ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
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.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 7
N 11
2016

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 004.052 DOI: 10.17587/prin.7.483-489

В. Е. Гвоздев, проф., зав. каф., e-mail: wega55@mail.ru,
Д. В. Блинова, доц., e-mail: blinova.darya@gmail.com,
А. С. Давлиева, аспирант, e-mail: aliyasr21@gmail.com,
В. В. Тесленко, магистрант, e-mail: sltl@yandex.ru,
Уфимский государственный авиационный технический университет

Построение базовых моделей эффективности функционирования аппаратно-программных комплексов на основе методов математической статистики

Рассмотрен подход к построению базовых моделей оценки пользователем эффективности функционирования аппаратно-программных комплексов с учетом неопределенности характеристик компонентов системы, представляющий собой развитие аппарата марковского анализа. Обсуждена концептуальная основа постановки задачи формирования базовой модели с учетом статистической неопределенности компонентов системы, а также приведена формальная постановка задачи. Даны результаты вычислительного эксперимента, соответствующие случаю, когда статистическая неопределенность учитывается в форме законов распределения случайных величин.

Ключевые слова: *базовая модель, аппаратно-программный комплекс, дефект, статистическая неопределенность, эффективность функционирования объекта, граф состояний, методы математической статистики, интенсивность переходов*

Введение

Современная тенденция к повышению "интеллектуальности" систем обработки данных и управления, являющихся неотъемлемой частью сложных систем, выдвигает на первый план повышение внешней (т. е. с точки зрения пользователя) оценки эффективности функционирования аппаратно-программных комплексов [1, 2]. Соответствие фактических потребительских свойств объекта* базовым потребительским свойствам**, определенным в техническом задании, является основой оценивания функциональной пригодности объекта. Одной из форм концентрированного выражения функциональной пригодности является показатель внешней оценки эффективности функционирования объекта [3]. Отклонение фактических показателей внешней эффективности объекта от показателей, соответствующих базовым свойствам объекта, является индикатором наличия в объекте дефектов различной природы, не устраненных на стадиях тестирования и верификации [4—8].

Сложность формирования базовых моделей эффективностей функционирования аппаратно-программных комплексов (АПК) обусловлена следующими обстоятельствами:

- множеством режимов функционирования, в которых понятие "функциональная пригодность" имеет различное содержание (гетерогенность характеристик состояния в разных режимах функционирования);
- неопределенностью внутренних маршрутов преобразования исходных данных в программных продуктах, состоящих из большой совокупности взаимодействующих модулей;
- многомерностью пространства входных данных и случайным характером формирования наборов исходных данных;
- неопределенностью определения диапазонов допустимых значений входных данных, что обусловлено неопределенностью состояния внешней среды АПК;
- ресурсной неопределенностью, которая характеризуется временем обработки входного набора данных (временная неопределенность); объемом

* Содержание термина "объект" понимают в смысле, определенном в ГОСТ 51901.5—2005: аппаратное средство, программное обеспечение или то и другое.

** Под базовыми потребительскими свойствами понимают свойства, соответствующие базовой линии. Содержание понятия "базовая линия" понимают в смысле, определенном в ГОСТ Р ИСО/МЭК 15288—2005: спецификация или продукт, которые были официально рассмотрены и согласованы, чтобы впоследствии служить основой для дальнейшего развития.

памяти, необходимым для размещения программного комплекса (программная неопределенность); емкостью памяти, необходимой для накопления и хранения данных при выполнении программ (информационная неопределенность);

- неопределенностью характера взаимного влияния аппаратной и программной составляющих объекта при разных воздействиях внешней среды и различных режимах использования АПК.

Отмеченные особенности формирования базовых моделей эффективности функционирования, а также обязательное требование устойчивого функционирования объекта (иными словами — требование воспроизводимости внешнего поведения объекта, т. е. схожести отклика, а именно — результатов функционирования при схожих исходных данных) позволяет сделать следующее заключение. В качестве методологической основы формирования моделей внешней эффективности функционирования АПК следует использовать методы, дающие целостное описание объектов без учета его внутреннего устройства, и, в частности, методы математической статистики [9, 10].

В настоящей работе рассмотрена задача построения на основе метода математической статистики модели внешней эффективности функционирования АПК в предположении гомогенности характеристик состояния системы при разных режимах использования объекта, а также с учетом статистической неопределенности характеристик компонентов структуры.

Концептуальная основа и формальная постановка задачи

Одним из широко используемых инженерных инструментальных средств исследования функционально сложных систем (к числу которых принадлежит АПК) со сложными стратегиями обслуживания является аппарат марковского анализа. Различные приложения этого аппарата описаны, например, в работах [11—15]. Результаты анализа системы (финальные вероятности пребывания объекта в каждом из состояний), во-первых, допускают понятную содержательную интерпретацию качества функционирования (функциональной пригодности) объекта, во-вторых, создают основу для оценивания средней эффективности объекта на основе аддитивного критерия [12]. Одним из существенных ограничений упомянутого подхода к построению моделей объектов является предположение о том, что интенсивности переходов между состояниями, в которых может находиться система, являются постоянными, заранее определенными величинами.

В литературе, посвященной статистическому анализу свойств технических систем, давно обоснована необходимость изучения не только точечных, но и интервальных оценок характеристических признаков [13, 16, 17]. Значимость таких исследований обусловлена следующими причинами. Если свойства компонентов структуры определяются на основе выборочных данных (т. е. по результатам "жестких" измерений), то по причине конечного объема выборок, кроме точечных статистических оценок параметров, следует принимать во внимание также интервалы, внутри которых с заданной доверительной вероятностью могут находиться истинные значения параметров. Использование экспертных оценок ("мягких" измерений) является особенностью получения исходных данных на ранних стадиях проектирования, особенно в случае, когда речь идет об объектах, для которых отсутствуют аналоги. В случае "мягких" измерений широкое распространение получило использование интервальных оценок в разных формах [18—22].

Аналогичные рассуждения применимы к системам, представляющим собою неделимое единство аппаратной и программной составляющих, т. е. аппаратно-программным комплексам.

Из представленных выше аргументов следует, что при построении базовой модели эффективности функционирования объекта необходимо учитывать отсутствие определенности свойств компонентов системы.

Постановка задачи, подходы к решению которой представлены далее, включает следующие формализованные положения.

- Рассматривается система с гомогенными во времени характеристиками состояния, которой ставится в соответствие ориентированный граф с числом вершин n, причем k-й вершине соответствует показатель эффективности функционирования $\text{Э}_k \ (k = \overline{1; n})$.
- В общем случае элементы несимметричной относительно главной диагонали матрицы размером $n \times n$ $\|\lambda^{(0)}\|$ представляют начальные значения интенсивностей переходов между вершинами графа.
- Определяются границы интервалов $\{\lambda_{ij}^{(\text{H})}, \lambda_{ij}^{(\text{B})}\}$ значений интенсивностей переходов, начинающихся в i-м и заканчивающихся в j-м узлах, т. е. $\lambda_{ij} \in \left[\lambda_{ij}^{(\text{H})}, \lambda_{ij}^{(\text{B})}\right] \ (i, j = \overline{1; n})$. В отдельных случаях λ_{ij} может принимать значение нуль, что означает отсутствие отношения между i-й и j-й вершинами.
- Определяются границы интервалов $\{\text{Э}_k^{(\text{H})}, \text{Э}_k^{(\text{B})}\}$ возможных значений показателей эффективности системы в случае, когда ей соответствует k-я вершина графа V_k, т. е. $\text{Э}_k \in \left[\text{Э}_k^{(\text{H})}, \text{Э}_k^{(\text{B})}\right] \ (k = \overline{1; n})$.
- Определяется множество начальных значений показателей эффективности $\text{Э}_k^{(0)}, \ k = \overline{1; n}$.

Исходя из перечисленных выше положений формальной модели целевого АПК, необходимо построить базовую модель оценки внешней эффективности функционирования АПК с учетом интервальной неопределенности характеристик компонентов структуры.

При этом рассматривают следующие допущения.

- Описание компонентов графа дается одним из следующих способов:
  - ◇ способ А — точечные оценки $\text{Э}_k \ (k = \overline{1; n})$ и интервальные оценки $\lambda_{ij} \in \left[\lambda_{ij}^{(\text{H})}, \lambda_{ij}^{(\text{B})}\right] \ (i, j = \overline{1; n})$;

Рис. 1. Схема вычислительного эксперимента

- ◇ способ Б — интервальные оценки $Э_k \in \in \left[Э_k^{(H)}, Э_k^{(B)}\right]$ $\left(k = \overline{1;n}\right)$ и точечные оценки интенсивностей переходов, представленные в виде матрицы $\|\lambda^{(0)}\|$;
- ◇ способ В — интервальные оценки $Э_k \in \in \left[Э_k^{(H)}, Э_k^{(B)}\right]$ $\left(k = \overline{1;n}\right)$ и интервальные оценки $\lambda_{ij} \in \left[\lambda_{ij}^{(H)}, \lambda_{ij}^{(B)}\right]$ $\left(i, j = \overline{1;n}\right)$.

В основе исследований, связанных с решением задачи построения базовых моделей эффективности функционирования аппаратно-программных комплексов, лежит вычислительный эксперимент, схема которого представлена на рис. 1.

Пример построения базовой модели эффективности функционирования с учетом статистической неопределенности характеристик компонентов системы

Решение задачи проиллюстрируем следующим примером. Дан размеченный граф состояний (рис. 2)

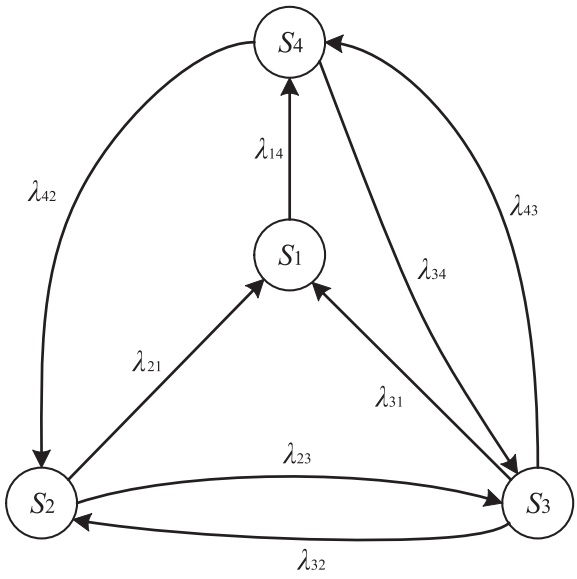


Рис. 2. Размеченный граф состояний:
S_i — i-е состояние системы

Таблица 1

Исходные данные

Задача	Интенсивности переходов							
	λ_{14}	λ_{21}	λ_{23}	λ_{31}	λ_{32}	λ_{34}	λ_{42}	λ_{43}
А	[0, 2]	[1, 3]	[2, 4]	[3, 5]	[4, 6]	[5, 7]	[6, 8]	[7, 9]
Б	1	2	3	4	5	6	7	8
В	[0, 2]	[1, 3]	[2, 4]	[3, 5]	[4, 6]	[5, 7]	[6, 8]	[7, 9]

Таблица 2

Задача	Показатели эффективности системы в k-м состоянии			
	$Э_1$	$Э_2$	$Э_3$	$Э_4$
А	1	2	3	4
Б	[0, 2]	[1, 3]	[2, 4]	[3, 5]
В	[0, 2]	[1, 3]	[2, 4]	[3, 5]

Требуется определить, как влияет способ задания характеристик компонентов системы (точечные/статистические) на значения статистических характеристик базовой модели внешней эффективности функционирования.

В ходе исследований исходные данные подбирались следующим способом: $\lambda_{ij}^{(0)}$ принималось равным середине интервала $\left[\lambda_{ij}^{(\text{H})}, \lambda_{ij}^{(\text{B})}\right]$; $Э_k^{(0)}$ принималось равным середине интервала $\left[Э_k^{(\text{H})}, Э_k^{(\text{B})}\right]$. Законы распределения $f(Э_k)$, $f(\lambda_{ij})$ полагались равномерными либо нормальными.

В табл. 1, 2 приведены исходные данные, соответствующие задачам А, Б, В.

В табл. 3 приведены статистические характеристики базовых моделей, соответствующих каждой из задач.

На рис. 3 приведены оценки дифференциальных функций распределения, соответствующие задачам А, Б, В, при равномерных законах распределения $f(\lambda_{ij})$, $f(Э_k)$. Число интервалов группирования при построении оценок определялось по правилу Старджесса: $n = 1 + 3{,}3\lg(\eta)$.

Заметим, что значение внешней эффективности базовой модели, соответствующее значениям $Э_k^{(0)}$, $\lambda_{ij}^{(0)}$

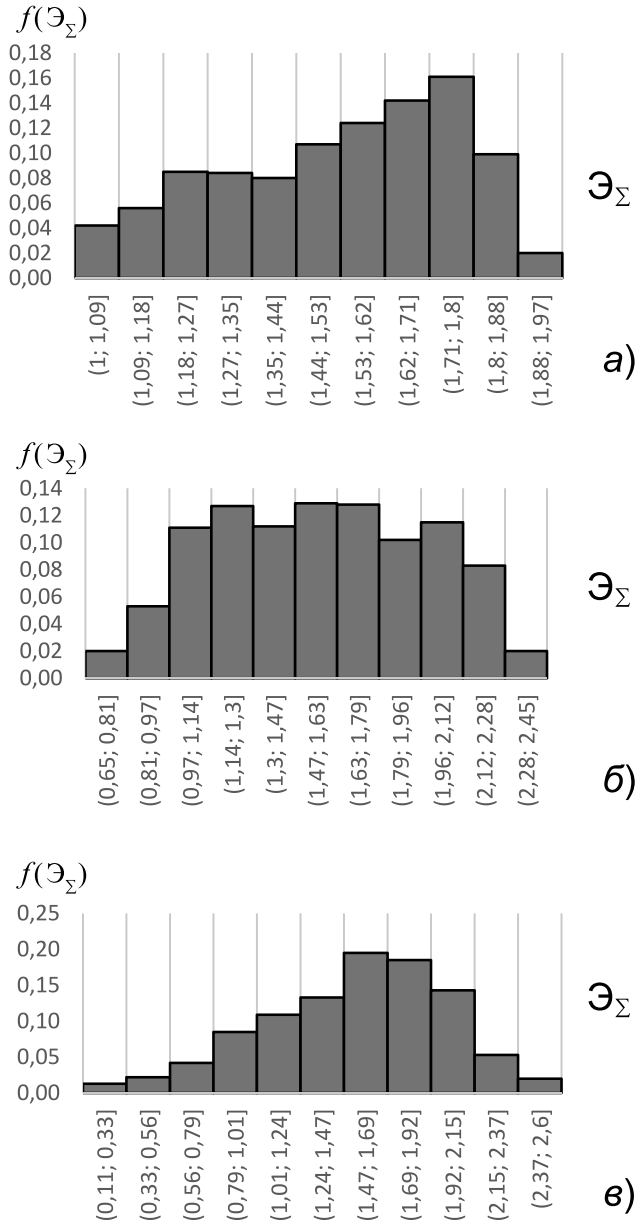


Рис. 3. Оценки дифференциальных функций распределения внешнего показателя эффективности при равномерных законах распределения $f(Э_k)$, $f(\lambda_{ij})$:
а — задача А; *б* — задача Б; *в* — задача В

Таблица 3

Статистические характеристики базовой модели в случае равномерных законов распределения $f(Э_k)$, $f(\lambda_{ij})$

Задача	Оценка математического ожидания $\widehat{M}[Э_k]$	Оценка среднеквадратического отклонения $\hat{\sigma}[Э_k]$	Оценка коэффициента асимметрии $\widehat{As}[Э_k]$	Оценка коэффициента эксцесса $\widehat{Ex}[Э_k]$	Оценка энтропии по Шеннону $\widehat{H}[Э_k]$
А	1,535	0,237	−0,486	−0,793	0,068
Б	1,566	0,412	−0,035	−1,038	0,080
В	1,528	0,474	−0,454	−0,229	0,082

Таблица 4

Статистические характеристики базовой модели в случае нормального закона распределения $f(Э_k)$, $f(\lambda_{ij})$

Задача	Оценка математического ожидания $\widehat{M}[Э_k]$	Оценка среднеквадратического отклонения $\hat{\sigma}[Э_k]$	Оценка коэффициента асимметрии $\widehat{As}[Э_k]$	Оценка коэффициента эксцесса $\widehat{Ex}[Э_k]$	Оценка энтропии по Шеннону $\widehat{H}[Э_k]$
А	1,554	0,098	−0,539	0,733	0,062
Б	1,556	0,181	−0,018	−0,001	0,078
В	1,555	0,201	−0,286	0,252	0,072

$(i, j = \overline{1; 4})$, для приведенного примера составило $Э^{(0)} = 1{,}559$.

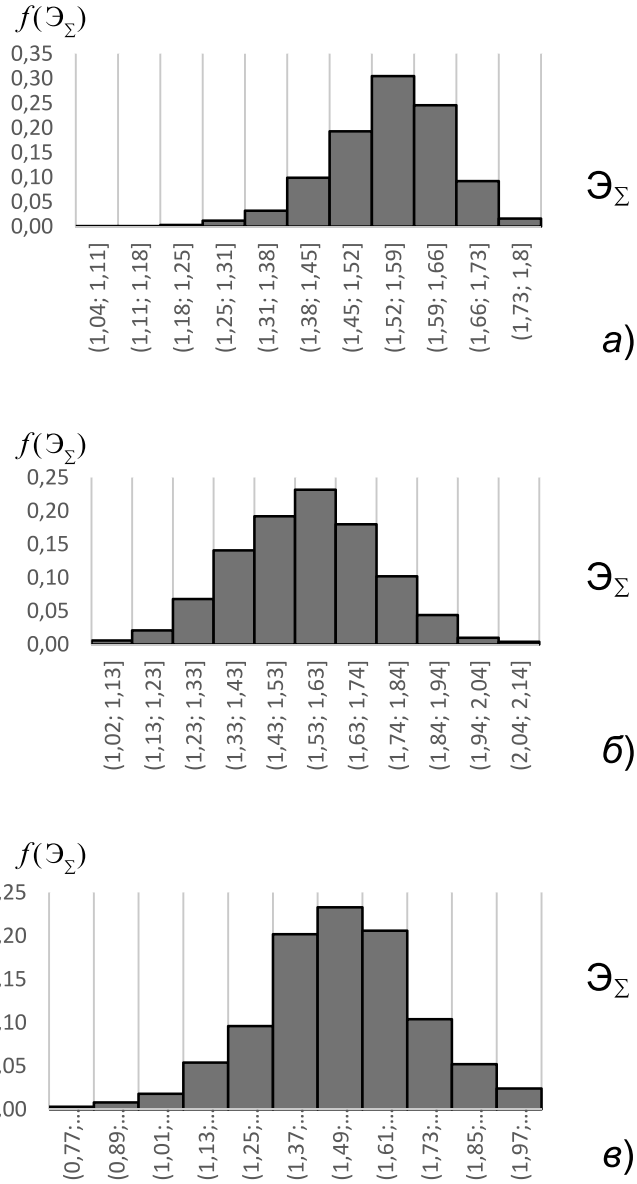


Рис. 4. Оценки дифференциальных функций распределения внешнего показателя эффективности при нормальных законах распределения случайных величин $f(Э_k)$, $f(\lambda_{ij})$:
а — задача А; *б* — задача Б; *в* — задача В

В табл. 4 приведены статистические характеристики базовых моделей, соответствующие задачам А, Б, В, в случае, когда функции распределения $f(Э_k)$, $f(\lambda_{ij})$ полагались нормальными. Параметры $N(x, M, s)$ (где $N(x, M, s)$ соответствует известной нотации нормального закона распределения случайной величины x с математическим ожиданием M и средним квадратическим отклонением s) определяли следующим образом. Значение математического ожидания M принимали равным середине интервала возможных значений случайной величины, значение s выбирали из условия $s = \left(\Pi^{(В)} - \Pi^{(Н)}\right)/8$, где $\Pi^{(В)}$ — левая граница возможных значений случайной величины, $\Pi^{(Н)}$ — соответственно нижняя граница.

На рис. 4 приведены оценки плотности распределения $f\left(Э_\Sigma^{(\xi)}\right)$, соответствующие решению задач А, Б, В при нормальных законах распределения $f(Э_k)$, $f(\lambda_{ij})$.

С использованием оценки дифференциальных функций распределения $f(Э_\Sigma)$, соответствующих различным способам задания характеристик

Таблица 5

Интервальные оценки вероятности показателя внешней эффективности функционирования

l	Задача	Тип закона распределения $f(Э_k)$, $f(\lambda_{ij})$	$P\left(Э_\Sigma^{(Н)} < Э_\Sigma < Э_\Sigma^{(В)}\right)$
1	А	Равномерный	0,444
		Нормальный	0,423
	Б	Равномерный	0,388
		Нормальный	0,714
	В	Равномерный	0,545
		Нормальный	0,62
2	А	Равномерный	0,852
		Нормальный	0,912
	Б	Равномерный	0,811
		Нормальный	0,961
	В	Равномерный	0,903
		Нормальный	0,936

компонентов системы, определялись оценки вероятности попадания показателей внешней эффективности $P(Э_\Sigma)$ в интервал, границы которого формировались по следующим правилам:

$$\hat{Э}_\Sigma^{(н)} = \hat{M}[Э_\Sigma] - l\hat{\sigma}[Э_\Sigma];$$

$$\hat{Э}_\Sigma^{(в)} = \hat{M}[Э_\Sigma] + l\hat{\sigma}[Э_\Sigma],$$

где l служило для расчета доверительного интервала и принимало значение 1 и 2.

Результаты расчетов приведены в табл. 5.

Обсуждение результатов моделирования

По результатам приведенного эксперимента можно заключить, что разброс характеристик узлов графа при прочих равных условиях в большей степени влияет на статистическую устойчивость показателей базовой модели внешней эффективности (увеличивается оценка среднеквадратического отклонения $\hat{\sigma}[Э_\Sigma]$). Учет разброса характеристик интенсивностей переходов в большей степени влияет на отклонение формы закона распределения от симметричного, причем знак коэффициента асимметрии указывает на то, что значения $Э_\Sigma$ бóльшие, чем математические ожидания $M[Э_\Sigma]$, появляются чаще, нежели значения меньшие, чем математическое ожидание.

Сравнение расчетов, соответствующих разным типам законов распределения $f(Э_k)$, $f(\lambda_{ij})$, позволяет заключить, что уменьшение неопределенности характеристик $Э_k$, λ_{ij} за счет учета центра группирования в виде $M[Э_k]$, $M[\lambda_{ij}]$ при использовании нормального закона распределения отразилось на характеристике неопределенности $H[Э_\Sigma]$. Получаемые оценки стали более устойчивыми. Уменьшились характеристика эксцесса, а также значения $\hat{H}[Э_\Sigma]$, что свидетельствует об уменьшении неопределенности оценок аддитивного показателя внешней эффективности.

На основе проведенных исследований предлагается при формировании технического задания для рассматриваемой системы указывать значения показателя эффективности не 1,56, а использовать диапазон, в котором может находиться показатель эффективности 1,45...1,65.

Заключение

Одним из основных положений разработки спецификации требований к АПК является описание ожидаемых потребительских свойств объекта (базовой модели объекта), а также определение ограничений на показатели эффективности функционирования объектов, которые могут находиться в разных состояниях. В силу целого ряда обстоятельств при формировании базовой модели эффективности функционирования необходимо учитывать неопределенность, присущую компонентам системы. В работе рассмотрен один из возможных способов построения базовых моделей эффективности функционирования АПК на основе методов математической статистики. Предлагаемый способ отличается от описанного в нормативных документах метода исследования посредством аппарата марковского анализа систем, пребывающих в разных состояниях, тем, что характеристики эффективности функционирования, соответствующие разным состояниям объекта, а также интенсивности переходов между состояниями представляются в виде статистических оценок. Это делает возможным дать статистическое описание свойств базовых моделей эффективности функционирования при различных комбинациях свойств компонентов системы.

Работа выполнена при поддержке гранта Российского фонда фундаментальных исследований № 16-08-00442.

