генетические алгоритмы многокритериальной оптимизации. Обзор // Информационные технологии. 2011. № 9. С. 2—8.

6. **Пупков К. А., Феоктистов В. А.** Алгоритм дифференциальной эволюции для задач технического проектирования // Информационные технологии. 2004. № 8. С. 25—31.

7. **Казаков П. В.** Оценка эффективности генетических алгоритмов многокритериальной оптимизации. Часть 2 // Информационные технологии. 2012. № 9. С. 42—46.

8. **Карпенко А. П., Овчинников В. А., Семенихин А. С.** Программная система PRADIS//FRONT для построения множества Парето в задаче многокритериальной оптимизации динамических систем с использованием параллельного генетического алгоритма // Информационные технологии. 2009. № 8. С. 27—33.

9. **Coello Coello C. A., Reyes Sierra M.** A Coevolutionary Multi-objective Evolutionary Algorithm // Proc. of the 2003 Congress on Evolutionary Computation (CEC'2003). 2003. Vol. 1. P. 482—489.

10. **Карпенко А. П., Митина Е. В., Семенихин А. С.** Когенетический алгоритм Парето-аппроксимации в задаче многокритериальной оптимизации // Информационные технологии. 2013. № 1. С. 22—32.

11. **Storn R., Price K.** Differential Evolution — A Fast and Efficient Heuristic for Global Optimization over Continuous Spaces // Journal of Global Optimization. 1997. N. 11. P. 341—359.

12. **Kukkonen S., Lampinen J.** An extension of generalized differential evolution for multi-objective optimization with constraints // Proc. of Parallel Problem Solving from Nature — PPSN VIII. 2004. P. 752—761.

13. **Mezura-Montes E., Velazques-Reyes J., Coello Coello C.** Comparing Differential Evolution Models for Global Optimization // Proc. of the 2006. Genetic and Evolutionary Computation Conference — GECCO 2006. 2006. Vol. 1. P. 485—492.

14. **Abbass H., Sarker R., Newton C.** PDE: A pareto-frontier differential evolution approach for multi-objective optimization problems // Proc. of the Congress on Evolutionary Computation 2001 (CEC'2001). 2001. Vol. 2. P. 971—978.

15. **Xue F., Sanderson A., Graves R.** Pareto-based multi-objective differential evolution // Proc. of the 2003 Congress on Evolutionary Computation — CEC 2003. 2003. Vol. 2. P. 862—869.

16. **Deb K., Pratap A., Agarwal S., Meyarivan T.** A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II // IEEE Transactions on Evolutionary Computation. 2002. Vol. 6, N. 2. P. 182—197.

17. **Zitzler E., Laumanns M., Thiele L.** SPEA2: Improving the Strength Pareto Evolutionary Algorithm // Proc. of the EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, 2002. P. 95—100.

18. **Iorio A., Li X.** Incorporating directional information within a differential evolution algorithm for multi-objective optimization // Proc. of the 2006 Genetic and Evolutionary Computation Conference — GECCO 2006. 2006. Vol. 1. P. 675—682.

19. **Robic T., Filipic B.** DEMO: Differential evolution for multiobjective optimization // Proc. of the Third International Conference on Evolutionary Multi-Criterion Optimization — EMO 2005. 2005. P. 520—533.

20. **Tusar T., Filipic B.** Differential Evolution Versus Genetic Algorithms in Multiobjective Optimization // Proc. of the Fourth International Conference on Evolutionary Multi-Criterion Optimization — EMO 2007. 2007. P. 257—271.

21. **Казаков П. В.** Оценка эффективности генетических алгоритмов многокритериальной оптимизации. Часть 1 // Информационные технологии. 2012. № 8. С. 2—6.

R. V. Kazakov, Lecturer, Bryansk State Technical University

The use Differential Evolution to Obtain Pareto-Set by the Multi-Objective Genetic Algorithms

The new manner to improve the efficiency of multi-objective genetic algorithms is considered. It is based on the principles of differential evolution in the process of creating new individuals. The use of general schemes of differential evolution DE/rand/1/bin etc. for multi-objective optimization are analyzed. It is drawn a conclusion that the general schemes mainly use the information about location of solutions in a decision space that is often insufficient for effective exploration of the objective space. Therefore the original scheme of differential evolution DE/rand/1X/bin is suggested. By this scheme in recombination process is taken an account of searching path into two spaces — decisions and objectives. This process is based on the rule that the selection of second parent and other (two or more) individuals is made randomly. Hence to increase the probability of creating individuals that dominate their parents various combinations of individuals selected for recombination may be examined. For example in general scheme DE/rand/1/bin it means that each of three randomly selected individuals can be selected as a second parent. If no one of various combinations of individuals in recombination process does not allow to get individual-offspring being better than their parents, the parent-individual one is selected for next generation. It was found that such scheme should allow to realize higher replacement of the population dominated individuals. Also it was found that the number of trials for offspring creation self-adapted and changed according to array of current population. The scheme DE/rand/1X/bin should be integrated into other multi-objective genetic algorithms to handle the obtained nondominated solutions. The experiments for evaluation scheme DE/rand/1X/bin were conducted on the benchmark problems DTLZ. The results was found that the use of such a scheme allowed to improve the results (on a some indicators) of algorithms SPEA2, NSGA-II and their modifications with a general scheme of differential evolution.

Keywords: *multi-objective optimization, Pareto's principles, Pareto front, multi-objective genetic algorithms, differential evolution*

References

1. **Sobol I. M., Statnikov R. B.** *Vybor optimalnykh parametrov v zadachakh so mnogimi kriteriyami.* M.: Drofa, 2006. 175 p.
2. **Nogin V. D.** *Prinyatie resheniy v mnogokriterialnoy srede: kolichestvennyy godkhod.* M.: FIZMATLIT, 2004. 176 p.
3. **Gladkov L. A., Kureychik V. V., Kureychik V. M.** *Geneticheskie algoritmy.* M.: FIZMATLIT, 2006. 320 p.
4. **Deb K.** *Multi-Objective Optimization Using Evolutmnary Algorithms.* Hoboken: Wiley, 2009. 536 p.
5. **Kazakov P. V.** Geneticheskie algoritmy mnogokriterialnoy optimizatsyi. Obzor. *Informatsyonnye tekhnologii.* 2011. N. 9. P. 2—8.
6. **Pupkov K. A., Feoktistov V. A**. Algoritm differentsyalnoy evolyutsii dlya zadach tekhnicheskogo proektirovaniya. *Informatsyonnye tekhnologii.* 2004. N. 8. P. 25—31.
7. **Kazakov P. V.** Ocenka effektivnosti geneticheskikh algoritmov mnogokriterialnoy optimizatsyi. Chast 2. *Informatsyonnye tekhnologii.* 2012. N. 9. P. 42—46.
8. **Karpenko A. P., Ovchinnikov V. A., Semenikhin A. S.** Programmnaya sistema PRADIS // FRONT dlya postroeniya mnozhestva Pareto v zadache mnogokriterialnoy optimizatsyi dynamicheskikh sistem s ispolzovaniem parallelnogo geneticheskogo algoritma. *Informatsyonnye tekhnologii.* 2009. N. 8. P. 27—33.
9. **Coello Coello C., Reyes Sierra M.** A Coevolutionary Multiobjective Evolutionary Algorithm. *Proc. of the 2003 Congress on Evolutionary Computation (CEC'2003).* 2003. Vol. 1. P. 482—489.
10. **Karpenko A. P., Mitina E. V., Semenikhin A. S.** Kogeneticheskiy algoritm Pareto-approksimacyi v zadache mnogokriterialnoy optimizatsyi. *Informatsyonnye tekhnologii.* 2013. N. 1. P. 22—32.
11. **Storn R., Price K.** Differential Evolution — A Fast and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization.* 1997. N. 11. P. 341—359.
12. **Kukkonen S., Lampinen J.** An extension of generalized differential evolution for multi-objective optimization with constraints. *Proc. of Parallel Problem Solving from Nature — PPSN VIII.* 2004. P. 752—761.
13. **Mezura-Montes E., Velazques-Reyes J., Coello Coello C.** Comparing Differential Evolution Models for Global Optimization. *Proc. of the 2006 Genetic and Evolutionary Computation Conference — GECCO 2006. 2006.* Vol. 1. P. 485—492.
14. **Abbass H., Sarker R., Newton C.** PDE: A pareto-frontier differential evolution approach for multi-objective optimization problems. *Proc. of the Congress on Evolutionary Computation 2001 (CEC'2001).* 2001. Vol. 2. P. 971—978.
15. **Xue F., Sanderson A., Graves R.** Pareto-based multi-objective differential evolution. *Proc. of the 2003 Congress on Evolutionary Computation — CEC 2003.* 2003. Vol. 2. P. 862—869.
16. **Deb K., Pratap A., Agarwal S., Meyarivan T.** A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation.* 2002. Vol. 6, N. 2. P. 182—197.
17. **Zitzler E., Laumanns M., Thiele L.** SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *Proc. of the EUROGEN 2001.* Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems. 2002. P. 95—100.
18. **Iorio A., Li X.** Incorporating directional information within a differential evolution algorithm for multi-objective optimization. *Proc. of the 2006 Genetic and Evolutionary Computation Conference — GECCO 2006.* 2006. Vol. 1. P. 675—682.
19. **Robic T., Filipic B.** DEMO: Differential evolution for multiobjective optimization. *Proc. of the Third International Conference on Evolutionary Multi-Criterion Optimization — EMO 2005.* 2005. P. 520—533.
20. **Tusar T., Filipic B.** Differential Evolution Versus Genetic Algorithms in Multiobjective Optimization. *Proc. of the Fourth International Conference on Evolutionary Multi-Criterion Optimization — EMO 2007.* 2007. P. 257—271.
21. **Kazakov P. V.** Ocenka effektivnosti geneticheskikh algoritmov mnogokriterialnoy optimizatsyi. Chast 1. *Informatsyonnye tekhnologii.* 2012. N. 8. P. 2—6.

УДК 519.6

А. Н. Вдовин, канд. биол. наук, вед. науч. сотр.,
"ТИНРО-Центр", г. Владивосток, e-mail: vdovin@.tinro-center.ru,
А. Н. Четырбоцкий, д-р физ.-мат. наук, вед. науч. сотр.,
ДВГИ ДВО РАН, г. Владивосток, e-mail: chetyrbotsky@yandex.ru,
В. А. Четырбоцкий, студент,
МГУ имени М. В. Ломоносова, Москва, e-mail: ve14232@gmail.com

Компьютерное моделирование динамики роста рыб (на примере южного одноперого терпуга Pleurogrammus azonus) Часть 1

В рамках положений системы типа "ресурс—потребитель" разработана модель динамики роста рыб. При ее построении учитывалась предельная длина рыбы, этапность ее жизненного цикла и влияние сезонных факторов. Выполнена параметрическая идентификация модели. Согласно ей проведена оценка статистических свойств параметров модели, установлена адекватность между эмпирическим распределением и ее модельным образом. На основании вычислительных экспериментов выявлена продолжительность периода этапности жизненного цикла.

Ключевые слова: *динамика роста, логистическое уравнение, параметрическая идентификация, задача поиска минимума, адекватность модели*

Введение

Для представления популяционной динамики биологических видов имеется огромное число многообразных количественных соотношений, при формулировке которых учитываются особенности их жизненного цикла. Основная часть из них характеризует динамику численности (биомассы) организмов (большой спектр таких моделей был рассмотрен А. Д. Базыкиным [1]). Существенно меньше число работ, где изучается динамика роста

и массы именно рыб [6, 12, 16—18]. Между тем, эти характеристики широко используют в рыбохозяйственной науке для оценки промысловой значимости определенных видов и при разработке стратегии рациональной эксплуатации сообществ морских организмов [7, 10]. В этих работах обсуждают вопросы применимости модификаций уравнений логистического роста (для длины) и уравнения Берталанфи (для массы), тогда как этапность жизненного цикла особей не учитывается. Существенным ограничением их применимости является годичная временная дискретность, что не позволяет учитывать сезонную динамику. В этой ситуации актуальной является задача разработки такой модели динамики роста, где этот цикл принимается во внимание и имеется возможность моделирования сезонной динамики.

Для разработки модели была использована статистически значимая выборка временного распределения длин южного одноперового терпуга (*Pleurogrammus azonus*). Предельный возраст его особей оценивается 96 месяцами, а предельный размер — 52 см [3]. Этот вид обитает в Тихоокеанских водах островов Хоккайдо и Хонсю, в южной части Охотского моря, в сопредельных водах Восточно-Китайского моря [11], в Японском и Желтом морях. Важность изучения его динамики обусловлена его промысловой значимостью в южной части Охотского моря и в северо-западной части Японского моря.

Для выполнения исследований использованы материалы, собранные в научно-исследовательских и промысловых рейсах, а также на рыбокомбинатах Приморского края (Японское море) в 1960—1986 гг. Возраст определяли по чешуе отработанными авторскими методиками [4]. Возраст был определен у 5298 экз. Сборы личинок проводили во время ихтиопланктонных съемок в октябре—декабре 1976—1983 гг. Всего было промерено 227 личинок. Исходя из того, что массовый выклев личинок в морских водах Приморья происходит в октябре, средний возраст особей в выборках определялся с точностью в пределах одного месяца. По средним значениям длины и возраста была построена кривая популяционного роста, опубликованная нами ранее [5]. Необходимо отметить, что исследования охватывали весь период онтогенеза.

На основании этого материала была сформирована выборка распределения роста рыб. Она характеризует достаточно равномерно распределенную по возрастам выборку из 61 наблюдения.

Модель сезонной динамики роста рыб

В практике изучения динамики роста рыб количественная формализация обычно выполняется на основании формулы роста Берталанфи

$$\dot{L} = k(1 - L/L_{\max}), \qquad (1)$$

где $L = L(t)$ — длина в момент времени t (точка над переменной отвечает взятию производной по времени); $L_{\max} = L(t_{\max})$ — соответствующий $t_{\max}$ предельный размер; k — неотрицательный коэффициент пропорциональности размерности $[L]/[t]$ (квадратные скобки указывают размерность переменных) [16—18]. Представляется разумной такая интерпретация (1). Выражение в скобках характеризует доступный для роста особи ограниченный "ресурс" ее длины. Его "трансформация" или "утилизация" непосредственно в саму длину выполняется с постоянной интенсивностью k. Решение (1) определяется соотношением

$$L(t) = L_{\max}[1 - \exp(-kt)], \qquad (1a)$$

которому на плоскости $\{t, L\}$ соответствует монотонно возрастающая от начала координат до точки $(t_{\max}, L_{\max})$ кривая.

В рамках положений модели (1) в динамике роста рыб отсутствует этапность их жизненного цикла, наличие которой заметно при анализе эмпирического распределения (рис. 1). Здесь для учета указанной особенности предлагается соотношение

$$\dot{L} = \psi(t)(L_{\max} - L)^{\gamma}L, \qquad (2)$$

где $\psi(t)$, γ — некоторая функция и показатель самой модели. Они подлежат оцениванию на основании экспериментальных данных. В соотношении (2) размерность $[L]$ = см, а $[\psi(t)] = 1/(\text{с} \cdot \text{см}^{\gamma})$. Нелинейность правой части (2) отражает определенную скачкообразность изменения кривой динамики (она следует из анализа рис. 1). Интерпретация (2) состоит в следующем. Полагается, что скорость прироста $\ln L$ определяется доступным для него "ресурсом" $(L_{\max} - L)^{\gamma}$. Показатель степени γ характеризует интенсивность "трансформации" ресурса в динамику $\ln L$. Он определяет различия темпов роста у молодых (следуя рис. 1, возраста до 1 года) и более взрослых рыб. Действительно, при $\psi(t) = \text{const}$, чем выше γ, тем ближе точка перегиба кривой $L(t)$ расположена к началу координат, иными словами, тем выше темп роста молодых особей, а у самых возрастных особей рост вообще прекращается.

Формализация задачи оценивания параметров модели (2) здесь принимает следующий вид:

$$\Phi(\gamma, \psi) = \sum_{i=1}^{61} [L_i^{(d)} - L(t_i)]^2 \to \min, \qquad (3)$$

где выборка $\{L_i^{(d)} : i = 1...61\}$ соответствует временному распределению роста особей южного одноперового терпуга. Для поиска искомого решения использовался метод Левенберга—Марквардта [2], который является комбинацией метода Ньютона и метода наискорейшего спуска. В системе MATLAB программную реализацию этого метода выполняет

процедура *lsqnonlin* [8]. Достоинством данного метода является такое конструирование матрицы Гессе, при которой она имеет невырожденную обратную матрицу [2]. Далее эта обратная матрица используется для вычисления направления и шага искомого экстремума.

При использовании процедуры *lsqnonlin* матрица ковариации оценок параметров вычисляется следующим образом. Одним из выходных параметров *lsqnonlin* является якобиан $J = \{\partial\Phi(t_i)/(\partial p_m)\}$. В методе Левенберга—Марквардта искомая матрица ковариации определяется выражением

$$V \approx 2\sigma_e^2 H^{-1}, \qquad (4)$$

где элементы матрицы Гессе $H = \{H_{mk} = 2\sum_{i=1}^{NN} J_{im}J_{ik}\}$ следуют соотношению

$$H_{mq} \approx 2\sum_{i=1}^{N} J_{im}J_{ik}.$$

Набор $\{L(t_i),\ i = 1...61\}$ соответствует решению уравнения (2), для чего использовался двухшаговый метод [9]: сначала на промежуточном временном слое $k + 0{,}5$ рассчитываются значения $L^{(k+0,5)} = L^{(k)} + f^{(k)}\Delta t$, а затем — $L^{(k+1)} = L^{(k)} + f^{(k+0,5)}\Delta t$. Метод является устойчивым при $\Delta t < 2(\partial f/\partial L)^{-1}$, где $f = \psi(t)(L_{\max} - L)^{\gamma}L$ — правая часть (2) и k — текущий номер временного слоя. В вычислительных экспериментах шаг по времени Δt подбирался исходя из требования устойчивости расчетов. Необходимо отметить, что набор $\{t_i,\ i = 1...61\}$ соответствует части выборки расчетных времен модели. Так, в расчетах было выполнено 200 временных шагов, а затем выбирались только те значения $\{L(t_i),\ i = 1...61\}$, которые соответствуют временам эмпирического распределения. В расчетах предельных возраст $t_{\max}$ принимали равным 96 месяцам, а предельный размер — $L_{\max} = 52$ см.

Универсальный способ задания $\psi(t)$ состоит в ее представлении конечным отрезком ряда Фурье. В условиях ограниченной выборки этот способ позволяет выявить только низкочастотные составляющие изучаемого процесса [13]. Здесь для выявления периодических составляющих использовался такой прием [14]. Полагалось, что $\psi(t)$ определяется заранее заданным числом ее гармонических составляющих, частоты и амплитуды которых следуют решению задачи (3). В вычислительных экспериментах их число варьировалось, а затем заново оценивались частоты и амплитуды гармоник (заново решалась задача (3)). Для решения (3) здесь начальное приближение было задано набором: $\gamma = 1$, $\psi_0 = 10^{-6}$, $\psi_1 = 10^{-7}$ и $\psi_2 = 10^{-7}$. На первой итерации значение $\Phi_1 = 2{,}281 \cdot 10^3$. На 12-й итерации шаг процедуры *lsqnonlin* оказался меньше заданного здесь шага 10^{-8}, что привело к останову. Тогда оказалось, что гармонике с периодом 18 месяцев соответствует наименьшее значение $\Phi(\gamma^*, \psi^*) = 913.289$. В этом случае $\gamma^* = 3{,}285 \pm 0{,}084$, $\psi^*(t) = \psi_0 + \psi_1\cos(\omega t) + \psi_2\sin(\omega t)$, $\omega = 2\pi/18$, $\psi_0 = (7{,}657 \pm 0{,}824) \cdot 10^{-7}$, $\psi_1 = (-3{,}734 \pm 0{,}481) \cdot 10^{-7}$ и $\psi_2 = (3{,}626 + 0{,}339 \cdot 10^{-7}$. В вычислительных экспериментах рассматривались случаи расширения числа параметров модели. В частности, рассматривался случай задания $\Psi(t)$ тремя гармоническими составляющими с периодами 11, 12 и 18 месяцев. Тогда число итераций поиска экстремума повысилось до 22 шагов, а значение функционала снизилось всего на 3 %.

Адекватность модели (3) выборочному распределению оценивалась на основании результатов анализа остатков (разницы между модельным и выборочным распределениями) и коэффициента корреляция между ними. Здесь среднее значение элементов вектора остатков равно $0{,}517 \pm 5{,}078$. Поскольку доверительный интервал содержит нуль, то согласно статистическим критериям этот средний остаток статистически незначим. Коэффициент же корреляции между эмпирическим и модельным распределениями оказался равным 0,958. Его близость к максимально возможному единичному значению указывает на тесную линейную связь между распределениями и их подобие. Результаты численных экспериментов представлены на рис. 1.

Сопоставление кривых показывает, что их пересечения соответствуют возрастам около 1,5 и 6 лет. До момента первого пересечения эмпирическое и модельное распределения практически не отличаются. Первые полтора года характеризуются стремительным темпом роста, замедление которого начинается только в октябре (в возрасте одного года) с началом осеннего выхолаживания вод. Между точками пересечения модельные темпы роста превышают эмпирические темпы, что обусловлено пре-

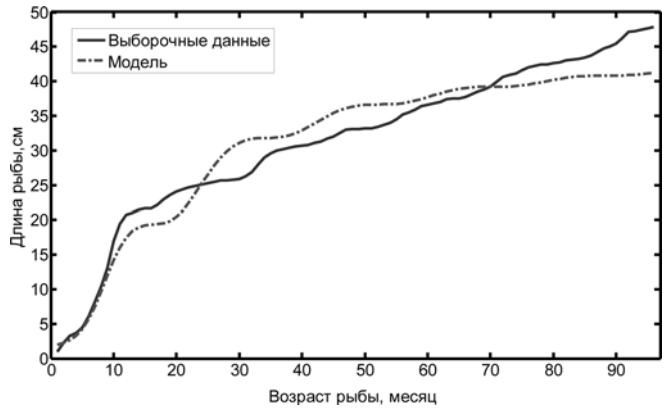


Рис. 1. Динамика усредненного по выборке данных роста рыбы (эмпирические данные и модель)

обладанием тугорослых особей в общей массе рыб. Далее имеет место обратная картина, которая характеризуется низкими амплитудами сезонной ритмики с выходом на плато. Следует подчеркнуть, что результаты модели объективнее, чем усредненные эмпирические данные, отражают особенности индивидуального роста рыб, поскольку динамика популяционного роста зависит от вариабельности этого показателя и селективной смертности рыб [5].

Модельный жизненный цикл терпуга кратен 18-месячному периоду. Представляется разумной такая интерпретация этой периодичности. Околододичный жизненный цикл терпуга длится от 9—11 до 13 месяцев, увеличиваясь с возрастом [3]. В свою очередь он делится на период активной жизнедеятельности (с апреля—мая по октябрь—ноябрь) и период относительного жизненного покоя. С возрастом активный период увеличивается за счет длительности нерестового периода. Период относительного жизненного покоя, напротив, начинает укорачиваться, поскольку рыбы старших возрастных групп быстрее восстанавливаются после нереста. Совокупность двух периодов относительного зимнего покоя и периода активной жизнедеятельности остается практически постоянным и составляет полтора года.

Результаты вычислительных экспериментов указывают на универсальность модельной динамики роста одноперого терпуга. Подтверждением чему является распределение кривых на рис. 2. Здесь каждая из кривых характеризует динамику выраженного в процентах от соответствующих максимальных значений относительного роста трех видов рыб (*1* — калифорнийского анчоуса *Engraulis mordax*; *2* — синего тунца *Thunnus thynnus*; *3* — терпуга).

Сопоставление кривых роста именно этих видов обусловлено различиями их жизненных стратегий: анчоус (кривая *1*) — короткоцикловый вид (продолжительность жизни около двух лет); терпуг (кривая *3*) — среднецикловый вид (продолжительность жизни восемь лет); тунец (кривая *2*) — длинноцикловый (живет 33 года). Кроме того, их возраст определяется надежно. Анализ распределений показывает подобие хода кривых. Кривая *1* (анчоус) показывает самые высокие темпы роста, что типично для короткоцикловых видов. Различия кривой *2* (тунец) и кривой *3* (терпуг) проявляются только в сезонной ритмике. Подобие кривых следует известному положению о самом высоком темпе роста рыб на их начальных этапах. Так, за пятую часть жизни все рассматриваемые виды вырастают больше чем на половину своего размера.

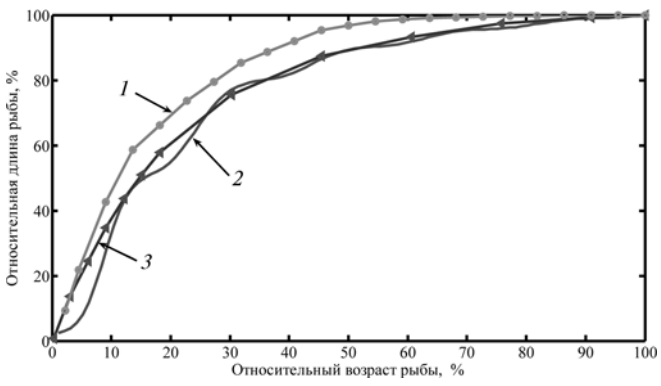


Рис. 2. Динамика относительного роста особей трех видов рыб (распределения 1 и 2 взяты из работы А. А. Яржомбека [15])

Заключение

На основании положений системы типа "ресурс—потребитель" здесь разработана модель динамики роста рыб. При ее построении учитывалась предельная длина рыбы, этапность ее жизненного цикла и влияние сезонных факторов. Выполнена параметрическая идентификация модели.

Показана адекватность модели эмпирическому распределению. Выполнена оценка продолжительности периода жизненного цикла.

Результаты вычислительных экспериментов указывают на универсальность модельной динамики роста одноперого терпуга.

Авторы выражают благодарность канд. биол. наук Г. В. Швыдкому за помощь в сборе и анализе материала, а также ценные консультации.

Список литературы

1. **Базыкин А. Д.** Нелинейная динамика взаимодействующих популяций. Ижевск: Институт компьютерных исследований. 2003. 368 с.

2. **Бард Й.** Нелинейное оценивание параметров. М.: Статистика, 1979. 349 с.

3. **Вдовин А. Н.** Биология и динамика численности южного одноперого терпуга (*Pleurogrammus azonus*) // Изв. ТИНРО. 1998. Т. 123. С. 16—45.

4. **Вдовин А. Н., Васильков В. П.** Определение возраста южного одноперого терпуга *Pleurogrammus azonus* JordanetMetz (*Hexagrammidae*) алгоритмическими методами распознавания образов с обучением // Вопр. ихтиологии. 1982. Т. 22, вып. 6. С. 1006—1014.

5. **Вдовин А. Н., Швыдкий Г. В.** Физиологические аспекты роста одноперого терпуга *Pleurogrammus azonus* в водах Приморья // Вопр. ихтиологии. 1993. Т. 33, вып. 1. С. 156—160.

6. **Винберг Г. Г.** Зависимость энергетического обмена от массы тела у водных пойкилотермных животных // Журн. общ. биологии. 1976. Т. 37. № 1. С. 56—70.

7. **Дгебуадзе Ю. Ю.** Оценки возраста и роста в популяционных исследованиях рыб // Актуальные проблемы современной ихтиологии. М.: Товарищество научных изданий КМК. 2010. С. 96—123.

8. **Дьяконов В., Круглов В.** Математические пакеты расширения MATLAB. Специальный справочник. СПб.: Питер, 2001. 480 с.

9. **Поттер Д.** Вычислительные методы в физике. М.: Мир, 1975. 392 с.

10. **Риккер У. Е.** Количественные показатели и модели роста рыб / Биоэнергетика и рост рыб. М.: Лег. и пищ. пром-сть, 1983. С. 346—405.

11. **Соколовский А. С., Соколовская Т. Г., Яковлев Ю. М.** Рыбы залива Петра Великого: 2-е изд., испр. и доп. Владивосток: Дальнаука, 2011. 431 с.

12. **Суханов В. В.** Математическая модель сезонной динамики весового роста молоди нерки (*Oncorhynchus nerka,Walb.*) // Вопр. ихтиологии. 1974. Т. 14, № 4. С. 575—580.

13. **Четырбоцкий А. Н.** Параметрическая идентификация математической модели формирования плазменно-электролитических покрытий // Информационные технологии, 2006. № 9. С. 60—67.

14. **Четырбоцкий А. Н.** Крупномасштабное математическое моделирование пространственно-временной динамики морского ледяного покрова (на примере Японского моря). Владивосток: Дальнаука, 2009. 195 с.

15. **Яржомбек А. А.** Закономерности роста промысловых рыб. М.: Изд-во ВНИРО, 2011. — 182 с.

16. **Akamaine T.** Non-linear and graphical method for fish stock analysis with statistical modeling // Aqua-BioSci. Monogr., 2009. V. 2, N. 3. P. 1—45.

17. **Bertalanffy L.** Principles and theory of growth // Fundamental aspects of normal and malignant growth. Amsterdam: Elsevier, 1960. P. 137— 259.