Список литературы

1. **Рожников В. А.** Психология программирования: цели, проблемы, перспективы // Общество: социология, психология, педагогика. 2014. № 3. С. 18—21.
2. **Dobson S.** Applications considered harmful for ambient systems // Proceeding ISICT '03 — Proceedings of the 1st international symposium on Information and communication technologies. 2003. P. 163—168.
3. **Гвоздев В. Е., Бежаева О. Я.** Принципы и подходы к проектированию требований к программным продуктам и программным проектам // Труды XVI международной конференции "Проблемы управления и моделирования в сложных системах" (30 июня — 03 июля 2014 г.). Самара, 2014. С. 196—203.
4. **Липаев В. В.** Функциональная безопасность программных средств. Москва: СИНТЕГ, 2004. 348 с.
5. **Липаев В. В**. Анализ и сокращение рисков проектов сложных программных средств. М.: СИНТЕГ, 2005. 224 с.
6. **Майерс Г.** Надежность программного обеспечения. М.: Мир, 1980. 360 с.
7. **Черкесов Г. Н.** Надежность аппаратно-программных комплексов. СПб.: Питер, 2005. 479 с.
8. **ESA-PSS-05-10** Software Engineering Standards, software verification and validation.
9. **Каган А. М., Линник Ю. В., Рао С. Р.** Характеризационные задачи математической статистики. М.: Наука, 1972. 656 с.
10. **Нейман Ю.** Вводный курс в теорию вероятности математической статистики. М.: Наука, 1968. 448 с.
11. **Бусленко Н. П.** Моделирование сложных систем. М.: Наука, 1978. 399 с.
12. **Вентцель Е. С.** Исследование операций: задачи, принципы, методология. 2-е изд. М.: Наука, 1988. 208 с.
13. **Вентцель Е. С.** Исследование операций. М.: Советское радио, 1972. 552 с.
14. **ГОСТ 51901.5—2005** Менеджмент риска. Руководство по применению методов анализа надежности.
15. **ГОСТ 51901.15—2005** Менеджмент риска. Применение марковских методов.
16. **Смирнов Н. В., Дунин-Барковский И. В.** Курс теории вероятностей и математической статистики для технических приложений. М.: Наука, 1969. 512 с.
17. **Дружинин Г. В.** Надежность автоматизированных систем. М.: Энергия, 1977. 536 с.
18. **Зак Ю. А.** Принятие решений в условиях нечетких и размытых данных: Fuzzy-технологии. М.: Книжный дом "ЛИБРОКОМ", 2013. 352 с.
19. **Pelaez C., Bowles J.** Using fuzzy cognitive maps as a system model for failure models and effect analysis // Information Sciences. 1996. Vol. 88. P. 177—199.
20. **Гвоздев В. Е., Ровнейко Н. И.** Численное моделирование процессов формирования требований к программному продукту // Вестник УГАТУ. 2012. Том 16. № 8 (53). С. 52—60.
21. **Гвоздев В. Е., Ровнейко Н. И.** Информационная поддержка анализа требований к программным проектам и продуктам на основе мер неопределенности // Управление проектами и программами. 2013. № 3 (35). С. 198—217.
22. **Гвоздев В. Е., Блинова Д. В., Ровнейко Н. И., Ямалова О. П.** Системное и структурное моделирование требований к программным продуктам и проектам // Программная инженерия. 2013. № 5. С. 2—10.

Construction of Basic Functioning Efficiency Models of the Hardware-Software Complexes, Based on the Mathematical Statistics Methods

V. E. Gvozdev, wega55@mail.ru, **D. V. Blinova,** blinova.darya@gmail.com, **A. S. Davlieva,** aliyasr21@gmail.com, **V. V. Teslenko,** sltl@yandex.ru, Ufa State Aviation Technical University, Ufa, 450000, Russian Federation

Corresponding author:

Gvozdev Vladimir E., Professor, Ufa State Aviation Technical University, 450000, Ufa, Russian Federation, e-mail: wega55@mail.ru

Received on July 01, 2016
Accepted on August 31, 2016

An approach to the basic models construction of the functioning efficiency hardware and software systems is considered, taking into account the uncertainty of system components characteristics, which is a development of the well-known Markov analysis apparatus. We discuss the conceptual statement basis of the basic model formation problem, taking into account the statistical uncertainty of the system components, and provide a formal problem statement. The results of computational experiments are provided, corresponding to the case when the statistical uncertainty is taken into account in the form of random variables distribution laws.

Keywords: *base model, the hardware-software complex, defect, statistical uncertainty, the functioning efficiency of the object, state graph, mathematical statistics method, the transitions intensity*

Acknowledgements: *This work was supported by the Russian Foundation for Basic Research, project nos. 16-08-00442*

For citation:

Gvozdev V. E., Blinova D. V., Davlieva A. S., Teslenko V. V. Construction of Basic Functioning Efficiency Models of the Hardware-Software Complexes, Based on the Mathematical Statistics Methods, *Programmnaya Ingeneria,* 2016, vol. 7, no 11, pp. 483—489.

DOI: 10.17587/prin.7.483-489

References

1. **Rozhnikov V. A.** Psihologija programmirovanija: celi, problemy, perspektivy (Programming Psychology: Goals, Challenges, Prospects), *Obshhesmtvo: sociologija, psihologija, pedagogika*, 2014, no. 3, pp. 18—21 (in Russian).
2. **Dobson S.** Applications considered harmful for ambient systems, *Proceeding ISICT '03 — Proceedings of the 1st international symposium on Information and communication technologies,* 2003, pp. 163—168.
3. **Gvozdev V. E., Bezhaeva O. Ja.** Principy i podhody k proektirovaniju trebovanij k programmnym produktam i programmnym proektam (The principles and approaches to the design requirements for software products and software), *Trudy XVI mezhdunarodnoj konferencii "Problemy upravlenija i modelirovanija v slozhnyh sistemah"*, 30 june — 03july 2014, Samara, 2014, pp. 196—203 (in Russian).
4. **Lipaev V. V.** *Funkcional'naja bezopasnost' programmnyh sredstv* (Functional safety software), Moscow, SINTEG, 2004, 348 p. (in Russian).
5. **Lipaev V. V.** *Analiz i sokrashhenie riskov proektov slozhnyh programmnyh sredstv* (Analysis and reduction of risks of complex software projects), Moscow, SINTEG, 2005, 224 p. (in Russian).
6. **Majers G.** *Nadezhnost' programmnogo obespechenija* (Software Reliability), Moscow, Mir, 1980, 360 p. (in Russian).
7. **Cherkesov G. N.** *Nadezhnost' apparatno-programmnyh kompleksov* (The reliability of software and hardware), Saint Petersburg, Piter, 2005, 479 p. (in Russian).
8. **ESA-PSS-05-10** Software Engineering Standards, software verification and validation.
9. **Kagan A. M., Linnik U. V., Rao S. R.** *Haracterizacionnie zadachi mathematicheskoi statistiki* (Characterization problems of mathematical statistics), Moscow, Nauka, 1972, 656 p. (in Russian).
10. **Neiman U.** *Vvodnii kurs v theoriu veroyatnosti mathematicheskoi statistiki* (Introductory course in probability theory mathematical statistics), Moscow, Nauka, 1968, 448 p. (in Russian).
11. **Buslenko N. P.** *Modelirovanie slozhnyh system* (Simulation of complex systems), Moscow, Nauka, 1978, 399 p. (in Russian).
12. **Ventcel E. S.** *Issledovanie operacij: zadachi, principy, metodologija*, 2-e izd. (Operations research: objectives, principles, methodology. 2nd ed), Moscow, Nauka, 1988, 208 p. (in Russian).
13. **Ventcel E. S.** *Issledovanie operacij* (Operations research), Moscow, Sovetskoe radio, 1972, 552 p (in Russian).
14. **GOST 51901.5—2005** Menedzhment riska. Rukovodstvo po primeneniju metodov analiza nadezhnosti (GOST 51901.5—2005 Risk management. On the application of reliability analysis methods Guide) (in Russian).
15. **GOST 51901.15—2005** Menedzhment riska. Primenenie markovskih metodov (GOST 51901.15—2005 risk management. Application of Markov techniques) (in Russian).
16. **Smirnov N. V., Dunin-Barkovskij I. V.** *Kurs teorii verojatnostej i matematicheskoj statistiki dlja tehnicheskih prilozhenij* (The course of the theory of probability and mathematical statistics for technical applications), Moscow, Nauka, 1969, 512 p. (in Russian).
17. **Druzhinin G. V.** *Nadezhnost' avtomatizirovannyh system* (The reliability of automated systems), Moscow, Energia, 1977, 536 p. (in Russian).
18. **Zak Ju.A.** *Prinjatie reshenij v uslovijah nechetkih i razmytyh dannyh: Fuzzy-tehnologii* (Decision-making in a fuzzy and blurry data: Fuzzy-technology), Moscow, LIBROKOM, 2013, 352 p. (in Russian).
19. **Pelaez C., Bowles J.** Using fuzzy cognitive maps as a system model for failure models and effect analysis, *Information Sciences*, 1996, vol. 88, pp. 177—199.
20. **Gvozdev V. E., Rovnejko N. I.** Chislennoe modelirovanie processov formirovanija trebovanij k programmnomu produktu (Numerical modeling of formation of requirements for software product), *Vestnik UGATU*, 2012, vol. 16, no. 8 (53), pp. 52—60 (in Russian).
21. **Gvozdev V. E., Rovnejko N. I.** Informacionnaja podderzhka analiza trebovanij k programmnym proektam i produktam na osnove mer neopredelennosti (Information support of the analysis of requirements for software projects and products based on the uncertainty of measures), *Upravlenie proektami i programmami*, 2013, no. 3 (35), pp. 198—217 (in Russian).
22. **Gvozdev V. E., Blinova D. V., Rovnejko N. I., Jamalova O. P.** Sistemnoe i strukturnoe modelirovanie trebovanij k programmnym produktam i proektam (Systemic and structural modeling requirements for software products and projects*), Programmnaya Inzheneria*, 2013, no. 5, pp. 2—10 (in Russian).

УДК 004.738 DOI: 10.17587/prin.7.490-497

Р. Э. Асратян, канд. техн. наук, вед. науч. сотр., e-mail: rea@ipu.ru,
Институт проблем управления им. В. А. Трапезникова РАН, г. Москва

Интернет-служба защищенной обработки информационных запросов в распределенных системах

Рассмотрены принципы организации сетевой службы, предназначенной для реализации защищенной обработки информационных запросов в распределенных информационных системах. Отличительными особенностями службы являются тесная интеграция функций аутентификации и защиты данных с функциями сетевого информационного взаимодействия, а также возможность отладки как клиентских, так и серверных компонентов системы вне сетевой среды.

Ключевые слова: *распределенные системы, web-технологии, интернет-технологии, информационное взаимодействие, информационная безопасность*

Введение

Уже более десяти лет сетевая архитектура .NET и технология web-сервисов занимают ведущее положение в разработках распределенных информационных систем. Решающую роль здесь играет мощная и многоязыковая поддержка этой архитектуры в целом ряде современных систем программирования (MS Visual Studio, Borland C++ builder, Oracle developer suit и т. п.), а также web-серверов (IIS, Apache, Oracle HTTP server). Этому же способствуют и удачные сетевые стандарты WSDL и SOAP, сопровождающие жизненный цикл сетевых приложений от разработки до эксплуатации и обеспечивающие большую гибкость в спецификациях методов (функций-членов) электронных сервисов [1, 2].

Тем не менее технология web-сервисов не свободна от недостатков, которые зачастую создают трудности для разработчиков распределенных систем. Как следствие, в последние годы появилась тенденция поиска альтернативных решений. В качестве основных причин можно назвать:

- отсутствие в архитектуре .NET встроенных средств защиты и аутентификации сетевых сообщений;
- отсутствие средств отладки клиентских и сервисных компонентов системы (и их взаимодействия) вне сетевой среды и web-сервера;
- возрождение интереса к традиционной HTTP-модели для классического взаимодействия на основе обмена сообщениями вместо характерной для архитектуры .NET модели, основанной на вызове методов удаленных объектов.

Рассмотрим ситуацию, в которой клиент посылает сложный (включающий десятки параметров) информационный запрос электронному сервису и получает в результате текстовый документ отчета, к которому могут быть приложены один или несколько графических документов. С ростом сложности информационных запросов разработчики все чаще выбирают формат XML-документа для представления как параметров запроса, так и результатов его выполнения. Модель обмена текстовыми и двоичными сообщениями представляется в данном случае более простой и адекватной, чем модель вызова удаленных методов через SOAP. Во всяком случае, первая из отмеченных моделей позволяет избежать явной избыточности, связанной с упаковкой XML-документов в объемлющую SOAP-структуру. Такая избыточность выражается, например, в перекодировке всех символов "<" в последовательности "<" и т. п. в случае передачи документа в форме символьной строки, а при наличии двоичных приложений к документу — в обязательном применении текстовой кодировки base64. Вместе с тем появились признаки того, что компактность представления данных в сети вновь стала заботить разработчиков (например, в относительно новом сетевом протоколе WebSocket [3] сделан акцент на экономию сетевого трафика, в том числе за счет возможности передачи данных в двоичной форме).

Основной метод организации защиты и аутентификации данных в архитектуре .NET, не требующий дополнительного программирования, основан на использовании протокола HTTPS в качестве "транспорта" для передачи SOAP-сообщений через сеть. Однако этот метод основан скорее на создании защищенных каналов взаимодействия (с помощью технологии SSL/TLS [4]), чем на защите отдельных сообщений. Поэтому его использование сопряжено с рядом важных ограничений. К их числу относится отсутствие возможности анализа и регистрации

сообщений (информационных запросов и ответов) вместе с удостоверяющими их электронными цифровыми подписями (ЭЦП) или возможности использования нескольких ЭЦП (например, исполнителя и руководителя) для одного сообщения и т. п. [5]. Это существенно усложняет задачу разработчика, так как внимание к проблемам информационной безопасности непрерывно возрастает [6, 7].

В настоящей работе рассмотрены принципы организации новой сетевой службы PMS (Protected Message Service), разработка которой стала попыткой преодоления перечисленных выше недостатков. Суть подхода заключается в тесной интеграции функций сетевого информационного обмена с функциями защиты и аутентификации данных. Внешне эта интеграция проявляется в том, что отмеченные функции входят в набор методов главного класса службы — класса "защищенное сообщение", отображающего электронный документ (информационный запрос или ответ), снабженный одной или несколькими удостоверяющими ЭЦП. В отличие от технологии web-сервисов описываемая служба опирается не на модель вызова методов удаленных объектов, а на модель обмена сообщениями. В данном случае это означает, что все сервисные обрабатывающие функции (методы) имеют одинаковую, жесткую спецификацию: они получают объект класса "защищенное сообщение" в качестве параметра и возвращают объект того же класса. Эти обрабатывающие функции группируются в одну или несколько динамических библиотек, которые подключаются к серверу PMS в момент его запуска (каждая библиотека может рассматриваться как отдаленный аналог web-сервиса в .NET), и становятся доступными для клиентских компонентов. Web-технология (HTTP, HTTPS, SOAP, WSDL и т. п.), так же как и web-серверы, вообще не используется.

Два коротких примера

Как и всякая сетевая служба, основанная на базовом сетевом протоколе TCP/IP [8], PMS поддержана клиентским и серверным программным обеспечением. Сервер PMS представляет собой постоянно активную программу, обслуживающую запросы на обработку от клиентов (по умолчанию используется порт 8132). Клиентское программное обеспечение представляет собой библиотеку функций PmsBase.dll, реализующих прикладной программный интерфейс (API) к PMS. Этот интерфейс является "лицом" PMS с точки зрения пользователя. Приведенные далее два коротких примера его использования позволят сразу же получить представление о нем.

На рис. 1 представлен фрагмент кода в нотации C#, иллюстрирующий простое обращение к обрабатывающей функции без применения средств защиты данных. Две первые строки кода определяют две переменные типа PmsMessage, представляющего собой главный класс PMS (защищенное сообщение). Первой переменной (Request) присваивается

```
PmsMessage Request =  new PmsMessage("<request> … </request>");
PmsMessage Reply;
PmsConnection MyConn = new PmsConnection();
MyConn.Connect("MyServer");
Reply=Request.Process (MyConn, "MyLib.MyFunc");
if(Reply != null)
    Console.WriteLine (Reply.GetString());
else
    Console.WriteLine ("Ошибка: " +MyConn.ErrMsg);
MyConn.Disconnect();
```

Рис. 1. Пример использования PMS без защиты данных

значение "объект, инициированный символьной строкой" (например, содержащей XML-документ информационного запроса). Вторая переменная (Reply) предназначена для хранения результата обработки. В третьей строке определяется и инициируется переменная класса PmsConnection, предназначенного для создания и прекращения сетевого соединения с сервером.

Собственно вызов обрабатывающей функции выполняется в четвертой и пятой строках. Сначала устанавливается соединение с сервером с помощью метода Connect (с указанием сетевого имени или адреса), а затем выполняется вызов удаленной функции с помощью применения метода Process к переменной Request на основе данного соединения. Предполагается, что к серверу PMS подключена библиотека MyLib.dll, содержащая код функции MyFunc.

В последующих строках кода осуществляется запись результата обращения в стандартный вывод (предполагается, что результат, как и запрос, имеет форму символьной строки). Пример заканчивается закрытием соединения с сервером с помощью метода Disconnect, так как в данном примере оно больше не нужно.

Рассмотрим другой пример кода, который выполняет обращение к той же самой сервисной функции, но с использованием средств защиты (рис. 2). Легко видеть, что второй пример кода получен путем

```
PmsMessage Request = new PmsMessage("<request> … </request>");
PmsMessage Reply;
PmsConnection MyConn = new PmsConnection();
PmsCertList  SenderCerts = PmsCertList( new string [] {"Иванов", "Петров"});
PmsCertList  ReceiverCert = PmsCertList( new string [] {"Admin"});
Request.AddSignatures(SenderCerts);
MyConn.Connect("MyServer");
Reply=Request.Process (MyConn, "MyLib.MyFunc", ReceiverCert);
if(Reply != null)
{
    string Signer;
    for(int i=0; Reply.Signatures.Length; i++)
        if(Reply.VerifySignature(i, out Signer) >= 0)
            Console.WriteLine ("Ответ подписал: " + Signer);
    Console.WriteLine (Reply.GetString());
}
else
    Console.WriteLine ("Ошибка: " +MyConn.ErrMsg);
MyConn.Disconnect();
```

Рис. 2. Пример использования PMS с защитой данных

добавления к первому нескольких строк, которые выделены серым фоном. Первые две из добавленных строк определяют две новые переменные класса PmsCertList. Этот класс предназначен для хранения в памяти списков сертификатов с открытыми ключами в стандарте X509 и содержит несколько конструкторов для загрузки сертификатов из файлов или из системных хранилищ с поиском по имени владельца или по серийному номеру. В переменную SenderCerts загружаются два сертификата, соответствующих именам владельцев "Иванов" и "Петров", для последующего формирования двух ЭЦП в запросе. Такой способ использования означает, что с этими сертификатами (вернее, с содержащимися в них открытыми ключами) обязательно должны быть связаны парные им закрытые ключи, иначе формирование ЭЦП закончится неудачно. В переменную ReceiverCert загружается один сертификат для выполнения шифрования.

Формирование ЭЦП выполняется путем применения метода AddSignatures к переменной Request, а шифрование данных перед передачей в сеть — всего лишь путем добавления параметра в вызов метода Process. Последующие добавленные строки обеспечивают последовательную проверку всех ЭЦП в полученном ответе сервера и запись в стандартный вывод сведений о подписантах.

Необходимо отметить, что из обоих фрагментов кода намеренно удалены операторы обработки исключений.

Принципы организации PMS

Как видно из приведенных выше примеров, функциональные возможности PMS на стороне клиента основаны на трех основных классах: PmsMessage (защищенное сообщение), PmsConnection (соединение с сервером PMS) и PmsCertList (список сертификатов). Общая особенность всех трех классов заключается в том, что они содержат очень мало элементов данных (полей) и сравнительно много функций-членов (методов). В табл. 1 приведены основные элементы данных в нотации C#.

Организация сервера PMS основана на следующих принципах.

- Сервер PMS основан на многопотоковой обработке: для обслуживания каждого информационного запроса создается отдельная обрабатывающая программная нить (поток), в рамках которой организуется выполнение вызванной обрабатывающей функции. Сервер берет на себя всю работу, связанную с подачей полученного по сети сообщения на вход функции, отправкой результатов ее выполнения в сеть и отслеживанием таймаута выполнения.

Таблица 1

Основные программные классы PMS

Класс	Основные элементы данных	Описание
PmsMessage	public byte [] Data; public Signature [] Signatures	Член Data представляет собой "контейнер" данных в форме массива байтов. В классе имеется множество методов для загрузки в член Data (и обратно) символьных строк, массивов строк, наборов файлов (вместе с их названиями) и т. п. Член Signatures представляет собой массив электронных подписей — объектов класса Signature. Каждый такой объект содержит свертку (digest) массива Data, зашифрованную закрытым ключом подписанта, и сертификат структуры X509, содержащий открытый ключ. Важно отметить, что пользователю совершенно необязательно знать устройство этого класса. Методы класса данных позволяют добавлять новые подписи, проверять и удалять их, а также отправлять данные на обработку в сервер PMS с возможностью шифрования запроса и дешифровки результата обработки
PmsConnection	Socket msock;@string Host; int Port; public string ErrStr	Основным членом класса является msock, в котором хранятся дескриптор сетевого соединения и ряд настроек типа таймаутов чтения из сети и записи в сеть. Члены Host и Port содержат адресные данные последнего соединения (для автоматического восстановления его в случае разрыва). Член ErrStr содержит последнее диагностическое сообщение, возникшее при неудачной попытке соединения с сервером или же полученное от сервера в результате обработки запроса
PmsCertList	X509Certificate2 [] Certs; public string ErrStr	Член Certs представляет собой массив сертификатов в стандарте X509. Класс содержит несколько конструкторов для загрузки сертификатов из системных хранилищ (с поиском по имени владельца или серийному номеру) или из файлов. Член ErrStr содержит последнее диагностическое сообщение, возникшее при неудачной попытке загрузки сертификатов

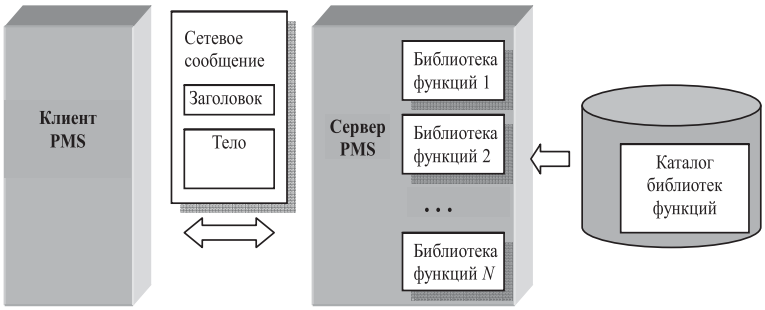


Рис. 3. Принципы организации сервера PMS

• Все множество сервисных обрабатывающих функций реализуется в форме одной или нескольких динамических библиотек. Никакие специальные конструкции типа web-сервисов не используются. Поиск и загрузка всех динамических библиотек выполняются при запуске сервера PMS из каталога, указанного в конфигурации сервера (рис. 3).

В каждой найденной библиотеке проводятся поиск и подключение всех функций, имеющих строго определенную спецификацию:

PmsMessage *имя_функции*
(PmsMessage Inpt, string [] Conf, ref string ErrMsg),

в которой Inpt — входное сообщение (запрос); Conf — конфигурационные данные в форме массива строк вида "имя=значение"; ErrMsg — возвращаемое диагностическое сообщение. С этого момента все библиотечные функции, соответствующие данной спецификации, начинают играть роль сервисных функций, доступных для клиентских программ. Все остальные функции попросту игнорируются.

• При загрузке каждой динамической библиотеки сервер PMS выполняет поиск ее конфигурационного файла, совпадающего по имени с файлом библиотеки, но имеющего расширение имени ".cfg" (своего рода аналог файла web.config для web-сервисов). Этот файл может содержать определения строковых конфигурационных параметров в форме "имя=значение" как для библиотеки в целом, так и для каждой функции отдельно. Конфигурационные данные сохраняются во внутренних структурах данных сервера и автоматически подаются в качестве значения параметра Conf на вход каждой функции при обращении к ней. Отметим, что некоторые важные системные параметры (например, таймаут выполнения) можно определять для каждой функции в отдельности (в отличие от web-сервисов).

• В условиях строго одинаковой спецификации всех обрабатывающих функций применение WSDL практически теряет смысл. Вместо этого в PMS сделан акцент на обеспечение возможности отладки клиентских и сервисных компонентов информационной системы (и их взаимодействия) вне сетевой среды на локальном компьютере. Эта возможность реализована путем введения особых режимов работы в метод Connect класса PmsConnection и в метод Process класса PmsMessage (см. рис. 1 и 2). Если при вызове метода Connect для какого-либо объекта типа PmsConnection ему передается не имя сервера PMS, а полная спецификация каталога, то вместо установления соединения с сервером автоматически выполняется загрузка динамических библиотек из этого каталога и подключение их к программе клиента (точно по тем же правилам, что и в сервере PMS). После этого выполнение метода Process с использованием этого объекта приведет попросту к вызову библиотечной функции в рамках клиентского процесса (рис. 4) с возможностью применения всех отладочных средств, имеющихся в среде разработки.

Важно подчеркнуть, что отлаженные компоненты могут быть перенесены в продуктивную сетевую среду без каких-либо изменений.