18. **Enberg K., Dunlop E. S., Jorgensen C.** Fish grows // Ecological models. 2008. P. 1564—1572.

A. N. Vdovin, Senior researcher, "TINRO-CENTER", Vladivostok, e-mail: vdovin@tinro-center.ru,
A. N. Chetyrbotsky, Senior researcher, Far-East Geology Institute RAS, Vladivostok, chetyrbotsky@yandex.ru,
V. A. Chetyrbotsky, Student, Moscow University, Moskva, ve14232@gmail.com

Mathematical of the Dynamics Fish Growth (for example, southern Atka mackerel Pleurogrammus azonus)

As part of the system of "resource—consumer" is developed the model of dynamics of fish growth. To construct it, we used: the maximum length of the fish, the stages of their life cycle and the impact of seasonal factors. The parametric identification of the model is performed. According to her is carried out the assessment of the statistical properties of the model parameters,is established the adequacy between the empirical distribution and its model image. On the based of computational experiments the assessment the assessment of duration of the life cycle period.

Keywords: *growth dynamics, the logistic equation, parameter identification, the task of finding the minimum value model*

Reference

1. **Bazykin A. D.** *Nelinejnaja dinamika vzaimodejstvujushhih populjacij*. Izhevsk: Institut Komp'juternyh issledovanij, 2003. 368 p.

2. **Bard J.** *Nelinejnoe ocenivanie parametrov*. M.: Statistika, 1979. 349 p.

3. **Vdovin A. N.** Biologija i dinamika chislennosti juzhnogo odnoperogo terpuga (Pleurogrammus azonus). *Izv. TINRO*. 1998. Vol. 123. P. 16—45.

4. **Vdovin A. N., Vasil'kov V. P.** Opredelenie vozrasta juzhnogo odnoperogo terpuga *Pleurogrammus azonus JordanetMetz* (*Hexagrammidae*) algoritmicheskimi metodami raspoznavanija obrazov s obucheniem. *Voprosy ihtiologn*. 1982. T. 22. V. 6. P. 1006—1014.

5. **Vdovin A. N., Shvydkij G. V.** Fiziologicheskie aspekty rosta odnoperogo terpuga Pleurogrammus azonus v vodah Primor'ja. *Vopr. Ihtiologii*. 1993. V. 33, N. 1. P. 156—160.

6. **Vinberg G. G.** Zavisimost' jenergeticheskogo obmena ot massy tela u vodnyh pojkilotermnyh zhivotnyh. *Zhurn. obshh. biologii*. 1976, V. 37, N. 1. P. 56—70.

7. **Dgebuadze Ju._Ju.** Ocenki vozrasta i rosta v populjaciormyh issledovanij ah ryb. *Aktual'nye problemy sovremennoj ihtiologii*. M.: Tovarishhestvo nauchnyh izdanij KMK, 2010. P. 96—123.

8. **D'jakonov V., Kruglov V.** *Matematicheskie pakety rasshirenija MATLAB*. Special'nyj spravochnik. SPb.: Piter, 2001. 480 p.

9. **Potter D.** *Vychislitel'nye metody v fizike*. M.: Mir, 1975. 392 p.

10. **Rikker U. E.** *Kolichestvennye pokazateli i modeli rosta ryb*. Biojenergetika i rost ryb. M.: Leg. i pishh. prom-st'. 1983. P. 346—405.

11. **Sokolovskij A. S., Sokolovskaja T. G., Jakovlev Ju. M.** Ryby zaliva Petra Velikogo: 2-e izd., ispr. i dop. Vladivostok: Dal'nauka, 2011. 431 p.

12. **Suhanov V. V.** Matematicneskaja model' sezonnoj dinamiki vesovogo rosta molodi nerki (Oncorhynchus nerka, Walb.). *Vopr. Ihtiologii*. 1974. V. 14, N. 4. P. 575—580.

13. **Chetyrbocky A. N.** Parametricheskaja identifikacija matematicheskoj modeli formirovanija plazmenno-jelektroliticheskih pokrytij. *Informacionnye tehnologii*. 2006. N. 9. P. 60—67.

14. **Chetyrbocky A. N.** *Krupnomasshtabnoe matematicheskoe modelirovanie prostranstvenno-vremennoj dinamiki morskogo ledjanogo pokrova (na primere Japonskogo morja)*. Vladivostok: Dal'nauka, 2009. 195 p.

15. **Jarzhombek A. A.** *Zakonomernosti rosta promyslovyh ryb*. M.: Izd-vo VNIRO, 2011. 182 p.

16. **Akamaine T.** Non-linear and graphical method for fish stock analysis with statistical modeling. *Aqua-BioSci. Monogr*. 2009. V. 2, N. 3. P. 1—45.

17. **Bertalanffy L.** Principles and theory of growth. *Fundamental aspects of normal and malignant growth*. Amsterdam: Elsevier, 1960. P. 137—259.

18. **Enberg K., Dunlop E. S., Jørgensen C.** Fish grows. *Ecological models*. 2008. P. 1564—1572.

УДК 004.43

В. Б. Коваленко[1], канд. техн. наук, ст. науч. сотр., e-mail: vereten@hotbox,
А. И. Дордопуло[2], канд. техн. наук, зав. лаб., e-mail: scorpio@mvs.tsure.ru,
В. А. Гудков[2], канд. техн. наук, ст. науч. сотр., e-mail: gudkov@mvs.tsure.ru,
А. А. Гуленок[2], канд. техн. наук, ст. науч. сотр., e-mail: andrei_gulenok@mail.ru,
Л. М. Сластен[2], канд. техн. наук, ст. науч. сотр., e-mail: lmslasten@yandex.ru
[1] Федеральное государственное бюджетное учреждение науки Южный научныи центр РАН, г. Ростов-на-Дону
[2] НИИ многопроцессорных вычислительных систем им. акад. А. В. Каляева ЮФУ, г. Таганрог

Использование софт-архитектур при решении задач цифровой обработки сигналов на реконфигурируемых вычислительных системах

Рассматривается макрообъектный подход к программированию реконфигурируемых вычислительных систем, позволяющий сократить время программирования задачи за счет уменьшения времени трансляции параллельной программы.

Ключевые слова: *многопроцессорные системы, суперкомпьютеры, реконфигурируемые вычислительные системы, архитектуры вычислительных систем, параллельное программирование, языки программирования*

Введение

Программирование реконфигурируемых вычислительных систем (РВС), обладающих полем связанных программируемых логических интегральных схем (ПЛИС), является сложной и трудоемкой задачей. На данный момент существуют два метода программирования РВС:

- программирование с применением стандартных средств программирования, основанных, как правило, на языках HDL-группы;
- использование языков высокого уровня, таких как COLAMO, Handel-C, SystemC.

Каждый из существующих на данный момент методов программирования РВС имеет свои достоинства и недостатки. Программирование РВС с использованием стандартных средств, предоставляемых производителями ПЛИС, позволяет достигать максимальной производительности, однако требует длительного времени для разработки решений. Кроме того, подобные средства позволяют программировать лишь отдельные ПЛИС, что приводит к возникновению сложностей при объединении нескольких ПЛИС в единую вычислительную структуру.

Более эффективным методом программирования РВС является применение языков высокого уровня, таких как COLAMO, Handel-C, SystemC и др. Сокращение времени программирования при этом достигается за счет существенного ускорения написания текста программ. Современные системы программирования РВС, основанные на языках высокого уровня, зачастую позволяют сократить время программирования до нескольких недель, однако трансляция программ на универсальные процессоры или промежуточные решения, предзагруженные в ПЛИС (Handel-C, SystemC), приводят к сокращению реальной производительности вычислительной системы, а трансляция на уровень логических ячеек ПЛИС (COLAMO) зачастую требует значительных временны́х затрат.

Еще один метод программирования РВС основан на использовании динамически перестраиваемых архитектур, предзагруженных в РВС, что позволяет сократить время разработки прикладной программы РВС за счет уменьшения времени трансляции прикладных программ. При этом производительность вычислительной системы остается на уровне, сопоставимом с производительностью специализированных вычислительных структур. Это достигается путем создания и предзагрузки в РВС вычислительных архитектур, способных решать задачи определенного класса. Опыт создания вычислительных структур на РВС показывает, что для каждого класса задач можно выделить конечный набор вычислительных узлов, с помощью которых может быть построена вычислительная архитектура, эффективно решающая задачи данного класса. Вы-

числительные узлы, управляемые посредством системы команд и связанные в единую вычислительную структуру, составляют софт-архитектуру [1] РВС. При решении задач на РВС с использованием софт-архитектур процесс программирования разбивается на два основных этапа:

- создание софт-архитектуры РВС на основе разработанных ранее вычислительных блоков;
- решение прикладной задачи путем настройки разработанной ранее софт-архитектуры в соответствии с алгоритмом решаемой задачи.

При этом единожды созданная софт-архитектура загружается в ПЛИС РВС, а трансляция прикладной программы осуществляется на уровень команд вычислительных блоков софт-архитектуры. Подобный подход позволяет сократить время отладки прикладных программ в 2—3 раза по сравнению с существующими методами программирования РВС за счет сокращения времени трансляции.

Возможность программирования РВС на уровне пользовательских софт-архитектур ранее была высказана в ряде работ [2—4], однако методы и средства создания и программирования софт-архитектур разработаны не были. В НИИ многопроцессорных вычислительных систем ЮФУ (НИИ МВС ЮФУ) в 2013 г. в рамках государственного контракта по теме "Разработка реконфигурируемой вычислительной системы РВС-7 и организация на ее основе производства реконфигурируемых вычислительных систем с производительностью до 10^{15} операций в секунду в одностоечном конструктиве 47U" был создан программный комплекс, позволяющий разрабатывать различные пользовательские софт-архитектуры и решать на них задачи цифровой обработки сигналов. Это позволило на практике апробировать принципы и методы программирования, сформулированные в более ранних работах [5].

Иерархия софт-архитектур

Софт-архитектуры имеют иерархическое строение, при котором устройства, расположенные на более низких уровнях, являются элементами для построения устройств на более высоком уровне. Иерархия элементов построения софт-архитектур приведена на рис. 1. Построение софт-архитектур осуществляется на основе макрообъектов, имеющих независимое управление и объединенных информационными и синхронизирующими связями. Макрообъект представляет собой программно-аппаратную заготовку, содержащую в себе не только схемотехнические элементы, но и средства их программирования, которые позволяют осуществлять перенастройку без управления извне.

На нижнем уровне иерархии софт-архитектуры находятся объекты, являющиеся неделимыми элементами при построении более сложных элементов архитектуры. При этом объекты являются единственными элементами архитектуры, разрабатываемыми схемотехнически. Все остальные элементы формируются только с помощью языка описания софт-архитектур SADL.

Рис. 1. Иерархия элементов софт-архитектуры

На следующем уровне располагаются узлы, которые формируются из объектов и созданных ранее узлов. Узлы не могут непосредственно участвовать в построении софт-архитектуры, так как не обладают возможностью самостоятельно настраиваться на решение задач и требуют внешнего управления. Использование узлов позволяет структурировать описание софт-архитектуры в целях упрощения понимания и сопровождения кода.

На следующем уровне располагаются макрообъекты. В качестве элементов макрообъектов выступают объекты и узлы. Формирование макрообъектов и связей между ними выполняется с учетом того, что макрообъект должен иметь возможность самостоятельно настраиваться на решение задач. По этой причине в состав макрообъекта включено устройство, осуществляющее управление информационными потоками и настройку макрообъекта на решение задачи.

Разработка софт-архитектур реконфигурируемых вычислительных систем

Транслятор языка SADL вошел в состав комплекса программного обеспечения (КПО) [6], реализующего программирование РВС на уровне логических ячеек ПЛИС с использованием транслятора языка COLAMO. Для программирования РВС с использованием софт-архитектур потребовалась модернизация структуры КПО. В частности, в структуру КПО были добавлены: транслятор языка описания софт-архитектур SADL, синтезатор конфигураций параллельно-конвейерных вычислитель-

ных структур из макрообъектов Steam! Constructor, библиотека элементов софт-архитектур. В настоящий момент программный комплекс состоит из следующих компонентов:

- транслятора языка SADL;
- транслятора языка высокого уровня COLAMO;
- транслятора языка Argus;
- синтезатора разработки многокристальных схемотехнических решений Fire!Constructor;
- синтезатора Steam!Constructor.

На рис. 2 представлена структура системного программного обеспечения, позволяющая создавать софт-архитектуры для РВС и решать с их помощью прикладных задач.

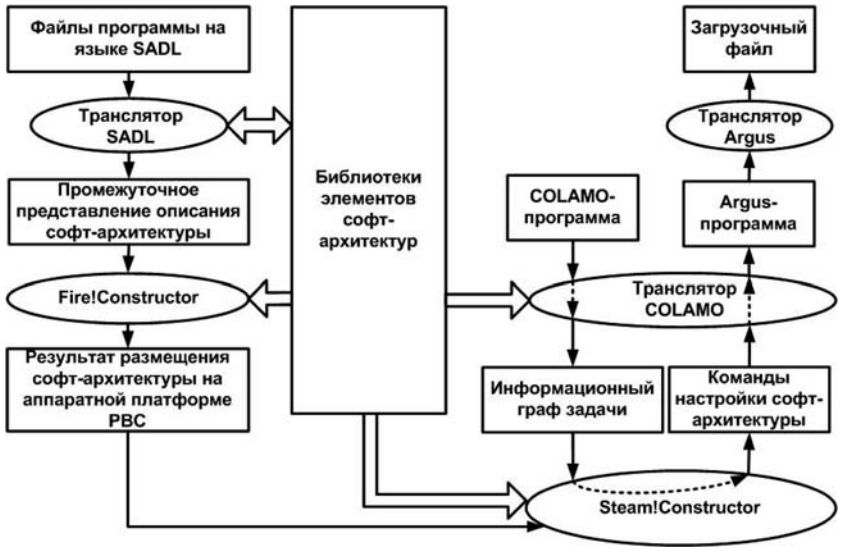


Рис. 2. Структура системного программного обеспечения для решения задач с использованием софт-архитектуры

На основе COLAMO-программы транслятор генерирует информационный граф задачи и указание, на какую из доступных архитектур он должен быть оттранслирован. Информационный граф передается синтезатору Steam!Constructor, который размещает его на софт-архитектуре. Результатом работы синтезатора Steam!Constructor является набор команд для настройки софт-архитектуры. На следующем этапе трансляции в соответствии с командами настройки и потоковой составляющей параллельной COLAMO-программы компилируются загрузочный файл и управляющая программа. Загрузочный файл содержит кодировки для управления софт-архитектурой, а управляющая программа предназначена для управления информационными обменами между вычислительной системой РВС и персональным компьютером.

Работоспособность приведенных методов и комплекса системного программного обеспечения апробирована на задачах цифровой обработки сигналов.

Рассмотрим более подробно процесс создания и программирования софт-архитектур на задаче нахождения спектра комплексного сигнала с использованием алгоритма быстрого преобразования Фурье (БПФ). Разработан следующий перечень элементов, требуемых для решения поставленных задач цифровой обработки сигналов:

- контроллер распределенной памяти, позволяющий управлять информационными потоками;
- функциональные устройства, обеспечивающие выполнение операции суммирования, вычитания, умножения;
- АЛУ, выполняющее операции суммирования или вычитания в зависимости от управляющего сигнала;
- набор мультиплексоров и демультиплексоров для управления информационными потоками внутри кадра;
- память ПЗУ, хранящая коэффициенты W для реализации алгоритма БПФ;
- синхронизирующая память, устанавливаемая между итерациями БПФ, необходимая для формирования информационных потоков в соответствии с графом алгоритма БПФ;
- память ОЗУ, в которой хранятся коэффициенты ядра свертки.

После того как перечень элементов определен, необходимо выполнить их схемотехническое описание. Файлы, содержащие схемотехническое описание каждого объекта, помещаются в библиотеку элементов софт-архитектуры и в дальнейшем используются синтезатором Fire!Constructor для размещения на аппаратной платформе.

На следующем этапе процесса создания софт-архитектуры для каждого из ее элементов разрабатываются файлы описания на языке SADL. В них указывается информация о входах и выходах элемента, его типе, а также описываются вычислительные возможности элемента. В качестве примера приведем описание АЛУ, выполняющего суммирование или вычитание:

```
add_sub_32:ALU;
add_sub_32 (In: in1, in2; Out: out1; InH: H1; VHDLFile:
AS_ALU_32.VHDL);
Var
    in1, in2, outl : integer size 32;
    H1 : integer size 32;
Begin
    Case h1 of
        0: out1 = in1 + in2;
        1: out1 = in1 – in2;
    End;
End;
```

В верхней строке описания элемента указываются его имя (add_sub_32) и тип (ALU). Далее в заголовке выполняется описание его входов/выходов, а также имя VHDL-файла, содержащего схемотехническое описание элемента. В разделе Var

указываются типы входов/выходов и типы переменных, использованных при описании шаблонов вычислений. Далее следует раздел описания шаблонов, которые может выполнять данный элемент.

В приведенном описании АЛУ по управляющему сигналу 0 выполняет суммирование двух чисел из потоков данных in1 и in2, по управляющему сигналу 1 — вычитание.

На основе ранее описанных элементов путем соединения их входов и выходов строятся более сложные объекты: узлы и макрообъекты. Ниже приведен пример узла, выполняющего базовую операцию БПФ:

```
UsesSA;                                  // используемая библиотека элементов софт-архитектуры
Node_BO :Node;                           // имя элемента и тип элемента
Node_BO(In:ReA, ImA, ReB, ImB;           // заголовок элемента с
Out:ReA1, ImA1, ReB1, ImB1; InH: h1);    // указанием типов входов/выходов
Var                                      // описание типов переменных
    AddFU1, AddFU2, AddFU3 : ADD_32X32;
    SubFU1, SubFU2, SubFU3 : SUB_32X32;
    MFU1, MFU2, MFU3, MFU4 : MUL_32X32;
    MemDSP1: KOEF_W64X2048;
    ALU_1, ALU_2 : ADD_SUB_32X32;
    ReA, ImA, ReB, ImB, ReA1, ImA1, ReB1, ImB1: integer size 32;
    ReW, ImW: integer size 32;
    h1: integer size 32;
    a1, a2, b1, b2, b3, b4 : integer size 32;
Begin                                    // описание структуры узла
    MemDSP1(Out: ReW, ImW;               // память коэффициентов БПФ
    Ins: ImB; InH: h1);
    AddFU1(In: ReA, ReB; Out: ReA1);     //сумматор
    AddFU2(In: ImA, ImB; Out: ImA1);     // сумматор
    SubFU1(In: ReA, ReB; Out: a1);       // вычитатель
    SubFU2(In: ImA, ImB; Out: a2);       // вычитатель
    MFU1(In: a1, ReW; Out: b1);          //умножитель
    MFU2(In: a2, ImW; Out: b2);          // умножитель
    MFU3(In: a2, ReW; Out: b3);          // умножитель
    MFU4(In: a1, ImW; Out: b4);          // умножитель
    AddFU3(In: b1, b2; Out: ReB1);       // сумматор
    SubFU3(In: b3, b4; Out: ImB1);       // вычитатель
End;
```

В заголовке описания указываются имя и тип объекта, а также описание его входов и выходов. Так как в состав объекта могут входить различные элементы, описанные ранее, то в разделе описания переменных необходимо указать тип каждого из использованных элементов. Кроме описания типов элементов в данном разделе приводится описание коммутационных переменных, объединяющих элементы в единую вычислительную структуру. В теле описания формируется вычислительная структура будущего узла путем соединения входов и выходов элементов. Из описания видно, что вычислительная структура узла Node_BO состоит из функциональных устройств (сумматоров, вычитателей и умножителей) и памяти DSPMem1, отвечающей за подачу коэффициентов БПФ в соответствии с сигналами управления, поступающими по шине H1. Графическое представление приведенного выше описания приведено на рис. 3.

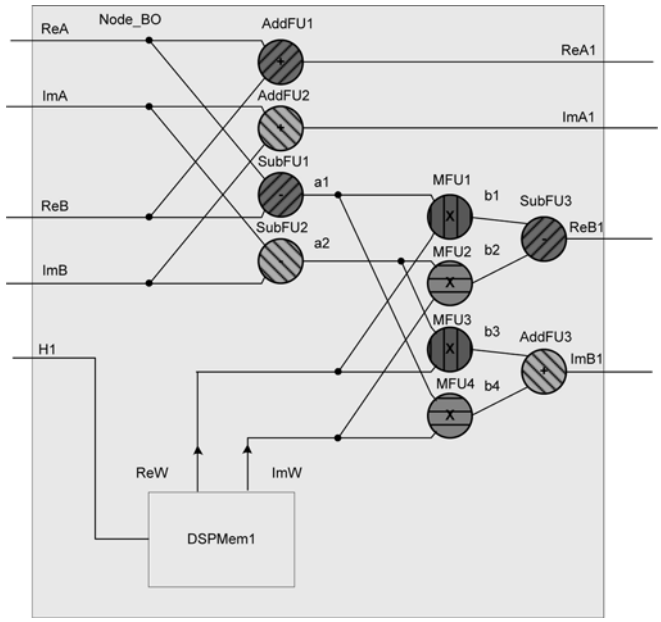


Рис. 3. Узел Node_BO

Софт-архитектура для решения задачи включает в себя следующие объекты (рис. 4):

- восемь контроллеров распределенной памяти (DMC1...DMC8), выполняющих хранение входных, выходных и промежуточных данных алгоритма;
- четыре узла (BONode1, ..., BONode4), реализующих вычисления в соответствии с "бабочкой" БПФ;
- четыре синхронизирующие памяти (DPMem1, ..., DPMem4), формирующие потоки данных в соответствии с алгоритмом БПФ;
- четыре мультиплексора (Mx1, ..., Mx4) и четыре демультиплексора (DMx1, ..., DMx4), с помощью которых программист организует перенаправление информационных потоков.

Узел Node_BO является базовым узлом при решении задач цифровой обработки сигналов. Компонуя различные узлы и элементы в единую вычислительную структуру, архитектор имеет возможность создавать уникальные софт-архитектуры. На рис. 4 приведена типовая структура макрообъекта для решения задач цифровой обработки сигналов, основанных на алгоритме БПФ. Вычислительный конвейер софт-архитектуры SA_DSP состоит из четырех ступеней, каждая из которых выполняет одну итерацию алгоритма БПФ. Во всех ступенях конвейера имеется память типа DPMem для вре-

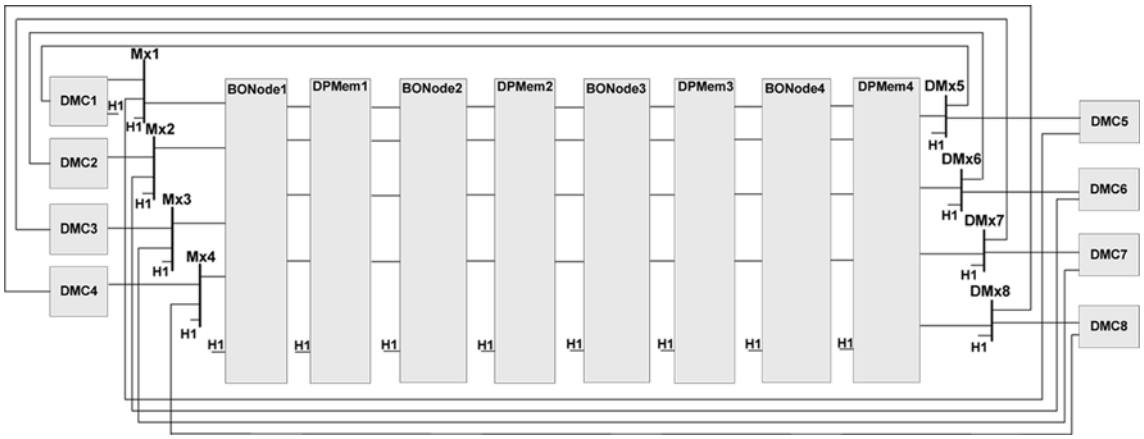


Рис. 4. Структура макрообъекта MO_DSP

менно́го согласования данных при выполнении базовых операций соседних итераций алгоритма БПФ. Задача нахождения спектра сигнала длиной 256 отсчетов с помощью приведенной софт-архитектуры решается за два кадра.

Перед первым кадром софт-архитектура настраивается на выполнение 1, 2, 3, 4 итераций БПФ. При этом промежуточные данные записываются в контроллер распределенной памяти с номерами 5, 6, 7, 8. После этого архитектура перестраивается на выполнение 5, 6, 7, 8 итераций алгоритма БПФ, а результирующие данные записываются в контроллер распределенной памяти с номерами 1, 2, 3, 4.

В соответствии с алгоритмом создания софт-архитектур, после того как архитектура сформирована и оттранслирована в промежуточное представление, она может быть размещена на вычислительном ресурсе РВС. Эту функцию выполняет синтезатор масштабируемых параллельно-конвейерных процедур на уровне логических ячеек ПЛИС Fire!Constructor. В качестве входных данных синтезатор использует: библиотечные файлы элементов софт-архитектуры, описание софт-архитектуры, описание аппаратной платформы РВС. Описания вычислительных модулей РВС включают информацию о числе и типах ПЛИС, а также о существующих физических связях между кристаллами ПЛИС вычислительного модуля.

В результате отображения софт-архитектуры на аппаратный ресурс РВС синтезатором Fire!Constructor осуществляется разбиение иерархичного информационного графа софт-архитектуры на подграфы в соответствии с описанием структуры РВС и выполняется трассировка информационных связей между кристаллами ПЛИС. Для каждого кристалла ПЛИС генерируется описание на языке VHDL, соответствующее фрагменту софт-архитектуры, представленному подграфом информационного графа, размещенного в данном кристалле ПЛИС. Кроме того, для каждой ПЛИС формируется UCF-описание, которое содержит назначение входных/выходных связей фрагмента софт-архитектуры с внешними выводами ПЛИС и дополнительную информацию, необходимую для синтеза конфигурационных файлов ПЛИС. Конфигурационные файлы ПЛИС формируются с помощью средств разработки схемотехнических решений, предоставляемых фирмами-изготовителями ПЛИС, или аналогичным программным обеспечением.

В качестве аппаратной платформы для размещения софт-архитектуры SA_DSP был использован базовый модуль 24V7-750, имеющий следующие технические характеристики:

- число ПЛИС Virtex-7 (58,5 млн вентилей) — 6 шт;
- число элементарных процессоров — 1350 шт.;
- производительность — 720 (340) Гфлопс;
- частота работы платы — 400 МГц;
- частота информационных обменов — 1200 МГц.

Этап получения конфигурационных файлов ПЛИС является последним в алгоритме создания софт-архитектуры. В результате загрузки конфигурационных файлов в РВС на базе реконфигурируемого вычислителя формируется архитектура специализированного вычислительного устройства, соответствующего разрабатываемой софт-архитектуре, на котором могут решаться различные параллельные прикладные задачи без перезагрузки конфигурационных файлов и без необходимости многократной трансляции на уровень логических ячеек ПЛИС при отладке программы.

Решение прикладных задач с использованием софт-архитектур

Процесс решения задач с использованием софт-архитектур начинается с разработки параллельной программы на языке COLAMO. Основная часть программы для вычисления по алгоритму БПФ с длинной сигнала 256 отсчетов представляет собой два кадра. Первый кадр настраивает архитектуру на выполнение 1...4 итераций БПФ, второй кадр настраивает архитектуру на вычисление 5...8 итераций. В качестве примера ниже приведено описание первого кадра программы.

```
ProgramTEST_FFT;
Include V7_Pleyada.Pleayda, V7_SADL.LibMacroObject_SA_Int;
Architecture SA_DSP;
Var DPRam0A, DPRam0B, DPRam0C, DPRam0D : Array Integer
[100 : Stream] mem;
Var DPRam5A, DPRam5B, DPRam5C, DPRam5D : Array Integer
[100 : Stream] mem;
Var DPRam1A, DPRam1B, DPRam1C, DPRam1D : integer com;
Var DPRam2A, DPRam2B, DPRam2C, DPRam2D : integer com;
Var DPRam3A, DPRam3B, DPRam3C, DPRam3D : integer com;
Var DPRam4A, DPRam4B, DPRam4C, DPRam4D : integer com;
Var ReA0, ImA0, ReB0, ImB0 : integer com;
Var ReA1, ImA1, ReB1, ImB1 : integer com;
Var ReA2, ImA2, ReB2, ImB2 : integer com;
Var ReA3, ImA3, ReB3, ImB3 : integer com;
Var ReA4, ImA4, ReB4, ImB4 : integer com;
Vari :number;
Var co1 = $141;Var co2 = $242; Var co3 = $343; Var co4 = $444;
Var coM1 = $0; Var coM2 = $1; Var coM3 = $2; Var coM4 = $3;
Cadr Cadr1;
for i:=0 to 63 do
    Begin
ReA0 := DPRam0A[i];
ImA0 := DPRam0B[i];
ReB0 := DPRam0C[i];
ImB0 := DPRam0D[i];
BONode(ReA0, ImA0, ReB0, ImB0, coM1, ReA1, ImA1, ReB1,
ImB1);
DPMem1(ReA1, ImA1, ReB1, ImB1, co1, DPRam1A, DPRam1B,
DPRam1C, DPRam1D);
BONode(DPRam1A, DPRam1B, DPRam1C, DPRam1D, coM2,
ReA2, ImA2, ReB2, ImB2);
DPMem2(ReA2, ImA2, ReB2, ImB2, co2, DPRam2A, DPRam2B,
DPRam2C, DPRam2D);
BONode(DPRam2A, DPRam2B, DPRam2C, DPRam2D, coM3,
ReA3, ImA3, ReB3, ImB3);
DPMem3(ReA3, ImA3, ReB3, ImB3, co3, DPRam3A, DPRam3B,
DPRam3C, DPRam3D);
BONode(DPRam3A, DPRam3B, DPRam3C, DPRam3D, coM4,
ReA4, ImA4, ReB4, ImB4);
DPMem4(ReA4, ImA4, ReB4, ImB4, co4, DPRam4A, DPRam4B,
DPRam4C, DPRam4D);
        DPRam5A[i] := DPRam4A;
        DPRam5B[i] := DPRam4B;
        DPRam5C[i] := DPRam4C;
        DPRam5D[i] := DPRam4D;
    End;
EndCadr;
```

Константы co1, .., co4 настраивают память коэффициентов БПФ, а константы coM1,..., coM4 настраивают работу синхронизирующей памяти в соответствии с алгоритмом решения задачи.

Результатом работы транслятора COLAMO являются:

- объектная модель вычислительного графа задачи, передаваемая в синтезатор Steam!Constructor;
- проект параллельно-конвейерной программы на языке Argus и конфигурационный файл, содержащий данные о настройке архитектуры на решение текущей задачи;
- управляющая программа на языке высокого уровня VisualC#, организующая обмен данными между персональным компьютером и РВС.

Объектная модель вычислительного графа задачи представляет собой XML-файл с описанием структуры каждого кадра многокадровой прикладной программы. Описание структуры кадра содержит информацию о так называемом плоском графе кадра: об информационных и операционных вершинах; о комплексных вершинах, которые соответствуют некоторому блоку элементарных операций; об информационных связях между всеми описанными вершинами "плоского" графа кадра. Кроме того, XML-описание также содержит информацию о последовательности выполнения кадров в программе, информацию о предварительной загрузке параметров для каждого кадра (в случае, если такая загрузка необходима) и другие параметры прикладной программы, необходимые для ее отображения на архитектуру РВС.

На основании полученных данных синтезатор Steam!Constructor выполняет автоматическое отображение информационных графов многокадровых прикладных программ на софт-архитектуру РВС. В процессе отображения прикладной программы на софт-архитектуру РВС выполняются: одновременное разбиение информационного графа каждого из кадров на подграфы согласно модифицированному описанию софт-архитектуры РВС, размещение данных подграфов в ПЛИС РВС и трассировка информационных каналов внутри каждой ПЛИС и между кристаллами ПЛИС согласно информационным связям текущего кадра. Для каждого кристалла ПЛИС формируется XML-структура, содержащая параметры, описывающие функционирование внутренних объектов данной ПЛИС, задействованных при отображении информационных графов. Аналогичные XML-структуры формируются и для входов/выходов кристаллов ПЛИС, задействованных при формировании информационных каналов. Полученные XML-структуры составляют файл результатов отображения, который передается синтезатором Steam!Constructor транслятору COLAMO для дальнейшей обработки. На основании переданного файла с результатами отображения транслятором COLAMO формируются загрузочный out-файл и управляющая программа. Весь процесс трансляции параллельной программы на софт-архитектуре занимает не более десяти минут, что существенно быстрее, чем при трансляции программы на уровень логических ячеек ПЛИС.

Заключение

Практическое использование комплекса программного обеспечения при создании и модификации софт-архитектуры SA_DSP позволило в два раза ускорить процесс отладки прикладных программ за счет уменьшения времени трансляции. Включение в состав программного обеспечения транслятора языка SADL позволило достаточно быстро разрабатывать и модифицировать пользо-

вательские софт-архитектуры. Использование синтезатора Fire!Conatructor дало возможность автоматического размещения софт-архитектуры на аппаратной платформе и избавило пользователя от необходимости детального изучения аппаратного ресурса РВС. При решении прикладных задач с использованием РВС транслятор COLAMO и синтезатор Water!Constructor автоматически и за короткое время (до 10 мин) преобразуют текст параллельной программы в набор кодов управления софт-архитектуры, что позволяет прикладному программисту получить результат за более короткий срок по сравнению с существующими подходами к программированию РВС.

Список литературы

1. **Gudkov V. A., Gulenok A. A., Kovalenko V. B., Slasten L. M.** Preprints of the 12th IFAC Conference on Programmable Devices and Embedded Systems PDES — 2013, Technical University of Ostrava, Czech Republic. 2013. P. 65—70.

2. **Каляев И. А., Левин И. И., Семерников Е. А., Шмойлов В. И.** Реконфигурируемые мультиконвейерные вычислительные структуры / Под общ. ред. И. А. Каляева. 2-е изд., перераб. и доп. Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. 344 с.

3. **Левин И. И.** Многопроцессорная система с программированием архитектуры на нескольких уровнях // Тр. Первой Всероссийской науч. конф. "Методы и средства обработки информации". М.: Изд. МГУ. 2003. С. 111—118.

4. **Семерников Е. А., Дмитренко Н. Н., Каляев И. А., Левин И. И.** Семейство многопроцессорных вычислительных систем с динамически перестраиваемой архитектурой // Вестник компьютерных и информационных технологий. 2009. Ч. 1. № 6. С. 2—8. Ч. 2. № 7. С. 2—10.

5. **Коваленко В. Б., Семерников Е. А.** Организация многоуровневого программирования реконфигурируемых вычислительных систем // Вестник компьютерных и информационных технологий. 2011. № 9. С. 3--10.

6. **Левин И. И., Дордопуло А. И., Сластен Л. М., Гудков В. А.** Средства программирования реконфигурируемых многопроцессорных вычислительных систем // Известия ТРТУ. 2006. № 16. Специальный выпуск. С. 16--20.

V. B. Kovalenko[1], Senior Researcher, e-mail: vereten@hotbox,
A. I. Dordopulo[2], Head of Laboratory, e-mail: scorpio@mvs. tsure.ru,
V. A. Gudkov[2], Senior Researcher, e-mail: gudkov@mvs.tsure.ru,
A. A. Gulenok[2], Senior Researcher, e-mail: andrei-gulenok@mail.ru,
L. M. Slasten[2], Senior Researcher, e-mail: emslasten@yandex.ru
[1] Southern Scientific Centre of the Russian Academy of Sciences
[2] Academical A. V. Kalyaev Scientific Research Institute of Miltiprocessor Computer at Southern Federal University

Use of Soft-Architectures for Digital Signal Processing Tasks Solved on Reconfigurable Computer Systems

The paper covers the macroobject approach to reconfigurable computer system (RCS) programming, which reduces the programming time of tasks owing to reduction of the translation time of parallel programs. The description of the main advantages and shortcomings of the existing RCS programming levels are given. We have analysed the notion of a soft-architecture, a macroobject, a node and objects of reconfigurable computer systems. We have given the structure of system software owing to which the suggested approach to RCS programming is provided. In addition we have described functions of the elements of the software suit and their interconnection during development of soft-architectures, and use of soft-architectures for implementation of application tasks. The development of the soft-architecture for digital signal processing tasks is analysed, and examples of SADL-descriptions of soft-architecture elements are given. The example of the soft-architecture used for implementation of digital signal processing tasks is given. Besides, we have given the description of one cadr of the parallel application, written in the high-level programming language COLAMO. It implements the FFT algorithm on the base of the DSP soft-architecture.

Keywords: *multiprocessor systems, supercomputers, reconfigurable computer systems, architectures of computer systems, parallel programming, programming languages*

References

1. **Gudkov V. A., Gulenok A. A., Kovalenko V. B., Slasten L. M.** Preprints of the 12th IFAC Conference on Programmable Devices and Embedded Systems PDES — 2013. Technical University of Ostrava, Czech Republic. 2013. P. 65—70.

2. **Kalyaev I. A., Levin I. I., Semernikov E. A., Shmoilov V. I.** Rekonfiguriruemiye multikonveyemiye vychislitelniye struktury. Pod obtsh. red. I. A. Kalyaev. 2nd-e izd., pererab i dop. Rostov-na-Donu: Izd-vo YuNC RAN, 2009. 344 p.

3. **Levin I. I.** Mnogoprotsessomaya systema s programmirovaniyem arkhitektury na neskolkykh urovnyakh. *Tr. Pervoi Vserossiiskoy nauch. konf. "Metody i sredstva obrabotki informatsii"*. M.: MGU, 2003. P. 111—118.

4. **Semernikov E. A., Dmitrenko N. N., Kalyaev I. A., Levin I. I.** Semeistvo mnogoprotsessornykh vychislitelnykh system s dinamicheski perestraiyayemoi arkhitekturoi. *Vestnik kompiuternykh i informatsionnykh tekhnologiy*. 2009. Ch. 1. N. 6. P. 2—8. Ch. 2. P. 2—10.

5. **Kovalenko V. B., Semernikov E. A.** Organizatsiya mnogourovnevogo programmirovaniya rekonfiguriruemykh vychislitelnykh system. *Vestnik kompiuternykh i informatsionnykh tekhnologiy*. 2011. N. 9. P. 3—10.

6. **Levin I. I., Dordopulo A. I., Slasten L. M., Gudkov V. A.** Sredstva programmirovaniya rekonfiguriruemykh mnogoprotsessornykh vychislitelnykh system. *Izv. TRTU*. 2006. N. 16. Spec. vyp. P. 16—20.

УДК 004.4, 004.65

А. В. Посконин, аспирант, e-mail: aposk@yandex.ru,
Московский государственный университет им. М. В. Ломоносова

Интеграция SQL-ориентированных СУБД и NoSQL-систем на уровне объектного отображения

Рассмотрены вопросы интеграции разнородных систем управления данными: традиционных SQL-ориентированных СУБД и новых NoSQL-систем. На примере реализованного прототипа программной библиотеки показаны возможности интеграции различных решений для управления данными на уровне объектного отображения, включая унифицированный язык запросов и отображение объектов языка программирования на различные модели данных. Приводятся результаты тестов производительности полученного прототипа.

Ключевые слова: *SQL, NoSQL, polyglot persistence, сервис-ориентированная архитектура, объектно-реляционное отображение, объектно-документное отображение, интеграция*

Введение

Современные приложения вынуждены работать с данными различной структуры, объема и степени важности, предъявляя к системам хранения данных достаточно жесткие (а порой и невыполнимые) требования. Отсутствие универсального решения для работы с данными привело к появлению на рынке большого числа узкоспециализированных систем, значительно отличающихся от традиционных SQL-ориентированных СУБД. Этот новый класс систем получил собирательное название "NoSQL" ("Not Only SQL"), подчеркивающее, что в ряде задач применение реляционных СУБД может быть не самым эффективным решением.

Стремительное развитие Интернет и Web-приложений вывело на первый план проблему масштабируемости при возрастающих нагрузках и объемах информации, которую NoSQL-системы стараются решить с использованием распределенных архитектур в противоположность классическим "односерверным" СУБД. Однако поддержка согласованности данных (и, особенно, поддержка ACID-транзакций) в условиях распределенности увеличивает не только сложность системы, но и время отклика из-за большого числа сетевых взаимодействий. По этой причине NoSQL-системы ослабляют гарантии согласованности данных и отказываются от ACID-транзакций в пользу масштабируемости, скорости и высокой доступности данных (см., например, [1]). Системы категории NoSQL, кроме того, основаны на нереляционных моделях данных (например, ключ-значение, документная модель и т. д.), ориентированы в основном на работу с неструктурированными или слабоструктурированными данными и используют отличные от SQL языки запросов и интерфейсы доступа к данным. Стоит также упомянуть о появлении нового поколения SQL-ориентированных СУБД, изначально создаваемых для работы в распределенной среде — так называемых NewSQL-системах. Эти СУБД поддерживают транзакционную семантику и язык SQL, но обеспечивают при этом приемлемую масштабируемость. Подробные обзоры существующих NoSQL- и NewSQL-решений можно найти, например, в работах [2—4].

Богатый выбор различных систем для управления данными обусловлен тенденцией к специализации инструментов: если раньше SQL-ориентированные системы рассматривались как универсальные СУБД, способные решать практически любые задачи, то сегодня для каждой задачи можно выбрать наиболее оптимальное и удобное решение (см., например, [5]). Например, если приложению приходится работать с сущностями, обладающими переменным набором атрибутов, то практичнее использовать NoSQL-системы с документной моделью данных, чем реализовывать модель Entity-Attribute-Value [6] средствами SQL. Документная модель данных позволяет хранить объекты с произвольным набором атрибутов, обычно представляемые в JSON и называемые "документами". При этом обычно допускаются списки и вложенные документы, поддерживаются индексы и выборки на основе полей документов. Сами документы обычно хранятся в коллекциях, не накладывающих каких-либо ограничений на набор атрибутов входящих в них документов (отсутствие строгой схемы данных). Одной из наиболее популярных систем такого типа является MongoDB. Эта система имеет

Рис. 1. Реляционная и документная модели данных

богатую функциональность и поддерживает атомарные операции на уровне одного документа, однако если приложение нуждается в полноценных ACID-транзакциях, то SQL-ориентированные системы по-прежнему вне конкуренции. На рис. 1 приведено соответствие между реляционной (SQL-ориентированной) и документной моделями данных.

Polyglot Persistence

Термин "Polyglot Persistence" [7] применяется в англоязычных источниках для обозначения практики использования нескольких различных систем хранения данных в рамках одного приложения. Потребность в этом обусловлена тем фактом, что современные приложения (особенно Web-приложения) вынуждены работать одновременно с несколькими видами данных. Например, это может быть информация о клиентах, каталог товаров, журналы событий и переходов по ссылкам, данные пользовательских сессий и т. д. Все эти данные не только обладают различной структурой и соотношением операций чтения/записи, но и предъявляют разные требования к надежности хранения и возможностям языка запросов. Таким образом, естественно применить для работы с этими данными несколько систем хранения (см., например, [8, 9]). Пример такого подхода приведен на рис. 2, показывающем применение различных систем управления данными для компонентов платформы электронной коммерции.

Однако применение такого подхода усложняет приложение, требуя использования различных языков запросов и интерфейсов для доступа к данным. Одним из решений этой проблемы является применение сервис-ориентированной архитектуры (Service-Oriented Architecture, SOA). При этом все детали доступа к данным эффективно скрываются за интерфейсом соответствующего сервиса, позволяя, к тому же, разбить приложение на более или менее независимые части, что упрощает управление разработкой. Этот подход, однако, требует значительных дополнительных усилий и накладных расходов на организацию взаимодействия между сервисами, что делает его применение нецелесообразным для небольших приложений.

Рис. 2. Polyglot Persistence

При разработке приложений, использующих системы хранения данных, практически всегда возникает проблема, известная как "impedance mismatch", а именно — необходимость каким-либо образом отображать объектную модель приложения на модель данных целевой системы хранения. В случае реляционных СУБД существует множество готовых решений и подходов для осуществления объектно-реляционного отображения (Object-Relational Mapping, ORM). Однако и для NoSQL-решений бывает необходимо осуществлять подобное отображение, например, объектно-документное отображение (Object-Document Mapping, ODM).

Применение готовых решений для объектного отображения, таких как ORM-библиотеки, упрощает и ускоряет разработку приложений, предоставляя высокий уровень абстракции и богатую функциональность. К сожалению, высокий уровень абстракции заметно снижает производительность и делает практически невозможными низкоуровневые оптимизации под конкретную используемую СУБД. По этой причине подобные решения редко применяются в высоконагруженных приложениях, где предпочтительнее использование более оптимизированных под конкретную задачу реализаций (см. [10]).

Интеграция на уровне объектного отображения

Слой объектного отображения представляется подходящим для интеграции разнородных систем хранения данных, однако нужно постараться избежать недостатков, присущих традиционным библиотекам объектного отображения. Для этого требуется пересмотр классической архитектуры ORM-библиотек, а именно:

- использование модульного подхода вместо монолитной системы: слой объектного отображения может быть собран из отдельных "строительных блоков", чтобы лучше отвечать потребностям разработчика. Кроме того, если стандартные реализации компонентов по каким-то причинам не подходят, они должны быть легко заменяемыми;
- максимально "чистая" объектная модель, свободная от логики взаимодействия с подсистемами хранения. Сущность, в отличие от случая ORM, может содержать не только поля простых типов,

но также списки, вложенные объекты и динамические свойства;
- настраиваемый уровень абстракции: поддержка интерфейсов разных уровней позволяет найти нужный компромисс между уровнем абстракции и производительностью;
- независимое отображение сущностей и поддержка ассоциаций поверх слоя отображения позволяет иметь целостную объектную модель, несмотря на то, что сущности могут отображаться на разные системы хранения;
- наличие унифицированного языка запросов с поддержкой базовой функциональности, не привязанного к SQL или другому языку запросов;
- возможность усилить гарантии целостности данных с помощью валидации на стороне приложения;
- поддержка кэширования, "ленивой" загрузки, Unit of Work и других успешно себя зарекомендовавших практик (см., например, [11]).

Приведенные принципы были применены при разработке прототипа программного каркаса MapperStack для интеграции разнородных систем хранения данных на уровне объектного отображения в Web-приложениях на PHP. В архитектуре MapperStack можно выделить два основных слоя: слой отображения, состоящий из объектов, реализующих интерфейс Mapper (например, методы getEntityId(), find(), fmdBy() и т. д.), и верхний слой MapperStack, координирующий работу слоя отображения и предоставляющий возможности интеграции, в том числе поддержку ассоциаций (рис. 3).

Каждый объект в слое отображения ответственен за отображения сущностей определенного класса. Каждая сущность должна иметь уникальный идентификатор (аналог первичного ключа в реляционной модели данных) и может содержать динамические свойства, списки и вложенные объекты, а также метаданные, определяющие правила отображения, например:

```
class BlogPost extends MapperStack\Object
{
    protected $id;
    protected $username;
    protected $text;
    protected $tags;
    protected $comments;
    public static function meta(EntityMetaClass $metaClass)
    {
    $metaClass->db('blog')
                ->collection('posts')
                ->field('id', 'integer', '_id')
                ->field('username', 'string')
                ->field('text', 'string')
                ->field('tags', 'string[]')
                ->object('comments', 'Blog\Comment[]')
                ->id('id');
    }
}
```

В настоящее время реализовано отображение сущностей на MySQL и MongoDB, а также унифицированный язык запросов с трансляцией в SQL и запросы к MongoDB. Поддерживаются операции filter, sort, skip, limit, project и другие, условия выборки задаются в виде логического выражения, поддерживаются предикаты (как механизм расширения языка запросов), связываемые и встроенные параметры. Кроме того, если целевая система хранения данных поддерживает документную модель данных, то в запросах можно использовать операции сопоставления элементов коллекции и переход по графу объектов, чтобы добраться до атрибутов вложенных объектов.

Рассмотрим пример запроса в терминах сущностей и результирующие запросы к целевым системам хранения данных:

```
$p := $m->query ( ' Domain\Product' )
        ->filter('price >= 1000 & & type == "shoes"')
        ->sort('price', 'asc')
        ->limit(3)
        ->getResult() ;
```

Этот запрос допускает трансляцию как в SQL, так и в язык запросов MongoDB (сопоставление по образцу):

```
SELECT*FROM "products'     db.products.find( {
    WHERE 'price' > 1000          $and : [
        AND 'type' = 'shoes'          { 'price' : { $gte : 1000} },
    ORDER BY 'price'              j 'type' : 'shoes' }
    LIMIT 3                   ] } ).sort( { 'price' : 1 } ).limit(3)
```

Рассмотрим теперь запрос, содержащий операцию сопоставления элементов массива (в данном случае в массиве comments ищется элемент, для которого выполняется условие в фигурных скобках, кроме того, присутствует предикат регулярного выражения). Такой запрос не может быть выражен средствами SQL, но поддерживается в MongoDB:

```
$posts = $m->query('Blog\BlogPost')
        ->filter('regexp(text, "/programming/i" & &
                comments {username == "ivan"} || rating > 100')
        ->getResult ();
```

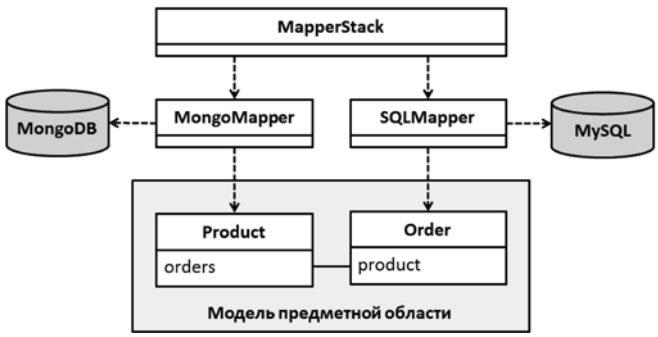


Рис. 3. Отображение сущностей с помощью MapperStack

Результирующий запрос к MongoDB будет выглядеть следующим образом (можно отметить, что запрос в MapperStack является более наглядным и привычным):

```
db.posts.find( { $or : [
                { $and : [
                    { 'text' : /programming/i },
                    { 'comments' :
                        { $elemMatch: { 'username' : 'ivan' } }
                    }
        ] },
        { 'rating' : { $gt : 100 } }
    ] } ) ;
```

Для непосредственной организации отображения в MapperStack реализован ряд компонентов, таких как классы для хранения метаданных, адаптеры для систем кэширования, абстракция отображения типа, механизмы для отслеживания изменений и преобразования данных, Unit of Work (откладывание всех операций модификации данных до вызова метода flush(), синхронизующего состояние с системой хранения и выполняющего операции с учетом ассоциаций [12]) и ряд других компонентов. Каждый компонент может быть легко заменен другой реализацией, кроме того, возможны полностью пользовательские реализации классов отображения. Дальнейшее развитие данного подхода может включать поддержку других типов систем, реализацию оптимистических блокировок, абстракцию MapReduce, поддержку модификации без чтения, агрегатных функций, наследования и т. д.

Производительность библиотеки

В заключение рассмотрим некоторые аспекты производительности библиотеки MapperStack на примере модуля объектно-реляционного отображения (SQLMapper). Для этого был разработан тестовый сценарий, включающий выборку сущностей, модификацию полей, удаление сущностей и фиксирование сделанных изменений в базе данных. В качестве эталона приведены характеристики для того же сценария, реализованного с помощью библиотеки Doctrine ORM — мощной и хорошо оптимизированной библиотеки объектно-реляционного отображения для PHP (см. таблицу).

Результаты исполнения тестового сценария

Библиотека	Время работы, мс	Пиковое потребление памяти, Кбайт
MapperStack SQLMapper (первый запуск)	1625	10190
MapperStack SQLMapper (дальнейшие запуски)	595	6940
Doctrine ORM (первый запуск)	1540	9472
Doctrine ORM (дальнейшие запуски)	560	6144

При первом запуске сценария происходит анализ и трансляция запросов с последующим кэшированием, чем и обусловлено несколько большее время работы. Кэш результатов запросов при тестировании не использовался. В плане производительности SQLMapper незначительно уступает специализированным ORM-библиотекам (главным образом вследствие большей модульности и слабой связанности компонентов программного каркаса). Стоит заметить, что с использованием компонентов MapperStack можно реализовать и более производительное отображение (например, работая напрямую с драйвером СУБД, пропуская этап преобразования типов и т. д.).

Заключение

Реализация модульного программного каркаса для интеграции различных систем хранения на уровне объектного отображения позволяет быстро разработать прототип приложения, используя стандартную функциональность, а затем при необходимости постепенно заменять их на более эффективные реализации исходя из решаемых задач. Интеграция различных систем хранения открывает новые возможности, позволяя использовать наиболее подходящие решения для работы с данными, сохраняя при этом целостную объектную модель приложения. Это актуально, прежде всего, для Web-приложений и мобильных сервисов, хранящих и обрабатывающих большие объемы разнородных данных.

Список литературы

1. **Merriman D.** On Distributed Consistency, 2010. URL: http://blog.mongodb.org/post/475279604/on-distributed-consistencv-part-1
2. **Cattell R.** Scalable SOL and NoSQL Data Stores, 2011. URL: http://www.cattell.net/datastores/Datastores.pdf
3. **Aslett M.** NoSQL, NewSQL and Beyond: The drivers and use cases for database alternatives // Отчет компании 451 Group, 2011. URL: https://451research.com/report-long?icid=1651
4. **Кузнецов С. Д., Посконин А. В.** Распределенные горизонтально масштабируемые решения для управления данными // Труды Института системного программирования РАН. 2013. Т. 24. С. 327—358.
5. **Stonebraker M., Cetintemel U.** "One Size Fits All": An Idea Whose Time Has Come and Gone // ICDE '05: Proceedings of the 21st International Conference on Data Engineering, Вашингтон, 2005.
6. **Entity-Attribute-Value** model. URL: http://en.wikipedia.org/wiki/Entity %E2 %80 %93attribute %E2 %80 %93value model
7. **Fowler M.** Polyglot Persistence, 2011. URL: http://www.martinfowler.com/bliki/PolyglotPersistence.html
8. **Кузнецов С. Д., Посконин А. В.** Возможно ли сотрудничество SQL и NoSQL? // Открытые системы. СУБД. 2013. N. 9. С. 38—41.
9. **Francia S., Hileman J.** Augmenting RDBMS with MongoDB for eCommerce. URL: http://www.nosqldatabases.com/main/2011/4/11/augmenting-rdbms-with-mongodb-for-ecommerce.html
10. **Посконин А. В.** Web-приложения и данные проблемы абстракции и масштабируемости // Труды Института системного программирования РАН. 2012. Т. 23. С. 159—171.
11. **Miller J.** Design Patterns for Data Persistence, 2009. URL: http://msdn.microsoft.com/en-us/magazine/dd569757.aspxy
12. **Fowler M.** Patterns of Enterprise Application Architecture. Бостон: Addison Wesley, 2002.

A. V. Poskonin, Postgraduate, e-mail: aposk@yandex.ru, Lomonosov Moscow State University

Integrating SQL-Based DBMSs with NoSQL Datastores at the Object-Mapping Layer

This paper will discuss and evaluate an approach to building an object-mapping layer that provides integration of multiple different data management systems, including SQL-based DBMSs and popular NoSQL systems. A prototype object-mapping framework was developed to illustrate basic concepts of such integration including unified entity-based query language, transparent object mapping and a high-level interface, that still allows low-level optimizations and delivers sufficient performance for most use cases. Unified query language makes it possible to translate a single query to different underlying datastores without a need to modify the query while object-mapping layer manages object construction and lifecycle, providing simple interface for persisting, modifying and deletion of objects. In conclusion, performance of the implemented framework is measured and compared with popular ORM solution.

Keywords: *SQL, NoSQL, polyglot persistence, service-oriented architecture, object-relational mapping, object-document mapping, integration*

References

1. **Merriman D.** *On Distributed Consistency.* 2010. URL: http://blog.mongodb.org/post/475279604/on-distributed-consistencv-part-1
2. **Cattell R.** *Scalable SQL and NoSQL Data Stores.* 2011. URL: http://www.cattell.net/datastores/Datastores.pdf
3. **Aslett M.** *NoSQL, NewSQL and Beyond:* The drivers and use cases for database alternatives, 451 Group, 2011. URL: https://451research.com/report-long?icid=1651
4. **Kuznetsov S. D., Poskonin A. V.** Raspredelennye gorizontal'no masshtabiruemye reshenija dlja upravlenija dannymi. *Trudy Instituta sistemnogo programmirovanija* RAN. 2013. V. 24 P. 327—358.
5. **Stonebraker M., Cetintemel U.** "One Size Fits All": An Idea Whose Time Has Come and Gone, ICDE '05. *Proceedings of the 21st International Conference on Data Enegineering, Washington*, 2005.
6. **Entity-Attribute-Value** model. URL: http://en,wikipedia.org/wiki/Entitv %E2 %80 %93attribute %E2 %80 %93value_model
7. **Fowler M.** *Polyglot Persistence*, 2011. URL: http://www.martin-fpwler.com/bliki/PolyglotPersistence.html
8. **Kuznetsov S. D., Poskonin A. V.** Vozmozhno li sotrudnichestvo SQL i NoSOL? *Otkrytye sistemy. SUBD.* 2013. N. 9. P. 38—41.
9. **Francia S., Hileman J.** *Augmenting RDBMS with MongoDB for eCommerce.* URL: http://www.nosqldatabases.com/main/2011/4/11/augmenting-rdbms-with-mongodb-for-ecommerce.html
10. **Poskonin A. V.** Web-prilozhenija i dannye: problemy abstrakcii i sshtabiruemosti. *Trudy Instituta sistemnogo programmirovanija RAN.* 2012. V. 23. P. 159—171.
11. **Miller J.** *Design Patterns for Data Persistence.* 2009. URL: http://msdn.microsoft.com/en-us/magazine/dd569757.aspx
12. **Fowler M.** *Patterns of Enterprise Application Architecture.* Boston: Addison Wesley, 2002.