• PMS в полной мере использует двоичную природу TCP/IP [8]. Взаимодействие между клиентом и сервером PMS осуществляется по специальному, достаточно простому протоколу, ориентированному на передачу двоичных сетевых сообщений в обоих направлениях. Каждое сообщение в общем случае содержит два массива байтов: заголовок сообщения и тело сообщения. Первые четыре байта заголовка или тела сообщения содержат целое число — его длину. При передаче запроса от клиента к серверу в заголовок сетевого сообщения помещается строка, содержащая полное имя вызываемой функции в формате "имя_библиотеки.имя_функции", а в тело сообщения упаковывается структура PmsMessage в открытой или зашифрованной форме, содержащая информационный запрос. Строка заголовка используется сервером для организации вызова соответствующей обрабатывающей функции. При передаче результата обработки от сервера к клиенту в заголовок сетевого сообщения помещается строка диагностического сообщения (значение параметра ErrMsg, сформированное обрабатывающей функцией), а в тело сообщения упаковывается структура PmsMessage, содержащая

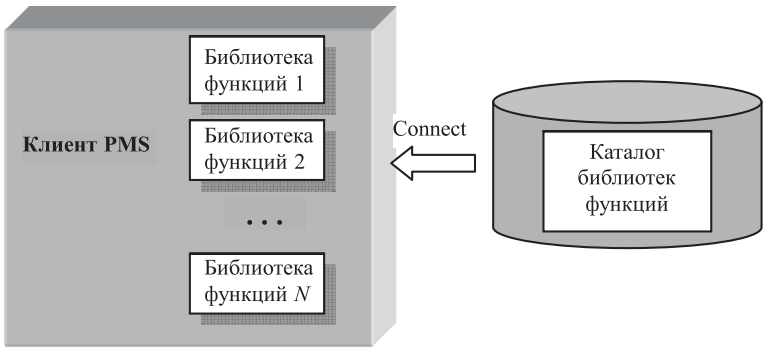


Рис. 4. Отладка функций-обработчиков вне сетевой среды

ответ сервера в открытой или зашифрованной форме. Полученное от сервера диагностическое сообщение автоматически присваивается члену ErrStr объекта класса PmsConnection на стороне клиента.

- Главную нагрузку по обеспечению информационной безопасности берет на себя сервер PMS, а не обрабатывающие функции. В частности, он реализует дешифрование информационного запроса перед его передачей на вход обрабатывающей функции (если запрос зашифрован) и шифрование результата обработки перед его отправкой в сеть. Кроме того, на сервер можно опционально возложить функции проверки ЭЦП у входящих сообщений и добавления собственной ЭЦП к исходящим. Важно подчеркнуть, что проверка ЭЦП на сервере может включать не только аутентификацию информационного запроса, но и разграничение доступа к обрабатывающим функциям путем проверки реквизитов подписантов (имя, организация, должность и т. п.) по имеющимся в конфигурационных данных спискам доступа. Отметим важное свойство PMS: такие списки доступа (как и таймаут выполнения) можно определять как для всей библиотеки функций, так и для каждой функции в отдельности.

Ввиду того что и клиентская библиотека PmsBase.dll, и сервер PMS выполняют функции защиты и аутентификации данных, конкретная реализация службы должна опираться на определенную криптосистему (или на несколько криптосистем с возможностью гибкого выбора одной из них). От применяемой криптосистемы, в частности, зависят структура и алгоритмы формирования ЭЦП (т. е. класса Signature) и ключевой информации, содержащейся в шифрованном сообщении.

Для проведения лабораторных экспериментов и получения сравнительных оценок быстродействия была проведена реализация службы PMS в среде программирования Microsoft VisualStudio 2010 для платформы .Net Framework 4.0 на основе криптосистемы "КриптоПро CSP" версии 3.6, соответствующей требованиям действующих в России ГОСТов в области криптографической защиты информации.

Временные оценки

Основная цель экспериментов с PMS заключалась в сравнении ее быстродействия с быстродействием web-сервисов в одинаковых условиях. Главное внимание уделялось вызовам сервисных функций с относительно малым (от нескольких мс до нескольких сотен мс) временем выполнения (при более длительной обработке разница между двумя технологиями практически нивелируется) с применением и без применения средств криптозащиты. В экспериментах с PMS использовали средства криптозащиты, интегрированные в клиентскую библиотеку и сервер PMS. В экспериментах с web-сервисами средства криптосистемы "КриптоПро CSP" подключали непосредственно к программе клиента и программе web-сервиса. При этом сервер PMS с модельными библиотечными функциями и Internet Information Server с модельными web-сервисами были установлены на одном и том же четырехъядерном сервере приложений с тактовой частотой 2,4 ГГц в операционной среде Window 2003 Server. В качестве клиентской рабочей станции использовался одноядерный компьютер с тактовой частотой 2,8 ГГц.

На рис. 5, 6 и 7 показаны характерные результаты экспериментов с очень быстрой сервисной функцией, выполняющей простое перекодирование полученного строчного сообщения в верхний регистр и возврат результата клиенту, при длине сообщения в 2, 50 и 100 Кбайт соответственно. На каждом рисунке приведены диаграммы времен выполнения операции на сервере с помощью web-сервиса (черный столбик) и с помощью PMS (серый столбик) для трех режимов: без применения криптозащиты, с применением ЭЦП и с применением ЭЦП

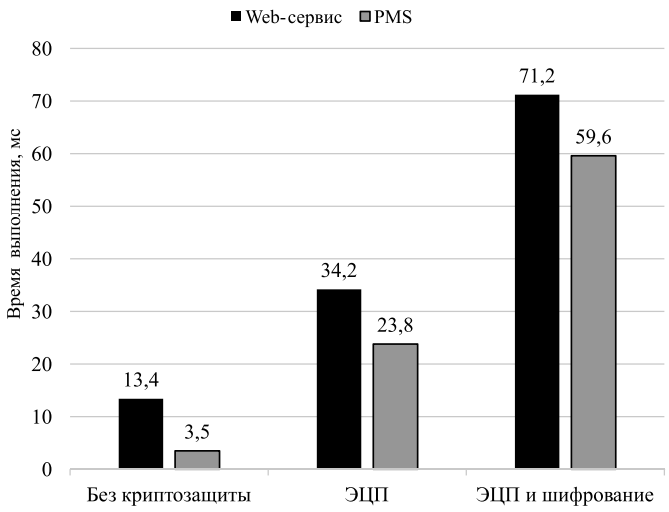


Рис. 5. Время обработки при длине сообщения 2 Кбайт

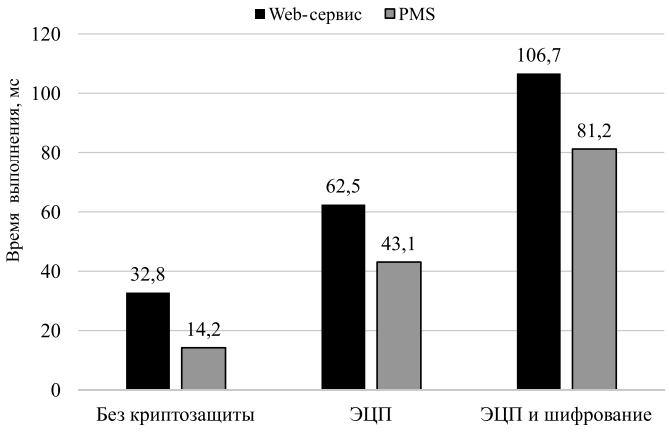


Рис. 6. Время обработки при длине сообщения 50 Кбайт

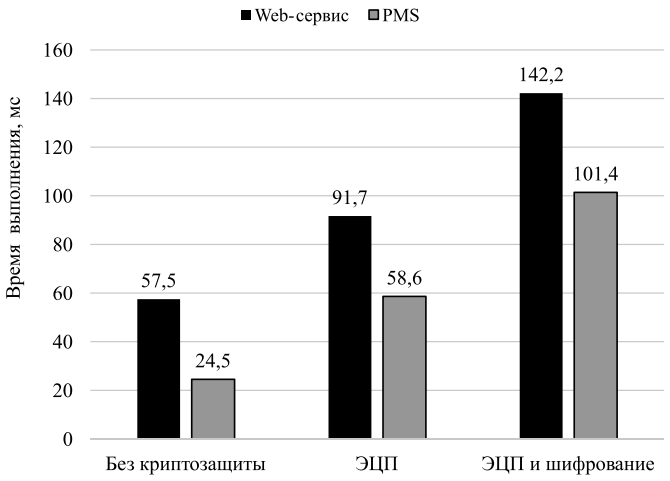


Рис. 7. Время обработки при длине сообщения 100 Кбайт

и шифрования сообщений. В каждом режиме время выполнения вычислялось как среднее значение для 100 последовательных вызовов сервисной функции.

Как видно на рисунках, в данной серии экспериментов PMS не уступает технологии web-сервисов в быстродействии и даже несколько превосходит ее. Причем в режиме без применения криптозащиты это превосходство является весьма значительным (что, по-видимому, целиком объясняется временными затратами, связанными с применением протокола HTTP/SOAP). При подключении криптозащиты время обработки возрастает и различие становится менее существенным (характерно, что абсолютная разница во времени выполнения при этом остается относительно стабильной).

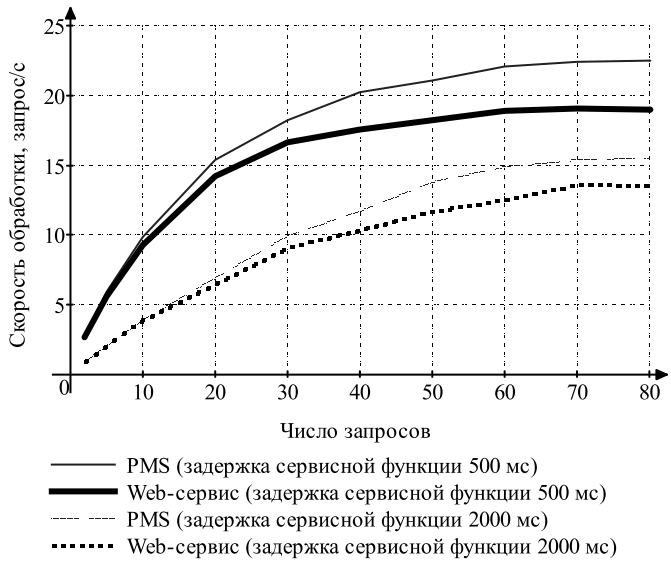


Рис. 8. Скорость обработки пакета одновременных запросов

На рис. 8 приведены результаты экспериментов с относительно медленными сервисными функциями со временем выполнения в несколько сотен мс. Главное внимание в этой серии экспериментов уделялось сравнению быстродействия технологий PMS и web-сервисов в условиях высокой нагрузки: при одновременной обработке пакетов информационных запросов и при применении криптозащиты. Скорость обработки вычислялась как частное от деления числа запросов в пакете на полное время его выполнения. Характерные кривые, представленные на рис. 8, отражают зависимость скорости обработки запросов от их числа в пакете для технологий PMS и web-сервисов при обращении к модельным сервисным функциям со временами выполнения 0,5 и 2 секунды с применением средств криптозащиты и при длине входного и выходного сообщений 50 Кбайт.

Как видно на рис. 8, все кривые ведут себя в целом одинаково. При увеличении числа запросов в пакете растет и скорость обработки, что объясняется положительным эффектом от многопоточной обработки запросов и в IIS, и в сервере PMS. Например, при 20 запросах в пакете и времени выполнения сервисной функции 500 мс скорость обработки достигает 15,5 запроса в секунду у PMS и 14,2 — у web-сервиса (при последовательной обработке скорость не могла бы превысить значения 2). Однако при дальнейшем увеличении размеров пакета рост скорости обработки замедляется, а потом и совсем останавливается вследствие достижения предельной производительности. Как видно на графиках, в этой серии экспериментов PMS несколько превышает web-сервисы по скорости обработки, но это превышение не является значительным.

В целом результаты экспериментов позволяют сформулировать следующие выводы.

- PMS в целом не уступает в быстродействии web-сервисам (как при использовании средств криптозащиты данных, так и без использования).
- Применение средств криптозащиты в PMS не разрушает положительного эффекта от многопоточной обработки запросов.
- Как и web-сервисы, PMS вполне позволяет поддерживать скорость обработки до нескольких и даже нескольких десятков запросов в секунду даже при использовании средств криптозащиты, что обычно бывает достаточным для большинства информационных систем.

В табл. 2 сопоставлены основные преимущества и недостатки web-сервисов в технологии .NET и PMS. Как видно из таблицы, только высокая скорость обработки является общим свойством двух технологий, в части же остальных характеристик списки преимуществ и недостатков почти зеркально противоположны.

Таблица 2

Основные преимущества и недостатки технологий web-сервисов и PMS

Технология	Преимущества	Недостатки
Web-сервисы	• Высокая гибкость в спецификациях сервисных функций • Высокая скорость обработки • Языковая и платформенная независимость	• Избыточный сетевой трафик, связанный с упаковкой двоичных данных и XML-документов в структуру HTTP/SOAP • Отсутствие собственных встроенных средств криптозащиты или возможности их встраивания (кроме использования HTTPS с вытекающими отсюда ограничениями) • Затрудненность отладки сервисных компонентов вне сетевой среды
PMS	• Высокая скорость обработки • Тесная интеграция средств сетевого взаимодействия и средств криптозащиты • Возможность отладки клиентских и сервисных компонентов и их взаимодействия вне сетевой среды • Возможность задания конфигурационных параметров (таймаут выполнения, список доступа и т. п.) для каждой сервисной функции в отдельности	• Жесткость спецификации сервисных функций (строгая ориентация на модель обмена сообщениями) • Платформенная зависимость (Windows, .NET Framework4) • Привязанность к определенной криптосистеме или группе криптосистем

Заключение

По-видимому, в настоящее время ни одна сетевая технология не может сравниться с технологией web-сервисов по гибкости и возможности применения в самых различных областях, связанных с распределенной обработкой. Тем не менее при создании информационных систем довольно часто встречается ситуация, когда готовые решения в области защиты и аутентификации информационных запросов в сети оказываются более важными для разработчика, чем возможность определения сервисных функций с любым числом и любыми типами параметров

Как видно из табл. 2, основные недостатки PMS связаны с жесткой спецификацией сервисных функций и с жесткой привязкой к определенной платформе и к одной или нескольким криптосистемам. Поэтому можно сделать вывод, что область ее эффективного применения включает распределенные информационные системы, все компоненты которых разрабатывают в рамках одного проекта со строгой координацией и администрированием (например, ведомственные системы или системы предприятий, имеющих региональные филиалы). Именно в таких разработках могут быть использованы такие преимущества PMS, как встроенные средства криптозащиты или возможность совместной отладки клиентских и серверных компонентов вне сетевой среды. Наоборот, в системах с независимой разработкой отдельных компонентов или подсистем (межведомственных, транснациональных и т. п.) гибкость в спецификациях сервисных функций и платформенная независимость web-сервисов оказываются наиболее востребованными.

Список литературы

1. **Шапошников И. В.** Web-сервисы Microsoft.NET. СПб: БХВ-Петербург, 2002. 336 с.
2. **Мак-Дональд М., Шпушта М.** Microsoft ASP.NET 3.5 с примерами на C# 2008 и Silverlight 2 для профессионалов. М.: Вильямс, 2009. 1408 с.
3. **Wang V., Salim F., Moskovits P.** The definitive guide to HTML5 WebSockets. NY: Apress, 2013. 208 p.
4. **Хант К.** TCP/IP. Сетевое администрирование. СПб.: Питер, 2007. 816 с.
5. **Асратян Р. Э., Лебедев В. Н.** Защита электронных сообщений в мультисетевой среде // Управление большими системами. 2013. Вып. 45. С. 95—111.
6. **Згоба А. И., Маркелов Д. В., Смирнов П. И.** Кибербезопасность: угрозы, вызовы, решения // Вопросы кибербезопасности. 2014. № 5. С. 30—38.
7. **Щеглов А. В.** Защита компьютерной информации от несанкционированного доступа. СПб.: Наука и техника, 2004. 384 с.
8. **Снейдер Й.** Эффективное программирование TCP/IP. Библиотека программиста. СПб.: Символ-Плюс, 2002. 320 с.

Internet Service for Protected Information Queries Processing in Distributed Systems

R. E. Asratian, rea@ipu.ru, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation

Corresponding author:

Asratian Ruben E., Leading Researcher, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation,
e-mail: rea@ipu.ru

Received on July 06, 2016
Accepted on July 13, 2016

Principles of organization of a network service, intended for protected queries processing in distributed information systems, are considered. We will call this service "Protected Message Service" (PMS). A distinctive feature of this service is the close integration of authentication and data protection functions with functions of network information exchange. From the client point of view the service architecture is based on two main program classes: "Protected message" and "Network Connection". These classes offer necessary functionality not only for creating and protecting messages, but also for transferring them to remote server via established network connections for processing. Contrary to web-services, based on remote function call model, PMS-service is based on message processing model: all service functions receive object of "Protected message" class as a parameter and return another object of the same class as a result of processing. Main advantages of proposed service include high speed of processing (small overhead for data transmission) and a possibility of debugging both client and server components of system out of the network environment. The experimental implementation of PMS in C# for Microsoft Framework 4.0 and the study of performance of new service (in comparison with web services in.NET architecture) were carried out and the area of its effective application is outlined. This area includes distributed systems in which requirements of high processing rate and information security are considered more important than flexibility in service functions specifications.

Keywords: *distributed systems, web-technologies, Internet-technologies, network interactions, data security*

For citation:

Asratian R. E. Internet Service for Protected Information Queries Processing in Distributed Systems, *Programmnaya Ingeneria*, 2016, vol. 7, no. 11, pp. 490—497.

DOI: 10.17587/prin.7.490-497

References

1. **Shaposhnikov I. V.** *Web-servisy Microsoft.NET* (Web-services of Microsoft.NET), Saint Petersburg, BHV-Peterburg, 2002, 336 p. (in Russian).
2. **MacDonald M., Szpuszta M.** *Microsoft ASP .NET 3.5 s primerami na C# 2008 i Silverlight 2 dlja professionalov* (Pro Microsoft ASP.NET 3.5 in C# 2008 includes Silverlight 2), Moscow, Willams, 2009, 1408 p. (in Russian).
3. **Wang V., Salim F., Moskovits P.** *The definitive guide to HTML5 WebSockets*, New York, Apress, 2013, 208 p.
4. **Hunt C.** *TCP/IP. Setevoe administrirovanie* (TCP/IP Network administration), Saint Petersburg, Piter, 2007, 816 p. (in Russian).
5. **Asratian R. E., Lebedev V. N.** Zashhita elektronnyh soobshhenij v mul'ti-setevoj srede (Protection of messages in multi-network environment), *Upravlenie bol'shimi sistemami,* 2013, vol. 45, pp. 95—111 (in Russian).
6. **Zgoba A. I., Markelov D. V., Smirnov P. I.** Kiberbezopasnost: ugrozy, vyzovy, reshenija (Cybersafety: threats challenges, decisions), *Voprosy kiberbezopasnosti*, 2014, no. 5, pp. 30—38 (in Russian).
7. **Shheglov A. V.** *Zashhita komp'juternoj informacii ot nesankcionirovannogo dostupa* (Protection of computer information against illegal access), Saint Petersburg, Nauka i tehnika, 2004, 384 p. (in Russian).
8. **Snader J.** *Effektivnoe programmirovanie TCP/IP* (Effective TCP/IP programming), Saint Petersburg, Simvol-Plus, 2002, 320 p. (in Russian).

УДК 004.4 DOI: 10.17587/prin.7.498-508

И. Б. Бурдонов, д-р физ.-мат. наук, вед. науч. сотр., e-mail: igor@ispras.ru,
А. С. Косачев, канд. физ.-мат. наук, вед. науч. сотр., e-mail: kos@ispras.ru,
Институт системного программирования РАН, г. Москва

Исследование графа автоматом

Модели, описывающие объекты и технологические процессы в виде графов и анализирующих их свойства автоматов, востребованы и активно используются при разработке аппаратных и программных средств различного назначения. Настоящая статья является первой в серии из трех публикаций, посвященных исследованию графов автоматами. Автомат двигается по ребрам графа, используя вершины графа как ячейки памяти для чтения/записи. Рассмотрены алгоритмы обхода графа (проход по всем ребрам) с оценкой их сложности в зависимости от вида графа и типа автомата. В последующих статьях будут представлены результаты по исследованию графов (в том числе недетерминированных или динамически меняющихся) коллективом автоматов.

Ключевые слова: *неориентированный граф, ориентированный граф, упорядоченный граф, нумерованный граф, неизвестный граф, обход графа, автомат, робот, полуробот*

Введение

Задача исследования графа возникает каждый раз, когда объект исследования моделируется графом. Вот лишь несколько примеров.

- Компьютерные сети моделируются графом (сетевая топология [1]), в котором вершины соответствуют компьютерам, а ребра — линиям связи.
- Гипертексту, в частности, веб-страницам в сети Интернет [2], также соответствует граф, в котором вершины — это страницы (или их элементы), а ребра — это гиперссылки. Заметим, что в этом случае ребра ориентированные.
- Сложные (составные) программные и аппаратные системы моделируются графами, в которых вершины соответствуют компонентам системы, а ребра — связям между ними.
- Автоматные и автоматоподобные (например, системы размеченных переходов — LTS — *Labelled Transition System*) модели программных и аппаратных объектов [3—5].

Наиболее удобным и распространенным способом представления такой модели является ориентированный граф переходов, в котором вершины изображают состояния объекта, а ребра — переходы. Возникает естественный вопрос: "Почему графы исследуются автоматами?" Главная причина в желании, чтобы исследование выполнялось автоматически, т. е. с помощью некоторого алгоритма. В этом случае исследование графа может выполнять компьютер. Формализацией алгоритма является машина Тьюринга, т. е. автомат, "ползающий" по ленте. При исследовании графа автомат "ползает" по графу, перемещаясь вдоль его ребер, а вершины играют роль ячеек ленты машины Тьюринга.

На практике, с одной стороны, нужно, чтобы автомат требовал поменьше памяти, а с другой стороны, размеры графа, т. е. число его вершин и ребер, обычно каким-то образом ограничены. Поэтому будем рассматривать и такие автоматы, которые не конечны на рассматриваемом классе графов, но конечны на любом подклассе графов ограниченного размера. Автомат, который конечен на всем классе графов, будем называть *роботом*. Если автомат не конечный, но граф не помещается в его память, то такой автомат будем называть *полуроботом*. По сути, это то же самое, что локальный алгоритм на графе. Если граф помещается в память автомата, то это *неограниченный автомат*.

Как правило, исследование графа сводится к *обходу графа*, под которым понимается построение и проход автоматом маршрута, содержащего все вершины и ребра графа. Иногда имеет смысл говорить о покрытии ребер графа не одним маршрутом, а набором маршрутов: либо вследствие недостаточной связности графа, либо для ускорения времени обхода, если эти маршруты проходятся параллельно несколькими автоматами. Обычно обход должен начинаться с выделенной начальной вершины графа, которую для краткости будем называть *корнем*.

Приведем только два, однако весьма показательных примера исследования графа, которое сводится к его обходу. Первый из них — обход лабиринта, который является моделью многих частных задач. Отсюда возникли как понятие обхода графа, так и сама теория графов, когда Эйлер решал проблему семи мостов Кенигсберга [6]. Второй пример — это тестирование, при котором граф представляет собой граф переходов автоматной или автоматоподобной модели тестируемой системы [7—10]. Ее начальное

состояние — это корень, автомат на графе — тестирующая система, много автоматов на графе — распределенная тестирующая система. Проход автомата из текущей вершины по выходящему из нее ребру моделирует тестовое воздействие на систему, находящуюся в текущем состоянии, а также наблюдение результата этого воздействия, в том числе измененного состояния системы.

Во многих случаях такая модель, в которой автомат двигается по ребрам графа, а вершины графа используются как память, из которой автомат читает и в которую он пишет, может быть "инвертирована" [11]. В такой "инвертированной" модели в каждой вершине неподвижно "сидит" один автомат, а по ребрам графа перемещаются сообщения, которые автоматы пересылают друг другу.

Настоящая статья — это первая из серии трех статей, посвященных исследованию графов автоматами. В этих статьях будут представлены результаты исследований различных алгоритмов обхода графов в зависимости от типа графа, а также от типа и числа автоматов. В данной, первой статье будет рассмотрено исследование графа одним автоматом. Граф может быть как неориентированным, так и ориентированным. Вторая статья будет посвящена исследованию ориентированных графов коллективом двигающихся автоматов, в том числе исследованию недетерминированных графов. В третьей статье будет описано исследование ориентированного графа коллективом неподвижных автоматов, которые обмениваются между собой сообщениями, пересылаемыми по ребрам графа. Кроме исследования свойств самого графа, будут рассмотрены параллельные вычисления на графе, которые могут выполнять автоматы. Граф может быть как статическим, так и динамическим, т. е. меняющимся во времени.

1. Основные понятия и термины

В настоящем разделе вводятся основные понятия и термины, которые будут далее использоваться при изложении полученных авторами результатов.