УДК 681.518: 339.13

Э. М. Димов, д-р техн. наук, проф.,
О. Н. Маслов, д-р техн. наук, проф., зав. каф., e-mail: maslov@psati.ru,
Ю. В. Трушин, канд. техн. наук, доц.,
Поволжский государственный университет телекоммуникаций и информатики, г. Самара

Выбор средств программного обеспечения процесса статистического имитационного моделирования

Рассматривается проблема выбора средств программного обеспечения процесса статистического имитационного моделирования (СИМ) сложных систем организационно-технического типа. Представлены примеры реализации СИМ-моделей с применением универсального программного языка Delphi и среды имитационного моделирования AnyLogic.

Ключевые слова: *метод статистического имитационного моделирования, средства программного обеспечения, универсальный язык Delphi, среда имитационного моделирования AnyLogic*

Введение

Разработка и реализация систем управления (СУ) иерархическими и многокритериальными сложными системами (СС) организационно-технического типа (социально-экономическими, экологическими, военными и т. п.), элементами которых являются лица, принимающие решения (ЛПР), до настоящего времени является актуальной проблемой, которая имеет важное практическое значение. Решению данной проблемы способствует создание

"имитационных систем" [1, 2], каждая из которых представляет собой человеко-машинную диалоговую СС и состоит из математических (алгоритмических) моделей и системы процедур, объединяющей эти модели с ЛПР, а также специального математического обеспечения, включающего систему алгоритмов для решения конкретных задач, в том числе оптимизационных.

Перспективы повышения эффективности действий ЛПР в составе СУ с помощью статистического имитационного моделирования (СИМ) по версии метода Димова—Маслова (МДМ) рассмотрены в работе [1]. Согласно методике СИМ по МДМ, выбор программного обеспечения (ПО) является одним из важных предварительных этапов реализации СИМ-модели [2]. Помимо стоимости, которая в условиях рынка имеет существенное значение, выбранный вариант ПО должен быть общедоступным и универсальным, приемлемым для ЛПР по эффективности и комфорту использования. Цель статьи — не претендуя на завершенность анализа, дать оценку вариантов ПО, пригодных для реализации СИМ-моделей по МДМ, наиболее распространенных в настоящее время.

Требования и сравнительные возможности средств ПО СИМ

Требования к СИМ-модели (комфортный для ЛПР режим взаимодействия с ней; ввод и вывод информации в удобной форме; возможность модификации и интеграции с другими компонентами СУ и т. д.) во многом определяются требованиями, которые предъявляются к ПО процесса СИМ. Главными из них являются:

- гибкость в смысле возможности моделировать СС с разным уровнем сложности [3];
- наличие хороших средств отладки, которые позволяют отслеживать отдельные объекты по всей модели, чтобы убедиться в правильности их обработки и проверять состояние модели при каждом возникновении нового события;
- дружественный интерфейс и работа с графикой.

Важным также является наличие механизма генерирования независимых случайных численных величин (СЧВ) с исходным равномерным распределением на интервале [0, 1], чтобы, во-первых, с заданным качеством получать в дальнейшем все требуемые законы распределения СЧВ; во-вторых, для каждого прогона СИМ-модели применять одни и те же исходные условия, используя фиксированные наборы СЧВ и перед каждым прогоном приводя статистические счетчики в исходное состояние.

При выборе ПО необходимо решить, для какой платформы оно будет предназначено, поскольку большинство программных средств СИМ работают сегодня под управлением Windows и на рабочих станциях UNIX, но все большее распространение получает операционная система Mac. При возможности работы пакета на разных платформах ему необходимо обеспечить совместимость с другими платформами и уделить внимание тому, какие операционные системы он поддерживает. Для оценки результатов моделирования в ПО должна быть возможность создавать стандартные отчеты и сохранять отработанные сценарии СИМ в базе данных.

Первые версии ПО, которые с 1955 г. разрабатывались на языках типа FORTRAN [4], уже имели признаки СИМ-моделей: входные переменные в них случайным образом изменялись, а выходные подвергались статистической обработке. С тех пор были созданы методики и инструменты, призванные упростить и ускорить создание СИМ-моделей. В 60-е годы прошлого века появились специализированные языки GPSS, SIMSCRIPT, GASP и SIMULA, а в 1987 г. — графические среды, позволяющие сформировать структуру процесса СИМ путем комбинирования визуальных блоков. Сегодня на рынке присутствует более 50 программных продуктов для разработки СИМ-моделей различных объектов и процессов, которые можно разделить на три группы [5]:

- универсальные инженерные языки программирования;
- специализированные языки имитационного моделирования;
- среды имитационного моделирования.

Универсальные языки программирования BASIC, FORTRAN, C/C++, Pascal, Java и др. [5] являются традиционным средством реализации СИМ-моделей. В основе моделирующего алгоритма здесь лежит технология метода Монте-Карло (ММК), которая использует процедуру "разыгрывания" СЧВ, соответствующих случайным факторам, воздействующим на объект. Алгоритм представляет собой последовательность процедур, которая описывает состояния моделируемой СС в определенные дискретные моменты времени. Процесс СИМ по МДМ при этом заключается в многократном периодическом "вычислении" состояния СС и ее элементов. Достоинством данного подхода является его универсальность как возможность реализации модели СС любой сложности. Вместе с тем, учет особенностей реальных объектов (в том числе бизнес-процессов), а также обилие противоречивых условий их существования, приводят к тому, что разработка СИМ-модели на универсальном языке требует от ЛПР высокой квалификации и значительных временных затрат [4, 6].

Специализированные языки имитационного моделирования реализуют стандартные операции типа "создать объект", "передать объект", "ожидать в течение указанного промежутка времени", "сгенерировать СЧВ с указанным законом распределения", "выполнить операцию" и т. д., которые представляют собой блоки в виде последовательности команд на одном из универсальных языков [4]. Они ком-

пактны и имеют широкий круг приложений, различаются способами учета времени, сложностью изменения структуры модели и способами проведения экспериментов, но все требуют специальной подготовки ЛПР, который должен написать программу в терминах языка для конкретного объекта СИМ. Многие из них основаны на теории систем массового обслуживания (СМО), где поступающие заявки становятся в очередь на обслуживание агрегатами. Основной объект в подобных языках — это пассивный транзакт (заявка на обслуживание), который может представлять собой ресурсы (работников, деталей, сырья), документы, сигналы и т. п. "Перемещаясь" по СИМ-модели, транзакты становятся в очереди к одноканальным и многоканальным агрегатным устройствам, захватывают и освобождают их, расщепляются, уничтожаются и т. п. Данный подход в интересах СИМ поддерживают GPSS/PC, GPSS/H, GPSS World, Object GPSS, Arena, SimProcess, Enterprise Dynamics, AutoMod и др.

Среды имитационного моделирования, в отличие от языков, не требуют программирования в виде последовательности команд или предварительного синтеза алгоритма моделирования, что является их очевидным достоинством. Вместо написания программы ЛПР составляет модель из графических блоков или заполняет специальные бланки (формы) имитатора. Модель при этом составляется путем комбинирования элементов, входящих во встроенную библиотеку. Использование такого подхода позволяет повысить наглядность СИМ, хотя большинство сред являются лишь графическими интерпретациями соответствующих специализированных языков моделирования. В крупные имитационные системы входят средства автоматического анализа результатов и оптимизации процесса СИМ, а также управления проведением экспериментов.

Работа имитатора может быть основана как на независимом языке (среда Arena разработана на языке SIMAN, система MapSys — на языке MapSim), так и на собственном, разработанном специально для данной среды (среда EcosimPro основана на языке EL — EcosimPro Language). В объектно-ориентированных имитаторах поведение отдельных блоков задается посредством написания функции на одном из универсальных языков программирования (в среде AnyLogic используется язык Java, в ProcessModel — BASIC). Недостатком большинства проблемно-ориентированных имитаторов является сложность формализации бизнес-процесса и интерпретации результатов СИМ. Кроме того, в отличие от языков моделирования и программирования, из-за предопределенности элементов СИМ-модели область ее приложения получается существенно ограниченной. Помимо среды AnyLogic на рынке сегодня присутствуют среды ARIS Business Simulator, ReThink, Simulink.

В соответствии с изложенным оптимальный выбор ПО для проведения СИМ представляет собой непростую задачу, практическое решение которой ЛПР не всегда удается найти. Дорогостоящие графические среды не обладают гибкостью, необходимой для моделирования многих бизнес-процессов. В то же время они обеспечивают самую высокую скорость разработки СИМ-модели. Специализированные языки неудобны в использовании и уступают имитаторам по степени наглядности, однако обладают высокой степенью гибкости. Универсальные языки программирования сложны в изучении и требуют значительных временных затрат при разработке СИМ-моделей, зато они позволяют моделировать практически любые СС, создавать, менять и корректировать свойства объектов ПО (меню, элементов управления, окон и т. д.), что позволяет им в целом успешно конкурировать с имитаторами. Рассмотрим конкретные примеры программной реализации СИМ-моделей по МДМ с использованием универсальных языков и сред имитационного моделирования.

Программная реализация на языке Delphi СИМ-модели бизнес-процесса оплаты клиентами услуг телекоммуникационной компании

Сравнительные достоинства языка Delphi обусловлены следующими его особенностями.

1. Компилятор в машинный код со скоростью более 120 тыс. строк в минуту обеспечивает производительность, необходимую и достаточную для построения приложений в архитектуре клиент—сервер; предполагает легкость разработки и быстрое время проверки готового программного блока при требуемом высоком качестве кода. Delphi обеспечивает разработку без ручного написания кода (хотя это тоже возможно), когда ЛПР выбирает из палитры компонентов готовые элементы и еще до компиляции может видеть результаты своей работы, перемещаться по данным и представлять их в том или ином нужном ему виде. В этом смысле проектирование в Delphi мало отличается от проектирования в интерпретирующей среде, но после компиляции получается код, который исполняется в 10...20 раз быстрее, чем при исполнении с помощью интерпретатора.

2. В объектно-ориентированной модели программных компонентов Delphi основной упор делается на максимальном повторном использовании кода — это позволяет ЛПР быстро строить приложения из заранее подготовленных объектов, а также создавать собственные объекты. Наличие процедуры Randomize, входящей в пакет стандартных библиотек Delphi, позволяет инициализировать генератор СЧВ. Ограничений по типам создаваемых объектов не существует, разработка интерфейса в Delphi является приемлемо простой задачей для ЛПР.

3. Клиент-серверная версия Delphi позволяет реализовать высокопроизводительные приложения при работе с разными источниками информации, обеспечивает прозрачность подключения новых механизмов доступа к данным.

При использовании СИМ-модели бизнес-процесса оплаты клиентами услуг крупной региональной телекоммуникационной компании [1, 2] после запуска приложения ЛПР видит на экране ЭВМ окно безопасности, в поле которого следует ввести пароль доступа. Затем нажатием кнопки "Войти" окна или клавиши "Enter" стандартной клавиатуры он получает доступ к основному окну программы (рис. 1), которое содержит главное меню, панель управления функционалом программы и поля для ввода входных параметров СИМ-модели.

Исходные данные разделены по категориям, каждой из которых соответствует отдельная закладка окна, чтобы ЛПР было легче ориентироваться в многочисленных параметрах СИМ-модели. В левой части главного окна программы расположены кнопки, дублирующие команды главного меню. Панель разделена тремя областями, которые активируются (раскрываются) или сворачиваются, в зависимости от выполняемых действий. Так, например, на этапе ввода исходных данных ЛПР не нужны элементы управления результатами СИМ и отчетами, поскольку они еще не получены. Все кнопки панели управления СИМ-модели имеют соответствующие клавишные комбинации, информацию о которых, а также другие справочные данные можно получить, вызвав из главного меню справку по программе.

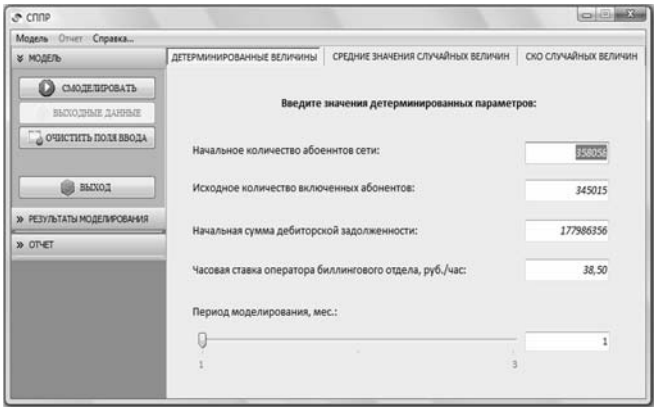
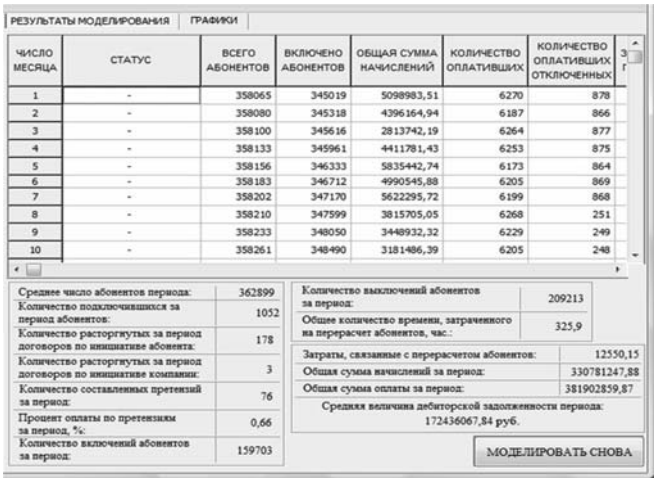


Рис. 1. Основное окно программы (СИМ-модели)



ЧИСЛО МЕСЯЦА	СТАТУС	ВСЕГО АБОНЕНТОВ	ВКЛЮЧЕНО АБОНЕНТОВ	ОБЩАЯ СУММА НАЧИСЛЕНИЙ	КОЛИЧЕСТВО ОПЛАТИВШИХ	КОЛИЧЕСТВО ОПЛАТИВШИХ ОТКЛЮЧЕННЫХ
1	-	358065	345019	5098983,51	6270	878
2	-	358080	345318	4396164,94	6187	866
3	-	358100	345616	2813742,19	6264	877
4	-	358133	345961	4411781,43	6253	875
5	-	358156	346333	5835442,74	6173	864
6	-	358183	346712	4990545,88	6205	869
7	-	358202	347170	5622295,72	6199	868
8	-	358210	347599	3815705,05	6268	251
9	-	358233	348050	3448932,32	6229	249
10	-	358261	348490	3181486,39	6205	248

Среднее число абонентов периода:	362899
Количество подключившихся за период абонентов:	1052
Количество расторгнутых за период договоров по инициативе абонента:	178
Количество расторгнутых за период договоров по инициативе компании:	3
Количество составленных претензий за период:	76
Процент оплаты по претензиям за период, %:	0,66
Количество включений абонентов за период:	159703

Количество выключений абонентов за период:	209213
Общее количество времени, затраченного на перерасчет абонентов, час.:	325,9
Затраты, связанные с перерасчетом абонентов:	12550,15
Общая сумма начислений за период:	330781247,88
Общая сумма оплаты за период:	381902859,87

Средняя величина дебиторской задолженности периода: 172436067,84 руб.

Рис. 2. Фрагмент окна результатов СИМ

Завершается моделирование появлением окна результатов СИМ (рис. 2), где в удобном для ЛПР виде отображаются выходные данные модели, которые он при необходимости (переключившись на вкладку "Графики" окна) может увидеть в графическом формате — как диаграммы. Предусмотрены печать таблицы выходных данных моделирования и графиков непосредственно из окна результатов, а также отправка результатов в виде отчета в приложение MS Excel, где можно просматривать и анализировать данные средствами этого табличного процессора. Цветовая гамма приложения выдержана в стандартных Windows-тонах, расположение меню и элементы управления соответствуют стандартам данной операционной системы, что способствует быстрому освоению программы ЛПР.

Программная реализация на языке Delphi СИМ-модели бизнес-процесса реанимирования нефтяных скважин

Данная СИМ-модель также является многооконным Windows-приложением, оснащенным главным меню, средствами управления окнами и средствами исследования данных. Импорт исходных данных осуществляется в одноименном окне, открытие которого реализуется выбором команды главного меню "Исходные данные → Импортировать" или нажатием комбинации клавиш Ctrl + I. Загруженные в модель данные (подробный перечень см. в работе [1]) имеют табличное представление и при необходимости могут быть отредактированы вручную или дополнены на выбор ЛПР, для чего предусмотрены соответствующие элементы управления. После того, как данные загружены и подготовлены, запуск СИМ реализуется командой "Моделирование → Запустить" главного меню или нажатием клавиш Ctrl + M.

По завершении СИМ в отдельном окне (аналогично рис. 2) представляется таблица с результатами реанимирования скважин, которая содержит выходные характеристики бизнес-процесса:

- наиболее эффективный вариант реанимирования;
- новое число качаний;
- наиболее подходящий вид промывочной жидкости;
- оптимальное время и вероятность успешности реанимирования для каждой скважины [1].

При нажатии клавишей мыши на соответствующую строку в таблице на экран выводится окно,

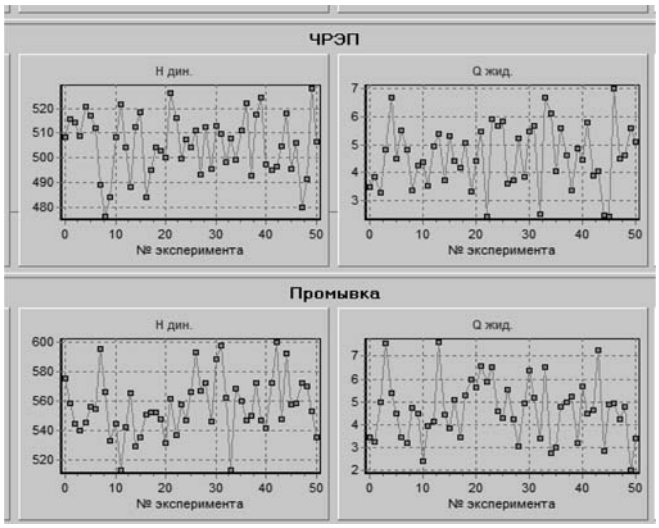


Рис. 3. Детализированные результаты СИМ (фрагмент вкладки "Графики")

содержащее детализированную информацию по заданной скважине: на вкладке "Расчеты" — статический и динамический уровень, дебит жидкости и нефти после реанимирования, предполагаемая стоимость смоделированных работ, общая стоимость работ и общее эффективное время мероприятий; на вкладке "Графики" — изменение моделируемых величин в форме линейных графиков (фрагмент данной вкладки в увеличенном виде см. на рис. 3).

Команда "План-графики → Сформировать" на основе данных СИМ позволяет автоматически формировать план-графики бригадам по реанимированию, содержащие: номер бригады, номер скважины, вариант реанимирования и его ожидаемые результаты (рис. 4), а также информацию об ответственном исполнителе, отметки о принятии к исполнению и, по завершении работ, дату исполнения — последние два поля автоматически обновляются при каждом открытии план-графика в СИМ-модели.

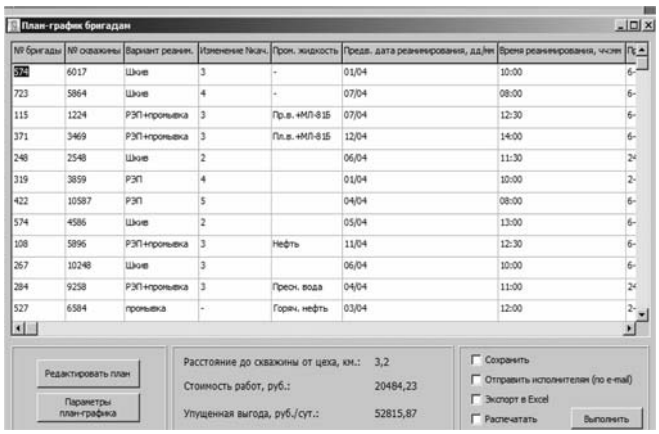


Рис. 4. Формируемый СИМ-моделью план-график бригадам по реанимированию нефтяных скважин

Сформированный план-график может быть сохранен, распечатан, отправлен исполнителям по электронной почте или экспортирован в MS Excel. Пользовательский интерфейс включает набор необходимых ЛПР элементов управления (меню, кнопки и т. д.), он прост и не избыточен, скрывает промежуточные вычисления и преобразования, реализован в стандартном Windows-оформлении и внешне соответствует интерфейсу корпоративной информационной системы, что делает его привычным и понятным специалисту-нефтянику.

Программная реализация в среде AnyLogic СИМ-модели бизнес-процесса "Выполнение заказа"

Среда AnyLogic, которая является одной из немногих отечественных разработок, получивших признание за рубежом [4], предназначена для графического создания СИМ-моделей с использованием объектно-ориентированного языка программирования Java. Она поддерживает три методики моделирования: системную динамику, дискретно-событийное и агентное моделирование, позволяя комбинировать эти подходы при разработке СИМ-модели. Разработчики AnyLogic непрерывно совершенствуют свой продукт, регулярно выпускают его новые версии и пополняют библиотеки готовых компонентов.

В рассматриваемом случае необходимо, во-первых, синтезировать структуру бизнес-процесса с учетом логики его функционирования, чтобы каждый блок бизнес-процесса, содержащий СЧВ, был отражен на схеме, показанной в верхней части рис. 5. Напомним, что каждый "прогон" СИМ-модели представляет собой последовательное моделирование всех СЧВ бизнес-процесса с учетом его логики в течение заданного периода моделирования [2; 7]. Показанная на рис. 5 схема выглядит следующим образом:

- *newClient* — начало бизнес-процесса, поступающие в СМО заявки (время между поступающими заявками — СЧВ);
- *queueToService* — очередь (существующая в общем случае) на оформление заказа. Задается ее вместимость, а также время (тайм-аут), по истечении которого не дождавшаяся обслуживания заявка покидает СМО {*rejectedClients*);
- *service* — этап оформления заказа, выполняемый отделом сбыта (длительность оформления заказа представляет собой СЧВ).

Элемент *rejectOrder* представляет возможный отказ клиента от заказа, это тоже СЧВ. "Отказавшиеся" клиенты накапливаются в блоке *rejectedClients.* Если заявка проходит дальше, она попадает в очередь на планирование производства (*queueToPlanning*), а затем — непосредственно в объект *planning* (длительность планирования производства, СЧВ).

Возможная нехватка запасов (СЧВ) реализуется с помощью элемента *noResources.* Если ресурсов

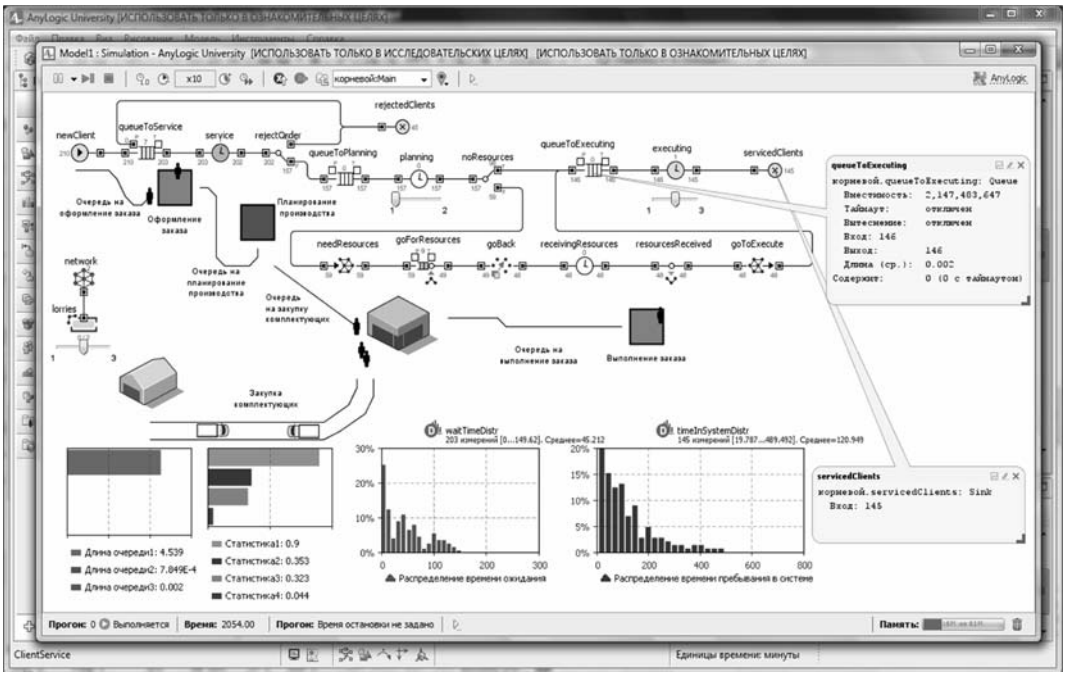


Рис. 5. Анимация СИМ-модели в среде AnyLogic

достаточно, заявка оказывается в очереди на выполнение заказа (*queueToExecuting*), после чего начинается само выполнение (СЧВ — длительность выполнения заказа). По завершении выполнения заказа обслуженные заявки накапливаются в блоке *servicedClients*. Если же ресурсов недостаточно, в блоке *receivingResources* моделируется их закупка (длительность и сумма закупки комплектующих СЧВ), после чего заявка поступает в очередь *queueToExecuting*.

Блоки *network*, *lorries*, *needResources*, *goForResources*, *goBack*, *resourcesReceived* и *goToExecute* используются для моделирования транспортной сети, т. е. процессов закупки комплектующих с помощью имеющихся транспортных средств. Во всех необходимых элементах предусмотрены механизмы генерации СЧВ по заданному закону распределения (можно задать тип закона и его параметры).

В СИМ-модели для оптимизации выделены следующие параметры: число специалистов отдела планирования; число специалистов и число оборудования производственного отдела; число транспортных средств. Под соответствующими блоками расположены "бегунки", с помощью которых можно в реальном времени изменять значения параметров от минимального до максимального. AnyLogic имеет широкие возможности для визуализации СИМ-моделей: в ПО присутствует достаточно большая библиотека изображений, кроме того, любые изображения можно нарисовать самому ЛПР, причем анимация возможна и в режиме 3D.

Анимацию построенной СИМ-модели иллюстрирует центральная часть рис. 5. С помощью ломаных линий здесь отображены четыре очереди: на оформление заказа, на планирование производства, на закупку комплектующих и на выполнение заказа. Заявки показаны как черные человеческие фигурки; квадраты — это блоки обслуживания заявок: отдел сбыта, плановый отдел, производственный отдел. Когда отдел простаивает, квадрат на рис. 5 имеет зеленый цвет, во время обслуживания — красный цвет. Наиболее важный отдел закупок изображен в более подробном виде: как грузовики, дорога, зоны погрузки и разгрузки. При работе СИМ-модели, когда возникает необходимость закупки комплектующих и при наличии свободных транспортных средств, одно из них отправляется за комплектующими — в левый нижний бокс на рис. 5. Если все грузовики заняты, заявка ожидает освобождения одного из них возле бокса в центре. "Играя" значениями трех параметров, можно наблюдать, как это влияет на очереди и скорость обслуживания — чтобы таким образом подбирать наилучшие (квазиоптимальные) параметры бизнес-процесса.

Нажав мышью на любой элемент СИМ-модели, можно вывести окно с параметрами и статистикой по нему (см. правую часть рис. 5). Также предусмотрена возможность сбора и отображения статистики работы модели в виде графиков и диаграмм (см. нижнюю часть рис. 5). Можно наблюдать, например, за длиной очереди, статистикой занятости любого обслуживающего устройства, распределением времени пребывания заявки в СМО и т. д. Для управления временем моделирования предусмотрена соответствующая панель — после запуска СИМ-модели можно остановить, ускорить или замедлить (для очень медленных или очень быстрых процессов) ее работу в целях более углубленного и

детального изучения объекта в удобном онлайн-режиме [7].

Построение СИМ-модели начинается путем создания ее в меню *Файл* → *Создать* → *Модель*, где задаются имя модели и место ее сохранения. Завершается создание модели нажатием кнопки "Далее" после выбора пункта меню "Начать с нуля". Активные объекты являются основными блоками иерархической модели AnyLogic, которая отображается в виде дерева в панели "Проекты": сама модель образует верхний уровень дерева; эксперименты, классы активных объектов и Java-классы образуют нижеследующий уровень; элементы, входящие в состав активных объектов, вложены в соответствующую подветвь дерева класса активного объекта и т. д.

Активный объект *Main* можно раскрыть в режиме редактора, чтобы начать построение модели с переноса из панели *Палитра* элементов библиотек. В основной библиотеке использованы следующие элементы:

- *Source* — создающий заявки;
- *Queue* — моделирующий очередь заявок;
- *Recourse Pool* — задающий набор ресурсов, которые могут захватываться и освобождаться заявками;
- *Service* — который захватывает для заявки заданное количество ресурсов, задерживает заявку, а затем освобождает захваченные ею ресурсы;
- *Delay* — задерживающий заявки на заданный промежуток времени;
- *Select Output* — направляющий входящие заявки в один из двух выходных портов в зависимости от выполнения заданного условия;
- *Select Output 5* — объект, который направляет входящие заявки в один из пяти выходных портов в зависимости от выполнения заданных (детерминистических или соответствующих заданным вероятностям) условий.

Каждому элементу из библиотеки необходимо задать его свойства в одноименном окне. Имеется возможность изменить название элемента, при необходимости выбрать его отображение на рабочем поле, задать закон распределения, фигуру анимации и т. д. Затем элементы соединяются между собой и проводится тестирование модели. Информация о выявленных ошибках содержится в окнах *Консоль* и *Ошибки*. После устранения ошибок и выполнения пробных прогонов ЛПР может приступить к последующим этапам СИМ.

Измерить время, проведенное заявкой между двумя точками схемы на рис. 5, можно с помощью элементов *Time Measure Start* и *Time Measure End*, которые необходимо установить после элемента *Source* и перед элементом *Sink*, чтобы собрать статистику времени пребывания заявки в процессе. Результаты СИМ из панели *Палитра* библиотеки *Статистика* выводятся на рабочее поле в виде гистограммы, которая в нашем случае (см. нижнюю часть рис. 5) показывает распределение времени обработки заявок. Далее в модель добавляются необходимые параметры и переменные: в данном случае параметрами являются число закупщиков, зарплата и себестоимость, как расчетные переменные заданы чистая прибыль и число выполненных заявок. В настройках также задается модельное время, в течение которого будет наблюдаться поведение СИМ-модели [7].

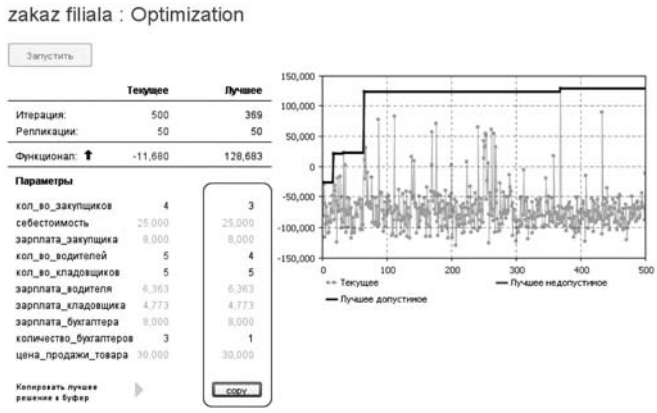


Рис. 6. К реализации оптимизационного СИМ-эксперимента в среде AnyLogic

Возможны два вида СИМ-эксперимента: простой и оптимизационный. В простом эксперименте задаются априори выбранные значения параметров, целью оптимизационного эксперимента является подбор наилучшего значения для выбранного параметра [2, 8]. В нашем случае модельное время составляет 10 080 мин (семь календарных дней), поскольку выполнение отдельных заявок может достигать нескольких суток. В результате работы СИМ-модели определяется оптимальное число сотрудников — закупщиков, кладовщиков, водителей, бухгалтеров для каждого филиала рассматриваемой торговой компании.

При простом эксперименте число сотрудников сразу появляется в окне с результатами СИМ. Целевой функцией при оптимизационном СИМ-эксперименте (рис. 6) является чистая прибыль филиала компании, которую нужно максимизировать. Для параметра "число закупщиков" имеется ограничение: от 1 до 7 чел., в очереди на обслуживание не должно скапливаться больше 10 заявок. По итогам СИМ оптимальное число закупщиков — 3, водителей — 4, кладовщиков — 5, бухгалтеров — 1.

Заключение

Степень соответствия требованиям и возможности рассмотренных средств ПО позволяют использовать их для реализации СИМ-моделей по МДМ в целях повышения эффективности работы СУ современными сложными системами организационно-технического типа [1, 2]. Для проведе-

ния СИМ в интересах управления бизнес-процессами выделены универсальный язык программирования Delphi и среда имитационного моделирования AnyLogic.

Высокая гибкость и скорость работы при относительно невысокой стоимости обеспечивает Delphi конкурентные преимущества перед другими языками программирования несмотря на сложность его изучения и отсутствие возможности анимации. Среду AnyLogic выгодно отличают наличие большого числа встроенных объектов и широкие возможности анимации моделируемых бизнес-процессов при приемлемых показателях скорости и гибкости, главным ее недостатком остается высокая стоимость. Представлены примеры использования данных средств ПО при разработке СИМ-моделей объектов в области телекоммуникаций, добычи природных энергоносителей и оптово-розничной торговли.

Авторы выражают признательность канд. техн. наук, доц. Р. Р. Халимову и С. В. Луковкину за участие в совместных исследованиях. Настоящая статья, к глубокому сожалению его постоянных соавторов, стала последней прижизненной работой канд. техн. наук, доц. Ю. В. Трошина.

Список литературы

1. **Димов Э. М., Маслов О. Н., Трошин Ю. В.** Снижение неопределенности выбора управленческих решений с помощью метода статистического имитационного моделирования // Информационные технологии. 2014. № 6. С. 51—57.

2. **Димов Э. М., Маслов О. Н., Пчеляков С. Н., Скворцов А. Б.** Новые информационные технологии: подготовка кадров и обучение персонала. Ч. 2. Имитационное моделирование и управление бизнес-процессами в инфокоммуникациях. Самара: Изд-во СНЦ РАН, 2008. 350 с.

3. **Лоу А. М., Кельтон В. Д.** Имитационное моделирование / Пер. с англ. СПб.: Питер; Киев: BHV, 2004. 847 с.

4. **Конюх В. Л., Зиновьев В. В., Игнатьев Я. Б.** Методы имитационного моделирования дискретных систем. Обзор программных продуктов. Кемерово: Изд. КемНЦ СО РАН. 2003. URL: http://www.gpss.ru/immod05/sl/konyuh/print.html (дата обращения 20.09. 2014).

5. **Димов Э. М., Луковкин С. В., Халимов Р. Р.** Анализ средств имитационного моделирования бизнес-процессов // Телекоммуникации. 2010. № 8. С. 43—48.

6. **Емельянов А. А., Власова Е. А., Дума Р. В.** Имитационное моделирование экономических процессов. М.: Финансы и статистика, 2002. 368 с.

7. **Димов Э. М., Маслов О. Н., Трошин Ю. В., Халимов Р. Р.** Динамика разработки имитационной модели бизнес-процесса // Инфокоммуникационные технологии. 2013. Т. 11, № 1. С. 63—78.

8. **Ануфриев Д. П., Димов Э. М., Маслов О. Н., Халимов Р. Р.** Сравнительная эффективность методов и средств информационной поддержки управленческих решений // Инфокоммуникационные технологии. 2014. Т. 12, № 1. С. 54—67.

Ad. M. Dimov, Professor, **O. N. Maslov,** Head of Chair, e-mail: maslov@psati.ru,
Yu. V. Troshin, Associate Professor
Povolzhskiy State University of Telecommunications and Informatics

Statistical Simulation Modeling Process Software Choice

The article considers the problem of statistical simulation modeling (SSM) software choice for complex organizational-technical systems. The examples of provided SSM-models applying universal language Delphi and simulation environment AnyLogic are presented. The SSM business process model of paying customers service of Telecommunications Company is the first example of a programming language Delphi. The second example — is SSM business process model of resuscitating (repairing) of oil wells. The "Order Fulfillment" business process of trading company modeled in the AnyLogic. Presented SSM models are destined for practical application by managers of telecommunication, oil-producing and trading business.

Keywords: *statistical simulation modeling method, software, universal language Delphi, simulation environment AnyLogic*

References

1. **Dimov Eh. M., Maslov O. N., Troshin Yu. V.** Snizhenie neopredelennosti vybora upravlencheskikh reshenij s pomoshh'yu metoda statisticheskogo imitatsionnogo modelirovaniya. *Informatsionnye tekhnologii*. 2014. N. 6. P. 51—57.

2. **Dimov Eh. M., Maslov O. N., Pchelyakov S. N., Skvortsov A. B.** *Novye informatsionnye tekhnologii: podgotovka kadrov i obuchenie personala*. Ch. 2. Imitatsionnoe modelirovanie i upravlenie biznes-protsessami v infokommunikatsiyakh. Samara: Izd-vo SNTS RAN, 2008. 350 p.

3. **Lou A. M., Kel'ton V. D.** *Imitatsionnoe modelirovanie*: Per. s angl. SPb.: Piter; Kiev: BHV, 2004. 847 p.

4. **Konyukh V. L., Zinov'ev V. V., Ignat'ev YA. B.** *Metody imitatsionnogo modelirovaniya diskretnykh sistem. Obzor programmnykh produktov*. Kemerovo: Izd. KemNTS SO RAN. 2003. URL: http://www.gpss.ru/immod05/sl/konyuh/print.html (data obrashheniya 20.09. 2014).

5. **Dimov Eh. M., Lukovkin S. V., Khalimov R. R.** Analiz sredstv imitatsionnogo modelirovaniya biznes-protsessov. *Telekommunikatsii*. 2010. N. 8. P. 43—48.

6. **Emel'yanov A. A., Vlasova E. A., Duma R. V.** *Imitatsionnoe modelirovanie ekonomicheskikh protsessov*. M.: Finansy i statistika, 2002. 368 p.

7. **Dimov Eh. M., Maslov O. N., Troshin Yu. V., Khalimov R. R.** Dinamika razrabotki imitatsionnoj modeli biznes-protsessa. *Infokommunikatsionnye tekhnologii*. 2013. V. 11, N. 1. P. 63—78.

8. **Anufriev D. P., Dimov Eh. M., Maslov O. N., Khalimov R. R.** Sravnitel'naya ehffektivnost' metodov i sredstv informatsionnoj podderzhki upravlencheskikh reshenij. *Infokommunikatsionnye tekhnologii*. 2014. V. 12, N. 1. P. 54—67.

УДК 621.391

С. В. Дворников[1], д-р техн. наук, проф., профессор кафедры,
С. С. Манаенко[1], канд. техн. наук, ст. препод., **С. С. Дворников**[2], бакалавр, студент,
А. А. Погорелов[1], канд. техн. наук, доц., нач. кафедры
[1] Военная академия связи, г. Санкт-Петербург
[2] Санкт-Петербургский государственный политехнический университет
e-mail: practicdsv@yandex.ru

Синтез фазоманипулированных вейвлет-сигналов

Представляются материалы исследования по вопросам синтеза фазоманипулированных сигналов на основе фрагментов, в качестве которых выступает вейвлет Гаусса первого порядка. Обосновываются энергетические параметры модулирующих фрагментов и анализируется помехоустойчивость вейвлет-сигналов.

Ключевые слова: *синтез сигналов, параметрическая скрытность, вейвлет Гаусса первого порядка, демодуляция фазоманипулированных сигналов*

Введение

Вопросам повышения помехоустойчивости всегда уделялось приоритетное внимание при разработке средств радиосвязи. Прежде всего это обусловлено необходимостью достоверной передачи информации в условиях шумов и помех различной природы.

Среди широко известных модуляционных форматов наиболее помехоустойчивым является двухпозиционная фазовая манипуляция [1], активно используемая на радиолиниях КВ—УКВ диапазонов.

Последние достижения в области кратномасштабного анализа позволяют предположить, что в настоящее время синтез радиоизлучений на основе вейвлет-функций (в дальнейшем вейвлетов) позволит получить модуляционные форматы, обладающие свойствами помехоустойчивости не хуже, чем у фазоманипулированных конструкций.

Кроме того необходимо учитывать, что обработка вейвлет-сигналов в базисах гармонических функций будет связана с определенными техническими сложностями, обусловленными их структурной несовместимостью.

Указанные обстоятельства позволят снизить вероятность несанкционированного доступа, основанного на применении методов гармонического анализа. Следовательно, даже на данном этапе можно утверждать, что вейвлет-сигналы обеспечат их пользователям дополнительную структурную скрытность [2] по отношению к системам радиомониторинга, использующим другие функциональные базисы обработки радиоизлучений.

В связи с этим в статье рассматриваются вопросы синтеза модуляционных форматов на основе вейвлетов семейства функций Гаусса первого порядка и исследуется их помехоустойчивость в канале с аддитивным белым гауссовым шумом.

Постановка задачи

Вопросы повышения помехоустойчивости за счет использования базисов формирования сигналов, отличных от гармонических, рассматривались в работе [3], где предлагалось осуществлять синтез сигналов в базисах функций сплайн-Виленкина — Крестенсона, которые являются обобщающими по отношению к базису функций Фурье.

Один из результатов работы заключался в следующем. Авторам удалось установить, что сигнал, сформированный в базисе указанных функций, будет иметь все внешние признаки модулированного колебания, синтезированного в гармоническом базисе. Но его анализ на основе процедур преобразования Фурье не позволит получить значение истинных параметров и, в итоге, правильно осуществить демодуляцию (извлечь вложенную информацию).

Рассмотрим, насколько указанные выводы применимы к вейвлет-модуляции.

Предложения по синтезу вейвлет-сигналов

В качестве исходного модуляционного формата определим двухпозиционную фазовую манипуля-

цию (ФМ-2). Здесь и далее под модуляционным форматом будем понимать совокупность вида модуляции (манипуляции) и скорости передачи сигнала.

В формате ФМ-2 скорость передачи определяется минимальной длительностью сигнала τ_c, в пределах которой его фаза остается постоянной. Указанный интервал в работе [1] определен как элемент сигнала ФМ-2. Сама же процедура синтеза заключается в следующем. Формируется несущее колебание определенной частоты, у которого в соответствии с заданной скоростью манипуляции происходит изменение фазы согласно исследуемой информационной последовательности логических нулей и единиц.

Учитывая, что модулированный сигнал ФМ-2 представляет собой совокупность повторяющихся элементов, соответствующих логическим нулям и единицам, предлагается следующий подход к его синтезу.

На первом этапе формируются элементы сигнала $s1(t)$ и $s0(t)$, соответствующие логическим нулю и единице (рис. 1), причем длительности элементов τ_c выбираются таким образом, чтобы соответствовать требуемой скорости манипуляции.

На втором этапе, согласно информационной последовательности, из сформированных элементов сигнала $s1(t)$ и $s0(t)$ синтезируется требуемая модуляционная конструкция. В результате получаем сигнал ФМ-2, аналогичный по структуре сигналу, синтезируемому по методу, описанному в работе [1].

Предложенный подход открывает возможности по синтезу сигналов ФМ-2 на основе импульсных фрагментов, в том числе вейвлетов.

Действительно, ведь в качестве элементов сигнала $s1(t)$ и $s0(t)$ можно определить и фрагменты локализованных функций, в частности, так называемого вейвлета Гаусса первого порядка $\psi(t)$, представляющего первую производную от функции Гаусса [4].

Аналитически вейвлет Гаусса первого порядка описывается следующим выражением:

$$\psi(t) = -t\exp(-t^2/2). \qquad (1)$$

Анализ выражения (1) указывает на локализованный характер функции, т. е. синтез непрерывного колебания на его основе в принципе невозможен. Однако вейвлет Гаусса вполне может быть использован в качестве фрагмента для синтеза модулированного колебания в соответствии с предложенным подходом. На рис. 2 представлены элементы сигнала $\psi1(t)$ и $\psi0(t)$, сформированные на основе вейвлета Гаусса.

Следует отметить, что свойства локализации вейвлета и его двусторонняя временна́я структура позволяют на его основе формировать колебания, которые также можно рассматривать как сигналы ФМ-2.

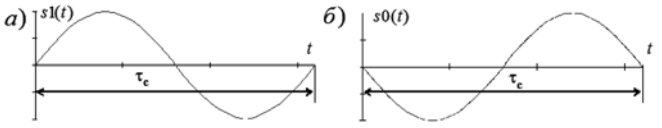


Рис. 1. Элементы сигнала ФМ-2:
а — логической единицы; *б* — логического нуля

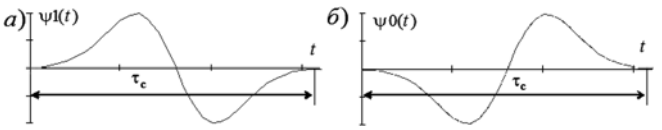


Рис. 2. Элементы вейвлет-сигнала:
а — логической единицы; *б* — логического нуля

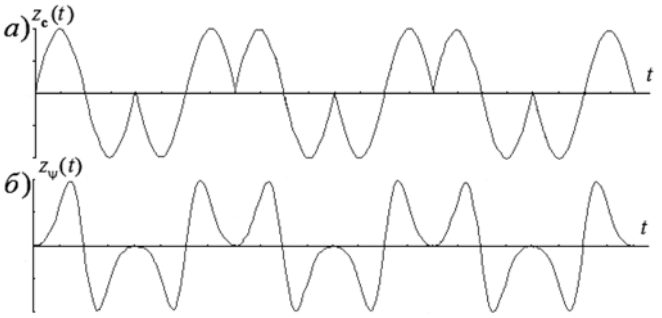


Рис. 3. Сигнал ФМ-2, синтезированный на основе фрагментов:
а — синусоид; *б* — вейвлетов

В частности, на рис. 3 показан фрагмент сигнала ФМ-2, сформированный на основе элементов синусоид и на основе вейвлетов.

Результаты эксперимента по анализу помехоустойчивости вейвлет-сигналов

Для оценки помехоустойчивости предложенной ФМ-2-вейвлет-конструкции был определен канал с аддитивным белым гауссовым шумом. При эксперименте учитывались следующие обстоятельства. В качестве исходной была определена мощность фрагмента сигнала на основе синусоиды, формируемого квадратурным методом из синфазной и квадратурной составляющих единичной амплитуды. В этом случае амплитуда временной развертки синусоиды составила $\sqrt{2}$.

Тогда, для обеспечения аналогичной мощности для фрагмента на основе вейвлета, его амплитуда была повышена до уровня 2,84.

На рис. 4 показан график зависимости вероятности ошибки на бит P_b от значения отношения мощности сигнала к спектральной плотности мощности шума (ОСШ) h^2 для сигналов ФМ-2 на основе фрагментов синусоид и вейвлетов, синтезированных в соответствии с предложенным подходом к формированию фазоманипулированных колебаний.

На рис. 4 кривая *1* — это потенциальная помехоустойчивость для сигнала ФМ-2 в гармоническом базисе, а кривая *2* — помехоустойчивость для вейвлет-сигнала, обрабатываемого в базисе вейвлет-

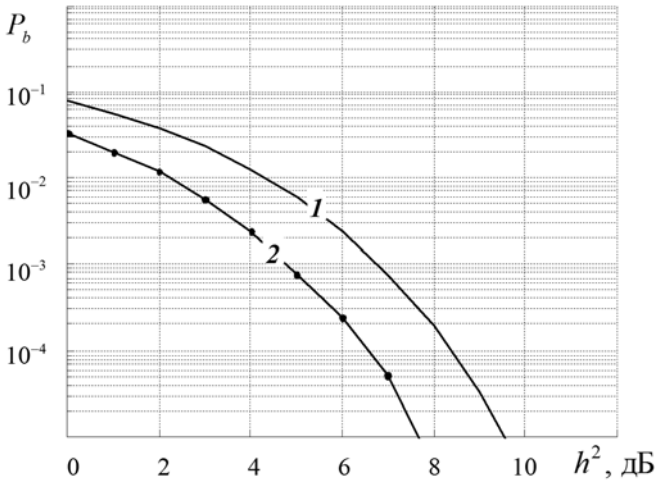


Рис. 4. Зависимость вероятности битовой ошибки от ОСШ при обработке вейвлет-сигнала в базисе вейвлет-функций

функций. Следует заметить, что указанную помехоустойчивость вейвлет-сигналы обеспечат только в том случае, когда на приемном конце корреляционная обработка будет осуществляться в базе вейвлет-функций (т. е. будут обеспечены условия когерентной обработки).

Совсем другие условия помехоустойчивости будут при обработке вейвлет-сигналов в базисах гармонических функций, т. е. при их несанкционированном приеме. На рис. 5 представлен график зависимости битовой вероятности ошибки от значения ОСШ при обработке вейвлет-сигнала (кривая *2*) и сигнала ФМ-2 (кривая *1*) в гармоническом базисе.

На рис. 5 кривая *1* — это потенциальная помехоустойчивость для сигнала ФМ-2 в гармоническом базисе, а кривая *2* — помехоустойчивость для вейвлет-сигнала, обрабатываемого в гармоническом базисе.

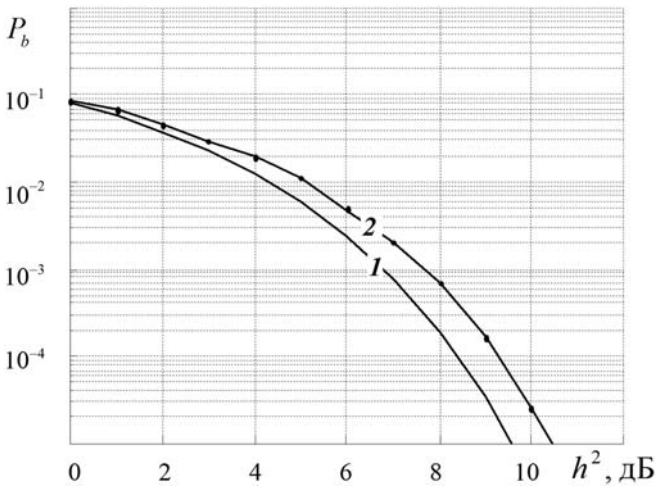


Рис. 5. Зависимость вероятности битовой ошибки от ОСШ при обработке сигналов в гармоническом базисе

Анализ полученных результатов позволяет сделать следующее заключение. При обеспечении требуемой битовой вероятности ошибки $P_b = 10^{-5}$ выигрыш в помехоустойчивости от когерентной обработки в базисе вейвлет-функций составит более 3 дБ.

Моделирование проводили в среде MATLAB. Графики строились из условия появления 100 ошибочных решений. Значение вероятности ошибки на бит рассчитывалось в соответствии с требованиями [1] по формуле

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right), \qquad (2)$$

где E_b — энергия, приходящаяся на бит; N_0 — спектральная плотность мощности шума.

В формуле (2) $Q(x) = \frac{1}{2\pi}\int_x^\infty \exp(-t^2/2)dt$ — функция плотности распределения вероятности (площадь под кривой интеграла вероятности).

Заключение

Полученные результаты позволяют сделать следующее заключение. Синтез сигналов на основе фрагментов вейвлетов позволяет получить модулированные колебания, помехоустойчивость которых на 2 дБ превосходит потенциально возможные показатели для сигналов ФМ-2, сформированных в базисе гармонических функций.

Если в качестве модулирующих фрагментов рассматривать вейвлеты Гаусса первого порядка, то выигрыш в обеспечении дополнительной структурной скрытности составит порядка 1 дБ. Однако можно предположить, что использование более сложных вейвлет-конструкций позволит увеличить указанный выигрыш. Авторы также предполагают, что указанный эффект может быть усилен при увеличении числа различных используемых вейвлетов, применяемых для синтеза сигналов, поскольку в этом случае возрастает сложность обработки таких конструкций, обусловливаемая отсутствием априорной информации о базисах их формирования.

Дальнейшие исследования видятся в разработке эффективных методов демодуляции вейвлет-сигналов.

Список литературы

1. **Прокис Дж.** Цифровая связь: Пер. с англ. / Под ред. Д. Д. Кловского. М.: Радио и связь. 2000. 800 с.

2. **Каневский З. М., Литвиненко В. П.** Теория скрытности. Воронеж: ВГУ, 1991. 144 с.

3. **Агиевич С. Н., Дворников С. В., Гусельников А. С.** и др. Описание сигналов в базисах функций сплайн-Виленкина — Крестенсона // Контроль—Диагностика. 2009. № 3. С. 52—57.

4. **Яковлев А. Н.** Введение в вейвлет-преобразования: учеб. пособие. Новосибирск: Изд-во НГТУ, 2003. 104 с.

S. V. Dvornikov[1], Professor, **S. S. Manaenko**[1], Associate Professor,
S. V. Dvornikov[2], Student, **A. A. Pogorelov**[1], Head of the Department
[1] Military Communications Academy, St. Petersburg
[2] St. Petersburg State Polytechnical University
e-mail: practicdsv@yandex.ru

Synthesis PSK Wavelet-Signal

On improving immunity has always been given priority in the development of radio communication. Among the well-known modulation formats is the most error-correcting binary phase shift keying is used extensively to radio HF VHF bands. Advances in the multiresolution analysis suggest that the synthesis signal on the basis of wavelets allow to obtain modulation formats with high noise immunity properties. It can be argued that the wavelet signals provide additional structural secrecy in relation to systems analysis using other functional bases of processing radio signals.

In this regard, the article discusses the synthesis of phase-shift keyed signals based on wavelet and study their immunity in an additive white Gaussian noise.

The very same synthesis procedure is as follows. Forming a carrier wave of a certain frequency, which in accordance with a predetermined manipulation speed is changed according to the value of the phase information sequence of logical ones and zeros. As the chip is proposed to use fragments of first-order Gaussian wavelet that represents the first derivative of the Gaussian function.

Localization properties of the wavelet in time and bilateral temporal structure allow it to form the basis of fluctuations, which can also be regarded as a phase-shift keyed signals.

To estimate the noise immunity of the proposed PSK-2 wavelet design was defined channel with AWGN. As the initial capacity of the fragment was determined based on the sine wave signal generated by the quadrature. Amplitude sine wave was timebase $\sqrt{2}$. To provide similar facilities for fragment-based wavelet, its amplitude was increased to a level of 2,84. High immunity to ensuring wavelet signals only when the receiving end of the correlation processing is performed in the wavelet base functions.

Synthesis of signals based on the fragments of wavelets allows to obtain the modulated vibration immunity which exceeds 2 dB potentially possible indicators for PSK-2 signals. If considered as modulating fragments of first-order Gaussian wavelets, the win in order to provide additional structural secrecy of the order of 1 dB.

Further studies are seen in the development of effective methods of wavelet signal demodulation.

Keywords: *synthesis signals, parametric stealth, Gauss wavelet first order demodulation of PSK signals*

References

1. **Prokis Dzh.** *Cifrovaja svjaz':* Per. s angl. Pod red. D. D. Klovskogo. M.: Radio i svjaz'. 2000. 800 p.

2. **Kanevskij Z. M., Litvinenko V. P.** *Teorija skrytnosti*. Voronezh.: VGU, 1991. 144 p.

3. **Agievich S. N., Dvornikov S. V., Guselnikov A. S.** i dr. Opisanie signalov v bazisah funkcij splajn-Vilenkina—Krestensona. *Kontrol'-Diagnostika*. 2009. N. 3. P. 52—57.

4. **Jakovlev A. N.** *Vvedenie v vejvlet-preobrazovanija:* ucheb. posobie. Novosibirsk: Izdatel'stvo NGTU, 2003. 104 p.

ЖУРНАЛ В ЖУРНАЛЕ

НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ

№ 2
ФЕВРАЛЬ
2015

ПОЗДРАВЛЯЕМ ЮБИЛЯРА!

А. И. Галушкин

Доктору технических наук, профессору, Заслуженному деятелю науки России, лауреату Премии Правительства России Александру Ивановичу Галушкину исполнилось 75 лет. Александр Иванович Галушкин закончил Московское высшее техническое училище им. Баумана по кафедре "Системы автоматического управления" в 1963 г. и в том же году был направлен в очную аспирантуру. В 1966 г. защитил диссертацию на соискание ученой степени кандидата технических наук, в 1969 г. получил ученое звание доцента. В 1974 г. на Ученом совете Вычислительного центра АН СССР защитил докторскую диссертацию.

А. И. Галушкин является автором более 400 научных работ, в том числе 25 монографий. В 1988 году получил звание профессора, в 1996 г. — звание Заслуженного деятеля науки Российской Федерации.

В период с 1965 по 1972 гг. А. И. Галушкин разработал общую методику синтеза многослойных нейронных сетей как элементов адаптивных систем управления. Основные научные результаты этого периода опубликованы в монографиях "Многослойные системы распознавания образов" (изд-во МИЭМ, 1970 г.) и "Синтез многослойных систем распознавания образов" (изд-во "Энергия", 1974 г.), где им впервые были предложены, описаны и изучены алгоритмы обучения многослойных нейронных сетей. В это же время были реализованы образцы систем, которые нашли применение при решении конкретных задач распознавания объектов по характеристикам сигналов.

Работы А. И. Галушкина по теории нейронных сетей являются общепризнанными в России и мире. Они представлены в монографиях: "Теория нейронных сетей" (М.: ИПРЖР, 2000 г.), "Neural network theory" (Springer, 2007), "Нейронные сети: основы теории" (М.: "Горячая линия", 2010 г.).

Предисловия к двум последним монографиям написали известные ученые с мировыми именами: Роберт Хехт-Нильсен — ведущий разработчик нейрокомпьютеров в США, Лотфи Заде — автор концепции нечеткой логики, Шун-иши Амари — директор Института исследований мозга в Японии.

А. И. Галушкин ведет активную научно-педагогическую деятельность, которую начал в 1968 г. в Московском Институте электронного машиностроения и сейчас продолжает в Московском физико-техническом институте на кафедре "Интеллектуальные информационные технологии и системы". Он является главным редактором журнала "Нейросетевые технологии", входящего в журнал "Информационные технологии".

В настоящее время А. И. Галушкин является начальником лаборатории интеллектуальных информационных систем Центра информационных технологий и систем исполнительных органов власти (ФГАНУ ЦИТиС).

Практически вся научная деятельность проф. А. И. Галушкина с 60-х годов прошлого века по настоящее время посвящена созданию высокоэффективных для текущего уровня развития технологии микроэлектроники программно-аппаратных эмуляторов нейронных сетей и применению их для решения конкретных задач.