1.1. Упорядоченный граф

Когда выполняется обход графа, на каждом шаге автомат должен указать ребро, по которому он "хочет" выйти из текущей вершины. Для этого используется нумерация таких ребер начиная с 1, при этом для каждой вершины своя нумерация ребер. Автомат указывает номер ребра в текущей вершине. В ориентированном графе ребро проходится только в направлении его ориентации, поэтому для данной вершины нумеруются только выходящие из нее ребра, а ребро имеет номер только в вершине, из которой выходит. В неориентированном графе ребро может проходиться в обоих направлениях, поэтому для данной вершины нумеруются все инцидентные ей ребра, и ребро имеет два номера: по одному в каждой из инцидентных

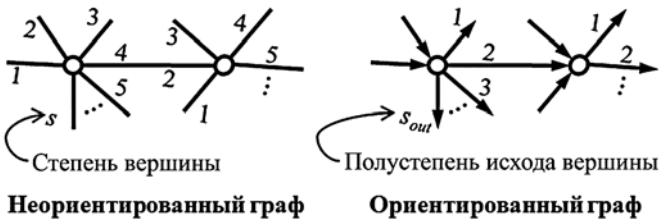


Рис. 1. Нумерация ребер в упорядоченном графе

ему вершин. Такой граф называется *упорядоченным* (рис. 1).

1.2. Структура на графе

Посмотрим, какая структура возникает на графе при его обходе (рис. 2).

Для наглядности представления будем использовать три "краски": белую, серую и черную. Непройденные ребра, т. е. ребра, по которым автомат не проходил, покрасим в белый цвет. Пройденные ребра будут серого или черного цвета. Пройденное ребро ориентируем в том направлении, по которому автомат первый раз его прошел. Эта операция не меняет заданную ориентацию ребер в ориентированном графе, поскольку ориентированные ребра проходятся только в направлении их заданной ориентации. Непройденная вершина, т. е. вершина, в которой автомат еще не был, будет окрашена в белый цвет. Пройденные вершины при этом серые или черные. В неориентированном графе пройденная вершина черная, если все инцидентные ей ребра не белые, т. е. автомат прошел по всем инцидентным вершине ребрам, иначе вершина серая. В ориентированном графе учитываются только те ребра, которые выходят из вершины: пройденная вершины черная, если автомат прошел по всем выходящим из вершины ребрам, иначе вершина серая.

Пройденный граф — подграф, порожденный пройденными ребрами (серыми и черными). *Прямое ребро* — это ребро, по которому автомат первый раз пришел в какую-то вершину. Они порождают прямое дерево — остов пройденного графа, ориентированный от корня. Остальные пройденные ребра будем называть *хордами* прямого дерева. Серое дерево — это поддерево прямого дерева, содержащее корень и все серые вершины, причем все его листья тоже серые. Ребра серого дерева серые, остальные прямые ребра черные.

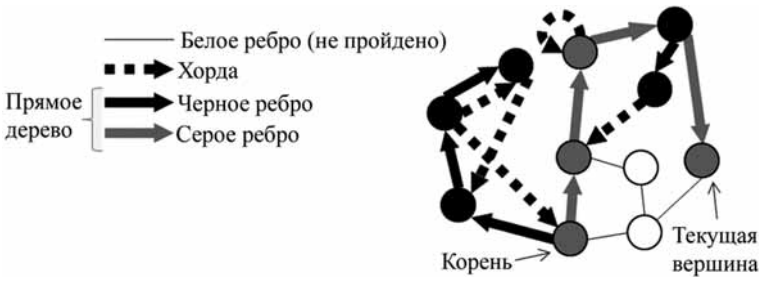


Рис. 2. Структура на графе во время обхода

1.3. Неизвестный граф, неизбыточные и свободные автоматы

Если граф известен (и помещается в память автомата), то можно вычислить его обход минимальной длины, а потом пройти по этому маршруту. Для неизвестного графа так сделать уже нельзя. Этот факт означает, что граф неизвестен? Он неизвестен заранее, до прохода по всем его ребрам. На каждом шаге, выбирая очередное ребро, автомат "знает" только часть графа. Что он "знает"? Во-первых, автомат "знает" пройденный граф, во-вторых, он что-то "знает" о белых ребрах, инцидентных серым вершинам. Далее возникает вопрос: "Что и когда автомат может "узнавать" об этих ребрах?"

Автомат будем называть *неизбыточным* [12], если он "узнает" степень вершины, когда первый раз в нее попадает. Поскольку пройденные ребра он "знает", всегда известно и число белых ребер, инцидентных вершине.

Автомат будем называть *свободным* [12], если он "узнает" о наличии или отсутствии белого ребра только при попытке пройти по нему. Если такое ребро есть, автомат идет по нему, а если нет, то получает отказ и остается на месте. Допуская определенные вольности речи, будем говорить о цвете вершин и ребер "с точки зрения" автомата: если вершина черная, но автомат еще не "знает" об этом, то для него она серая; соответственно, понимается серое дерево. Неизбыточный автомат "узнает" о том, что вершина черная, когда проходит последнее инцидентное ей белое ребро, а свободный автомат — когда получает отказ на попытку пройти по ребру при условии, что все ребра с меньшими номерами уже пройдены. Свободный автомат работает немного дольше потому, что может сложиться ситуация, когда ему нужно лишний раз прийти в вершину, чтобы убедиться, что она черная.

Поскольку автомат не "знает", куда ведут белые ребра, все равно, какое белое ребро он выберет в текущей серой вершине. Можно считать, что он проходит белые ребра из серой вершины в порядке возрастания их номеров. В ориентированном графе этот факт означает, что если из вершины выходят k пройденных ребер, то они имеют номера от 1 до k. В неориентированном графе это не обязательно так, потому что какое-то ребро ab может быть пройдено первый раз не из вершины a, а из вершины b.

Для прохода по ребру из вершины a в вершину b автомат называет номер ребра в вершине a. Если граф неориентированный и такое ребро присутствует, то после прохода по нему автомат дополнительно "узнает" номер ребра в вершине b.

Будем оценивать алгоритмы обхода с точки зрения времени обхода (для одного автомата это, по сути, длина маршрута, который он проходит). Поэтому обход известного графа (неограниченным автоматом) будет нас интересовать только для оценки минимальной длины обхода, с которой будем сравнивать обход неизвестного графа различными автоматами.

1.4. Конечность степени/полустепени вершин графа

Исследование неизвестного графа двигающимся автоматом похоже на машину Тьюринга. Вершина графа соответствует ячейке ленты, а движение по ребру — движению по ленте влево или вправо. Таким образом, лента обычной машины Тьюринга — это граф специального вида.

Однако здесь возникает затруднение для робота (конечного автомата). Дело в том, что номер ребра является выходным символом автомата. Для того чтобы автомат был конечным, в неориентированном графе степень вершины, а в ориентированном графе полустепень исхода вершины должны быть ограничены сверху. Возникает вопрос: "Как обойти это ограничение?"

Для ответа на поставленный вопрос преобразуем вершину ориентированного графа в цепочку вершин, каждая из которых имеет полустепень исхода не больше 2 (рис. 3). Для машины Тьюринга такое действие означает, что ячейка ленты превращается в цепочку ячеек конечной, но не ограниченной длины, по которой можно перемещаться в вертикальном направлении. Получается что-то вроде ленты переменной ширины.

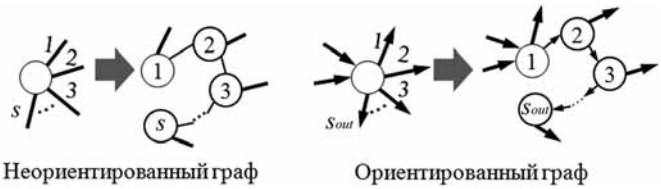


Рис. 3. Преобразование вершины в цепочку вершин

1.5. Нумерованный граф

Если граф помещается в память автомата, то по мере обхода автомат может запоминать в своей памяти весь пройденный граф. Для этого достаточно нумеровать вершины в порядке их обнаружения, т. е. когда автомат по белому ребру приходит в белую вершину, а далее записывать в вершине ее номер. Тогда при проходе любого ребра автомат может "узнать" номера обеих вершин, инцидентных этому ребру. На самом деле номер вершины — это единственное, что нужно хранить в самой вершине. Все остальное можно хранить в памяти автомата, индексируя номером вершины. Если вершины пронумерованы заранее и их номера в вершинах записаны, то такой граф будем называть *нумерованным*. Тогда автомат будет только читать из вершины ее номер и не будет в нее писать.

1.6. Классификация алгоритмов обхода графов

Резюмируя представленные выше определения будем констатировать, что различные алгоритмы обхода графа автоматами классифицируются в зависимости от рассмотренных параметров графа и автомата. Во-первых, граф может быть ориентиро-

ванным или неориентированным. От этого существенно зависят как алгоритмы обхода, так и оценки их сложности. Во-вторых, граф известный или неизвестный. Известный граф нужен только для того, чтобы оценить минимальную длину обхода. Только для неизвестного графа рассматриваются различные автоматы обхода. Эти автоматы могут быть неизбыточными или свободными в зависимости от способа общения с графом. По размеру их памяти автоматы делятся на роботы, полуроботы и неограниченные автоматы. Представляется также интересным различать первый обход графа и повторный, при котором можно использовать пометки на графе, оставленные автоматом при первом проходе.

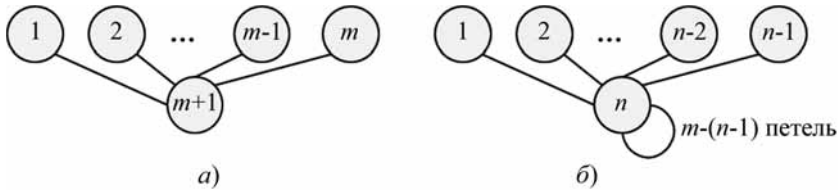


Рис. 4. Достижимость минимальной длины обхода неориентированного графа

Далее естественным представляется рассмотреть один или несколько автоматов на графе, ответить на вопрос: "Зачем нужно несколько автоматов?" Наличие нескольких автоматов решает либо "проблему времени", уменьшая время обхода, либо "проблему ограниченности памяти" компьютера за счет распределенного обхода. В настоящей, первой статье ограничимся одним автоматом, а коллективы автоматов будут рассмотрены в двух следующих статьях.

Как уже отмечалось во Введении, есть две модели исследования графов автоматами. В первой модели автоматы двигаются по ребрам графа, а вершины используются как ячейки памяти. Во второй модели автоматы неподвижно "сидят" в вершинах графа, а по ребрам графа пересылаются сообщения от автомата к автомату. Первая модель будет рассмотрена в первых двух статьях, а вторая — в третьей статье.

Кроме того, граф может быть недетерминированным. Такой случай будет рассмотрен во второй статье, или динамическим, т. е. меняющимся со временем. Этот случай будет рассмотрен в третьей статье.

2. Неориентированный граф

В этом разделе представлены результаты по исследованию автоматом неориентированных конечных графов.

2.1. Минимальная длина обхода

Обход неориентированного графа существует тогда и только тогда, когда граф является связным. Минимальная длина обхода, т. е. обхода известного графа неограниченным автоматом, не более $2m - 1$, где m — число ребер. Действительно, если каждое ребро удвоить, то граф становится эйлеровым (степени всех вершин четные). Его эйлеров цикл соответствует обходу исходного графа длиной $2m$. Вместе с тем последнее ребро можно не проходить, поскольку это был бы его повторный проход. Пример достижимости оценки представлен на рис. 4, *а*.

Если задано не только число ребер m, но и число вершин n, то получается оценка $m + n - 2$. Действительно, пусть S — минимальное множество ребер, которые нужно удвоить, чтобы получился эйлеров граф, а r — число ребер в S. Такое множество S определяется как минимальное паросочетание в полном графе, вершины которого — все вершины исходного графа с нечетной степенью, а вес ребра — длина минимального пути между вершинами. В рассматриваемом случае минимальная длина обхода равна $m + r$, если корень не инцидентен ребрам из S, или $m + r - 1$ в противном случае. В первом случае проходится эйлеров цикл, а во втором — эйлеров путь с началом в корне. В S нет циклов, поскольку любой цикл можно удалить без ущерба для четности всех вершин. Поэтому $r \leqslant n - 2$, если корень не инцидентен ребру из S, и $r \leqslant n - 1$ в противном случае. Как следствие, минимальная длина обхода равна $m + n - 2$. Пример достижимости этой оценки представлен на рис. 4, *б*.

2.2. Алгоритм Тэрри

Для обхода неизвестных неориентированных графов существует хорошо известный алгоритм Тэрри [13]. Он реализуется свободным роботом, если степень вершины ограниченна, или полуроботом для любых графов. По сути, это DFS-алгоритм, т. е. обход в глубину (*Depth-first search*). В DFS-алгоритме серое дерево всегда состоит только из одного серого пути, который свободный робот размечает на графе: серое ребро отмечается в его конце. Каждое ребро автомат проходит 2 раза. Возникает естественный вопрос: "Как работает алгоритм Тэрри?" В его основе следующие два основных правила: 1) после прохода по хорде немедленно возвращаемся по ней назад (*откат по хорде*); 2) если идти некуда (текущая вершина стала черной), то идем по входящему серому ребру, т. е. в предыдущую вершину серого пути (*откат по дереву*). Если предыдущей вершины нет, т. е. черным стал корень, то конец обхода.

2.3. Оптимизации алгоритма Тэрри

В табл. 1 сведены возможные оптимизации алгоритма Тэрри. Эта таблица демонстрирует различие между типами автоматов, а также дополнительные возможности повторных обходов графа.

Оптимизации A, B и C — это проход по петле только один раз. Для этого автомат должен распознавать петли. Оптимизацию A (распознавание всех петель) может делать неограниченный автомат или полуробот, запоминая номер вершины, из которой он выходит по ребру, и сравнивая с номером вершины, куда попал. Они совпадают только для петли. Если граф ненумерованный, полуробот может сам

Таблица 1

Оптимизации алгоритма Тэрри

Обход выполняет	Оптимизация						Тип автомата
	A	*B*	*C*	*D*	*E*	*F*	
Неограниченный автомат или полуробот	Да	Да	Да	Да	Да	Нет	Неизбыточный
	Да	Да	Да	Нет	Нет	Нет	Свободный
Робот	Нет	Да	Да	Нет	Да	Нет	Неизбыточный
	Нет	Да	Да	Нет	Нет	Нет	Свободный
Свободный робот на повторном обходе №	3	—				2	—

Обозначения: A — нет отката по всем петлям;
B — нет отката по петлям с одинаковым номером в начале и конце;
C — нет отката по петлям в корне (или в конечном множестве вершин);
D — нет отката по черному пути;
E — нет отката по черному пути, если все его ребра имеют максимальные номера (или число ребер, имеющих большие номера, ограниченно);
F — нет отката по самой правой ветви прямого дерева

нумеровать вершины, присваивая вершине очередной номер тогда, когда первый раз в нее попадает. Робот не может использовать номер вершины (он неограничен), но может не проходить петли в корне, если специально пометить корень. В общем случае робот, используя ограниченное множество разных пометок, может распознавать петли в ограниченном множестве вершин (оптимизация *C*). Робот может также распознать петли во всех вершинах, если петля имеет не два разных номера в вершине, а один (оптимизация *B*). Когда робот проходит по белому ребру, которое ведет в другую вершину, то в ней это ребро еще помечено как белое. А если он "видит", что ребро уже не белое, значит, это петля.

Робот может также распознавать все петли при повторных обходах, т. е. используя пометки, оставленные на предыдущих обходах графа. Это можно сделать начиная с третьего обхода. На первом обходе размечаются все хорды. Сделать это можно, так как по каждой хорде автомат идет два раза подряд: первый раз ребро помечается как хорда в своем конце, а второй раз — в начале. При втором обходе, перед тем как идти по белой хорде, автомат помечает вершину. Если, пройдя белое ребро, автомат приходит в вершину с пометкой, то это петля, и автомат отмечает это ребро. В любом случае, поскольку автомат возвращается по хорде, он снимает пометку в вершине. Поэтому начиная с третьего обхода петли проходятся по одному разу — это отмеченные ребра.

Оптимизации *D*, *E* и *F* касаются отката по дереву. Когда серый путь становится черным, по нему можно не делать отката. Любую такую ситуацию может распознать неизбыточный неограниченный автомат или полуробот (оптимизация *D*). Для этого он должен "помнить" число белых ребер, инцидентных вершинам серого пути. Путь становится черным, когда это число становится равным нулю. Свободный автомат (даже неограниченный) ничего не "знает" о числе белых ребер: пока он не попробовал выйти из вершины по ребру, он не "знает", присутствует такое ребро или нет. Соответственно, для него все вершины серого пути серые, кроме, быть может, последней вершины, ставшей черной. У робота память ограниченна, поэтому неизбыточный робот может "помнить" только ограниченное число белых ребер, инцидентных вершинам серого пути. Если число таких ребер не превосходит ограничения робота, то робот не делает отката по дереву (оптимизация *E*). Следует заметить, что свободный робот при повторном обходе может не делать отката по всей самой правой ветви прямого дерева (оптимизация *F*). Для этого при первом обходе размечаются хорды, а при втором обходе в каждой вершине в первую очередь проходятся хорды, а уже потом прямые ребра. Поэтому все ребра самой правой ветви прямого дерева проходятся в последнюю очередь, что робот и запоминает.

Сочетание оптимизаций *C* и *E* для неизбыточного автомата (в табл. 1 выделено серым фоном ячеек) гарантирует достижение оценки $2m - 1$. Последнее ребро проходится один раз: либо петля в корне (оптимизация *C*), либо прямое ребро (оптимизация *E*).

2.4. BFS- и "жадный" алгоритмы

Алгоритм BFS (*breadth-first search*), т. е. алгоритм обхода в ширину, тоже реализуется свободным роботом, если степень вершины ограниченна, или полуроботом для любых графов. Здесь следует отметить, чем он отличается от алгоритма Тэрри. Робот проходит хорды по одному разу. Пройдя по хорде, робот не возвращается, а снова пытается идти по белым ребрам. Вследствие этого в сером дереве может "вырасти" новая ветвь.

Когда текущая вершина становится черной, робот ищет серую вершину, двигаясь по серому дереву. Если робот находится в листе дерева, то он делает откат по дереву, т. е. идет по нему вниз к корню, либо до серой вершины, либо до развилки. Ребра, которые проходятся, становятся черными, т. е. удаляются из серого дерева. Если робот доходит до развилки и она оказалась черной, он выполняет *поиск серой вершины*: идет вверх по дереву до серой вершины. Если робот находится не в листе дерева, он сразу идет вверх до серой вершины. Конец обхода — когда в сером дереве остался один корень и он "почернел".

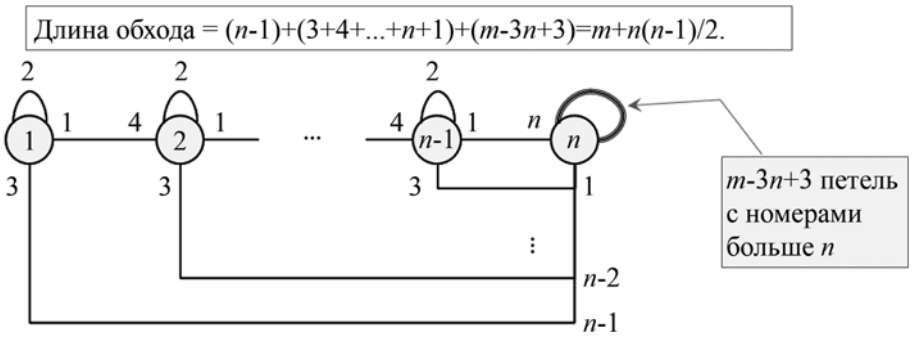


Рис. 5. Достижимость оценки длины обхода BFS-алгоритма

Когда робот движется вверх по дереву в поиске серой вершины, он может выбирать, по какому серому ребру ему двигаться, если из вершины выходит несколько таких ребер. Этот выбор зависит от алгоритма робота. Он, например, может выбирать серое ребро с наименьшим номером или перебирать серые ребра по циклу. На оценку длины обхода в наихудшем случае это не влияет.

Длина обхода оценивается как $m + \boldsymbol{O}(n^2)$. Докажем это. Робот проходит по каждой хорде ровно один раз. По каждому прямому ребру, число которых равно $n - 1$, придется пройти 4 раза: 1) первый проход по белому ребру; 2) возврат в начало ребра, чтобы пометить его как прямое в его начале; 3) повторить проход вперед; 4) при откате по ребру, который для каждого прямого ребра выполняется 1 раз, поскольку после этого ребро удаляется из серого дерева.

Кроме этого, выполняются поиски серых вершин. Такой поиск начинается в двух случаях, а именно после прохода по белой хорде в нелистовую вершину, которая в силу этого становится черной; после отката по дереву, если робот попадает в черную развилку дерева. Число первых случаев не более n, так как вершина только один раз становится черной. Число вторых случаев тоже не больше n, так как откат по серому ребру делается один раз (после чего ребро становится черным). Каждый поиск — это проход простого пути (без самопересечения), т. е. имеет длину не более n.

С учетом изложенного выше длина обхода не больше $[m - (n - 1)] + [4(n - 1)] + [n^2] = m + \boldsymbol{O}(n^2)$. Эта оценка достижима на графе, который изображен на рис. 5.

Отличие "жадного" алгоритма от алгоритма обхода в ширину заключается в том, как осуществляется поиск серой вершины. Вместо движения по серому дереву вниз, а потом вверх робот по всему пройденному графу сначала вычисляет кратчайший путь до серой вершины, а потом идет по этому пути. Конечно, такое вычисление без движения по графу может делать только неограниченный автомат. В памяти полуробота и тем более робота граф полностью не помещается, поэтому им для вычисления кратчайшего пути придется двигаться по графу, в том числе по хордам. Поэтому для таких автоматов "жадный" алгоритм не имеет смысла.

Длина обхода неограниченным автоматом, работающим по "жадному" алгоритму, тоже не более $m + \boldsymbol{O}(n^2)$. Может быть, в этой оценке вместо $\boldsymbol{O}(n^2)$ должно стоять $\boldsymbol{O}(n)$, но пока это только гипотеза.

2.5. Сводка результатов по обходу неориентированного графа

В табл. 2 представлена сводка результатов по обходу неориентированного графа (кроме оптимизаций для алгоритма Тэрри). Что в связи с этими данны-

Таблица 2

Длина обхода неориентированного графа

Тип обхода и графа	Тип автомата		Обход как функция от	
			m	m и n
Первичный обход известного графа	Неограниченный автомат		$2m - 1$	$m + n - 2$
Повторный обход любого графа	Автомат любого типа			
Первичный обход неизвестного графа	Свободный робот	Алгоритм DFS	$2m$	$2m$
		Алгоритм BFS	—	$m + \boldsymbol{O}(n^2)$
	Неизбыточный робот	Алгоритм DFS	$2m - 1$	$2m - 1$
	Неограниченный неизбыточный автомат	"Жадный" алгоритм	—	$m + \boldsymbol{O}(n^2)$?

ми следует отметить? Понятно, что обход неизвестного графа длиннее, чем известного. Однако в терминах числа ребер разница всего лишь на единицу для свободного автомата при первом проходе, а именно $2m$ вместо $2m - 1$. Если задано еще и число вершин n, то алгоритм поиска в ширину и "жадный" алгоритм лучше алгоритма Тэрри при достаточно большом числе ребер по сравнению с числом вершин — вместо $2m$ будет $m + \boldsymbol{O}(n^2)$. Все алгоритмы реализуются роботом, но для "жадного" алгоритма это не имеет смысла (и оценка, очевидно, значительно выше).

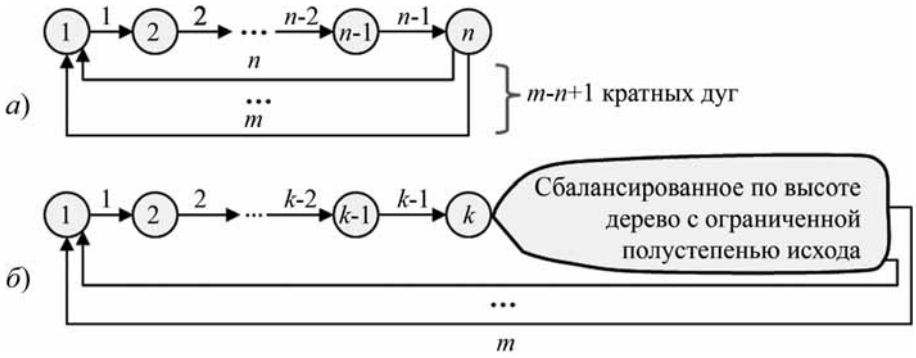


Рис. 7. Примеры достижимости минимальной оценки

3. Ориентированный граф

В этом разделе представлены результаты по исследованию автоматом ориентированных конечных графов. Ориентированное ребро будем, как это принято, называть *дугой*. Дугу можно проходить только в направлении ее ориентации. Напомним, что в ориентированном графе вершина становится черной, когда автомат прошел не по всем инцидентным ей ребрам, как в неориентированном случае, а по всем *выходящим* из вершины дугам. В черную вершину могут входить белые дуги.

3.1. Сильно-связные графы и графы 1-го и 2-го рода

Для существования обхода из любой вершины как начальной требуется сильная связность графа. Если начальная вершина фиксирована (корень), то граф должен быть 1-го рода [12] — цепочка компонентов сильной связности и разделяющих дуг, начальная вершина — в первом компоненте (рис. 6.)

Если граф неизвестен, то он должен быть графом 2-го рода [12]: простой путь, ведущий в последний компонент сильной связности. Пройденный подграф является графом 1-го рода [12], а в конце обхода, конечно, совпадает со всем графом.