Редакция журнала, друзья и коллеги сердечно поздравляют Александра Ивановича с 75-летием, желают крепкого здоровья и новых творческих успехов!

УДК 004.032.26

А. И. Галушкин, д-р техн наук, проф., зам. зав. каф.,
Московский физико-технический институт, г. Долгопрудный, e-mail: neurocomputer@yandex.ru

Мемристоры в развитии высокопроизводительной вычислительной техники

Излагается мнение автора о перспективах развития данного направления сверхпроизводительной вычислительной техники в связи с появлением мемристоров.

Ключевые слова: *мемристоры, суперЭВМ, аналоговые вычисления, многослойные нейронные сети, клеточные нейронные сети*

Введение

Ниже представлена концепция разработки нейрокомпьютеров с применением мемристоров, ориентированная на большое и очень большое число вычислительных элементов и, как следствие, на большое число мемристоров.

На рис. 1 условно представлена сфера применения мемристорных систем, сформировавшаяся за последние несколько лет. Несомненно, эта сфера будет дополняться и расширяться. В данной работе изложены основные принципы применения мемристоров при создании и применении нейрокомпьютеров, как раздела сверхвысокопроизводительной вычислительной техники.

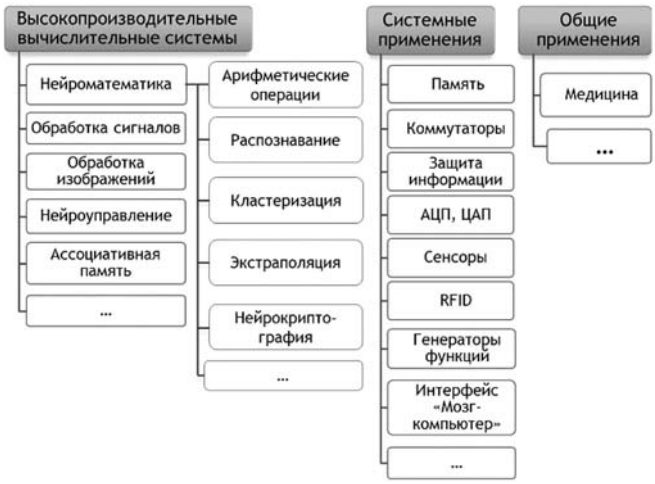


Рис. 1. Функциональные задачи для мемристорных систем (уровень 2014 г.)

1. Предпосылки: архитектура вычислительных систем

В настоящее время высказываются серьезные сомнения в перспективности КМОП-технологии и других технологических решений для решения задачи перехода от петафлопного к экзафлопному уровню производительности вычислительных систем. В работе [1] представлено мнение двух известных экспертов: Харста Саймона и Томаса Стерлинга о том, что существуют серьезные сомнения в достижении к 2020 г. производительности экзафлопного уровня (10^{18}). Основные сомнения касаются энергопотребления, которое для 10-петафлопной системы составляет 10 МВт, а для экзафлопной системы по расчетам должно достигать 2 ГВт. В этой же работе отмечается необходимость рассмотрения кроме КМОП-технологий и других технологий, а также других типов архитектур, отличных от фон-Неймановской.

В настоящее время во всем мире реализуется несколько десятков подобных проектов. Одним из характерных проектов является проект, представленный в работе [2].

В России разработана концепция создания вычислительных систем экзафлопного уровня с использованием классических принципов построения кластерных суперЭВМ, а также концепция развития высокопроизводительных вычислений на базе супернейрокомпьютеров [3].

Необходимо отметить два основных свойства современных суперЭВМ:

- масштабируемость;
- двухслойность архитектур.

С нашей точки зрения для перехода на экзафлопный уровень вычислений необходима реализация по крайней мере еще одного шага: изменения логического базиса алгоритмов решения задач и, соответственно, элементной базы с переходом от носителя информации в виде уровней токов и напряжений в электрических схемах к носителю информации в виде частоты узких импульсов, подобно тому, как это имеет место в реальной нервной системе.

Основными предпосылками принципиального изменения архитектуры вычислительных систем при переходе к системам экзафлопной производительности являются следующие:

- необходимость резкого повышения надежности за счет отказа от фон-Неймановской архитектуры вычислительных систем из элементов булевой логики И, ИЛИ, НЕ, когда в системе из значительного числа этих элементов происходит катастрофический отказ при отказе любого элемента;
- необходимость резкого снижения энергопотребления за счет отказа от существующего позиционнного метода представления информации и перехода к другим, обеспечивающим снижение энергопотребления, и как следствие, дополнительное повышение надежности;
- необходимость резкого повышения однородности схемотехники элементов вычислительных систем, что должно привести к повышению эффективности при том резком увеличении интеграции элементов, которое будет иметь место при переходе к экзафлопной производительности.

На рис. 2 отмечены основные этапы развития сверхвысокопроизводительной вычислительной техники за последние десятилетия. Отмечается, что мемристоры являются одним из эффективных путей развития данного вида техники на ближайшее десятилетие.

На рис. 3 условно представлена динамика развития технологий нейрокомпьютеров, начиная с 50-х годов прошлого столетия. В 50-е и 60-е годы прошлого столетия для изготовления нейрокомпьютеров в основном использовали аналоговую технологию при незначительной доле программной реализации на универсальных ЭВМ. С появлением микропроцессоров, а также мощных универсальных цифровых ЭВМ, аналоговые ЭВМ вследствие их ограниченной точности, не-

Время	Производительность	Технология
1990	1 Гигафлоп/с	Транспьютерные системы
1999	1 Терафлоп/с	Кластерные суперЭВМ
2009	1 Петафлоп/с	Графические процессоры
2020	1 Экзафлоп/с	Мемристоры

Рис. 2. Этапы развития сверхвысокопроизводительной вычислительной техники за последние десятилетия

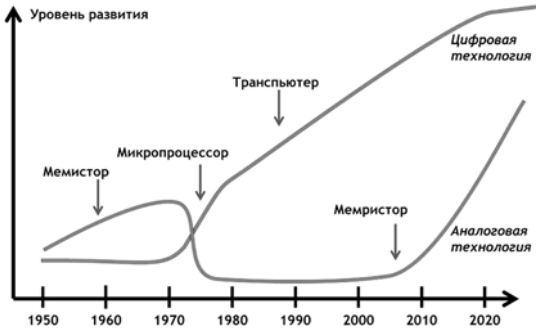


Рис. 3. Развитие технологий нейрокомпьютеров. Мемристоры — детище нанотехнологий

смотря на большое быстродействие, теряют свою роль, оставаясь в период 80-х, 90-х и даже 2000-х годов предметом разработки достаточно большого числа специализированных нейрочипов, наряду с большим числом разработок специализированных цифровых нейрочипов.

Разработка мемристоров возрождает аналоговую обработку, резко понижая энергопотребление, увеличивая скорость обработки при ее контролируемой точности, характерной для высокопараллельных нейросетевых структур с ограниченным числом слоев логической обработки.

2. Предпосылки: теория нейронных сетей, нейроматематика и нейроуправление

Российская школа теории нейронных сетей зародилась в середине 60-х годов прошлого столетия [4, 5]. С самого начала этих работ применительно к адаптивным системам автоматического управления нейронная сеть как объект управления была выбрана в структуре, показанной на рис. 4. Нейронная сеть представляет собой высокопараллельную многослойную структуру с настраиваемыми последовательными, перекрестными и обратными связями.

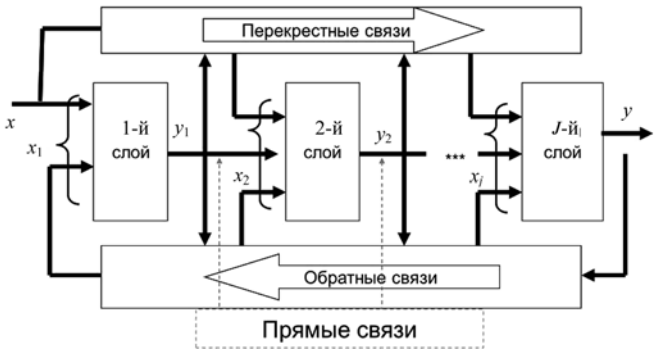


Рис. 4. Многослойная нейронная сеть (концепция 1967 г.)

Ниже представлена основная аксиоматика теории нейронных сетей, характерная для российской научной школы:

- Вероятностная модель мира.
- Нейронные сети — логический базис решения любых задач.
- Разработка нейросетевых алгоритмов, специфичных для конкретной выбранной задачи.
- Отказ от использования субъективных нейросетевых парадигм.
- Нейронная сеть — частный вид многомерного нелинейного динамического объекта управления.
- Любая идея нейросетевого алгоритма должна быть ориентирована на эффективную аппаратную реализацию в соответствие с текущей или перспективной технологией.
- Любая идея в части нейросетевых технологий не эффективна, если не может быть перенесена из одно- или двумерной иллюстрации на многомерную иллюстрацию.

Работы в области нейронных сетей получили развитие в работах [6—8].

В 2007 г. на международной конференции по нейронным сетям (IJCNN — 07, Orlando, USA) была организована российская секция "Overview of Soviet/Russian Neural Network R & D: The Untold Story" под руководством Роберта Хехт-Нильсена. Кроме ряда российских докладов была представлена монография [9] с предисловиями Лотфи Заде, Амари и Р. Хехт-Нильсона, которые отметили высокий уровень российских научных работ в этой области. Работа [10] является итоговой в этой области.

Нейроматематика — новый раздел вычислительной математики, связанный с разработкой неиросетевых алгоритмов решения сложных формализуемых и неформализуемых задач. Основные результаты российской научной школы в области нейроматематики представлены в работах [11—13].

Нейроуправление — это новый раздел теории управления, связанный с применением нейрокомпьютеров для идентификации и управления динамическими системами. Нейроуправление является направлением теории управления различными сложными системами:

- сильно нелинейными;
- с изменяемыми параметрами;
- с изменяемой структурой;
- многомерными;
- распределенными.

Работы российской научной школы представлены в работах [14, 15] и более десяти монографиях по прикладным задачам нейроуправления различными объектами.

3. Предпосылки: реализации нейрокомпьютеров

На рис. 5 представлены этапы развития реализаций нейрокомпьютеров в России.

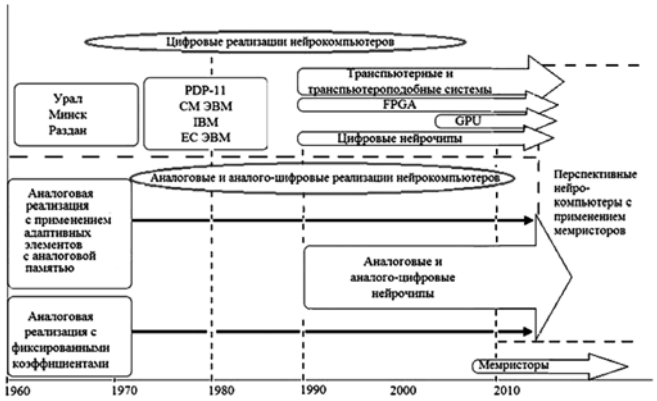


Рис. 5. Развитие реализаций нейрокомпьютеров в России

Реализация аналоговых нейрокомпьютеров с использованием адаптивных элементов с аналоговой памятью в 60-е годы прошлого века оказалась слишком громоздкой и дорогостоящей. Кроме этого существенным недостатком подобных реализаций нейрокомпьютеров являлось достаточно большое время перестройки коэффициентов нейронных сетей.

На рубеже 60-х и 70-х годов прошлого столетия для применения аналоговых нейрокомпьютеров был выбран класс задач, для которых сбор архива машинных данных для обучения был достаточно длительным и дорогостоящим. При этом необходимость в перестройке коэффициентов нейронных сетей либо отсутствовала, либо была необходима через значительный период времени (1 месяц, 1 год и т. д.). Для этих целей были реализованы аналоговые нейрокомпьютеры с коэффициентами, проставляемыми вручную (рис. 6, 7).

Период 70-х, 80-х годов прошлого столетия был периодом программной реализации нейрокомпьютеров на микропроцессорах и универсальных ЭВМ различного типа (PDP-11, IBM, их общегражданских и специальных отказоустойчивых аналогов).

Начало 90-х годов прошлого века характеризовалось сразу несколькими направлениями разработок нейрокомпьютеров, связанных с развитием технологии микроэлектроники:

- программные реализации нейрокомпьютеров на базе транспьютерных систем (десятки, сотни процессоров с сопроцессорами обработки сигналов и изображений);
- нейрокомпьютеры на базе ПЛИС (программируемых логических интегральных схем);
- цифровые, цифроаналоговые и аналоговые нейрочипы.

Эти разработки в значительной степени отражены в монографии [18].

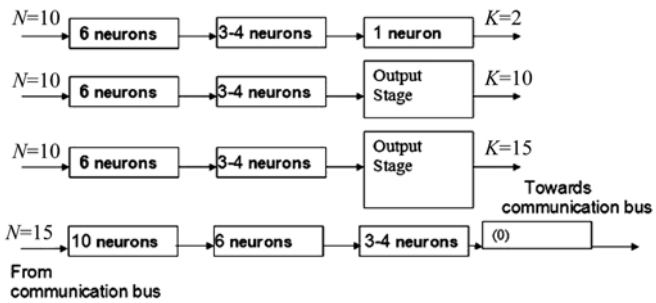


Рис. 6. Функциональная структура нейрокомпьютера (1970 г.)

Рис. 7. Общий вид нейрокомпьютера (1974 г.). Размерность входных признаков — 30, число классов 8

Период 2005—2007 гг. характеризуется активным становлением суперЭВМ на базе графических процессоров и началом разработки для них программных эмуляторов больших нейронных сетей [19].

В настоящее время (см. рис. 5) научно-технический задел для разработки будущих нейрокомпьютеров с применением мемристоров составляют:

— программные реализации нейронных сетей на транспьютерных и кластерных суперЭВМ;

— реализация нейрокомпьютеров на ПЛИС;

— программные реализации нейрокомпьютеров на суперЭВМ с использованием графических процессоров;

— многолетние разработки цифровых, аналого-цифровых и аналоговых нейрочипов.

4. Выбор технологии изготовления мемристоров. Оценка производительности вычислительных систем с применением мемристоров

Мемристоры изготавливают на базе следующих элементов:

- электрохимические элементы;
- оксид титана;
- оксид тантала;
- полимеры;
- ферроэлектрики;
- углеродные нанотрубки;
- кремний;
- аморфный кремний;
- поликремний.

В результате разработок реализованы мемристоры, мемристорные матрицы и мемристорные системы с различными физическими свойствами, которые необходимо оценивать с точки зрения их применимости в конкретных структурах вычислительных систем.

Применимость той или другой технологии изготовления мемристорных систем необходимо оценивать как с точки зрения существующих критериев оценки производительности, так и с точки зрения оценки производительности будущих нейрокомпьютеров с применением мемристоров.

Производительность нейрокомпьютеров с применением мемристоров необходимо оценивать с точки зрения следующих количественных показателей:

— число эмулируемых нейронов;

— число эмулируемых связей;

— число переключений связей в секунду;

— потребляемая мощность;

— число физических переключений до момента отказа (дополнительно).

Расчет производительности мемристорных систем при анализе технологий необходимо выполнять на нескольких уровнях:

- на уровне элементарных операций;
- на уровне базовых нейросетевых систем на мемристорах;
- на уровне СБИС с учетом внутренней коммуникационной среды;
- на уровне плат с учетом внутриплатной коммуникационной среды;
- на уровне блоков с учетом внеплатной коммуникационной среды;
- на уровне супернейрокомпьютера с учетом межблочной коммутационной среды.

Необходимо высказать некоторое предостережение тем, кто будет оценивать производительность мемристорных систем. В истории развития вычислительной техники известны примеры, когда оптимистическая оценка производительности на уровне элементарных операций предлагаемой технологии оказывалась несостоятельной при распространении оценки производительности на более общие архитек-

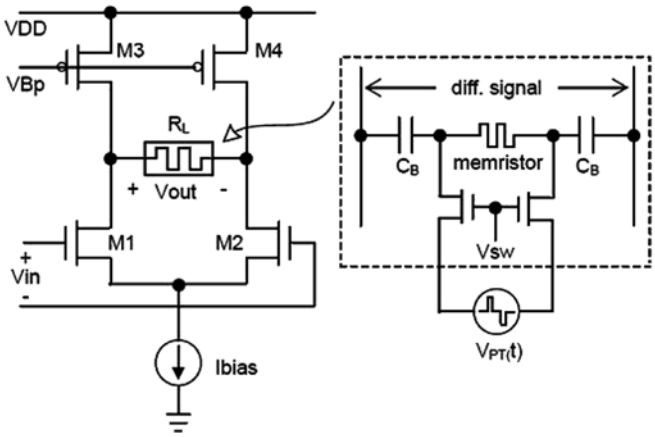


Рис. 8. Реализация синапса нейрона с применением мемристора

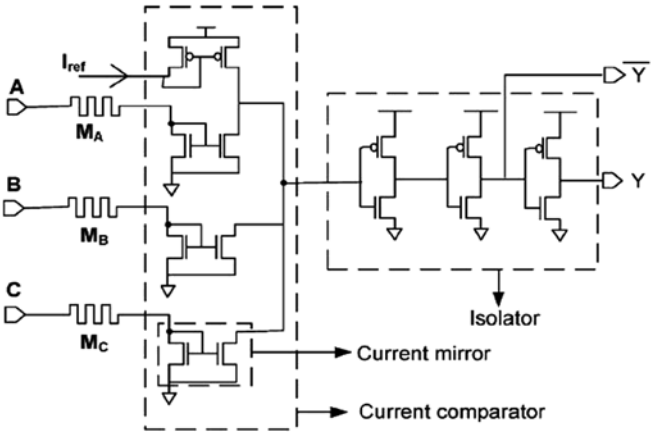


Рис. 9. Схема трехвходового нейрона с применением мемристоров

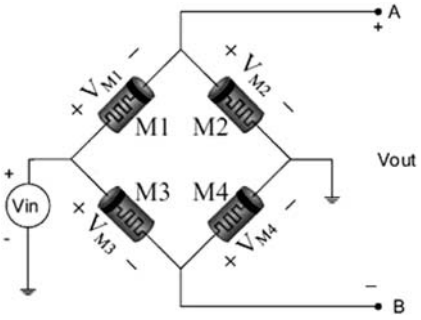


Рис. 10. Схема мемристорного моста

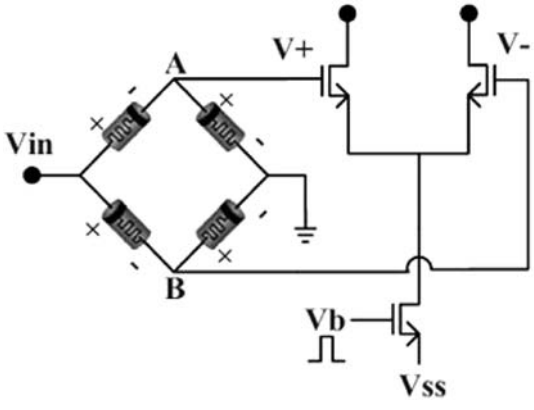


Рис. 11. Схема синапса с мостом из мемристоров

турные элементы, платы, блоки, стойки и т. д. и при расширении круга реализуемых алгоритмов (т. е. в попытке сделать вычислительную систему более универсальной). Так произошло с оптическими ЭВМ, в которых элементарные операции (перемножение матриц) выполнялись очень быстро, однако системы так и не стали универсальными. То же произошло с транспьютерными системами с сопроцессорами обработки сигналов и изображений IMSA100, IMSA110 фирмы *Inmos*, в которых операции умножения—сложения выполнялись очень быстро, но в конкретных структурах одномерных и двумерных аппаратно реализованных Z-фильтров.

Исследование производительности мемристорных систем необходимо начинать с разработки инструментальных систем исследования мемристоров и мемристорных матриц, включая следующие этапы.

1. Исследование, разработка и экспериментальная реализация принципов стыковки мемристорных матриц с элементами КМОП-технологии.

2. Разработка инструментальной системы для исследования мемристорных матриц с выходом на персональную ЭВМ, включая драйвер.

3. Разработка контроллера на ПЛИС для стыковки мемристорных систем с персональными ЭВМ, с драйвером.

4. Экспериментальная и теоретическая оценка понижения энергопотребления в мемристорных системах при переходе к представлению сигналов в виде частотно-модулированной последовательности импульсов.

5. Реализация нейрона с применением мемристоров

В схемной реализации нейрона мемристоры выполняют функцию синапсов — перестраиваемых весовых коэффициентов. Возможно несколько вариантов применения мемристоров для этой цели. Ниже представлены несколько известных вариантов, требующих доработки с точки зрения включения нейрона в схему нейронной сети и с точки зрения включения схемы нейронной сети в общую схему СБИС-нейрочипа.

В работе [20] представлен один из вариантов реализации синапса нейрона с применением мемристоров (рис. 8). Вариант принципиальной схемы многовходового нейрона представлен на рис. 9 [21]. На рис. 10 представлена мостовая схема соединения мемристоров, обеспечивающая реализацию положительных и отрицательных значений весовых коэффициентов в нейронной сети [22]. В той же работе в развитие мостовой схемы соединения мемристоров представлена схема синапса с мостом из мемристоров (рис. 11).

В работе [23] представлен другой вариант соединения мемристоров в схеме синапса (рис. 12) и предлагается схема изменения знака весового коэффициента в синапсе, представленная на рис. 13.

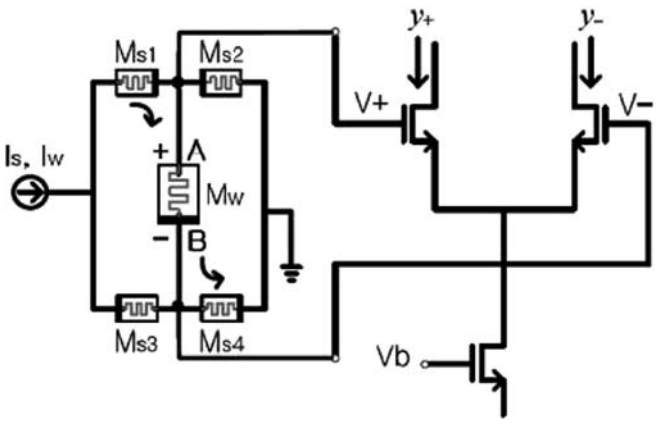


Рис. 12. Вариант реализации синапса с применением мемристоров

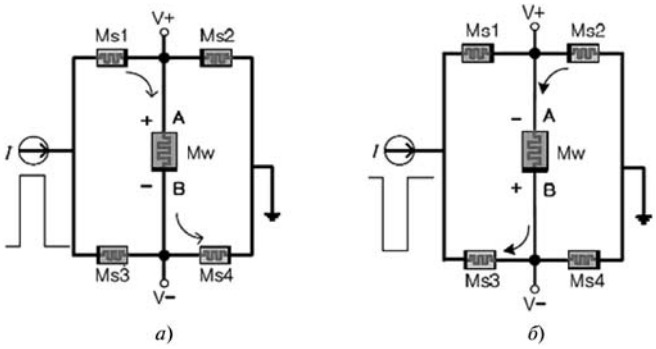


Рис. 13. Предлагаемая схема изменения знака весового коэффициента в синапсе с применением мемристоров:
а — положительная конфигурация; *б* — отрицательная конфигурация

В соответствии с мостовыми схемами соединения мемристоров, представленными в работе [22], в работе [24] показана схема нейрона с несколькими мемристорными входами.

6. Реализация нейронных сетей с применением мемристоров

На рис. 14 представлена простейшая двухслойная нейронная сеть, ориентированная на реализацию с применением мемристоров [22], а на рис. 15 (см. третью сторону обложки) ее реализация с применением мемристоров.

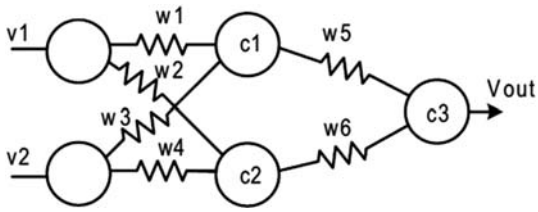


Рис. 14. Типовая двухслойная нейронная сеть

Анализ публикаций 2008—2014 гг. показал, что попытки реализации нейронных сетей с применением мемристоров касаются в основном многослойных нейронных сетей следующих видов:

- с полными последовательными связями;
- с обратными связями (рекуррентные нейронные сети);
- клеточные нейронные сети, ориентированные на обработку изображений.

Рассматриваются нейронные сети, ориентированные на достаточно узкий класс прикладных задач:

- бинарные нейронные сети с бинарными многомерными входными сигналами и бинарными (0, 1) коэффициентами;
- RBF-нейронные сети;
- CMAC-нейронные сети.

В работе [25] предложена (рис. 16) схема реализации весового коэффициента с мемристорами для клеточной нейронной сети. В работе [26] предложена другая схема применения мемристоров в клеточной нейронной сети.

Разработка и реализация принципиальных схем нейронных сетей различных структур с применением мемристоров является предметом дальнейших исследований в ближайшие годы.

7. Настройка нейронных сетей с применением мемристоров

С точки зрения автора, работы по реализации алгоритмов настройки нейронных сетей с применением мемристоров имеют самый начальный характер и в основном касаются реализации простейших алгоритмов типа правила Хебба [27]. Ориентация мемристоров как детища наноэлектроники на реализацию нейронных сетей с большим и очень большим числом нейронов и настраиваемых коэффициентов сделает задачу разработки и реализации алгоритмов настройки важной, сложной, в значительной степени определяемой архитектурой будущих вычислительных систем с применением мемристоров.

Замечание. *Отметим, что правило Хебба, в его логическом варианте в России было реализовано диалоговым интерактивным режимом работы в мостовой системе прецизионного измерения сопротивления (рис. 17), выпускаемой во время второй мировой войны сотнями тысяч штук.*

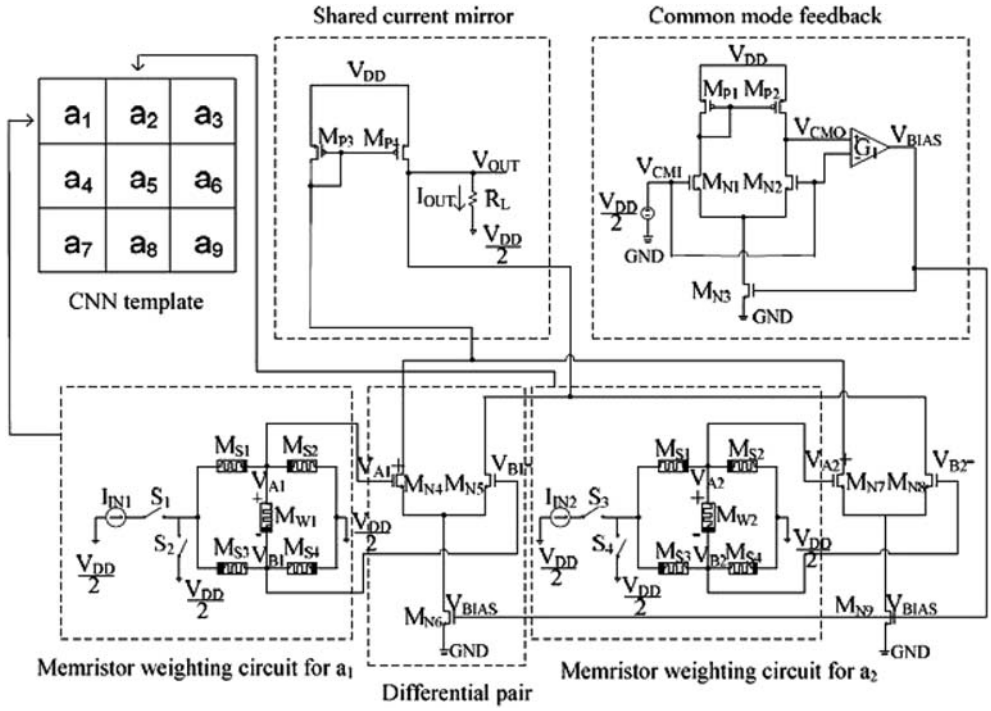


Рис. 16. Схема реализации весового коэффициента с мемристорами для клеточной нейронной сети

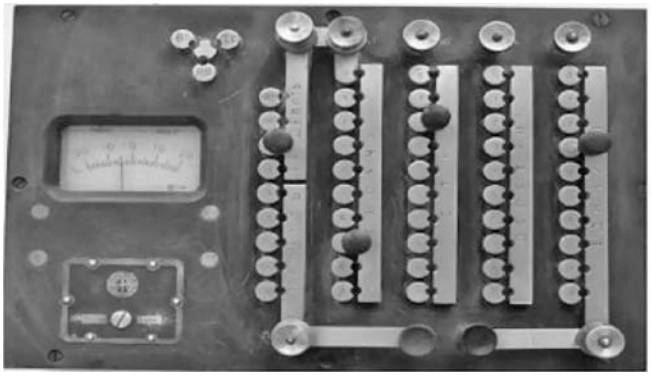


Рис. 17. Отечественный прецизионный измеритель сопротивления, функционирующий по правилу — аналогу правила Хебба

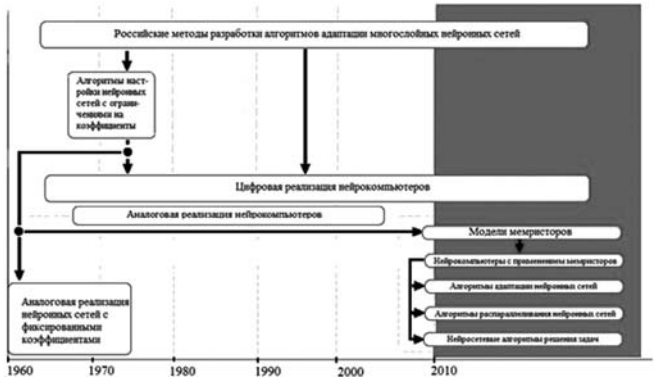


Рис. 18. Общая структура работ по созданию алгоритмов настройки сетей с применением мемристоров

В работе [28] представлен аналитический подход к сравнению простейших алгоритмов настройки применительно к мемристорным системам. Эти методы требуют развития как с точки алгоритмов адаптации, используемых в MATLAB (Neural Network Toolbox), так и с точки зрения алгоритмов адаптации нейронных сетей, учитывающих ограничения на весовые коэффициенты [5, 6, 8—10]. Условно, если в 60-е годы прошлого века алгоритмы настройки формировали для нейронных сетей как объекта управления:

$$y = f\Sigma a_3 \cdot f\Sigma a_2 \cdot f\Sigma a_1 \cdot x$$

с ограничениями на коэффициенты 1, 2, 3-го слоев нейронной сети, то для мемристорных систем эти алгоритмы должны формироваться как для более сложного объекта:

$$y = f\Sigma a_3(z) \cdot f\Sigma a_2(z) \cdot f\Sigma a_1(z) \cdot x.$$

Здесь x — входной сигнал нейронной сети; y — выходной сигнал нейронной сети; f — функция активации нейрона; $a_1(z)$, $a_2(z)$, $a_3(z)$ — передаточные функции мемристоров, формируемые моделями в системах PSpice, Cadence, MATLAB.

На рис. 18 представлена общая структура работы по созданию алгоритмов настройки нейронных сетей с применением мемристоров. Для нейрокомпьютеров с использованием моделей мемристоров на базе созданных алгоритмов настройки многослойных нейронных сетей с учетом ограничений на коэффициенты необходимо разработать специфические для мемристорных систем алгоритмы:

— адаптации нейронных сетей;

— распараллеливания нейросетевых алгоритмов с контуром адаптации на архитектуру вычислительных систем с применением мемристоров.

В связи с этим представленная в работе [22] схема обучения нейронной сети с применением мемристоров с реализацией блока настройки на хост-ЭВМ вряд ли жизнеспособна для вариантов с большим числом мемристоров (рис. 19, см. третью сторону обложки).

Распараллеливание алгоритмов адаптации и требование повышения быстродействия должны привести к внутрикристальной реализации алгоритма обучения, причем на цифровом и аналоговом уровнях.

8. СБИС — нейрочипы с применением мемристоров

Разработка СБИС-нейрочипов с применением мемристоров является одним из важнейших направлений работ в области создания перспективных нейрокомпьютеров. При этом необходима аналитическая работа для обеспечения качественной разработки мемристорных систем, в том числе:

- анализ разработок адаптируемых систем с аналоговой памятью 60-х—70-х годов прошлого столетия в целях использования в современных разработках мемристорных систем;
- анализ разработок аналоговых и аналого-цифровых нейрочипов последних десятилетий в целях использования в современных разработках мемристорных систем.

При этом анализ разработок аналоговых и аналого-цифровых нейрочипов должен касаться следующих разделов:

- аналоговые нейрочипы;
- аналого-цифровые нейрочипы;
- клеточные нейрочипы;
- нейрочипы с частотно-импульсным представлением сигналов;
- оптические и оптоэлектронные нейрочипы;
- молекулярные нейрочипы;
- специализированные аналоговые и аналого-цифровые нейрочипы:
  - АЦП;
  - CMAC;
  - обработки изображений;
  - нейроуправления;
  - ассоциативной памяти;
  - обработки речевой информации;
- другие.

Типичным примером домемристорной разработки цифровых нейрочипов с частотно-импульсной модуляцией сигналов является схема, представленная на рис. 20 [29]. Необходимо отметить несколько попыток реализации СБИС-нейрочи-

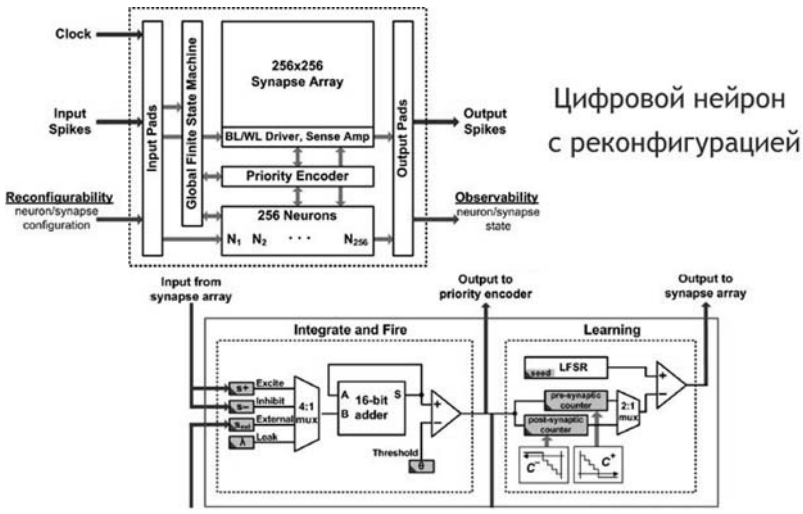


Рис. 20. Пример домемристорных разработок цифровых нейрочипов с частотно-импульсной модуляцией сигналов

пов с применением мемристоров (цифровых и аналого-цифровых) [30—32] — рис. 21—23.

Особое внимание нужно обратить на разработку нейрочипа с применением трехтерминального ферроэлектрического мемристора (рис. 24) [33] (в отличие от наиболее широко распространенных в настоящее время двухтерминальных мемристоров), в значительной степени идеологически повторяющего разработки мемристоров 60-х годов прошлого века [16, 17].

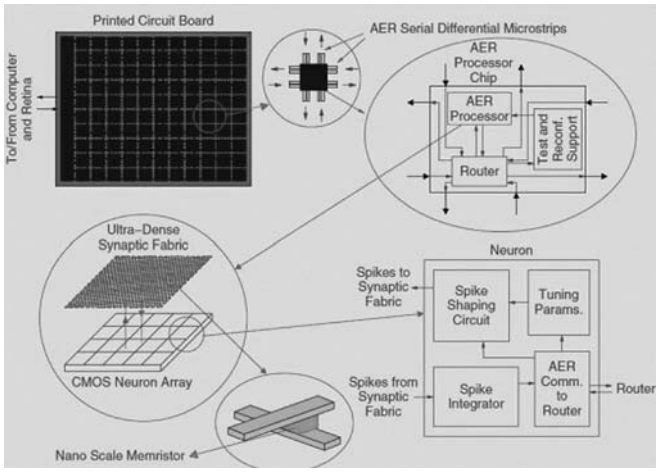


Рис. 21. Вариант архитектуры цифрового нейрочипа с применением мемристоров

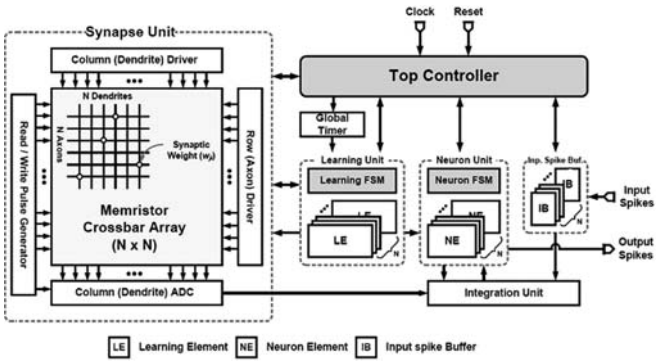


Рис. 22. Вариант архитектуры цифрового нейрочипа с применением мемристоров

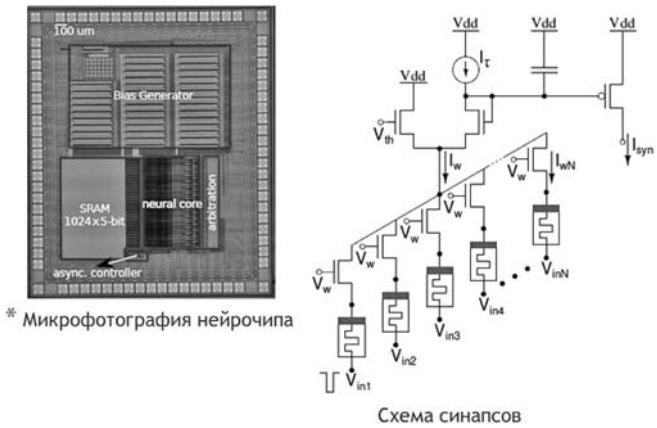


Рис. 23. Аналого-цифровой нейрочип с применением мемристоров

9. Архитектура вычислительных систем с применением мемристоров

На рис. 25 представлена общая архитектура перспективного супернейрокомпьютера с применением мемристоров. Классические варианты хост-ЭВМ и кластерного ядра дополняются в данной архитектуре двумя типами блоков:

- система на базе ПЛИС (БМК) с частотно-импульсным представлением сигналов как программно-аппаратный эмулятор нейронных сетей;
- аналого-цифровая часть, реализованная на СБИС-нейрочипах с применением мемристоров.

Результатами работ по разработке технических средств супернейрокомпьютера с применением мемристоров должны быть:

- архитектура и экспериментальные образцы каскадируемых СБИС на базе мемристорных матриц для использования в перспективных супернейрокомпьютерах;
- архитектура и экспериментальные образцы базовых каскадируемых плат в конструктивах PCI, microPCI, VME с использованием СБИС на базе мемристоров для использования в перспективных супернейрокомпьютерах;

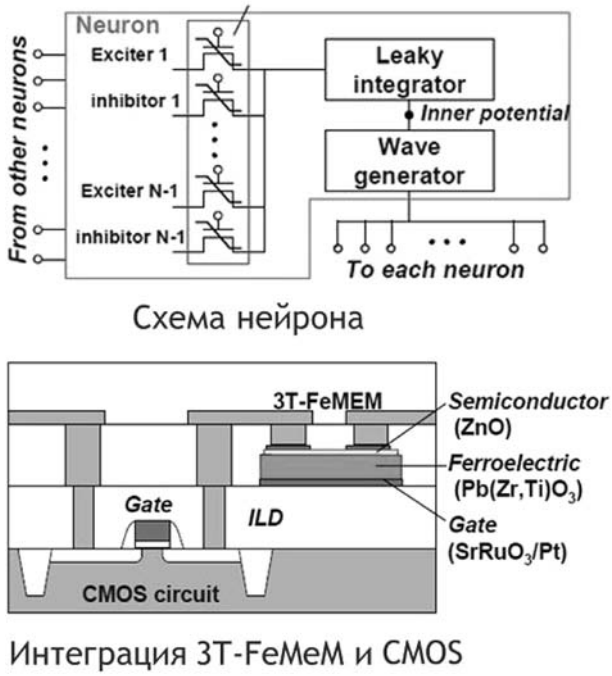


Рис. 24. Нейрочип с применением трехтерминального ферроэлектрического мемристора

Рис. 25. Архитектура перспективного супернейрокомпьютера с использованием мемристоров

- архитектура и экспериментальные образцы базовых каскадируемых блоков с использованием плат в конструктивах PCI, microPCI, VME для использования в перспективных супернейрокомпьютерах;
- оформление патентов на архитектуры СБИС, плат и блоков на базе мемристорных матриц.

Структура системного программного обеспечения перспективного супернейрокомпьютера с применением мемристоров должна состоять из следующих разделов:

- общее системное программное обеспечение (ОС, компиляторы и пр.);
- алгоритмы и программы распараллеливания нейронных сетей различной структуры;
- вариант MPI;
- вариант системы типа Foundation для ПЛИС с применением мемристоров;
- алгоритмы и программы настройки нейронных сетей с учетом ограничений мемристоров.

Для разработанных вариантов архитектуры должна быть проведена общая оценка качества вычислительных систем с использованием мемристоров, в том числе:

- разработка и апробация методики количественной оценки производительности суперкомпьютеров на базе мемристоров;
- разработка методики обеспечения и оценки необходимой точности вычислений при переходе в супернейрокомпьютере к аналоговой обработке;
- оценка понижения энергопотребления в супернейрокомпьютере при переходе к представлению сигналов в виде частотно-модулированной последовательности узких импульсов.

10. Прикладные задачи

Текущее состояние интереса к решению прикладных задач на вычислительных системах с применением мемристоров условно можно представить схемой, изображенной выше на рис. 1.

Необходимо отметить соответствие структуры и параметров реализуемых нейронных сетей с применением мемристоров классу решаемых задач.

Структура нейронных сетей:

- многослойные нейронные сети;
- 2D и 3D клеточные нейронные сети;
- реккуррентные нейронные сети;
- хаотические нейронные сети.

Разрядность коэффициентов нейронных сетей:

- 1;
- K;
- управляемая разрядность;
- плавающая запятая.

Скорость изменения коэффициентов:

- нулевая;
- средняя;
- большая.

Большие нейронные сети — главная задача реализации нейрокомпьютеров с применением мемристоров. Как следствие — выбор для решения на мемристорных системах следующих сложных задач:

- обнаружение атак на информационные ресурсы в больших распределенных вычислительных сетях;
- нейрокриптография;
- уравнения математической физики, включая генерацию адаптивных сеток;
- обработка видеоизображений и медиаинформации (проблема *big data*);
- создание 3D-моделей мозга;
- управление плазмой;
- обработка геномной и протеомной информации;
- любые другие применения нейрокомпьютеров, где необходимо увеличить отношение производительности к стоимости или энергопотреблению.

Заключение

Всю историю развития вычислительной техники сопровождают работы по эмуляции алгоритмов искусственного интеллекта, принятия решений и нейронных сетей.

Российские работы по эмуляции нейронных сетей на каждом этапе развития вычислительной техники были связаны с использованием максимально производительных образцов:

- 60-е годы прошлого столетия — аналоговые реализации;
- 70—80 годы прошлого столетия — программные цифровые реализации;
- 90-е годы прошлого столетия — транспьютерные системы, цифровые, аналого-цифровые и аналоговые нейрочипы, ПЛИС;
- 2000-е годы — различные нейрочипы, графические процессоры;
- в настоящее время — различные нейрочипы, мемристорные системы.

Идеология построения нейрокомпьютеров менялась слабо, менялись и развивались технологии реализации.

Научно-технический задел по созданию отечественных супернейрокомпьютеров экзафлопной производительности с применением мемристоров представлен такими разделами.

- *Теория нейронных сетей* как методика синтеза структур из нейронов различного вида, алгоритмов адаптации весовых коэффициентов в этих структурах в процессе решения различных задач.
- *Нейроматематика* как раздел вычислительной математики, связанный с решением в нейросетевом логическом базисе различных сложных формализуемых и неформализуемых задач.
- *Нейроуправление* как раздел теории управления, связанный с применением нейрокомпьютеров в качестве систем идентификации сложных динамических систем и нейрокомпьютеров для управления.
- *Нейрокомпьютеры и нейрочипы*.

Российский опыт работ в области нейросетевых технологий и их применения обобщен в монографии [34].

Когда сейчас говорят: "мемристоры — будущее искусственного мозга", то это вызывает серьезные сомнения. На самом деле, мемристоры — это очередной этап эволюционного, гармоничного развития вычислительной техники, в котором за несколько предыдущих десятилетий проведена фундаментальная подготовка к построению высокопроизводительных вычислительных систем, реализующих следующие принципы:

- переход от классической фон-Неймановской архитектуры к распределенной;
- возврат в части реализации алгоритмов решения задач к аналоговой обработке в целях увеличения быстродействия при контролируемой точности;
- переход от представления сигналов в виде уровней токов и напряжений к представлению сигналов в виде частоты последовательности узких импульсов;
- переход к архитектурам, в которых функции памяти и обработки хотя бы частично совмещены в отличие от классических ЭВМ с полностью распределенными функциями памяти и обработки;
- переход к нейроматематике — нейросетевым алгоритмам решения задач;
- переход к нейроуправлению — нейросетевым алгоритмам и системам управления сложными динамическими объектами.

Модель мозга и его разделов — задачи искусственного интеллекта — важные, но вторичные задачи вычислительной техники, в том числе и мемристорных систем. Мемристорная система — это просто более эффективный эмулятор нейронных сетей по сравнению с предыдущими типами суперЭВМ. На рис. 26 показано развитие базовых вычислительных платформ для реализации эффективных программных и программно-аппаратных эмуляторов нейронных сетей.

Несмотря на то, что вычислительную технику за весь большой период существования и развития всегда использовали для реализации мыслительных функций, только в период 2007—2009 гг. и далее появились работы, в которых показано, что даже самые современные суперЭВМ типа Blue Gene, K-Fujitsu и других еще крайне ограничены по производительности для моделирования разделов мозга в реальном времени. Эти исследования показывают, как далека современная (и думаем, что даже перспективная на ближайшее время) технология от технологии реализации реального живого мозга, и как далеки мы от настоящего, глубокого понимания принципов работы живого мозга, которые мы сможем использовать при построении искусственных систем.

С этой точки зрения мемристоры — это не шаг к модели мозга, а небольшой шаг в развитии высокопроизводительной вычислительной техники, который позволит создать более эффективные эмуляторы нейронных сетей и более быстро и качественно решать инженерные задачи, следовательно, можно сделать следующие выводы:

— мемристоры — эволюционное развитие микроэлектроники в сторону нанотехнологий;

— нейрокомпьютеры с применением мемристоров — эволюционное развитие высокопроизводительной техники.

Мемристорные системы имеют такое же далекое отношение к моделям мозга, как нейрокомпьютеры 60-х гг. прошлого века, IBM Computer 70-х и т. д.

На рис. 27 условно представлено мнение автора о взаимоотношении разработок сверхвысокопроизводительной

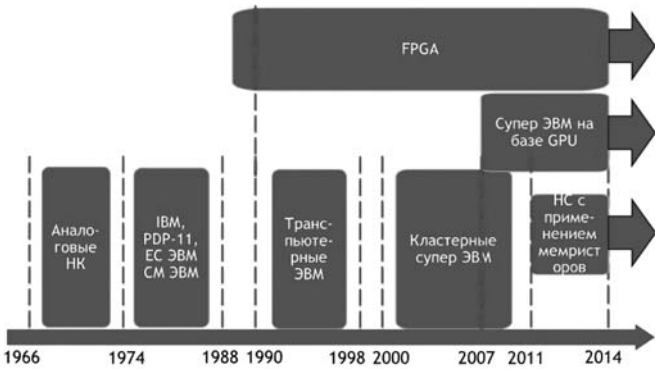


Рис. 26. Создание наиболее эффективных программно-аппаратных эмуляторов нейронных сетей — главная задача нейросетевых технологий

Рис. 27. Структура взаимоотношений высокопроизводительной техники и нейрофизиологии

техники, в том числе и супернейрокомпьютеров, и нейрофизиологии. Исходя из данной схемы эти два направления соединяют четыре проблемы:

— эмуляция нейронных сетей для решения технических задач;

— интерфейс "мозг — компьютер";

— нейроимплантанты;

— искуственный мозг.

В части эмуляции нейронных сетей для решения технических задач с точки зрения автора роль нейрофизиологии для развития вычислительных систем практически нулевая. Высокопроизводительные вычислительные системы используют в нейрофизиологии для эмуляции больших нейросетевых образований. В настоящее время эта процедура, реализуемая на ЭВМ IBM Blue Gene и Fujitsu K, показывает, что возможности современных суперЭВМ для моделирования разделов мозга крайне незначительны.

В части интерфейса "мозг — компьютер" нейрофизиология может оказать значительную помощь вычислительной технике в части новых методов управления вычислительными системами.

В части нейроимплантантов вычислительные системы являются основой их технической реализации.

Проблема "Искусственный мозг" является в настоящее время, с точки зрения автора, чисто искусственной, так как технологии реализации вычислительных систем резко отличаются от технологий реализации живого мозга и переносимость результатов с одного на другое является чисто условной.

Необходимо отметить основные перспективные направления работ по созданию нейрокомпьютеров с применением мемристоров с ориентацией на большие нейронные сети:

- выбор технологии с точки зрения обеспечения максимальной производительности;
- разработка вариантов архитектуры и схемотехники СБИС-нейрочипов с применением мемристоров;
- разработка вариантов архитектуры и схемотехники нейроплат, нейроблоков и нейростоек нейрокомпьютеров с применением мемристоров;
- разработка алгоритмов адаптации нейронных сетей, адекватных архитектурам с применением мемристоров и с учетом моделей мемристоров;
- разработка нейросетевых алгоритмов решения задач, адекватных архитектурам с применением мемристоров;
- разработка алгоритмов распараллеливания нейросетевых структур, адекватных архитектурам с применением мемристоров;
- разработка методики оценки производительности, адекватной архитектуре с применением мемристоров.

Список литературы

1. **Бёрд Киви.** К точке критического перехода. 3D-News (Daily Digital Digest). 29.05.2013.
2. **McKenziy A., Branch D. W., Forsythe C., James C. D.** Toward Exascale Computing through Neuromorphic Approaches // Sandia. Report, Sand 2010—6312. September. 2010.
3. **Галушкин А. И.** Стратегия развития современных супернейрокомпьютеров на пути к экзафлопным вычислениям // Информационные технологии. Приложение. 2012. № 3. 32 с.
4. **Галушкин А. И.** Многослойные системы распознавания образов. М.: Изд. МИЭМ, 1970.
5. **Галушкин А. И.** Синтез многослойных систем распознавания образов. М.: Энергия, 1974.
6. **Галушкин А. И.** Теория нейронных сетей. Серия "Нейрокомпьютеры и их применение". Кн. 1. М.: ИПРЖР. 2000, 416 с.
7. **Галушкин А. И., Цыпкин Я. З.** Нейронные сети: история развития теории. Серия "Нейрокомпьютеры и их применение". Кн. 5. М.: ИПРЖР. 2001. 840 с.
8. **Галушкин А. И.** Теория нейронных сетей. Пекин: Изд-во Университета Синьхуа (на китайском языке). 2003.

9. **Galushkin A. I.** Neural Network Theory. Berlin-Heidelberg: Springer, 2007.
10. **Галушкин А. И.** Нейронные сети: основы теории. М.: Горячая линия — Телеком, 2010.
11. **Нейроматематика.** Серия "Нейрокомпьютеры и их применение". Кн. 6. М.: ИПРЖР, 2002.
12. **Галушкин А. И.** Нейроматематика (проблемы развития) // Нейрокомпьютер. 2003. № 1.
13. **Галушкин А. И.** Методика решения задач в нейросетевом логическом базисе. Информационные технологии. Приложение. 2006. № 6. 32 с.
14. **Терехов В. А., Ефимов Д. В., Тюкин И. Ю.** Нейросетевые системы управления. Серия "Нейрокомпьютеры и их применение". Кн. 8. М.: ИПРЖР, 2004.
15. **Галушкин А. И.** Основы нейроуправления. Информационные технологии. Приложение. 2002. № 10. 32 с.
16. **Аналоговые** запоминающие и адаптивные элементы / Под ред. Б. С. Сотскова. М.: Энергия, 1973.
17. **Трейер В. В., Елизаров А. Б.** Электрохимические интегрирующие и аналоговые запоминающие элементы. М.: Энергия, 1973.
18. **Галушкин А. И.** Нейрокомпьютеры. Серия "Нейрокомпьютеры и их применение". Кн. 1. М.: Радиотехника, 2000.
19. **Нейрокомпьютеры:** от программной к аппаратной реализации. М.: Горячая линия — Телеком, 2006.
20. **Shin S., Kim K., Kang S. M.** Nemristors and Their Applications for Nanocomputing // EEE Trans. on Nanotechnology, Mar. 2011.
21. **Harika Manem, Jeyavijayan Rajendran, Garrett S. Rose.** Stochastic Gradient Descent Inspired Training Technique for a CMOS / Nano Memristive Trainable Threshold Gate Array // IEEE Transactions on circuits and systems — I: regular papers. V. 59, N. 5, May 2012.
22. **Adhikari S. P., Yang C., Kim H., Chua L. O.** Memristor Bridge Synapse-Based Neural Network and Its Learning // IEEE Transactions on neural network and laerning systems. September 2013. V. 23, N. 9.
23. **Pd. Sah M., Yang C., Hyongsuk Kim, Leon O. Chua.** Meristor Circuit for Artificial Synaptic Weighting of Pulse Inputs // Circuits and Systems (ISCAS). IEEE Intemational Symposium. 2012.
24. **Maheshwar Pd. Sah, Changju Yang, Ram Kaji Budhathoki, Hyongsuk Kim.** Features of memristor emulator-based artificial neural synapses // Circuits and Systems (ISCAS), IEEE International Symposium. 2012.
25. **Young-Su Kim, Keong-Sik Mim.** Synaptic weighting circuits for Cellular Neural Networks // Cellular Nanoscale Networks and Their Applications (CNNA). 13th International Workshop. 2012.
26. **Walls T. J., Likharev K. K.** Self-Organization in Autonomous, Recurrent, Firing-Rate CrossNets With Quasi-Hebbian Plasticity // IEEE Tansactions on Neural Networks and Learning Systems / April. V. 25. N. 4. 2014.
27. **Hebb D. O.** The organization of Behavior: A Neuropsyhological Theory. New York: Wiley, June 1949.
28. **Gorchetchnikov A., Versace M., Ames H., Chandler B., Leveille J., Livitz G., Mingolla E., Snider G., Amerson R., Carter D., Abdalla H., Qureshi M. S.** Review and unification of learning framework in Cog Ex Machina platform for memristive neuromorphic hardware // Proc. of International Joint Conference on Neural Networks, San Jose, California, USA, July 31—August 5, 2011.
29. **Seo Jae-sun, Brezzo B., Liu Y., Parker B. D., Esser S. K., Montoye R. K., Rajendran B., Tierno J. A., Chang L., Modha D. S., Friedman D. J.** A 45 nm CMOS neuromorphic chip with a scalable architecture for leaming in networks of spiking neurons // Custom Integrated Circuits Conference (CICC), 2011.
30. **Serrano-Gotarredona T., Prodromakis T., Linares-Barranco B.** A proposal for hybrid memristor-CMOS spiking neuromorphic learning systems // IEEE Circuits and Systems Magazine. 2013. Vol. 13, Is. 2. P. 74—88.
31. **Yongtae Kim, Yong Zhang, Peng Li.** A digital neuromorphic VLSI architecture with memristor crossbar synaptic array for machine learning // IEEE International SOC Conference (SOCC), 2012.
32. **Azghadi M. R., Moradi S., Indiveri G.** Programmable neuromorphic circuits for spike-based neural dynamics // IEEE 11th International New Circuits and Systems Conference (NEWCAS), 2013.
33. **Kaneko Y., Nishitani Yu., Ueda M., Tsujimura A.** Neural network based on a three-terminal ferroelectric memristor to enable on-chip pattern recognition // Symposium VLSI Technology (VLSIT). 2013.
34. **Галушкин А. И., Симоров С. Н.** Нейросетевые технологии в России (1982—2010 г.). М.: Горячая линия — Телеком, 2011.

A. I. Galushkin, Prof., Deputy Head of Chair,
Moscow Institute of Physics and Technology, Dolgoprudny, neurocomputer@yandex.ru

Memristor in High-Performance Computing Development

The author of this article is not a physicist or a mathematician. He is an engineer and has 50 years of experience in neural network technologies and their application. This is a rather narrow class of over high performance computing. This article presents the author's opinion about the prospects of this trend of super computing with the memristor invention.