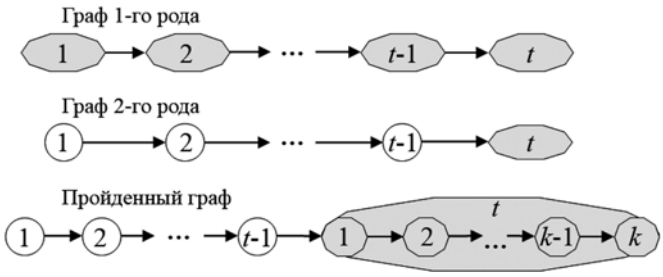


Рис. 6. Связность графов

3.2. Минимальная длина обхода

Минимальный обход имеет длину $\boldsymbol{O}(nm)$, где n — число вершин, m — число дуг. При любом линейном упорядочивании множества дуг после прохода очередной дуги идем в начало следующей дуги, проходя простой путь длиной не более $n - 1$. Пример достижимости оценки представлен на рис. 7, *а*. Здесь из вершины n в начальную вершину ведет много кратных дуг.

На рис. 7, *б*, показан пример достижимости оценки для ограниченной полустепени исхода вершин. Это ограничение может быть меньше, чем число кратных дуг в примере на рис. 7, *а*. Поэтому несколько последних вершин образуют сбалансированное по высоте дерево с ограниченной полустепенью исхода, из листьев которого дуги ведут уже в начальную вершину. Вообще же при ограничении полустепени исхода значением s оценка становится $\boldsymbol{O}(n^2)$, так как $m \leqslant sn$.

Более точную оценку можно получить, если вместо числа вершин n использовать диаметр графа d как максимальную длину пути в графе. Очевидно, что $d \leqslant n - 1$. Оценка получается $\boldsymbol{O}(dm)$.

3.3. Три алгоритма

Алгоритм DFS может применяться и в случае ориентированного графа. Дуга серого пути помечается в ее начале. Однако в этом случае возникает вопрос с откатом, так как автомат не может вернуться по дуге в обратном направлении. Вместо этого он должен пройти некоторый ориентированный путь из конца дуги в ее начало. Для этого строится лес *обратных деревьев*: по одному на каждый компонент сильной связности пройденного графа (рис. 8.).

В каждом компоненте обратное дерево является остовом, ориентированным к корню компонента. В этот корень входит разделяющая дуга из предыдущего компонента, в первом компоненте это корень графа. Очевидно, все корни обратных деревьев лежат на сером пути. Обратная дуга отмечена в ее начале.

Откат выполняется по дуге ab, если это хорда или последняя дуга серого пути. В первом случае серый путь заканчивается в a, а во втором случае — в b. Для отката всегда есть цикл, состоящий из дуги ab и двух путей: 1) путь по обратным дугам от вершины b до пересечения с серым путем, вершины c, лежащей на пути не выше a (если ab — последняя дуга серого пути, то $c \neq b$); 2) отрезок серого пути от вершины c

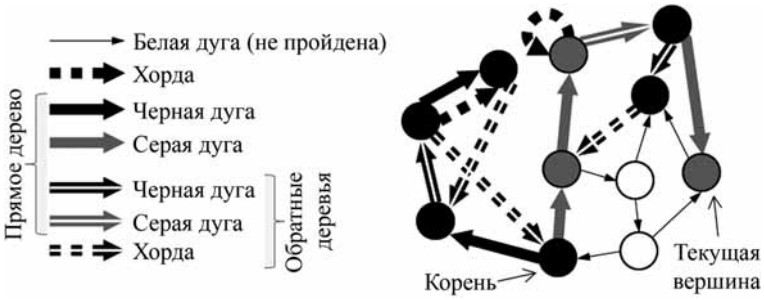


Рис. 8. Структура на графе во время обхода

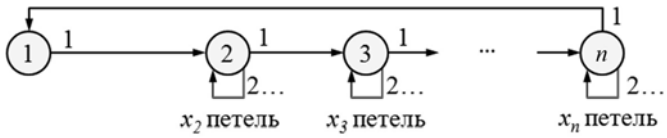
до вершины *a*. Вот по этому циклу автомат и ищет вершину *a*. Неограниченный автомат всегда "знает" вершину *a* — это либо последняя вершина серого пути (откат по хорде), либо предпоследняя вершина (откат по дереву). Полуробот может нумеровать вершины серого пути 1, 2, ... и запоминать номер последней вершины. Для этого ему достаточно памяти порядка log*n*. Если полуробот проходит хорду, то номер ее начала — это запомненный номер вершины. Если полуробот попадает в белую вершину, то ей присваивается номер на единицу больше и запоминается. Для отката по дереву в предпоследнюю вершину серого пути полуробот "знает", что ее номер на единицу меньше запомненного, а после отката запоминает этот уменьшенный на 1 номер. Откат не делается, если запомненный номер равен 1, т. е. это номер корня, в этом случае конец обхода.

При откате по хорде *ab* нужно корректировать лес обратных деревьев, если на отрезке серого пути *ca* имеется корень обратного дерева отличный от *c*. В каждой вершине этого отрезка начиная с первого попавшегося корня обратного дерева, обратной дугой становится выходящая серая дуга, а в последней вершине *a* обратной дугой становится пройденная хорда *ab*. Такую коррекцию может делать и полуробот.

Неизбыточный автомат мог бы делать откат по серому пути не на одну, последнюю дугу пути, а на весь постфикс серого пути начиная с последней серой вершины. Правда, полуроботу для этого придется несколько раз проходить цикл дуг: сначала — чтобы узнать номер последней серой вершины на сером пути, затем — чтобы отметить дуги постфикса как черные, и третий раз — чтобы придти в последнюю серую вершину.

В BFS-алгоритме нет отката по хорде, и серое дерево состоит не из одного пути, оно может ветвиться. После прохода по хорде автомат двигается по обратным дугам до пересечения с серым деревом, а затем по той или иной ветви серого дерева до серой вершины. Кроме того, автомат должен откорректировать обратные деревья и пометить пройденную дугу как хорду. Неограниченный автомат может это сделать в своей памяти, но памяти полуробота для этого может не хватить, поэтому ему нужно использовать память вершин. Полуробот должен, во-первых, двигаться по обратным дугам не до первого пересечения с серым деревом, а до корня обратного дерева, а во-вторых, из корня двигаться в начало хорды, корректируя обратные деревья и в конце отмечая хорду в ее начальной вершине. В силу этого, как правило, BFS-алгоритм для полуробота не дает преимуществ по сравнению с DFS-алгоритмом. Следует, однако, заметить, что возможны случаи, когда "лишняя" работа после прохода хорды отсутствует, например, если хорда — это петля. Пример приведен на рис. 9, где L_{dfs} и L_{bfs} — длина обхода по DFS- и BFS-алгоритму соответственно.

"Жадный" алгоритм не использует ни серое дерево, ни прямое или обратное деревья. Каждый раз, когда текущая вершина оказывается черной, автомат выполняет поиск серой вершины: строит по пройденному графу кратчайший путь в серую вершину и проходит его. Как и в случае неориентированного графа, "жадный" алгоритм имеет смысл только для неограниченного автомата.



$x_i > 0.\ x_2+x_3+\ldots+x_n = m-n$

$L_{dfs} = n + n(x_2+x_3+\ldots+x_n) = n + n(m-n) = \Theta(mn).$

$L_{bfs} = n + (1+x_2+1+x_3+\ldots+1+x_n) = n + n-1 + m-n = \Theta(m).$

Рис. 9. Пример преимущества BFS-алгоритма над DFS-алгоритмом для ориентированного графа

Очевидно, что оценка длины обхода каждого из этих трех алгоритмов в наихудшем случае совпадает с оценкой минимального обхода $\boldsymbol{O}(nm)$ или $\boldsymbol{O}(dm)$.

3.4. Робот

Задачу обхода ориентированного графа роботом впервые поставил М. О. Рабин в устной лекции в 1967 г. Некоторые работы по этой теме приведены в табл. 3.

Со временем выяснилось, что основная трудность для робота — это реализация отката, т. е. возврата из конца дуги в ее начало. Для этого используется цикл дуг, о котором упоминалось в подразд. 3.3. Проходя такой цикл, робот при откате по хорде корректирует обратные деревья и помечает хорду, а при откате по дереву удаляет дугу из серого дерева.

Здесь, однако, возникает вопрос: "Как найти начальную вершину дуги?" Она специально никак не помечена, и робот не может запоминать ее номер, поскольку его память ограниченна. Самый простой способ отката предполагает проход по циклу дуг столько раз, какова длина цикла: одна из вершин цикла помечается, и при каждом проходе цикла пометка сдвигается на следующую вершину цикла. Первый робот, основанный на DFS-алгоритме и использующий такой простой откат, предложен в 1970 г. [14]. Суммарно на откаты дополнительно тратилось n^3 проходов дуг. В следующем году был

Таблица 3

Обход роботом неизвестного ориентированного графа

Год	Авторы	Длина обхода	Алгоритм	Тип отката по циклу, оценка отката для цикла длины k
1967	M. O. Rabin	Постановка задачи		
1970	И. Б. Бурдонов	$\boldsymbol{O}(nm + n^3)$	DFS	Простой откат, $\boldsymbol{O}(k^2)$
1971	И. Б. Бурдонов	$\boldsymbol{O}(nm + n^2\log n)$	BFS	Простой откат, $\boldsymbol{O}(k^2)$
1994	Y. Afek, E. Gafni	$\boldsymbol{O}(nm + n^2\log n)$	DFS	Логарифмический, $\boldsymbol{O}(k\log k)$
2003	И. Б. Бурдонов	$\boldsymbol{O}(nm + n^2\log\log n)$ $\boldsymbol{O}(dm + dn\log\log n)$	BFS	Суммарно: по хордам, $\boldsymbol{O}(dm)$; по дереву, $\boldsymbol{O}(dn + dn\log\log n)$
2003	И. Б. Бурдонов	$\boldsymbol{O}(nm + n^2\log^* n)$ $\boldsymbol{O}(dm + dn\log^* n)$	BFS, повторный обход	Суммарно: по хордам, $\boldsymbol{O}(dm)$; по дереву, $\boldsymbol{O}(dn + dn\log^* n)$

Примечание: $\log^* n$ — решение неравенства $1 \leqslant \log^t(n) < 2$ относительно t, где $\log^t = \log \circ \log \circ \ldots \circ \log$ (знак суперпозиции $\circ$ применяется $t - 1$ раз)

предложен робот на основе BFS-алгоритма, что дало суммарное время отката $n^2\log n$ [14].

В 1994 г. появилась работа Афека и Гафни [11], в которой для DFS-алгоритма применялся логарифмический откат. Согласно него все вершины в цикле помечаются, а затем, с учетом четности их числа, пометки удаляются через одну и вычисляется четность числа оставшихся пометок. Такой подход позволил получить такую же оценку $n^2\log n$. В 2003 г. опубликована работа [14], в которой BFS-алгоритм "скрещен" с логарифмическим откатом, что дало оценку $n^2\log\log n$.

Суммарное время откатов сокращается за счет того, что помечаются не все вершины цикла, а только *значимые*. В связи с этим естественен вопрос: "Что такое значимая вершина?" Единственное требование: искомая вершина значимая. Откат по хорде делается тогда, когда робот первый раз по ней прошел, поэтому значимы только начальные вершины однократно пройденных дуг цикла. Это дает суммарную оценку откатов по хордам порядка nm.

Откат по дереву выполняется не на одну дугу, а до серой вершины или развилки дерева, они и считаются значимыми. Для BFS-алгоритма это и дает наилучшую на настоящее время оценку $n^2\log\log n$. В том же 2003 г. предложен способ отката по дереву, который назван *цифровым* [15], с уменьшенной суммарной оценкой (последняя строка табл. 3). К сожалению, это не значит, что теперь второе слагаемое в оценке длины обхода тоже уменьшается. Дело в том, что пройденный граф, как отмечено выше, — это граф 1-го рода, поэтому в нем не один остов, ориентированный к корню, а лес деревьев по одному в каждом компоненте сильной связности. В силу этого длина обхода в общем случае не меняет свой порядок при цифровом откате. Уменьшенная оценка получается, только если специально упорядочить граф, т. е. не при произвольной нумерации дуг, исходящих из вершины. В то же время такое упорядочивание можно сделать при первом обходе роботом, тогда повторный обход выполняется быстрее.

Заключение

Алгоритмы обхода графа одним автоматом, двигающимся по ребрам графа, могут использоваться (и уже используются) в задачах исследования систем, моделируемых графами. К таким системам относятся, например, отдельные компьютерные сети и глобальная сеть Интернет. Кроме того, важнейшей областью применения является тестирование программных и аппаратных систем на основе формальных моделей, сводимых к графам переходов. Практическое использование таких алгоритмов предъявляет к ним требования по объему памяти и по времени работы. Размер памяти отражается в понятиях неограниченного автомата, полуробота и робота (конечного автомата), а время работы оценивается как длина обхода, выполняемого автоматом.

Дальнейшим обобщением этой задачи является исследование графов коллективом взаимодействующих автоматов. Ключевыми вопросами здесь являются типы графов, включая недетерминированные графы и графы, динамически меняющиеся со временем, а также способы взаимодействия автоматов между собой и с графом. Модель коллектива двигающихся автоматов будет рассмотрена во второй статье этой серии, а модель неподвижных автоматов в вершинах графа, обменивающихся между собой сообщениями, пересылаемыми по ребрам графа, будет рассмотрена в третьей статье.

Список литературы

1. **Сетевая топология.** URL: https://ru.wikipedia.org/wiki/сетевая топология (дата обращения 31.05.2016).
2. **Райгородский А. М.** Математические модели интернета // Квант. 2012. № 4. С. 12—16.
3. **Milner R.** A Calculus of Communicating Processes // LNCS. Springer-Verlag. 1980. Vol. 92.
4. **Hoare C. A.R.** Communicating Sequential Processes. Englewood Cliffs, NJ: Prentice Hall International, 1985.
5. **Lynch N. Tuttle M.** An Introduction to Input/Output Automata // CWI Quart. 1989. Vol. 2, N 3. P. 219—246.
6. **Euler L.** Solutio Problematis ad Geometriam Situs Pertinentis // Commentarii Academiae Scientarum Imperialis Petropolitanae. 1736. N 8. P. 128—140.
7. **Bernot G., Gaudel M. C., Marre B.** Software Testing based on Formal Specifications: a Theory and a Tool // Software Engineering Journal. 1991. Vol. 6. N 6. P. 387—405.
8. **Fujiwara S., Bochmann G. V., Khendek F.** et al. Test Selection Based on Finite State Models // IEEE Transactions on Software Engineering. 1991. Vol. 17. N 6. P. 591—603.
9. **Tretmans J.** A Formal Approach to Conformance Testing // Proceedings of the IFIP TC6/WG6.1 Sixth International Workshop on Protocol Test systems VI. 1993. P. 257—276.
10. **Lee D., Yannakakis M.** Principles and Methods of Testing Finite State Machines — A Survey // Proceedings of the IEEE 84. 1996. N 8. P. 1090—1123.
11. **Afek Y., Gafni E.** Distributed Algorithms for Unidirectional Networks // SIAM J. Comput. 1994. Vol. 23. N 6. P. 1152—1178.
12. **Бурдонов И. Б., Косачев А. С., Кулямин В. В.** Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай // Программирование. 2003. № 5. С. 59—69.
13. **Оре О.** Теория графов. М.: Наука, 1968. С. 59, 68.
14. **Бурдонов И. Б.** Обход неизвестного ориентированного графа конечным роботом // Программирование. 2004. № 4. С. 11—34.
15. **Бурдонов И. Б.** Проблема отката по дереву при обходе неизвестного ориентированного графа конечным роботом // Программирование. 2004. № 6. С. 6—29.

Graph Learning by Automaton

I. B. Bourdonov, igor@ispras.ru, **A. S. Kossatchev,** kos@ispras.ru,
Institute for System Programming RAS, Moscow, 109004, Russian Federation

Corresponding author:

Bourdonov Igor B., Leading Researcher, Institute for System Programming RAS, 109004, Moscow, Russian Federation, e-mail: igor@ispras.ru

Received on July 22, 2016
Accepted on August 22, 2016

Graph models of technical objects and processes are used widely in hardware and software dealing with those objects and processes. So, various graph analysis algorithms are important to development of such software.

This article is the first one in a series of publications devoted to graph learning algorithms performed by automata. We consider a case of a graph explored by an automaton, which moves through graph edges (according to their orientation in case of directed graph or in any direction in undirected case), can store some data in graph vertices and read them when visiting a vertex later. The edges leading from a vertex are enumerated (such a graph is called an ordered one) and the automaton staying in a vertex specifies the edge number to move on this edge. We study and design algorithms executed by the automaton that performs graph learning or exploration — construction of a path containing all the graph edges, such a path is called a traversal. We provide exploration algorithm complexity estimations, which depend on the type of graph (directed or undirected, with the structure known or unknown at the start of exploration) or the type of automaton used (with various bounds on available resources and operations). Most attention is given to the case of unknown graph, the case of known structure of the graph is investigated to get the estimation of minimal traversal length. For unknown graph we consider irredundant and free automata. Irredundant automaton can execute an operation returning the number of all edges leading from the current vertex, so it can know all the valid edge numbers just after entering a vertex. Free automaton has no such operation; it can only try to use various numbers to move from the current vertex. If there exists an outgoing edge with such a number, the move is successful and the automaton enters the end vertex of this edge; if there are no such outgoing edges, the automaton remains in the current vertex. Automata types can differ by the size of their internal memory: a robot is a finite automaton with bounded memory; an automaton is a semi-robot if its memory size depends on the graph size, however the graph can't be located in automaton memory; an unbounded automaton can store the entire structure of the graph in its internal memory. We also consider different types of exploration: a primary exploration is performed on the graph without any information in its vertices, a secondary exploration is performed on a graph already explored and having some data stored in its vertices.

In the next papers we plan to present results concerning algorithms of graph exploration performed by several automata working in parallel. A graph explored can be nondeterministic or even dynamic, changing its structure during the exploration.

Keywords: *graph learning, graph exploration, graph traversal, undirected graph, directed graph, labyrinth, ordered graph, numerated graph, unknown graph, automaton, robot*

For citation:

Bourdonov I. B., Kossatchev A. S. Graph Learning by Automaton, *Programmnaya Ingeneria,* 2016, vol. 7, no 11, pp. 498—508.

DOI: 10.17587/prin.7.498-508

References

1. **Network** topology, available at: https://en.wikipedia.org/wiki/Network_topology (data of access 31.05.2016).
2. **Raygorotskiy A. M.** Matematicheskie modeli internet (Mathemathical models of Internet), *Kvant*, 2012, no. 4, pp. 12—16 (in Russian).
3. **Milner R.** A Calculus of Communicating Processes, *LNCS*, *Springer-Verlag*, 1980, vol. 92.
4. **Hoare C. A.R.** *Communicating Sequential Processes*, Englewood Cliffs, NJ,Prentice Hall International, 1985.
5. **Lynch N. Tuttle M.** An Introduction to Input/Output Automata, *CWI Quart*, 1989, vol. 2, no. 3, pp. 219—246.
6. **Euler L.** Solutio Problematis ad Geometriam Situs Pertinentis, *Commentarii Academiae Scientarum Imperialis Petropolitanae*, 1736, no. 8, pp. 128—140.
7. **Bernot G., Gaudel M. C., Marre B.** Software Testing based on Formal Specifications: a Theory and a Tool, *Software Engineering Journal*, 1991, vol. 6, no. 6, pp. 387—405.
8. **Fujiwara S., Bochmann G. V., Khendek F., Amalou M., Ghedamsi A.** Test Selection Based on Finite State Models, *IEEE Transactions on Software Engineering*, 1991, vol. 17, no. 6, pp. 591—603.
9. **Tretmans J.** A Formal Approach to Conformance Testing, *Proceedings of the IFIP TC6/WG6.1 Sixth International Workshop on Protocol Test systems VI*, 1993, pp. 257—276.
10. **Lee D., Yannakakis M.** Principles and Methods of Testing Finite State Machines — A Survey, *Proceedings of the IEEE 84*, 1996, no. 8, pp. 1090—1123.
11. **Afek Y., Gafni E.** Distributed Algorithms for Unidirectional Networks, *SIAM J. Comput.*, 1994, vol. 23, no. 6, pp. 1152—1178.
12. **Bourdonov I. B., Kossatchev A. S., Kuliamin V. V.** Irredundant Algorithms for Traversing Directed Graphs: The Deterministic Case, *Programming and Computer Software*, 2003, vol. 29, no. 5, pp. 245—258.
13. **Ore O.** *Theory of graphs*, Rhod Island, American Math. Society, 1962.
14. **Bourdonov I. B.** Traversal of an Unknown Directed Graph by a Finite Robot, *Programming and Computer Software*, 2004, vol. 30, no. 4, pp. 188—203.
15. **Bourdonov I. B.** Backtracking Problem in the Traversal of an Unknown Directed Graph by a Finite Robot, *Programming and Computer Software*, 2004, vol. 30, no. 6, pp. 305—322.

ИНФОРМАЦИЯ

УДК 004.02 DOI: 10.17587/prin.7.509-514

Д. А. Малахов, аспирант, e-mail: 79155155577@ya.ru,
В. А. Серебряков, д-р физ.-мат. наук, проф., e-mail: serebr@ultimeta.ru,
Московский государственный университет им. М. В. Ломоносова

Методы кластеризации OWL-объектов

Проанализированы подходы к распределению OWL-объектов по нескольким группам так, чтобы наиболее близкие по смыслу объекты находились в одной группе. Предложен способ кластеризации OWL-объектов по их описаниям на естественном языке. Описана область применения этого подхода. Приведен пример использования на практике представленных в статье алгоритмов для горизонтального масштабирования семантического хранилища.

Ключевые слова: *кластеризация онтологии, масштабирование семантического хранилища, кластеризация S-тегов, визуализация онтологий, k-means++, модель S-тега, OWL-объект, разбиение RDF-связей*

Введение

Использование модели RDF [1] вместо реляционной модели является альтернативным подходом к хранению данных. Такой выбор оправдан, когда данные не имеют статической схемы. Он может быть полезен при интеграции данных из разных источников. Модель RDF представляется перспективной, когда на стадии разработки системы хранения данных их схема неясна и конфигурируется на других этапах жизненного цикла системы. Модель RDF предлагает представлять данные в виде RDF-троек:

<субъект, предикат, объект>.

Каждый элемент RDF-тройки характеризуется некоторым уникальным идентификатором URI [1]. Язык OWL [1] позволяет с помощью RDF-троек описывать онтологии, включающие в себя классы и объекты, и задавать правила логического вывода связей между объектами. В ряде задач наличие гибкой схемы данных и использование правил вместо хранения всевозможных связей является преимуществом языка OWL и предоставляет возможность хранить и обрабатывать данные эффективнее других средств.

Семантическим хранилищем будем называть систему, отвечающую за хранение и обработку данных в формате RDF и использующую правила логического вывода связей между объектами языка OWL. Такая система оперирует OWL-объектами. Объект OWL идентифицируется с помощью URI и характеризуется набором RDF-троек, являясь в них субъектом или объектом. Объекты OWL имеют описание на естественном языке, могут быть связаны друг с другом с помощью RDF-троек и правил языка OWL.

Кластеризация OWL-объектов представляет собой процесс разбиения множества OWL-объектов на непересекающиеся группы. Кластеризации OWL-объектов позволяет решить следующие задачи:

- горизонтальное масштабирование семантического хранилища;
- визуализация онтологий;
- создание инструментальных механизмов для выделения новых связей.

Горизонтальное масштабирование семантического хранилища

Объемы семантических хранилищ растут, и, как правило, со временем появляется необходимость в повышении их производительности при обработке запросов пользователей. В связи с этим актуальным становится решение задачи горизонтального масштабирования семантического хранилища [2], а именно — добавление новых серверов с перераспределением нагрузки между ними. В случае семантического хранилища появляется необходимость распределить данные из одного такого хранилища в другие, как это приведено на рис. 1.

Распределенным семантическим хранилищем будем называть систему, оперирующую несколькими семантическими хранилищами. Масштабирование семантического хранилища является актуальной задачей, о чем свидетельствуют результаты, представленные в работе [3]. Предлагаемый в этой работе метод One-Class Clustering Tree характеризует OWL-объекты, которые должны быть в семантическом плане связаны друг с другом. Иной подход состоит в разбиении хранилища с помощью некоторого "семантического хеша", который считается для RDF-троек [4]. Кроме отмеченных направлений, развивается направление кластеризации данных с использованием онтологий [5]. Общим для всех представленных решений является то обстоятельство, что они основаны на кластеризации графов, которая использует связи между объектами.