Keywords: *memristor, supercomputer, multilayer, neural network, analog computation*

References

1. **Kivi B.** K tochke kriticheskogo perehoda. *3D-News (Daily Digital Digest)*. 29.05.2013 g.
2. **McKenzie A., Branch D. W., Forsythe C., James C. D.** Toward Exascale Computing through Neuromorphic Approaches // *Sandia Report, Sand 2010 — 6312.* September 2010.
3. **Galushkin A. I.** Strategija razvitija sovremennyh supernejrokomp'juterov na puti k jekzaflopnym vychislenijam. *Informacionnye tehnologii*. Prilozhenie. 2012. N. 3. 32 p.
4. **Galushkin A. I.** *Mnogoslojnye sistemy raspoznavanija obrazov*. M.: MIJeM, 1970.
5. **Galushkin A. I.** *Sintez mnogoslojnyh sistem raspoznavanija obrazov*. M.: Jenergija, 1974.
6. **Galushkin A. I.** *Teorija nejronnyh setej*. Kn. 1. Serii "Nejrokomp'jutery i ih primenenie". M.: IPRJR, 2000.
7. **Galushkin A. I., Cypkin Ja. Z.** *Nejronnye seti: istorija razvitija teorii*. Kn. 5. Seriy "Nejrokomp'jutery i ih primenenie". M.: IPRJR, 2001, 840 c.
8. **Galushkin A. I.** *Teorija nejronnyh setej*. Pekin: Izdatel'stvo Universiteta Sin'hua. (na kitajskom jazyke). 2003.
9. **Galushkin A. I.** *Neural Network Theory*. Berlin Heidelberg: Springer. 2007.
10. **Galushkin A. I.** *Nejronnye seti: osnovy teorii*. M.: Gorjachaja linija — Telekom, 2010.
11. **Nejromatematika.** Kn. 6 Seriya *"Nejrokomp'jutery i ih primenenie"*. M.: IPRJR, 2002.
12. **Galushkin A. I.** Nejromatematika (problemy razvitija). *Nejrokomp'juter*. 2003. N. 1.
13. **Galushkin A. I.** Metodika reshenija zadach v nejrosetevom logicheskom bazise. *Informacionnye tehnologii*. Prilozhenie. 2006. N. 6. 32 p.
14. **Terehov V. A., Efimov D. V., Tyukin I. Yu.** *Nejrosetevye sistemy upravlenija*. Kn. 8. Seriya "Nejrokomp'jutery i ihprimenenie". M.: IPRJR, 2004. 32 p.
15. **Galushkin A. I.** Osnovy nejroupravlenija. *Informacionnye tehnologii*. Prilozhenie. 2002. N. 10.
16. ***Analogovye*** *zapominajushhie i adaptivnye jelementy* / pod red. B. S. Sotskova. M.: Jenergija, 1973.

17. **Trejer V. V., Elizarov A. B.** *Jelektrohimicheskie integrirujushhie i analogovye zapominajushhie jelementy*. M.: Jenergija, 1973.
18. **Galushkin A. I.** Nejrokomp'jutery. Kn. 1. Seriya "Nejrokomp'jutery i ih primenenie". M.: IPRJR, 2000.
19. ***Nejrokomp'jutery:*** ot programmnoj k apparatnoj realizacii. M.: Gorjachaja linija — Telekom, 2006.
20. **Shin S., Kim K., Kang S. M.** Memristors and Their Applications for Nanocomputing. *IEEE Trans. on Nanotechnology*, Mar. 2011.
21. **Manem H., Rajendran J., Garrett S.** Rose Stochastic Gradient Descent Inspired Training Technique for a CMOS / Nano Memristive Trainable Threshold Gate Array. *IEEE Transactions on circuits and systems* — I: regular papers. May 2012. V. 59, N. 5.
22. **Adhikari S. P., Yang C., Kim H.** Memristor Bridge Synapse-Based Neural Network and Its Learning. *IEEE Transactions on neural networks and learning systems*. 2012. V. 23, N. 9.
23. **Maheshwar Pd. Sah, Changju Yang, Hyongsuk Kim and Leon O Chua.** Meristor Circuit for Artificial Synaptic Weighting of Pulse Inputs. *Circuits and Systems (ISCAS),* 2012, IEEE International Symposium.
24. **Maheshwar Pd. Sah, Changju Yang, Ram Kaji Budhathoki, Hyongsuk Kim.** Features of memristor emulator-based artificial neural synapses. *Circuits and Systems (ISCAS), 2012 IEEE International Symposium*.
25. **Young-Su Kim, Kyeong-Sik Min.** Synaptic weighting circuits for Cellular Neural Networks. *Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop*.
26. **Walls T. J., Likharev K. K.** Self-Organization in Autonomous, Recurrent, Firing-Rate CrossNets With Quasi-Hebbian Plasticity. *IEEE Transactions on Neural Networks and Learning Systems*. 2014. V. 25, N. 4.
27. **Hebb D. O.** *The organization of Behavior: A Neuropsyhological Theory*. New York: Wiley, June 1949.
28. **Gorchetchnikov A., Versace M., Ames H., Chandler B., Leveille J., Livitz G., Mingolla E., Snider G., Amerson R., Carter P., Abdalla H., Qureshi M. S.** Review and unification of learning framework in Cog Ex Machina platform for memristive neuromorphic hardware // *Proc. of International Joint Conference on Neural Networks*. San Jose, California, USA, July 31—August 5, 2011.
29. **Jae-sun Seo** et al. A 45 nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. IEEE. *Custom Integrated Circuits Conference (CICC)*, 2011.
30. **Serrano-Gotarredona T., Prodromakis T., Linares-Barranco B.** A Proposal for Hybrid Memristor-CMOS Spiking Neuromorphic Learning Systems. *IEEE Circuits and Systems Magazine*. V. 13, Is. 2.
31. **Yongtae Kim, Yong Zhang, Peng Li.** A digital neuromorphic VLSI architecture with memristor crossbar synaptic array for machine learning. *IEEE International SOC Conference (SOCC),* 2012.
32. **Azghadi M. R., Moradi S., Indiveri G.** Programmable neuromorphic circuits for spike-based neural dynamics. *IEEE 11th International New Circuits and Systems Conference (NEWCAS)*, 2013.
33. **Kaneko Y., Nishitani Yu., Ueda M., Tsujimura A.** Neural network based on a three-terminal ferroelectric memristor to enable on-chip pattern recognition. *Symposium VLSI Technology* rVLSIT). 2013.
34. **Galushkin A. I., Simorov S. N.** *Nejrosetevye tehnologii v Rossii* (1982—2010 g.). M.: Gorjachaja linija — Telekom, M. 2011.

УДК 519.856.2

В. В. Федосов, канд. техн. наук, доц., г. Москва, e-mail: vlr.fdsv@gmail.com,
А. В. Федосова, канд. физ.-мат. наук,
Национальный университет, Богота, Колумбия
(Universidad Nacional de Colombia, e-mail: afedosova@unal.edu.co)

Использование нейросетей для графической оценки загрязнения территории выбросами группы источников

Взамен математического описания выбросов загрязнений источниками предложено использовать нейрофункции, генерируемые по данным рисованных контурных графиков облаков выбросов.

Нейрофункции эффективно выводят графику выборочного или общего загрязнения территории и пригодны для дальнейших расчетов или оптимизации.

Ключевые слова: *источники промышленных выбросов, облака выбросов, загрязнение территории, контурные графики, нейронные сети, фильтрация выбросов*

Введение

Промышленная экология рассматривает загрязнения территории или среды источниками вредных выбросов (например промпредприятиями) как сложные системы, содержащие параметры источников (мощности, размещение, дальность разброса), выбросов (типы, масса и размеры частиц, наборы компонентов или состав), среды (ветер, влажность, температура), динамику и пр. Выбросы в значительной степени формируются за счет пылевых, капельных или газовых частиц. Несмотря на обширные базы данных и трудоемкость создания таких систем, конечные результаты расчетов загрязнений территорий следует признать только оценочными.

При моделировании ключевым является функциональное описание выбросов источниками. Его концепция, математический аппарат, введение усложнений (упрощений) определяются целями задачи и квалификацией исследователя. Последующие расчеты облаков выбросов ведут к построению соответствующих карт загрязнения, обсуждению адекватности моделирования и практической ценности результатов. Такой подход достаточно апробирован и ценен уже тем, что выявляет неясные заранее наложения загрязнений выбросами группы источников. Он применим как для прямых, так и для оптимизационных расчетов загрязнений.

Между тем авторская математика часто субъективна, достаточно трудоемка и ограничена в вариативности описания облаков выбросов. Формулы содержат значительное число условных коэффициентов. При наличии в модели множества источников выбросов их работа описывается единообразно, хотя на практике облака даже близко расположенных

источников нередко формируются по-разному. Формульные модели, как правило, чувствительны к малейшим изменениям входной информационной базы, но их трудно интерпретировать в результатах. Принятый в модели математический аппарат сложно корректировать, так как он уже "прошит" в программировании. Результаты работы моделей с разными математическими подходами трудно сопоставимы.

Противоречие состоит в том, что от такой аналитики хотелось бы уйти, но наличие функционального описания выбросов для моделирования остается необходимым.

В работе ставились следующие задачи:

- формировать облака выбросов без математических формул;
- входную базу данных связывать не с источниками, а сразу с облаками выбросов;
- напрямую встраивать в облака данные мониторинга или прогноза загрязнений;
- сохранить возможность получения функционального описания облаков выбросов.

Такие возможности открывает использование нейросетей. Современные системы (например, MATLAB) имеют полный набор процедур проектирования, обучения и генерирования нейросетей. Главные достоинства использования нейросетей в данной работе следующие:

- отмена авторской математики;
- значительное упрощение входной базы данных модели;
- работа напрямую с облаками выбросов;
- автоматическое получение функционального описания выбросов.

На рис. 1 приведена схема реформирования традиционного аналитического подхода.

Рис. 1. Схема применения аналитических и нейрофункций в описании выбросов

Концептуально обе ветви содержат по одному авторскому элементу, следовательно, предположительное качество моделирования сопоставимо. В то же время второй подход менее трудоемок (напрямую рисовать облака легче, чем описывать их формулами), более свободен для моделирования и менее требователен к квалификации исследователя. Важно также, что подготовка данных к расчетам в какой-то мере становится типовой.

В работе показана возможность отображения облаков выбросов нейрофункциями и на их основе построения карт загрязнения территории. Проектирование нейрофункций и представление загрязнений в графике вели в MATLAB.

Алгоритм и численные эксперименты

В основе алгоритма лежит способ представления облака разброса загрязнений источником в пределах территории непосредственно в графике и дальнейшей генерации нейрофункции на основе созданной графики.

В картографии для плоского изображения рельефных поверхностей широко применяют контурные графики, на которых каждый из контуров (линий) связывается с определенным значением высоты. Контуры могут быть полностью замкнутыми в пределах территории либо обрываться на какой-то из границ. Создание контурного графика является быстрой и элементарной процедурой.

Если контуры изобразить в цвете (связать цвет контура со значением предполагаемой функции загрязнения), то попиксельное считывание такого многоконтурного рисунка образует массив координат точек территории с переменным значением функции. Последовательность размещения точек в массиве значения не имеет. На основе данных этого массива возможно проектирование и обучение нейрофункции. В принципе, нейрофункции являются аналогами процедур аппроксимации, однако последние организуют дискретное представление результатов, в то время как, например, для задач полубесконечной оптимизации загрязнений [1, 2] требуется непрерывное представление. В качестве функции нейросеть уже обеспечивает бесконечный набор координат точек территории.

Проектирование, обучение и тестирование нейронных сетей описывали в терминах *Neural Networks Tools* MATLAB [3]. Вариантом архитектуры выбрана четырехслойная полносвязная сеть с числом нейронов слоев [2-6(8)-4(6)-1], функция создания сети *"newff"*, передаточные функции *"logsig"*, *"purelin"*, обучающий алгоритм *"trainlm"*.

При обработке массивов точек контуров наиболее удачно генерации нейросетей протекали при понижении исходного параметра *"Performance"* примерно на два порядка. Генерации нейрофункций требовали неопределенного числа пусков (в работе оно составило 3—8).

Прямое создание облаков выбросов требует иной информационной базы. Если, например, аналитическое описание выбросов относительно несложными параболическими функциями требует указания мощности источника, его координат, коэффициента дальности разброса, указания силы и направления ветра [2], то взамен этого для генерации нейрофункции достаточен только вектор значений загрязнения на контурах. Такая "экономия" достигается по каждому источнику или компоненту выбросов, если их состав сложный. Непосредственные координаты источников перестают быть аргументом выбросов, так как источник отождествляется сразу с облаком его выбросов. Хотя косвенно предположить место расположения источника в облаке можно, если иметь карту линий тока ветра данной территории.

Одному облаку можно ставить в соответствие выбросы двух, более и даже всех источников. Правда, последнее малоприемлемо, так как "рисованная" карта загрязнения территории всеми источниками является, по-существу, решением задачи. При сложном составе выбросов облако можно ассоциировать с одним компонентом, с выборкой компонентов либо суммой всех компонентов.

Расчет суммарного загрязнения в точках $s(X, Y)$ территории (T) вели по формуле

$$z(s_T) = \sum_{j=1}^{N} Ft(j) \times \mathrm{sim}(\mathrm{net}(j, s_T)), \qquad (1)$$

где $z(j, s_T)$ — значения нейрофункции $net(j, s_T)$ на сетке территории s_T; j — номер источника; sim() — функция, моделирующая работу нейросети; $Ft(j)$ — коэффициент фильтрации выбросов j источника [0...1]; N — число источников выбросов.

С помощью облаков представили воздействие на территорию 24 источников выбросов. Размеры территории d [600 × 300]. Рисованным контурам облаков ставили в соответствие три вектора условных значений выбросов: **v1** [10, 30, 50, 60, 70]; **v2** [80, 120, 160, 200, 230], **v3** [140, 210, 270, 350, 420]. По выбросам каждого источника генерировали нейрофункцию.

Сравнительная графика

На рис. 2—4 приведена графика облаков выбросов источников ($N = 24$) по результатам работы (выходов) обученных нейронных сетей. Облака представлены дискретной графикой, однако непрерывность нейрофункций позволяет объявить требуемую дискретность уровней вывода графики.

Облака разброса имели разный характер (с выраженной асимметрией, с подразумеваемым разным ветровым смещением, много- или унимодальными экстремумами).

Описать такое многообразие выбросов аналитически было бы проблемным. Генерация нейрофункций всех источников открывает далее возможность многих комбинаций представления карт загрязнения территории: суммарные общие, суммарные по выборкам, суммарные по компонентам выбросов и т. д.

На рис. 5 (см. четвертую сторону обложки) показана карта загрязнения территории выборкой 1 источников (2, 4, 6, 7, 9, 13—15, 18, 21, 24) при $Ft() = 1$ (фильтрация выбросов отсутствует). Загрязнения характеризуются двумя значительными возвышениями в юго-западной и одним локальным в центральной частях территории. Расположенность облака может предполагать концентрацию размещения группы 1 источников на юго-западной части и отсутствие ветровых смещений. Если ветровые смещения имели место, то предполагается: а) юго-западное размещение источников при умеренном также юго-западном ветровом смещении; б) северо-восточное и восточное размещение источников при значительном северо-восточном смещении. Полностью свободен от загрязнения небольшой (~5 %) участок в северо-западной части территории.

На рис. 6 (см. четвертую сторону обложки) показана карта загрязнения территории выборкой 2 источников (1, 3, 5, 8, 10, 11, 12, 16, 17, 19, 20, 22, 23) при $Ft() = 1$. Эпицентром загрязнения является значительное пятно в центральной части. Если предположить отсутствие или крайне слабые северо-восточные ветровые смещения, то мощности источников группируются именно в центральной части. Подобная карта также не исключает группировки источников в юго-западной части при наличии сильных смещений в сторону центра.

На рис. 7 (см. четвертую сторону обложки) показана карта общего загрязнения территории всеми источниками при $Ft() = 1$. Выявлены координаты центрального пятна загрязнений с уровнем $\geqslant 1550$ у. е., от которого на юго-запад уходит шлейф с уровнем $\geqslant 1300$ у. е. Оцифрованные склоны характеризуют падение загрязнения в направлениях границ территории.

Как известно, фильтрация выбросов остается эффективным средством понижения загрязняющей нагрузки территории. Если считать, что выбросы источников подвергаются фильтрации, то карты загрязнения меняются: в большей степени за счет интенсивностей выбросов, в меньшей — по конфигурациям облаков.

На рис. 8 (см. четвертую сторону обложки) показана карта загрязнения выборкой 1 источников при объявленном векторе фильтрации.

Фильтрация существенно ослабила загрязняющую нагрузку территории. Значение загрязнений в максимумах понизилось от ~1000 до ~300. В целом конфигурация общего облака сохранена. Характер распределения загрязнений практически выровнен. Общий усредненный коэффициент фильтрации составил 0,459. Фильтрация выборки 1 источников внесет корректив в общую карту загрязнения территории всеми источниками.

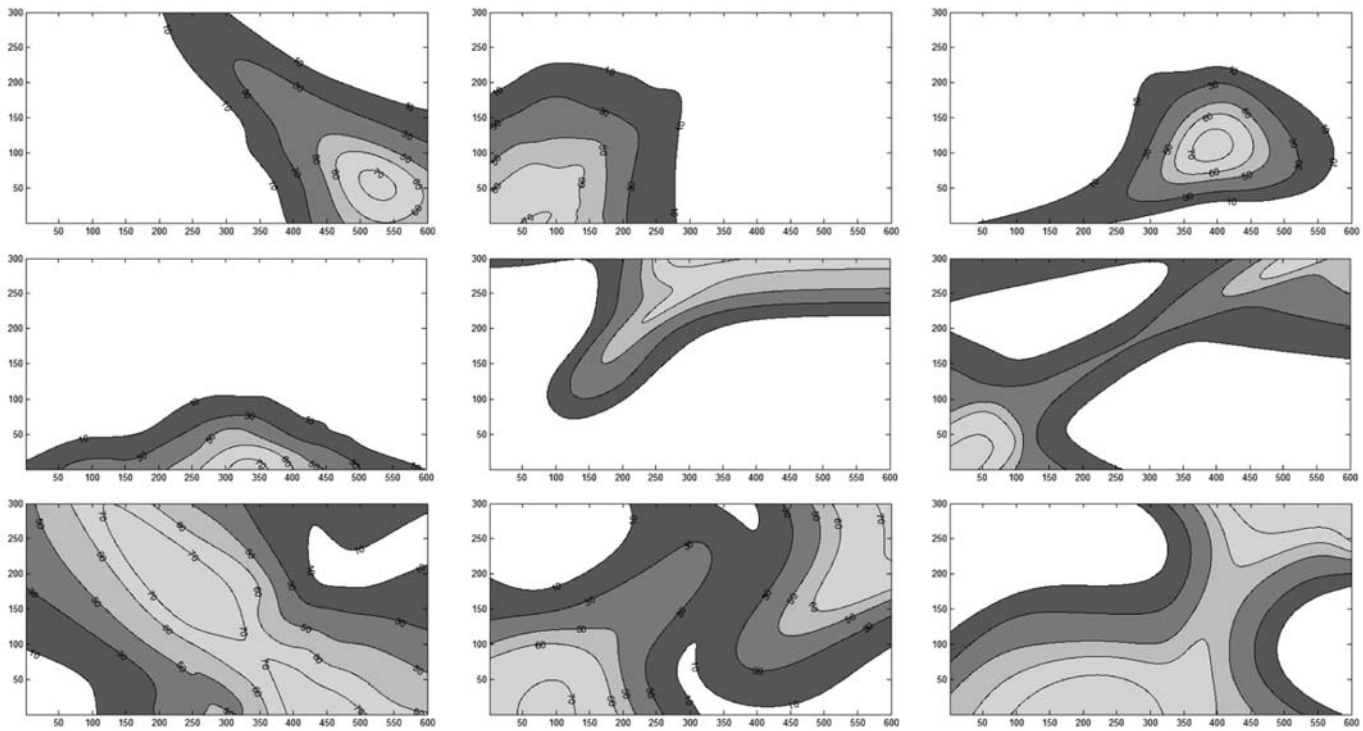


Рис. 2. Карты облаков выбросов источников 1—9

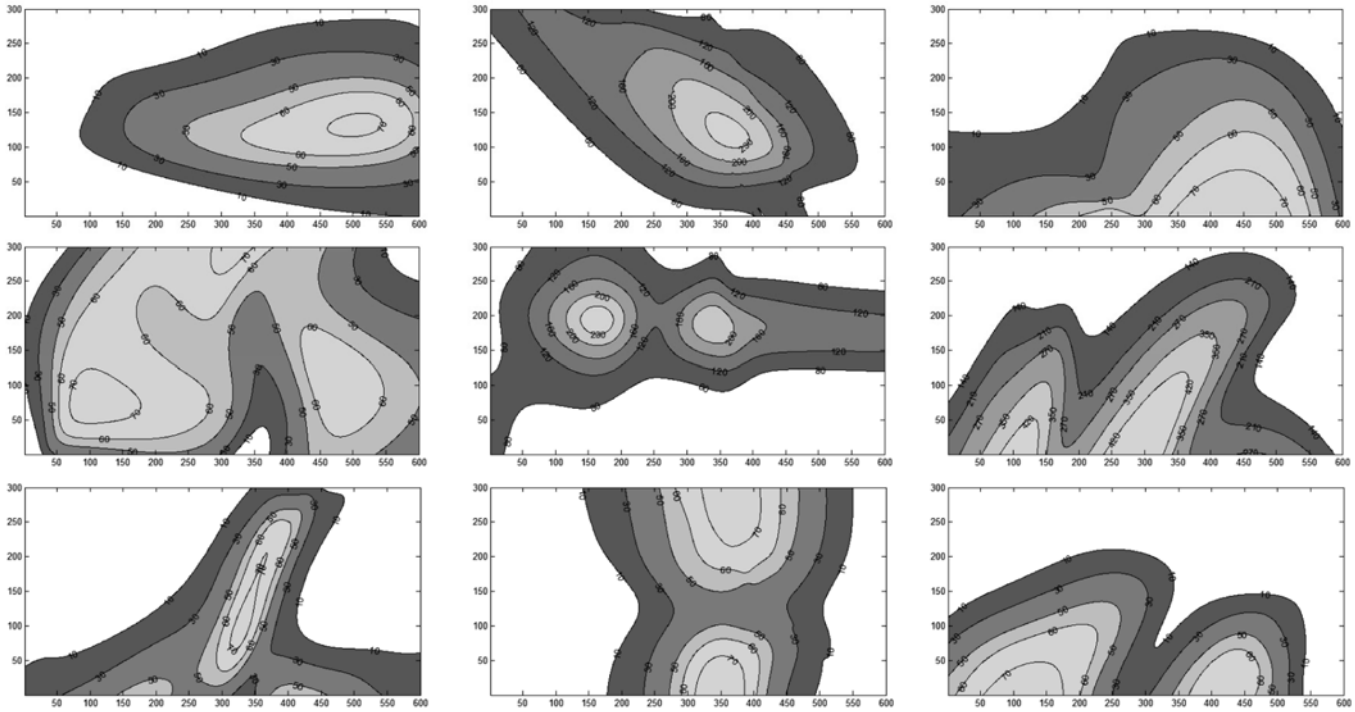


Рис. 3. Карты облаков выбросов источников 10—18

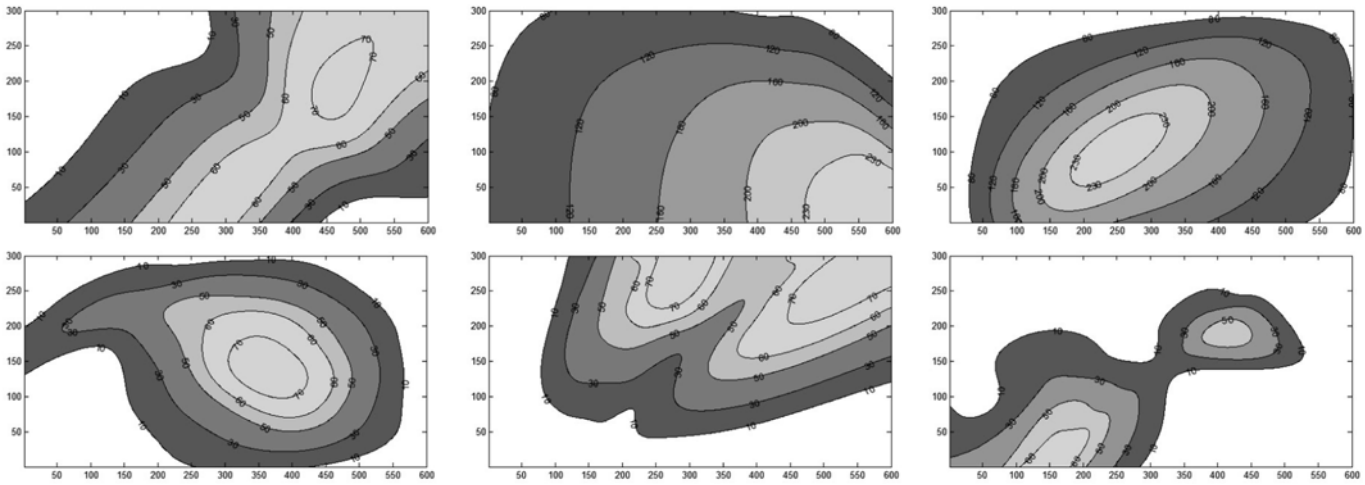


Рис. 4. Карты облаков выбросов источников 19—24

В приведенных примерах размеры изображений облаков выбросов источников совпадали с размерами территории. Естественно, это не обязательно. Если рассматривается превышающая по размерам территория, то перед суммированием облака отдельных источников можно относительно перемещать, вращать, делать частичные наложения и пр.

Полученные нейрофункции облаков выбросов позволяют в дальнейшем формировать многие критерии и оценки загрязнения частей или всей территории, имеющие различный технико-экономический смысл. На их основе могут проектироваться целевые функции или ограничения для задач оптимизации (линейное или нелинейное программирование).

Нейрофункции эффективны при описании простых, сложных и нестандартных облаков выбросов отдельных источников или групп. Уже то, что контурные графики будущих нейрофункций легко и бесконечно редактируемы, может стать решающим преимуществом перехода на нейрофункции.

Применение нейрофункций существенно дополняет аналитический подход новыми возможностями, совершенствующими моделирование в сложной области — расчета, оптимизации и визуализации разбросов загрязнений промышленными источниками.

Заключение

Введение нейрофункций взамен математического описания выбросов загрязнений источниками значительно сокращает входящую информационную базу моделирования, допускает прямое использование мониторинга или прогноза загрязнений и в целом упрощает графическое отображение облаков выбросов.

Нейрофункции эффективно выводят графику выборочного или общего загрязнения территории, позволяют учитывать необходимые дополнительные параметры (например переменную фильтрацию) выбросов источников и пригодны для прямых или оптимизационных расчетов загрязнений.

Список литературы

1. **Федосов В. В., Федосова А. В.** Полубесконечная модель фильтров многокомпонентных выбросов для группы промышленных источников территории // Безопасность жизнедеятельности. 2013. № 2. С. 42—46.

2. **Федосов В. В., Федосова А. В.** Моделирование контроля и ограничения промышленных выбросов при наличии ветровых смещений // Вестник компьютерных и информационных технологий. 2011. N. 9. С. 29—35.

3. **Дьяконов В. П., Круглов В. В.** MATLAB 6.5 SP1/7/7 SP1/7 SP2 + Simulink 5/6. Инструменты искусственного интеллекта и биоинформатики. М.: СОЛОН-ПРЕСС, 2006. 456 с.

V. V. Fedosov, Associate Professor, e-mail: vlr.fdsv@gmail.com, **A. V. Fedosova,**
Universidad Nacional de Colombia, e-mail: afedosova@unal.edu.co

The Use of Neural Functions for Graphical Evaluation Contamination Emissions Group Sources

Under the control of pollution of the territory, region or environment as a result of emissions from industrial sources, distributed monitoring, forecasts or estimates. Monitoring is necessary to spend, but few informative for understanding the relationship and management of complex systems. Projections should be based on a representative database, but the results are only probabilistic nature. Calculations pollution as complex systems that take into account a large number of variables that are to manage the impact on one or more of the evaluation criteria.

In the simulation of industrial ecology is a key functional description of the emission sources. His concept of the mathematical apparatus, the introduction of complications (simplications) determined by the objectives and tasks of the qualifications of the investigator. However, authoring mathematics is often subjective, is quite laborious and limited variability describe clouds emissions. Need a tool cloud formation emissions without mathematical formulas, but lets you embed cloud monitoring data and forecast pollution.

Such opportunities opened by the use of neural networks. This results in: the abolition of the author of mathematics; considerable simplification of the input database model; work directly with clouds emissions; automatically obtain the functional description emissions.

In this paper we propose a simple and fast way of learning neural function according cartoon clouds contour plots emissions. Shows examples of the formation of clouds emissions of different configurations. Neural function effectively graphics output selective or general pollution of the territory and are suitable for further calculation or optimization.

Keywords: *sources of industrial emissions, cloud emission, pollution area, contour plots, neural Networks, filtering emissions*

References

1. **Fedosov V. V., Fedosova A. V.** Polubeskonechnaja model' fil'trov mnogokomponentnyh vybrosov dlja gruppy promyshlennyh istochnikov territorii. *Bezopasnost' zhiznedejatel'nosti.* 2013. N. 2. P. 42—46.

2. **Fedosov V. V., Fedosova A. V.** Modelirovanie kontrolja i ogranichenija promyshlennyh vybrosov pri nalichii vetrovyh smeshhenij. *Vestnik komp'juternyh i informacionnyh tehnologij.* 2011. N. 9. P. 29—35.

3. **D'jakonov V. P., Kruglov V. V.** MATLAB 6.5 SP1/7/7 SP1/7 SP2 + Simulink 5/6. Instrumenty iskusstvennogo intellekta i bioinformatiki. M.: SOLON-PRESS, 2006. 456 p.

Адрес редакции:

107076, Москва, Стромынский пер., 4

Телефон редакции журнала **(499) 269-5510**

E-mail: it@novtex.ru

Технический редактор *Е. В. Конова.*

Корректор *Т. В. Пчелкина.*

Сдано в набор 08.12.2014. Подписано в печать 23.01.2015. Формат 60×88 1/8. Бумага офсетная.

Усл. печ. л. 8,86. Заказ IT215. 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".

119071, г. Москва, Ленинский пр-т, д. 19, стр. 1.