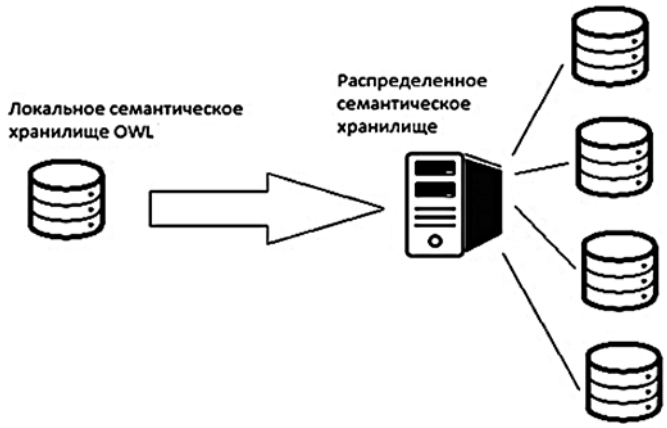


Рис. 1. Схема горизонтального масштабирования

Визуализация онтологий

Кластеризация OWL-объектов может представлять интерес не только для решения задачи масштабирования. Под визуализацией OWL-онтологии будем понимать отображение объектов онтологии и связей между ними. Когда OWL-объектов не так много, то можно одновременно отобразить все объекты и связи между ними в рассматриваемой проблемной области. Однако, если OWL-объектов много, то необходимо выделить только часть из них. Одним из подходов к решению задачи визуализации в этом случае является подход, основанный на разбиении графа с помощью матричной кластеризации по матрице смежности [6].

Создание инструментальных механизмов для выделения новых связей

Предварительное разбиение OWL-объектов на группы можно использовать для последующего выделения связей, аналогично тому, как это сделано для выделения онтологий из текста [7]. Под связями между OWL-объектами в данном случае понимают наборы RDF-троек, где в качестве *субъекта* выступает URI одного OWL-объекта, а в качестве *объекта* выступает URI другого OWL-объекта. Выделение связей является ключевой задачей, которую необходимо решать для интеграции данных разных форматов.

Трудности реализации существующих решений

Семантические хранилища предназначены для обработки SPARQL- запросов. Результатом SPARQL-запроса являются OWL-объекты, релевантные этому запросу. При горизонтальном масштабировании семантического хранилища важно, чтобы результаты SPARQL-запросов были корректными. В результате разбиения группы OWL-объектов часть связей неизбежно нарушается. Таким образом, при кластеризации OWL-объектов важно как можно меньше нарушать связи, используемые при выполнении SPARQL-запросов. Далее будем называть такие связи полезными.

Практически любое семантическое хранилище не является статическим. Появляются новые OWL-объекты, удаляются старые OWL-объекты. Появляются новые связи между OWL-объектами. После кластеризации существующих OWL-объектов, как правило, возникает необходимость отнесения новых OWL-объектов к той или иной группе.

Под устойчивостью разбиения семантического хранилища будем понимать способность уже полученного его разбиения незначительно меняться при незначительном изменении семантического хранилища. Если разбиение является недостаточно устойчивым, то со временем набор полезных в рассматриваемой проблемной области (далее — полезных), но нарушенных этим разбиением связей будет расти. Это обстоятельство потребует повторной кластеризации. Более устойчивое разбиение является предпочтительным, так как позволяет реже проводить повторную кластеризацию.

Связи OWL-объектов могут быть представлены в виде графа. Найденные существующие решения основаны на разбиении графа связей OWL-объектов. Как правило, кластеризация OWL-объектов как графа их связей является оправданной, но существуют некоторые ограничения.

Связей не должно быть много. Сложность алгоритмов кластеризации графов связей OWL-объектов имеет нелинейный характер, поэтому обработка достаточно большого числа связей является затруднительной.

Кроме того, при достаточно большой связности набор OWL-объектов сложно корректно разбить. Любое разбиение будет нарушать большое число полезных связей. В этом случае также не стоит ждать устойчивости разбиения при появлении новых связей.

Связей не должно быть мало. Если OWL-объекты слабо связаны, то после кластеризации получим группы объектов, объединенные вместе почти случайным образом. Такое разбиение не будет устойчивым при появлении новых связей. Таким образом, придется более пристально следить за качеством разбиения и часто проводить повторную кластеризацию.

Не должно быть много информационного шума. Информационным шумом будем называть связи OWL-объектов, которые не являются полезными. Такие связи появляются, как правило, неявным образом, как следствие других связей. Информационный шум негативным образом сказывается на разбиении, но с ним можно бороться установкой приоритетов для связей.

Пример. Рассмотрим книги *A* и *B*. Записи *A* и *B* связаны с городом *C*, в котором они опубликованы. Получается, что объекты *A* и *B* связаны неявным образом через объект *C*. Скорее всего, эта связь никогда не будет использована для выполнения запроса. Аналогично *A* и *B* могут быть связаны через издательство, год издания, автора, классификатор.

Как заранее определить какие связи будут использоваться, а какие нет? Как определить приоритеты связей? Как проверить, что заданные приоритеты подходят для кластеризации? Ответы на эти вопросы к настоящему времени недостаточно формализованы.

Не должно быть ошибочных связей. Если часть связей выделена ошибочно, полученное в результате кластеризации связей OWL-объектов разбиение может быть существенно смещенным. При использовании механизмов по автоматическому выделению связей существует вероятность появления большого числа ошибочных связей и информационного шума.

Пример. Рассмотрим библиографические записи B_1, B_2, B_3. Записи B_1, B_2 связаны с классификатором C_1. Запись B_3 связана с классификатором C_2, но на самом деле запись B_2 ошибочно связали с C_1, вместо этого B_2 должна быть связана с C_2. В результате ошибки и отсутствия связи B_2 и C_2 после кластеризации получим кластеры, в которых B_2 и B_3 разделены. Чтобы исправить ошибку, необходимо заново проводить кластеризацию.

S-тег

Соблюдение перечисленных ограничений усложняется необходимостью подобрать и формализовать критерии качества связей, использующихся для кластеризации.

Под предметной областью будем понимать набор текстовых документов, связанных по смыслу. В случаях, когда метод кластеризации с помощью связей неприменим, можно использовать кластеризацию описаний на естественном языке OWL-объектов по предметным областям.

Основой предлагаемого решения является предположение, что разбиение OWL-объектов по предметным областям позволяет сохранить большее число полезных связей. В этом случае OWL-объект относится к той предметной области, к которой относится его описание.

Задача кластеризации описаний OWL-объектов может быть сведена к задаче кластеризации S-тегов, выделенных из этих описаний.

Определение. Алфавитом будем называть любое конечное непустое множество. Элементы этого множества называются символами данного алфавита.

Пример. В качестве алфавита может выступать алфавит любого естественного языка.

Пусть задан алфавит A.

Определение. Термином алфавита A будем называть любой конечный непустой набор символов алфавита A.

Пример. Слова и словосочетания выбранного алфавита естественного языка являются терминами этого алфавита.

Пусть задано множество терминов T алфавита A.

Определение. Любое непустое подмножество T будем называть S-тегом на множестве T.

Пример. Таким образом, если выбран естественный язык с алфавитом A, выбрано множество слов и словосочетаний T, то предложения, состоящие из терминов множества T, являются S-тегами на множестве T.

Пусть задано множество S-тегов ST.

Определение. Контекстом термина $t \in T$ будем называть множество

$$Context_t = \left\{ct \in T | \exists st \in ST \left(t \in st \wedge ct \in st\right)\right\}.$$

Пример. Контекстом слова или словосочетания являются слова или словосочетания, встретившиеся с ним в одном предложении. Термины, которые имеют слишком большой контекст, как правило, не несут смысловой нагрузки, например, такими терминами являются стоп-слова.

Определение. Пусть $\forall t \in T$ задано множество $R_t \subset T$, такое, что $t \in R_t$. Множество R_t будем называть множеством сужений термина $t \in T$.

Пример. Для каждого слова или словосочетания можно задать множество его синонимов и гипонимов. Гипонимом принято называть слово, выражающее частную сущность по отношению к другому, более общему понятию.

Определение. Расширениями термина $t \in T$ будем называть множество

$$E_t = \left\{t_1 \in T | t \in R_{t_1}\right\}.$$

Пример. Для каждого слова или словосочетания можно задать множество его синонимов и гиперонимов на основе заданных множеств синонимов и гипонимов других слов. Гиперонимом принято называть слово, выражающее общую сущность по отношению к другому, более частному понятию. Если слово w_1 имеет в своем множестве сужений слово w_2, то слово w_1 для слова w_2 является либо синонимом, если $w_1 \in R_{w_2}$, либо гиперонимом.

Определение. Классом термина $t \in T$ будем называть множество

$$Class_t = \left\{st \in ST | t \in st \vee \left(\exists tr \in R_t \wedge tr \in st\right)\right\}.$$

Пример. Рассмотрим некоторое слово или словосочетание w_1 естественного языка с алфавитом A. Тогда классом w_1 будет являться множество всех предложений, в которых встретилось сужение для w_1.

Определение. Сужениями S-тега $st \in ST$ будем называть множество

$$R_{st} = \left\{str \in ST | \forall t \left(t \in st \rightarrow str \in Class_t\right)\right\}.$$

Пример. У каждого слова существует набор слов, являющихся эквивалентными или более узкими по смыслу. То же самое может быть характерно и для предложений. Рассмотрим предложение s_1, оно является сужением для предложений, состоящих из более общих терминов.

Пусть задано множество объектов O.

Пример. В качестве объектов O могут выступать описания OWL-объектов.

Пусть $\forall st \in ST$ задано множество $O_{st} \subset O$.

Пример. В качестве O_{st} для предложения *st* могут быть взяты описания OWL-объектов, из которых выделено предложение *st*.

Определение. Классом S-тега *st* будем называть множество:

$$Class_{st} = \left\{o \in O \mid o \in O_{st} \vee \left(\exists str \in R_{st} \wedge o \in O_{rst}\right)\right\}.$$

Пример. Аналогично тому, как слову можно поставить в соответствие набор предложений, в которых это слово встречается, так и предложению можно поставить в соответствие описания OWL-объектов, в которых встречаются сужения этого предложения. Предложение, для которого рассматривается его класс, может не содержаться в описании OWL-объекта, но существует более узкое по смыслу предложение, выделенное из описания OWL-объекта. Поэтому можно считать, что это предложение тоже характеризует это описание OWL-объекта.

На рис. 2 представлена визуализация связей между терминами, S-тегами и объектами. Явно указано, что объект $O3 \in Class_{S2}$. Кроме того, объекты $O1 \in Class_{S2}$ и $O2 \in Class_{S2}$, так как $S1 \in R_{S2}$. Объект $O4 \notin Class_{S2}$, так как $S3 \notin R_{S2}$. Это следует из того, что термин $T2$ и термины из R_{T2} не включены в S-тег $S3$.

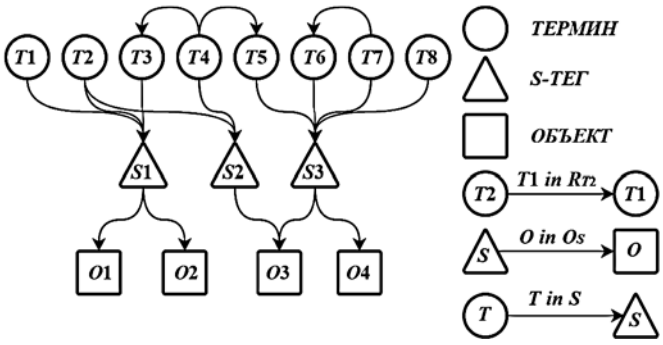


Рис. 2. Взаимодействие терминов, объектов и S-тегов

Кластеризация S-тегов

Под кластеризацией S-тегов будем понимать разбиение S-тегов на группы близких друг к другу S-тегов.

Если термин имеет слишком большой контекст, то он слабо подходит для использования при кластеризации S-тегов, так как является плохим отличительным признаком.

Поэтому множество терминов

$$\left\{t \in T \mid Context_t > Context\_Limit\right\}$$

будет исключено из *T*. Некоторый заданный параметр *Context_Limit* обозначает максимально допустимый размер контекста. Фильтрация терминов с большим контекстом позволяет решить проблему часто встречающихся слов. Служебные термины всегда имеют большой контекст, а ключевые термины, которые тоже часто встречаются, имеют значительно меньший контекст. Подобный подход позволяет убрать общие термины, отделив их от ключевых терминов.

Используя множество расширений E_t для термина $t \in T$ и S-тега $st \in ST$, таких, что $t \in st$, можно посчитать степень w_{st_t} термина *t* в S-теге *st*. Если $st \notin Class_t$, то $w_{st_t} = 0$, иначе

$$w_{st_t} = \max_{tt \in st \wedge t \in E_{tt}} \frac{1}{\sqrt{|E_{tt}|}}.$$

Степень термина характеризует вклад, который вносит термин в смысл S-тега. Если термин не является расширением ни одного из терминов, включенных в этот S-тег, то степень термина в этом теге можно считать равной нулю. Если термин является расширением некоторого термина, включенного в S-тег, то чем больше таких терминов, тем меньше должна быть степень каждого из них. При этом если термин является расширением нескольких терминов, включенных в S-тег, то его степенью должна стать максимальная степень.

Поставим S-тегу *st* в соответствие вектор $\mathbf{v}^{st}$ размерности $|T|$, где каждой компоненте вектора $\mathbf{v}_i^{st} = w_{st_{t_i}}$ соответствует $t_i \in T$.

Множество таких векторов будем обозначать *V*. Таким образом, можно свести задачу кластеризации множества S-тегов к задаче кластеризации векторов в евклидовом пространстве.

Приняв мерой близости S-тегов st_1 и st_2 меру Евклида, получим:

- чем больше элементов во множестве

$$\left\{t \in T \mid st_1 \in Class_t \wedge st_2 \notin Class_t \vee st_2 \in Class_t \wedge st_1 \notin Class_t\right\},$$

тем больше расстояние между st_1 и st_2;

- чем больше $\left(w_{st1_t} - w_{st2_t}\right)$, тем больше расстояние между st_1 и st_2.

Метод кластеризации k-means++ [8] за линейное время способен разбивать множество векторов *V*. Более близкие по евклидовой мере векторы при этом с большей вероятностью оказываются в одном кластере.

Таким образом, с помощью k-means++ можно разбить множество S-тегов на *N* групп.

Чтобы оценить качество полученной кластеризации, могут быть использованы следующие метрики:

- средняя косинусная мера сходства векторов кластеров (Q_1);
- среднее число ненулевых признаков в векторах кластеров (Q_2);
- среднее число признаков, по которым пересекаются векторы кластеров (Q_3);
- средняя частота признаков, по которым пересекаются векторы кластеров, выраженная в процентах (Q_4).

Кластеризация OWL-объектов

Умея проводить кластеризацию S-тегов, можно провести кластеризацию OWL-объектов. Будем обозначать множество OWL-объектов множеством объектов *O*. Для этого необходимо из описаний OWL-объектов выделить S-теги.

Способы выделения S-тегов из описания следующие:

- разбить текст на предложения, которые будут являться S-тегами;
- разбить текст на абзацы, которые будут являться S-тегами;
- использовать информацию о ключевых словах текста, выделять в S-тег только часть предложения или абзаца.

Получив множество S-тегов и проведя их кластеризацию, получим N кластеров *Clusters*. Каждому кластеру $c \in Clusters$ соответствует вектор его центра $\mathbf{v}_c$, где $\mathbf{v}_{c_i}$ является i-й ненулевой компонентой вектора $\mathbf{v}_c$. Необходимо распределить OWL-объекты из множества O по этим кластерам.

Каждый OWL-объект $o \in O$ можно отнести в кластер $c \in Clusters$, где будет максимальным число $\left|\{st | st \in c \wedge o \in Class_{st}\}\right|$. Если существуют несколько кластеров-претендентов, выбирают кластер $c \in Clusters$, имеющий n ненулевых компонент, для которого минимальным является среднеквадратическое отклонение компоненты вектора $\mathbf{v}_c$:

$$K_c = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(v_{c_i} - \frac{1}{n}\sum_{j=1}^{n} v_{c_j}\right)^2}.$$

Значение K_c меньше для тех кластеров, центры которых имеют ненулевые компоненты, слабо отличающиеся друг от друга. В противном случае кластер содержит много векторов, слабо похожих друг на друга. Такое может произойти, например, когда обобщающее свойство некоторого термина в кластере было преувеличено. Величина K_c позволяет "штрафовать" такие кластеры.

Пример использования описанных алгоритмов

В качестве демонстрации эффективности и практической пользы описанных алгоритмов была проведена кластеризация 3 млн библиографических записей Британской Национальной Библиотеки [9]. В данном случае существующие методы кластеризации не подходят, так как рассматриваемые OWL-объекты слабо связаны. В то же время кластеризация OWL-объектов может:

- улучшить визуализацию OWL-объектов, отображая рядом объекты, описания которых близки;
- способствовать выделению связей между OWL-объектами;
- помочь распределить объекты по разным хранилищам с учетом добавления новых связей в будущем.

Было получено 6 млн S-тегов, содержащих 90 тыс. терминов английского языка. В качестве словаря гипонимов и синонимов был использован WordNet [10].

Объекты OWL были разбиты на $N = 10$ кластеров.

Порог фильтрации терминов был выбран 1 %. Оценки качества:

- $Q1$ около 0,35;
- $Q2$ около 50 000;
- $Q3$ около 30 000;
- $Q4$ около 0,007 %.

Пример. В качестве примера того, как были связаны S-теги, можно рассмотреть S-теги, отнесенные в один кластер:

1) Discovering Mathematics: Bk. 1 (Harold Alan Shaw, Frances E. Wright);

2) Discovering Mathematics: Bk. 2: Investigations and Projects (Nigel Peace);

3) Discovering Mathematics 2: a Course for Secondary Schools (H. A. Wright, F. E. Shaw).

Первый и второй S-теги относятся к близким по смыслу библиографическим записям, имеющим разных авторов, разное время и место издания. Первый и третий S-теги также относятся к близким по смыслу библиографическим записям, но имеют одинаковых авторов, причем третий S-тег относится к имеющей ошибку в написании авторов библиографической записи. При более глубоком анализе кластеров могли бы быть выделены связи между этими книгами.

Заключение

Рассмотрены подходы к кластеризации OWL-объектов. Продемонстрирована практическая значимость и актуальность решения этой задачи. Представлены вопросы, которые решают с помощью кластеризации OWL-объектов. Существующие решения используют связи между OWL-объектами, что приводит к ряду ограничений, которые не позволяют считать эти решения универсальными. С помощью предложенного авторами подхода можно обойти часть таких ограничений.

Для осуществления кластеризации описана модель S-тега, позволяющая использовать синонимы и гипонимы. Представлен алгоритм кластеризации S-тегов с демонстрацией возможностей модели.

В качестве примера использования предложенного подхода была проведена кластеризация множества OWL-объектов, обладающего небольшим числом связей. К выбранному множеству неприменимы существующие методы кластеризации, использующие связи между OWL-объектами.

В рамках дальнейших исследований планируется реализация системы индексирования и поиска для S-тегов, а также изучение способов автоматического выделения S-тегов из текста.

Список литературы

1. **McGuinness D. L., Van Harmelen F.** OWL web ontology language overview. W3C recommendation. 10 February 2004. URL: https://www.w3.org/TR/owl-features/

2. **Басипов А. А., Демич О. В.** Семантический поиск: проблемы и технологии //Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2012. № 1. С. 104-111. URL: http://cyberleninka.ru/article/n/semanticheskiy-poisk-problemy-i-tehnologii

3. **Vijayalakshmi A., Babu S., Student P. G.** An incremental and distributed inference method for large-scale ontologies using one-class clustering tree // International Journal of Engineering Research and General Science. 2015. N 3. P. 818—824. URL: http://pnrsolution.org/Datacenter/Vol3/Issue6/98.pdf

4. **Lee K., Liu L.** Scaling queries over big rdf graphs with semantic hash partitioning // Proceedings of the VLDB Endowment. 2013. Vol. 6. N 14. P. 1894—1905.

5. **David C., Olivier C., Guillaume B.** A survey of RDF storage approaches // ARIMA Journal. 2012. Vol. 15. P. 11—35.
6. **Легостаев Г. О.** Визуализация семантических моделей данных // Вопросы современной науки и практики. Университет им. В. И. Вернадского. 2010. № 1—3. С. 73—82. URL: http://vernadsky.tstu.ru/pdf/2010/01/11.pdf
7. **Вороной С. М., Егошина А. А.** Предварительная кластеризация текстовых документов для повышения качества автоматического построения онтологий // Бионика интеллекта. 2013. № 1. URL: http://masters.donntu.org/2014/fknt/filonova/library/article3.htm
8. **Arthur D., Vassilvitskii S.** k-means++: The advantages of careful seeding // Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007. P. 1027—1035.
9. **British** library collection metadata. URL: http://www.bl.uk/bibliographic/download.html
10. **About** WordNet. URL: https://wordnet.princeton.edu/

Methods of OWL Objects Clustering

D. A. Malakhov, 79155155577@ya.ru, **V. A. Serebriakov**, serebr@ultimeta.ru, Lomonosov Moscow State University, Moscow, 119991, Russian Federation

Corresponding author:

Malakhov Dmitriy A., Postgraduate Student, Lomonosov Moscow State University, Moscow, 119991, Russian Federation, e-mail: 79155155577@ya.ru

Received on July 20, 2016
Accepted on July 28, 2016

This paper is devoted to solving the problem of clustering OWL-objects. The popularity and the relevance of this problem are demonstrated in the article. The solution to the problem is also suitable for tasks such as semantic repository scaling, semantic data visualization and extraction of links between OWL-objects. Existing solutions use links between OWL-objects, but there are several limitations. Therefore, these solutions are not universal. Proposed approach overcomes some of these limitations.

Model of S-tag has been introduced in this paper. S-tag allows one to use any thesaurus to add semantics into the classic clustering process. The S-tag clustering algorithm has been introduced here with a demonstration model features. A natural language sentence is the implementation of the S-tag model. Sentences of the OWL-object description can be used as S-tags attached to the OWL-objects. OWL-objects can be distributed across clusters of their S-tags. This approach is suitable when the separation of objects depends on the semantics of OWL-object descriptions.

A set of OWL-objects with a small number of links has been clustered as an example of this approach. The clustering of this set makes finding of new links between OWL-objects possible. This set can't be clustered by existing solutions which use links between OWL-objects.

It is planned to implement of S- tags and search engine indexing. Algorithms for the qualitative detection and extraction of tags in the text have to be also developed. In addition, there are plans for further studying of the properties of the S-tag model and the possibility of using this model in other areas.

Keywords: *ontology clustering, scaling of the semantic storage, S-tag clustering, visualization of ontologies, k-means++, model of S-tag, OWL-object, partition RDF-links*

For citation:

Malakhov D. A., Serebriakov V. A. Methods of OWL Objects Clustering, *Programmnaya Ingeneria*, 2016, vol. 7, no. 11, pp. 509—514.

DOI: 10.17587/prin.7.509-514

References

1. **McGuinness D. L., Van Harmelen F.** OWL web ontology language overview. W3C recommendation. 10 February 2004, available at: https://www.w3.org/TR/owl-features/
2. **Basipov A. A., Demich O. V.** Semanticheskij poisk: problemy i tehnologii (Semantic Search: Issues and Technologies),*Vestnik Astrahanskogo gosudarstvennogo tehnicheskogo universiteta. Serija: Upravlenie, vychislitel'naja tehnika i informatika*, 2012, no. 1, pp. 104—111, available at: http://cyberleninka.ru/article/n/semanticheskiy-poisk-problemy-i-tehnologii (in Russian).
3. **Vijayalakshmi A., Babu S., Student P. G.** An incremental and distributed inference method for large-scale ontologies using one-class clustering tree, *International Journal of Engineering Research and General Science*, 2015, no. 3, pp. 818—824, available at: http://pnrsolution.org/Datacenter/Vol3/Issue6/98.pdf
4. **Lee K., Liu L.** Scaling queries over big rdf graphs with semantic hash partitioning, *Proceedings of the VLDB Endowment*, 2013, vol. 6, no. 14, pp. 1894—1905.
5. **David C., Olivier C., Guillaume B.** A survey of RDF storage approaches, *ARIMA Journal,* 2012, vol. 15, pp. 11—35.
6. **Legostaev G. O.** Vizualizacija semanticheskih modelej dannyh, *Voprosy sovremennoj nauki i praktiki. Universitet im. V. I. Vernadskogo,* 2010, no. 1—3, pp. 73—82, available at: http://vernadsky.tstu.ru/pdf/2010/01/11.pdf (in Russian).
7. **Voronoj S. M., Egoshina A. A.** Predvaritel'naja klasterizacija tekstovyh dokumentov dlja povyshenija kachestva avtomaticheskogo postroenija ontologij, *Bionika intellekta: nauch.-tehn. zhurnal*, 2013, no. 1, available at: http://masters.donntu.org/2014/fknt/filonova/library/article3.htm (in Russian).
8. **Arthur D., Vassilvitskii S.** k-means++: The advantages of careful seeding, *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics*, 2007, pp. 1027—1035.
9. **British** library collection metadata, available at: http://www.bl.uk/bibliographic/download.html
10. **About** WordNet, available at: https://wordnet.princeton.edu/

УДК 519.72 DOI: 10.17587/prin.7.515-521

А. М. Кукарцев, ст. преподаватель, e-mail: amkukarcev@yandex.ru,
Сибирский государственный аэрокосмический университет
имени академика М. Ф. Решетнева, г. Красноярск

О частотных свойствах действий группы Джевонса на булевых функциях*

Рассмотрены частотные свойства действия группы Джевонса на булевы функции. Приведено доказательство сохранения частотных и/или эквивалентных частотным (энтропийным) характеристикам информационных объектов во всех их допустимых алфавитах одновременно, а также доказательство существования канонического разложения любого элемента группы Джевонса. Предложено использование этих свойств для решения ряда сложных математических задач (уравнений) действия группы Джевонса и ее подгрупп на множестве булевых функций. Показаны методы формирования некоторых информационных объектов с одинаковыми и/или эквивалентными частотными (энтропийными) характеристиками во всех их допустимых алфавитах одновременно. Полученные информационные объекты могут использоваться для анализа и модификации алгоритмов обработки информации.

Ключевые слова: *энтропия, действие группы на множестве, частотный анализ, группа Джевонса, булевы функции*

Введение

Многие алгоритмы обработки информации (или информационных объектов, далее — ИО) основаны на статистических методах [1, 2]. Энтропия [3], как оценка, является характеристикой как входных ИО таких алгоритмов, так и выходных. Некоторые алгоритмы используют не только энтропию, но и более детальные оценки — частоты символов. Информационный объект может быть построен на различных алфавитах. В частности, ИО длины, кратной степени двойки, может быть построен на алфавите из одного, двух, четырех, восьми и т. д. символов. Построение ИО с одинаковыми частотными характеристиками **во всех допустимых его алфавитах одновременно** является сложной задачей.

Информация в современных персональных ЭВМ представлена преимущественно в бинарном виде. Специальные методы кодирования позволяют установить инъекцию ИО произвольной длины в ИО длины, кратной степени двойки. При таком кодировании появляется естественная биекция между ИО длины, кратной степени двойки, и булевой функцией (далее — БФ) [4, 5]. Номера битов в двоичном виде фактически являются аргументами БФ, а биты ИО — значениями. Некоторые виды преобразований БФ сохраняют частотные характеристики соответственных им ИО. Такими преобразованиями являются отрицания и/или перестановки аргументов БФ. Оба множества действующих элементов (отрицаний и перестановок аргументов) описываются группами инвертирования [4] и перестановок переменных [6]. Совместно отрицания и перестановки аргументов БФ описываются более крупной структурой — группой Джевонса [4]. Структурно она является полупрямым произведением указанных групп [7].

Цель настоящей статьи: исследовать зависимость частотных характеристик информационных объектов от действий на соответственные им булевы функции элементами группы Джевонса. Эти свойства (зависимые частотные характеристики) могут быть положены в основу методов генерации ИО, обладающих одинаковыми частотными и спектральными распределениями во всех их допустимых алфавитах одновременно. Указанные частотные свойства позволяют исследовать действия группы Джевонса на множестве БФ. Это позволяет решать [8, 9] некоторый класс труднорешаемых математических задач, описанных в работах [10—12].

В заключение даются оценки эффективности генерации таких информационных объектов. Прикладные результаты рассматриваемого теоретического аппарата могут быть использованы для анализа и оптимизации параметров алгоритмов обработки информации, реализованных в различных инженерно-технических решениях [2, 13].

* Работа выполнена при поддержке гранта Президента РФ (проект МД-3952.2015.9).

Определения и обозначения

Пусть n — целое неотрицательное число, $k = 2^n$, i, j — целые неотрицательные индексы (счетчики), не превосходящие k.

Определим $E = \{0, 1\}$ — множество, состоящее из двух элементов. Декартово произведение этого множества на себя определим как $E^n = \underbrace{E \times \cdots \times E}_{n}$. Элементом такого множества будет бинарный вектор (далее — БВ). Для его координат будем использовать числовую нотацию L2R (*left to right*), т. е. большие координаты находятся левее. Обозначим произвольный элемент множества как $x \in E^n$, $x = \{x_{n-1}, \ ..., \ x_0\}$. Для векторов-аргументов БФ также будем применять нотацию L2R.

Булева функция n аргументов $f(x_{n-1}, x_{n-2}, \ ..., \ x_1, x_0)$ реализует отображение $E^n \to E$ [4]:

$$y_f = \{f(11\ldots11), \ f(11\ldots10), \ ..., f(00\ldots01), \ f(00\ldots00)\}, \quad (1)$$

$$(y_f)_j = f(x_{n-1}, x_{n-2}, \ ..., \ x_1, x_0), j = \sum_{i=n-1}^{0} x_i 2^i.$$

Пусть БФ задана через таблицу истинности [5]. Если однозначно определить порядок следования аргументов, то каждой БФ можно поставить в соответствие БВ y_f (1) длины k (столбец значений).

Определение 1. Алфавит A_i — множество, построенное как декартово произведение E^{2^i}.

Бинарный вектор y_f может быть разбит на символы в любом из алфавитов A_i, где i пробегает все значения [0; n]. При этом разбиение начинается с первого значения y_i, и символы не перекрываются. Длина y_f (число символов) в каждом из алфавитов определяется как $l_i = 2^{n-1}$.

Определение 2. Символ алфавита — элемент алфавита A_i, встречаемый в векторе y_f.

Определение 3. Частота символа — число случаев встречи заданного символа из алфавита A_i в векторе y_f.

Определение 4. Частотное распределение y_f (спектр) над алфавитом A_i — отношение $Q_i(y_f) \subset A_i \times [0; k]$, т. е. множество пар символ—частота.

Определение 5. Спектральное распределение y_f над алфавитом A_i — отношение $R_i(y_f) \subset [0; k] \times [0; k]$ (производное множество от $Q_i(y_f)$), элементы которого показывают, как часто повторяются частоты в y_f. В элементе отношения сначала указывается частота из $Q_i(y_f)$, а затем число различных символов, обладающих такой частотой в y_f.

Группу инвертирования переменных (или группу линейных сдвигов [4]) обозначим как $E_n = (E^n, \oplus)$. Группу перестановок переменных (симметрическую группу [6]) обозначим как S_n. Группу Джевонса обозначим как $D_n = E_n \leftthreetimes S_n$ [4]. В силу специфики целевых объектов условимся, что симметрическая группа действует на множестве чисел [0; n − 1].

Множество булевых функций от n аргументов будем обозначать как $B(n)$.

Каноническое разложение элемента группы Джевонса

При действии элементами группы Джевонса на БФ частотные и спектральные распределения эквивалентных им БВ преимущественно сохраняются. Далее для удобства будем подразумевать эквивалентность БФ и БВ. Поведение частотных и спектральных распределений БФ при действии элементов группы Джевонса зависит от самого элемента. Следующие лемма и теорема позволяют разложить элемент группы Джевонса так, чтобы была возможность манипулировать частотными и спектральными распределениями БФ. Такое разложение единственно, и перебор его множителей позволяет однозначно перебрать все элементы группы Джевонса. Далее по тексту будем называть такое разложение каноническим. Перед доказательством леммы и теоремы потребуется ряд определений из работы [14], описывающих механизм действия симметрической группы на БВ.

Важно разделять групповую операцию в симметрической группе и ее действие на БВ. Эмпирически показано, что для задач действия группы Джевонса на БФ удобно задавать гомоморфизм внешнего полупрямого произведения группы Джевонса из симметрической группы в группу автоморфизмов группы инвертирования переменных через само действие элемента на БФ. Такое действие может быть задано не единственным способом. Эмпирически показано, что можно выделить два набора действий: типа А и типа Б. Оба типа действий выражаются друг через друга. При действии элемента группы Джевонса на БВ предпочтителен тип А [14], но при действии на БФ — тип Б [14]. Поэтому невозможно выделить какой-то один тип действий как основной, а второй — как сводимый к нему. В то же время разработчик может выбирать тип действия по своему усмотрению в зависимости от решаемой задачи.

Действием типа А элемента группы $\pi \in S_n$ на БВ $x \in E^n$ будем называть вычисление результата $x' \in E^n$ по следующему правилу: в верхней строке подстановки находятся "старые" индексы, а в нижней строке — "новые":

$$x'_{\pi(i)} = x_i. \quad (1.А)$$

Действием типа Б элемента группы $\pi \in S_n$ на БВ $x \in E^n$ будем называть вычисление результата $x' \in E^n$ по следующему правилу: в верхней строке подстановки находятся "новые" индексы, а в нижней строке — "старые":

$$x'_i = x_{\pi(i)}. \quad (1.Б)$$

Отсюда получим два представления группы Джевонса для $z_0, z_1 \in E_n$ и $\pi_0, \pi_1 \in S_n$:

$$(z_0\pi_0)(z_1\pi_1) = \left(z_0 z_1^{\pi_0^{-1}} \pi_0\pi_1\right), \quad (2.А)$$

$$(z_0\pi_0)(z_1\pi_1) = \left(z_0 z_1^{\pi_0^{-1}} \pi_1\pi_0\right). \quad (2.Б)$$

Элемент группы Джевонса содержит в себе подстановку. Чтобы получить каноническое представление элемента, необходимо использовать специфичное разложение подстановки. При этом известных результатов о разложении подстановки на транспозиции в общей теории [6, 7] недостаточно, потому что в них не учитывается взаимный состав элементов транспозиций.

Лемма 1 (О монотонном представлении подстановки). Пусть k есть количество независимых циклов, включая циклы длины 1, нетривиальной подстановки π группы S_n степени n. Тогда она может быть единственным образом представлена как произведение из $n - k$ транспозиций следующего вида:

$$\pi = \left(0, \pi_0^{-1}(0)\right)\cdots\left(i_0, \pi_{i_0}^{-1}(i_0)\right)\cdots\left(i, \pi_i^{-1}(i)\right)\cdots$$
$$\cdots\left(i_1, \pi_{i_1}^{-1}(i_1)\right)\cdots\left(n-2, \pi_{n-2}^{-1}(n-2)\right),$$

причем $0 \leqslant i_0 < i < i_1 < n-1$. В произведение включаются только транспозиции, соответствующие точкам $i : \pi(i) \neq i$ и $i < \pi_i^{-1}(i)$. Промежуточные подстановки вычисляются рекурсивно как $\pi_{i+1} = \left(i, \pi_i^{-1}(i)\right)\pi_i$, при этом $\pi_0 = \pi$.

Доказательство. Представим подстановку в виде таблицы $\pi = \begin{pmatrix} \ldots & i & \ldots \\ \ldots & \pi(i) & \ldots \end{pmatrix}$, и пусть $i \neq \pi(i)$. Подстановка раскладывается в произведение независимых циклов, в один из которых входит точка i. Тогда верно представление:

$$\pi = (\ldots)\ldots\left(\ldots, \pi^{-1}(i),\, i,\, \pi(i), \ldots\right)\ldots(\ldots),$$

или в виде таблицы

$$\pi = \begin{pmatrix} \ldots & \pi^{-1}(i) & \ldots & i & \ldots \\ \ldots & i & \ldots & \pi(i) & \ldots \end{pmatrix}.$$

Определим произведение $\rho[i, \pi] = \left(i, \pi^{-1}(i)\right)\pi$ и вычислим его:

$$\begin{pmatrix} \ldots & i & \ldots & \pi^{-1}(i) & \ldots \\ \ldots & \pi^{-1}(i) & \ldots & i & \ldots \end{pmatrix}\begin{pmatrix} \ldots & \pi^{-1}(i) & \ldots & i & \ldots \\ \ldots & i & \ldots & \pi(i) & \ldots \end{pmatrix} =$$
$$= \begin{pmatrix} \ldots & i & \ldots & \pi^{-1}(i) & \ldots \\ \ldots & i & \ldots & \pi(i) & \ldots \end{pmatrix},$$

или в виде цикловой структуры

$$\rho[i, \pi] = (\ldots)\ldots\left(\ldots, \pi^{-1}(i),\, \pi(i), \ldots\right)\ldots(\ldots).$$

Тогда верны следующие утверждения: имеет место тождество $\rho[i, \pi](i) \equiv i$ и, в случае $\pi(i) \neq i$, длина одного из циклов $\rho[i, \pi]$ на единицу меньше, чем π.

Для удобства примем, что $\pi_0 = \pi$ и $\pi_{i+1} = \rho[i, \pi_i]$. Откуда $\pi_{i+1} = \left(i, \pi_i^{-1}(i)\right)\pi_i$ или $\pi_i = \left(i, \pi_i^{-1}(i)\right)\pi_{i+1}$. Последнее соотношение позволяет представить подстановку как произведение транспозиции, двигающей точку i, и остаток, стабилизирующий ее. Каждый цикл длины $l > 1$ исходной подстановки раскладывается на произведение из $l - 1$ транспозиций независимо от других циклов. Откуда общее число транспозиций в произведении будет равно сумме длин всех циклов за вычетом их числа, т. е. $n - k$.

Запустим рекурсию по всем значениям i начиная с нуля:

$$\pi_0 = \left(0, \pi_0^{-1}(0)\right)\pi_1 = \left(0, \pi_0^{-1}(0)\right)\left(1, \pi_1^{-1}(1)\right)\pi_2 = \ldots$$
$$= \left(0, \pi_0^{-1}(0)\right)\left(1, \pi_1^{-1}(1)\right)\ldots\left(n-2, \pi_{n-2}^{-1}(n-2)\right)\pi_{n-1}.$$

Так как на каждом шаге рекурсии длина какого-то из циклов уменьшается на единицу (или не изменяется, но тогда текущая точка неподвижна в исходной подстановке), то в π_{n-1} неподвижными будет $n - 1$ точек, т. е. тривиальная подстановка. Откуда получаем разложение

$$\pi = \left(0, \pi_0^{-1}(0)\right)\left(1, \pi_1^{-1}(1)\right)\ldots\left(i, \pi_i^{-1}(i)\right)\ldots\left(n-2, \pi_{n-2}^{-1}(n-2)\right),$$

из которого нужно убрать тождественные части вида $\pi_{i+1} \equiv \pi_i$. Транспозиция является инволюцией, и поэтому порядок точек в цикле не важен, тогда для наглядности пусть $i < \pi_i^{-1}(i)$. Единственность разложения следует из построения. Что и требовалось доказать.

Например, для $n = 8$, $p \in S_8$:

$$\pi = \begin{pmatrix} 76543210 \\ 42637150 \end{pmatrix} = (7, 4, 3)(6, 2, 1, 5)(0)$$

всего циклов $k = 3$. Всего транспозиций в произведении будет $n - k = 5$. Для удобства представления сведем шаги рекурсии в таблице.

В результате вычислений по данным этой таблицы получим разложение $\pi = (1, 2)(2, 6)(3, 4)(4, 7)(5, 6)$.

Пусть $b_{n-1}, \ldots, b_i, \ldots, b_0, z \in E_n$, причем $b_i = \{0, \ldots, 0, \ldots, z_i, \ldots, 0, \ldots, 0\}$, т. е. содержит на позиции i значение координаты i БВ z, а на остальных — нули.

Теорема 1 (О каноническом представлении элемента группы Джевонса). Любой элемент группы Джевонса $(z\pi) \in D_n$ представим единственным образом в виде произведения:

$$\left(b_{n-1}\left((n-1)\, j_{n-1}\right)\right)\cdots\left(b_{i_1}\left(i_1 j_{i_1}\right)\right)\cdots$$
$$\cdots\left(b_i\left(i j_i\right)\right)\cdots\left(b_{i_0}\left(i_0 j_{i_0}\right)\right)\cdots\left(b_0\left(0\, j_0\right)\right),$$

где $0 \leqslant i_0 < i < i_1 \leqslant n-1$. Элементы симметрической группы $(i, j_i) : i \leqslant j_i$ имеют порядок не более 2 и, в случае транспозиции, соответствуют разложению по лемме 1 (для типа А будет π^{-1} и для типа Б — π).

Доказательство. Вычислим указанное в условии произведение. Элемент группы E_n первой пары произведения будет равен $b_{n-1} b_{n-2}^{\left(n-1, j_{n-1}\right)^{-1}}$ независимо от типа (по формуле (2.А) или (2.Б)). Из условия $0 \leqslant i_0 < i < i_1 \leqslant n-1$ (условие монотонности для подстановки) верно, что $b_{n-2}^{\left(n-1, j_{n-1}\right)^{-1}} = b_{n-2}$, и элемент E_n

Шаги рекурсии вычисления представления подстановки

Индекс шага	Шаг рекурсии	Транспозиция
0, 1	$\pi_1 \equiv \pi_0 \equiv \pi = (7,\ 4,\ 3)(6,\ 2,\ 1,\ 5)$	$\left(1,\ \pi_1^{-1}(1)\right) = (1,\ 2)$
2	$\pi_2 = (1,\ 2)\pi_1 = \begin{pmatrix}76543210\\76543120\end{pmatrix}\begin{pmatrix}76543210\\42637150\end{pmatrix} = \begin{pmatrix}76543210\\42637510\end{pmatrix} = (7,\ 4,\ 3)(6,\ 2,\ 5)$	$\left(2,\ \pi_2^{-1}(2)\right) = (2,\ 6)$
3	$\pi_3 = (2,\ 6)\pi_2 = \begin{pmatrix}76543210\\72543610\end{pmatrix}\begin{pmatrix}76543210\\42637510\end{pmatrix} = \begin{pmatrix}76543210\\45637210\end{pmatrix} = (7,\ 4,\ 3)(6,\ 5)$	$\left(3,\ \pi_3^{-1}(3)\right) = (3,\ 4)$
4	$\pi_4 = (3,\ 4)\pi_3 = \begin{pmatrix}76543210\\76534210\end{pmatrix}\begin{pmatrix}76543210\\45637210\end{pmatrix} = \begin{pmatrix}76543210\\45673210\end{pmatrix} = (7,\ 4)(6,\ 5)$	$\left(4,\ \pi_4^{-1}(4)\right) = (4,\ 7)$
5, 6	$\pi_5 = (4,\ 7)\pi_3 = \begin{pmatrix}76543210\\46573210\end{pmatrix}\begin{pmatrix}76543210\\45673210\end{pmatrix} = \begin{pmatrix}76543210\\75643210\end{pmatrix} = (6,\ 5)$	$\left(5,\ \pi_5^{-1}(5)\right) = (5,\ 6)$

первой пары произведения будет равен $b_{n-1}b_{n-2}$. Другими словами, при вычислении произведения слева направо подстановки в силу монотонности действуют тривиально на соответствующие БВ. Значит, в рамках всего произведения будет сформирован z.

Результирующая подстановка будет по-разному определяться для типа А и типа Б. Для типа Б (по формуле (2.Б)) получим $(n-2,\ j_{n-2})(n-1,\ j_{n-1})$, и в рамках всего произведения будут собираться (без циклов длины один) транспозиции справа налево, что в итоге даст разложение по лемме 1. Для типа А будет аналогичное разложение, но записанное в обратном порядке. Так как произведение составлено из транспозиций, обратный порядок описывает разложение подстановки π^{-1}.

Так как z представляется через порождающие единственным образом и монотонное представление подстановки π по лемме 1 единственно, то это влечет единственность канонического разложения ($z\pi$). Что и требовалось доказать.

Частотные свойства действия группы Джевонса на БФ

Каноническое разложение элемента группы Джевонса по теореме 1 позволяет действовать множителями этого разложения на БФ. Определим $c_{n-1},\ ...,\ c_0 \in E^n$, причем каждый c_i имеет на позиции i значение 1, а на остальных — 0, т. е. все множество c_i представляет собой порождающее множество группы E_n.

Теорема 2 (Об инвариантности частотных спектров при действии E_n). Если элемент $c_i \in E_n$ действует на БФ $f \in B(n)$, то частотные распределения БВ y_f инвариантны относительно этого действия для алфавитов $A_{i'} : i' \le i, i \in [0; n-1]$, а спектральные распределения y_f инвариантны относительно этого же действия для алфавитов A_i.

Доказательство. Элемент c_i, действуя на БФ $f(x)$, фактически реализует эквивалентное действие $y_f^{\pi_{c_i}}$, где $\pi_{c_i} \in S_k$. Рассмотрим цикловую структуру подстановки π_{c_i}. Во-первых, реализуется инверсия $x^{c_i} = \left\{x_{n-1},\ ...,\ \overline{x_i},\ ...,\ x_0\right\}$, и тогда верно $\left(x^{c_i}\right)^{c_i} = x$, и, как следствие, π_{c_i} — инволюция и все точки подстановки подвижны. Поэтому вся подстановка состоит из 2^{n-1} независимых циклов длины 2 вида $\left(\left\{x_{n-1},\ ...,\ x_i,\ ...,\ x_0\right\}, \left\{x_{n-1},\ ...,\ \overline{x_i},\ ...,\ x_0\right\}\right)$. Во-вторых, инверсия координаты аргумента есть фактически арифметическое сложение вида $\left(x,\ x+2^i\right)$ в случае $0 \to 1$ (в случае $1 \to 0$ — арифметическое сложение вида $\left(x+2^i,\ x\right)$ — т. е. то же самое). Поэтому сохраняется взаимное положение точек подстановки или пара $x,\ x'$ перейдет в пару $x+2^i,\ \ x'+2^i$ в подмножествах точек числом 2^i. При этом символы алфавитов вплоть до A_i не меняются, а лишь переставляются в y_f. Как следствие, частотные распределения y_f для этих алфавитов инвариантны относительно действия. В-третьих, каждый символ алфавита с индексом больше i (обозначим как $i' > i$) может рассматриваться как упорядоченная комбинация символов алфавита A_i, так как размер каждого символа любого алфавита $A_{i'}$ кратен размеру символа алфавита A_i. Размер такой комбинации будет $2^{i'}/2^i = 2^{i'-i}$, т. е. степень двойки. Действие элемента c_i приводит к тому, что меняются местами четные и нечетные символы алфавита A_i, тем самым выполняется одна и та же перестановка символов в каждой упорядоченной комбинации (в каждом символе) любого алфавита $A_{i'}$. Так как перестановка одна и та же, то фактически выполняется биекция символов любого алфавита $A_{i'}$ на какие-то другие символы этого же алфавита. Поэтому, независимо от выбора i', спектральные распределения для y_f инварианты для $i' > i$ относительно действия. Из инвариантности частотных распределений следует инвариантность спектральных

распределений y_f для $i' \leqslant i$. Что и требовалось доказать.

Энтропия является функцией, зависящей только от частот, а не от символов частотного распределения. Значение частот и их количество определяются спектральным распределением, поэтому верно следствие.

Следствие 1. Энтропия БВ y_f инвариантна во всех алфавитах A_i, $i \in [0; n-1]$ относительно действия любого элемента E_n на БФ f.

Теорема 3 (Об инвариантности частотных спектров при действии S_n). Если транспозиция $(i, j) \in S_n$, $i, j \in [0; n-1] : i < j$ действует на БФ $f \in B(n)$, то частотные распределения БВ y_f инвариантны относительно этого действия для алфавитов $A_{i'} : i' \leqslant i$, а спектральные распределения y_f инвариантны относительно этого же действия для алфавитов $A_{i'}$ и $A_{j'} : j' > j$.

Доказательство. Транспозиция (i, j), действуя на БФ $f(x)$, фактически реализует эквивалентное действие $y_f^{\pi_{(i,j)}}$, где $\pi_{(i,j)} \in S_k$. Рассмотрим цикловую структуру подстановки $\pi_{(i,j)}$. Во-первых, реализуется перестановка координат $x^{(i,j)} = \{x_{n-1}, \ldots, x_i, \ldots, x_j, \ldots, x_0\}$, и тогда верно $\left(x^{(i,j)}\right)^{(i,j)} = x$ и, как следствие, $\pi_{(i,j)}$ — инволюция, и только половина точек подстановки подвижны (где $x_i \neq x_j$). Поэтому вся подстановка состоит из 2^{n-2} независимых циклов длины 2 вида

$$\left(\{x_{n-1}, \ldots, x_j, \ldots, x_i, \ldots, x_0\}, \{x_{n-1}, \ldots, x_i, \ldots, x_j, \ldots, x_0\}\right).$$

Во-вторых, транспозиция координат аргумента есть фактически арифметическое сложение вида $\left(x, x - 2^j + 2^i\right)$ в случае $x_i = 0$, $x_j = 1$ (иначе, если $x_i = 1$, $x_j = 0$, то вида $(x + 2^j - 2^i, x)$) или тривиальное преобразование в случае $x_i = x_j$. Поэтому сохраняется взаимное положение точек подстановки или пара x, x' перейдет в пару $x - 2^j + 2^i$, $x' - 2^j + 2^i$ в подмножествах точек числом 2^i. При этом символы алфавитов вплоть до A_i не меняются, а лишь переставляются в y_f. Как следствие, частотные распределения y_f для этих алфавитов инвариантны относительно действия. В-третьих, каждый символ алфавита с индексом больше i (обозначим как $i' : j \geqslant i' > i$) может рассматриваться как упорядоченная комбинация символов алфавита A_i, так как размер каждого символа любого алфавита $A_{i'}$ кратен размеру символа алфавита A_i. Размер такой комбинации будет $2^{i'}/2^i = 2^{i'-i}$, т. е. степень двойки. Вплоть до алфавита A_j включительно упорядоченные комбинации символов A_i будут отображаться в общем случае не взаимно однозначно, потому что перестановка будет менять качественный состав комбинации (изменять символы алфавита A_i) и, как следствие, не будет взаимной однозначности символов $A_{i'}$. Начиная с алфавита $A_{j'}$, где $j' > j$, упорядоченные комбинации символов A_i отображаются уже взаимно однозначно, потому что перестановка фактически реализуется над множеством символов алфавита A_i внутри символов алфавитов A_{j+1} и, как следствие, алфавитов с большим индексом. Фактически выполняется биекция символов любого алфавита $A_{j'}$ на какие-то другие символы этого же алфавита. Поэтому, независимо от выбора j', спектральные распределения для y_f инварианты для $j' > j$ относительно действия. Из инвариантности частотных распределений следует инвариантность спектральных распределений y_f для $i' \leqslant i$. Что и требовалось доказать.

Энтропия является функцией, зависящей только от частот, а не от символов частотного распределения. Значение частот и их число определяются спектральным распределением, поэтому верно представленное ниже следствие.

Следствие 2. Энтропия БВ y_f инвариантна для алфавитов индексов до i включительно и больше j относительно действия транспозиции (i, j), $i, j \in [0; n-1] : i < j$ группы S_n на БФ f.

Для исследования зависимости энтропии ИО от действия на эквивалентную БФ элементом группы Джевонса необходимо выполнить его каноническое разложение. Далее применить теорему 2 и теорему 3 к каждому множителю канонического разложения. При этом множители необходимо анализировать в порядке, обратном указанному в теореме 1, т. е. от меньших индексов к большим.

Заключение

Приведенные теоремы доказывают, что спектральные распределения инвариантны при рассмотренных действиях групп (с учетом ограничений теоремы 3). Под эффективностью генерации ИО для прикладного применения определим числа получаемых различных ИО. Для этого воспользуемся доказанным фактом из теории перечислений Пойа [4, 10]; для подавляющего большинства БФ подгруппы инерций в группах инвертирования и перестановок переменных тривиальны.

При действиях группой E_n число различных ИО с одинаковой энтропией во всех допустимых его алфавитах одновременно равно:

$$count_{E_n} \approx 2^n. \tag{3}$$

Для действий группы S_n оценки сложны и не всегда соответствуют (3), так как в каких алфавитах будет сохраняться энтропия — зависит от действующего элемента.

Предлагаемый математический аппарат можно использовать для генерации ИО при тестировании и разработке различных алгоритмов обработки информации, основывающихся на частотных и спектральных распределениях.

Список литературы

1. **Шеннон К.** Работы по теории информации и кибернетике. Пер. с англ. /под ред. Р. Л. Добрушина и О. Б. Лупанова с пред. А. Н. Колмогорова. М.: Издательство иностранной литературы, 1963. 832 с.
2. **Сэломон Д.** Сжатие данных, изображений и звука. Пер. с англ. В. В. Чепыжова. М.: Техносфера, 2004. 368 с.
3. **Хэмминг Р. В.** Теория кодирования и теория информации: Пер. с англ. М: Радио и связь, 1983.
4. **Логачев О. А., Сальников А. А., Ященко В. В.** Булевы функции в теории кодирования и криптологии. М.: МЦМНО, 2004. 470 с.
5. **Марченко С. С.** Замкнутые классы булевых функций. М.: ФИЗМАТЛИТ, 2000. 128 с.
6. **Супруненко Д. А.** Группы подстановок. Мн.: Навука і тэхніка, 1996. 366 с.
7. **Каргаполов М. И., Мерзляков Ю. И.** Основы теории групп. 3-е изд., перераб. и доп. М.: Наука, 1982. 288 с.
8. **Кукарцев А. М., Кузнецов А. А.** О применении частотного анализа для решения проблемы расстояний в группе Джевонса, индуцирующей действие на множестве булевых функций // Международная конференция "Мальцевские чтения" 10 — 12 ноября 2014 г. Тезисы докладов. Новосибирск, 2014. С. 67.
9. **Кукарцев А. М., Кузнецов А. А.** О применении частотного анализа для решения некоторых групповых уравнений индукции действия группы Джевонса и ее подгрупп на множестве булевых функций // Дискретные модели в теории управляющих систем: IX Международная конференция, Москва и Подмосковье, 20 — 22 мая 2015 г.: Труды. Москва, 2015. С. 136—138.
10. **Глухов М. М., Ремизов А. Б., Шапошников В. А.** Обзор по теории k-значных функций. Часть 1. Справочное пособие/ Ред. Н. Р. Емельянов. Заказ № 163ф. М.: Типография в/ч 33965, 125040, Москва А-40, 1988. 153 с.
11. **Якубайтис Э. А.** Субклассы и классы булевых функций // Автоматика и вычислительная техника. № 1. Рига: Зинатне, 1974. С. 1—8.
12. **Golomb S. W.** On classification of Boolean functions // IRE, Trans.circuit theory. 1959. N 6, Spec. Suppl., P. 176—186.
13. **Бентли Дж.** Жемчужины программирования. 2-е издание. Пер. с англ. Д. Солнышков / под ред. А. Пасечник. СПб.: Питер, 2002. 272 с.
14. **Кукарцев А. М., Кузнецов А. А.** О конструктивном представлении группы Джевонса для инженерно-технических решений обработки информации // Программная инженерия. 2015. № 11. С. 25—33.

On Frequency Characteristics Jevons Group Action on Boolean functions

A. M. Kukartsev, amkukarcev@yandex.ru, Siberian State Aerospace University named after academician M. F. Reshetnev, Krasnoyarsk, 6600014, Russian Federation,

Corresponding author:

Kukartsev Anatolii M., Senior Lecturer, Siberian State Aerospace University named after academician M. F. Reshetnev, Krasnoyarsk, 6600014, Russian Federation,
e-mail: amkukarcev@yandex.ru

Received on June 13, 2016
Accepted on August 29, 2016

Many data processing algorithms are based on statistical methods. Entropy, as the assessment, is characteristic of input and output information objects of such algorithms. Some algorithms use not only entropy but a more detailed assessment — frequencies of symbols. An information object can be created using different alphabets. In particular the object of length 2^n can be constructed on an alphabet of one, two, four, eight, etc. symbols. Construction of information objects with the same frequency characteristics in all its permissible alphabets simultaneously is a difficult problem. The data in modern computers is presented mostly in binary form. Special coding methods allow you to set an injection of information objects of any length into objects of length 2^n. With this coding appears a natural bijection between the information object of length 2^n and a Boolean function. Bit numbers in the binary form are actually arguments of a Boolean function, and bits of the information object are its values. Some transformations of Boolean functions preserve frequency characteristics of theirs information objects. Such transformations of the Boolean function are complements and/or permutations of its arguments. Both sets of acting elements are described by groups of inversion and permutation of variables. Together complements and permutations of arguments are presented by a larger structure — the Jevons group, which is a semi-direct product of these groups.

We study the frequency properties of the Jevons group actions on Boolean functions in this article. We prove that frequency characteristics of the information object preserve in all permissible alphabets simultaneously. Also we show that there is a canonical decomposition of any element of the Jevons group. We offer to use these properties for solution of complex mathematical problems related with actions of the Jevons group and it subgroups on the set of Boolean functions. We show methods of creating information objects with the identical and/or equivalent frequency characteristics in all its permissible alphabets simultaneously. The obtained objects can be used for analysis and modification of data processing algorithms.

Keywords: *Shannon entropy, action on the set, frequency analysis, the Jevons group, Boolean functions*

Acknowledgements: *This work was supported by the Grant of Russian Federation President, project nos. MD-3952.2015.9*

For citation:

Kukartsev A. M. On Frequency Characteristics Jevons Group Action on Boolean Functions, *Programmnaya Ingeneria,* 2016, vol. 7, no. 11, pp. 515—521.

DOI: 10.17587/prin.7.515-521

References

1. **Shennon K.** *Raboty po teorii informacii i kibernetike* (Works on information theory and cybernetics). Per. s angl. Ed. R. L. Dobrushina i O. B. Lupanova s pred. A. N. Kolmogorova, Moscow, Izdatel'stvo inostrannoj literatury, 1963, 832 p. (in Russian).

2. **Salomon D.** *A guide to data compression methods*, New York, Springer-Verlag, 2006, 1092 p.

3. **Hamming Richard W.** *Coding and information Theory*, Englewood Cliffs, New Jersey, Prentice Hall, 1980.

4. **Logachjov O. A., Sal'nikov A. A., Jashhenko V. V.** *Bulevy funkcii v teorii kodirovanija i kriptologii* (Boolean functions in coding theory and cryptology), Moscow, MCMNO, 2004, 470 p. (in Russian).

5. **Marchenko S. S.** *Zamknutye klassy bulevyh funkcij* (Post's lattice), Moscow, FIZMATLIT, 2000, 128 p. (in Russian).

6. **Suprunenko D. A.** *Gruppy podstanovok* (Groups of permutations), Minsk, Navuka i tehnika, 1996, 366 p. (in Russian).

7. **Kargapolov M. I., Merzljakov Ju. I.** *Osnovy teorii grupp* (Basics of group theory), 3-e izd., pererab. i dop., Moscow, Nauka, 1982, 288 p. (in Russian).

8. **Kukartsev A. M., Kuznetsov A. A.** O primenenii chastotnogo analiza dlja reshenija problemy rasstojanij v gruppe Dzhevonsa, inducirujushhej dejstvie na mnozhestve bulevyh funkcij (The use of frequency analysis to solve the problem of distances in the group Jevons inducing action on the set of Boolean functions), *Mezhdunarodnaja konferencija Mal'cevskie chtenija 10 — 12 November 2014, Tezisy dokladov,* Novosibirsk, 2014, pp. 67—67. (in Russian).

9. **Kukartsev A. M., Kuznetsov A. A.** O primenenii chastotnogo analiza dlja reshenija nekotoryh gruppovyh uravnenij indukcii dejstvija gruppy Dzhevonsa i ejo podgrupp na mnozhestve bulevyh funkcij (The use of frequency analysis to solve some group induction equations action Jevons group and its subgroups on the set of Boolean functions), *Diskretnye modeli v teorii upravljajushhih sistem: IX Mezhdunarodnaja konferencija, Moskva i Podmoskov'e 20 — 22 May 2015, Trudy,* Moskva, 2015, pp. 136—138. (in Russian).

10. **Gluhov M. M., Remizov A. B., Shaposhnikov V. A.** *Obzor po teorii k-znachnyh funkcij. Chast' 1. Spravochnoe posobie* (Review on the Theory of k-valued functions. Part 1. A Reference Guide). Ed. N. R. Emel'janov. Zakaz № 163f, Moscow, Tipografija v/ch 33965, 1988, 153 p. (in Russian).

11. **Jakubajtis Je. A.** Subklassy i klassy bulevyh funkcij (Sub class and the class of Boolean functions), *Avtomatika i Vychislitel'naja Tehnika*, 1974, no. 1, pp. 1—8 (in Russian).

12. **Golomb S. W.** On classification of Boolean functions, *IRE, Trans.circuit theory, Spec.Suppl.*, 1959, no. 6, pp. 176—186.

13. **Bentley J.** *Programming pearls (2nd edition)*, New Jersey, Addison-Wesley, 1999. 256 p.

14. **Kukartsev A. M., Kuznetsov A. A.** O konstruktivnom predstavlenii gruppy Dzhevonsa dlja inzhenerno-tehnicheskih reshenij obrabotki informacii (Constructive representation of the Jevons group for engineering solutions of information processing), *Programmnaya Ingeneria*, 2015, no. 11, pp. 25—33 (in Russian).

ИНФОРМАЦИЯ

3—7 апреля 2017 г. в Казанском (Приволжском) федеральном университете состоится международная научная конференция, одиннадцатая в серии ежегодных конференций, посвященных развитию и применению параллельных вычислительных технологий в различных областях науки и техники

"Параллельные вычислительные технологии (ПаВТ) 2017

Главная цель конференции — предоставить возможность для обсуждения перспектив развития параллельных вычислительных технологий и представления результатов, полученных ведущими научными группами в использовании суперкомпьютерных технологий для решения задач науки и техники.

Тематика конференции покрывает все аспекты применения высокопроизводительных вычислений в науке и технике, включая приложения, аппаратное и программное обеспечение, специализированные языки и пакеты.

В первый день работы конференции будет объявлена 26-я редакция списка Top50 самых мощных компьютеров СНГ.

Во все дни работы конференции будет действовать суперкомпьютерная выставка, на которой ведущие производители аппаратного и программного обеспечения представят свои новейшие разработки в области высокопроизводительных вычислений.

Официальный сайт конференции: http://agora.guru.ru/pavt2017/

УДК 681.5.015.42 DOI: 10.17587/prin.7.522-528

К. В. Мартиросян, канд. техн. наук, доц., e-mail: kv1961@live.ru,
А. В. Мартиросян, аспирант, e-mail: martalex11@mail.ru,
Северо-Кавказский федеральный университет, г. Ставрополь

Синтез распределенной системы управления пространственно-неоднородным гидрогеологическим объектом

Представлены подходы к решению задачи синтеза распределенной системы управления гидрогеологическим объектом. Приведена математическая модель объекта управления, описывающая гидродинамические процессы, протекающие в месторождении минеральных вод в процессе его эксплуатации. Предложено описание граничных условий с помощью уравнений Дарси. Учтена пространственная неоднородность коэффициента перетекания. Рассчитаны параметры регулятора распределенной системы. Представлены результаты компьютерного эксперимента, показывающие адекватность модели.

Ключевые слова: *системы управления, распределенные системы, системный синтез, математическое моделирование, модель гидрогеологического объекта*

Введение

В работе представлены подходы к применению методов синтеза регулятора для решения задачи организации оптимального управления технологическим процессом. В качестве объекта управления рассмотрен пространственно-неоднородный гидрогеологический объект. Наиболее известными методами синтеза управления являются метод аналитического конструирования оптимальных регуляторов (АКОР), метод аналитического конструирования агрегированных регуляторов (АКАР) и частотный метод синтеза регулятора. Одной из областей применения таких методов является организация управления добычей минеральной воды. Поиск оптимальных решений в данной области имеет большое значение для регионов, располагающих действующими месторождениями гидроминеральных ресурсов [9]. Подробное описание использования упомянутых выше методов для решения задач этой области представлено в трудах И. М. Першина и А. В. Малкова [1].

Так как гидрогеологический объект является пространственно-неоднородным (распределенным), то необходим синтез распределенного регулятора. В ходе исследований было выявлено, что для синтеза регулятора данного типа наиболее эффективен частотный метод [4].

Целью исследования, результаты которого представлены далее, является разработка системы управления пространственно-неоднородным гидрогеологическим объектом. Модель распределенной системы (месторождения) как объект управления на входе обрабатывает данные о гидродинамических параметрах, об интенсивности входного воздействия (объема добычи) и о пространственной структуре месторождения. Результатом моделирования является пространственное распределение давления воды в пластах месторождения, а также временные характеристики развития процесса [1, 2]. Обработка результатов моделирования позволяет рассчитать параметры регулятора объекта управления, обеспечивающие устойчивость системы. Таким образом, синтез регулятора дает возможность определить параметры режима эксплуатации месторождения [6, 7].

Описание объекта управления

В качестве объекта управления предложено рассматривать Нагутское месторождение. Данное месторождение является одновременно одним из наиболее крупных месторождений минеральных вод России, а также одним из наиболее сложных по структуре. Сложность структуры обусловлена многочисленными трещинами в пластах. Выбор данного месторождения обусловлен необходимостью исследования изменений в гидродинамических процессах и структуре месторождения в ходе эксплуатации.

Месторождения с такой сложностью строения трудно промоделировать, так как полный учет параметров и факторов, влияющих на гидродинамические процессы (стыки разных пород, трещины и разрывы в пластах месторождения, количество осадков), невозможен. Для устранения этих трудностей коэффициент перетекания b был задан в виде мас-

сива значений, что позволило более точно представить структуру месторождения и повысить точность моделирования. Для получения достоверных результатов моделирования был определен набор учитываемых параметров, в который входят: понижение уровня жидкости в пласте S; коэффициент фильтрации k; скорость потока F; коэффицент упругоемкости пласта η^*. В данном случае сравниваются результаты моделирования, полученные с использованием ретроспективных данных с эксплуатируемого месторождения, и реальные данные об изменениях в месторождении за принятый отрезок времени. Если погрешность моделирования составляет менее 15 %, то модель считают достоверной.

Математическая модель объекта

Составим математическую модель Нагутского месторождения минеральных вод. Предлагается описывать гидролитосферные процессы месторождения с помощью дифференциального уравнения в частных производных, имеющего следующий вид:

$$\frac{\partial S_j}{\partial t} = \frac{1}{\eta_j{}^*}\left(\frac{\partial\left(k_{xj}\partial S_j\right)}{\partial x^2} + \frac{\partial\left(k_{yj}\partial S_j\right)}{\partial y^2} + \frac{\partial\left(k_{zj}\partial S_j\right)}{\partial z^2}\right),$$

где S_j — понижение уровня, м; j — номер пласта; k_x, k_y, k_z — коэффициенты фильтрации по соответствующим осям, м/сут; η_j^* — упругоемкость пласта, м^2/сут.

Для того чтобы наиболее точно отразить гидролитосферные процессы в месторождении, предлагается составить трехмерную четырехслойную модель, где в качестве слоев выбраны следующие пласты месторождения:

- грунтовые воды;
- эльбурганский водоносный горизонт;
- сеноман-сантонский водоносный горизонт;
- титонский водоносный горизонт.

Одной из особенностей Нагутского месторождения минеральных вод является наличие нескольких мощных водонепроницаемых пластов между водоносными горизонтами. Учитывая строение месторождения, было принято решение рассматривать каждый набор разделяющих пластов как единый пласт с минимальным коэффициентом водопроводимости.

На рис. 1 показано расположение группы скважин на Нагутском месторождении минеральных вод.

Составим математическую модель для первого пласта месторождения минеральных вод:

$$\frac{\partial S_1}{\partial t} = \frac{1}{\eta_1{}^*}\left(\frac{\partial\left(k_{x_1}\partial S_1\right)}{\partial x^2} + \frac{\partial\left(k_{y_1}\partial S_1\right)}{\partial y^2} + \frac{\partial\left(k_{z_1}\partial S_1\right)}{\partial z^2}\right); \qquad (1)$$

$$0 < x < L_x, 0 < y < L_y, 0 < z < L_{z_1},$$

где L_x — длина моделируемого участка месторождения, м; L_y — ширина моделируемого участка месторождения, м; L_{z_i} — ширина i-го водоносного пласта месторождения, м.

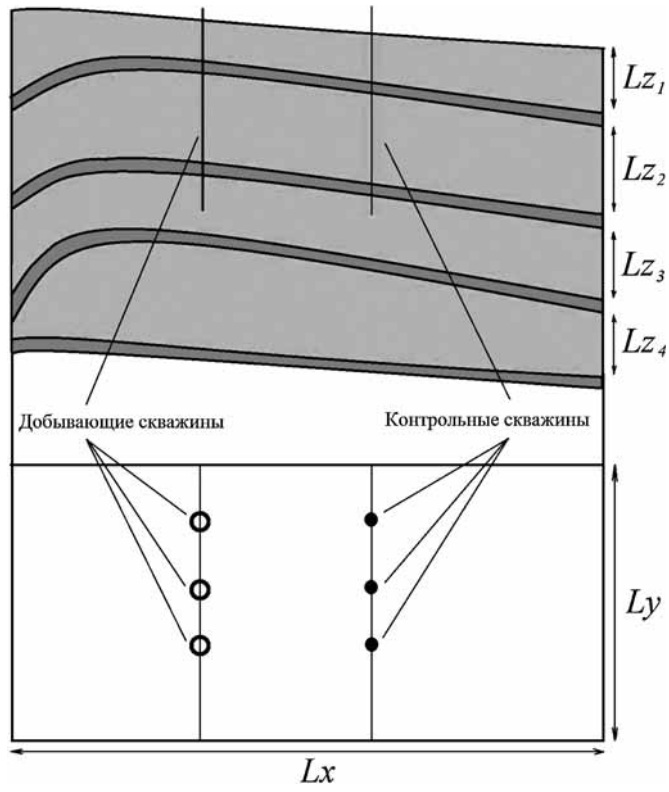


Рис. 1. Схема Нагутского месторождения

Начальные условия представлены в следующем виде:

$$S_1(x, y, z, 0) = 0.$$

Граничные условия имеют следующий вид:

$$\frac{\partial S_1\left(x, y, z = L_{z_1}, t\right)}{\partial z} = 0.$$

Таким образом, для каждого пласта можно записать соответствующую систему уравнений.

Составим математическую модель для второго пласта месторождения минеральных вод:

$$\frac{\partial S_2}{\partial t} = \frac{1}{\eta_2{}^*}\left(\frac{\partial\left(k_{x2}\partial S_2\right)}{\partial x^2} + \frac{\partial\left(k_{y2}\partial S_2\right)}{\partial y^2} + \frac{\partial\left(k_{z2}\partial S_2\right)}{\partial z^2}\right) -$$

$$-F_{x2}\frac{\partial S_2}{\partial x};\ 0 < x < L_x,\ 0 < y < L_y,\ 0 < z < L_{z_2},$$

где F_{xi} — скорость потока в i-м пласте, м/сут.

В уравнении (1) скорость потока F_x опущена, так как оно описывает слой грунтовых вод. Подпитка грунтовых вод обеспечивается инфильтрацией атмосферных осадков и поверхностных вод, а не за счет постоянного движения воды, поэтому предлагается рассматривать грунтовые воды как пласт месторождения с отсутствующей скоростью потока,

но учитывать его водопроницаемость и перетекание в прилегающие пласты [5].

Начальные условия уравнения представлены в следующем виде:

$$S_2(x,\ y,\ z,\ 0) = 0.$$

Граничные условия имеют следующий вид:

$$k_{zj}\frac{\partial S_j(x,\ y,\ L_{zj},\ t)}{\partial z} = k_{zj+1}\frac{\partial S_{j+1}(x,\ y,\ L_{zj+1},\ t)}{\partial z};$$

$$\frac{\partial S_2(x,\ y,\ z,\ t)}{\partial z} = 0;\ 0 < x < L_x,\ 0 < y < L_y.$$

Математическая модель третьего пласта месторождения минеральных вод имеет вид:

$$\frac{\partial S_3}{\partial t} = \frac{1}{\eta_3 *}\left(\frac{\partial(k_{x3}\partial S_3)}{\partial x^2} + \frac{\partial(k_{y3}\partial S_3)}{\partial y^2} + \frac{\partial(k_{z3}\partial S_3)}{\partial z^2}\right) -$$

$$-F_{x3}\frac{\partial S_3}{\partial x} - \hat{S}(t)\delta(x_i,\ y_i,\ z_i);$$

$$0 < x < L_x,\ 0 < y < L_y,\ 0 < z < L_{z_3},$$

где $\hat{S}(t)$ — управляющее воздействие на объект управления; в качестве управляющего воздействия на объект управления принято понижение уровня; управляющее воздействие приложено в третьем пласте, так как это эксплуатационный пласт, из которого идет добыча; x_i, y_i, z_i — координаты i-й водозаборной скважины; $\delta(x,\ y,\ z)$ — функция, значение которой равно 1 при $x = x_i$, $y = y_i$, $z = z_i$, а в остальных случаях значение $\delta(x,\ y,\ z) = 0$.

Начальные условия представлены в следующем виде:

$$S_3(x,\ y,\ z,\ 0) = 0.$$

Граничные условия заданы в следующем виде

$$\frac{\partial S_3(L_x,\ y,\ z,\ t)}{\partial x} = 0, S_3(0,\ y,\ z,\ t) = 0;$$

$$\frac{\partial S_3(x,\ 0,\ z,\ t)}{\partial y} = \frac{\partial S_3(x,\ L_y,\ z,\ t)}{\partial y} = 0.$$

Математическая модель четвертого пласта месторождения минеральных вод имеет следующий вид:

$$\frac{\partial S_4}{\partial t} = \frac{1}{\eta_4 *}\left(\frac{\partial(k_{x4}\partial S_4)}{\partial x^2} + \frac{\partial(k_{y4}\partial S_4)}{\partial y^2} + \frac{\partial(k_{z4}\partial S_4)}{\partial z^2}\right) -$$

$$-F_{x4}\frac{\partial S_4}{\partial x};\ 0 < x < L_x,\ 0 < y < L_y,\ 0 < z < L_{z_4}.$$

Начальные условия заданы в следующем виде:

$$S_4(x,\ y,\ z,\ 0) = 0.$$

Граничные условия имеют следующий вид:

$$\frac{\partial S_4(x,\ y,\ z = L_{z_4},\ t)}{\partial z} = 0.$$

Таким образом, можно объединить представленные выше уравнения и составить общую модель. Итак, математическую модель Нагутского месторождения минеральных вод предлагается представить в виде системы дифференциальных уравнений в частных производных следующего вида:

$$\begin{cases}
\frac{\partial S_1}{\partial t} = \frac{1}{\eta_1 *}\left(\frac{\partial(k_{x1}\partial S_1)}{\partial x^2} + \frac{\partial(k_{y1}\partial S_1)}{\partial y^2} + \frac{\partial(k_{z1}\partial S_1)}{\partial z^2}\right); \\
\frac{\partial S_2}{\partial t} = \frac{1}{\eta_2 *}\left(\frac{\partial(k_{x2}\partial S_2)}{\partial x^2} + \frac{\partial(k_{y2}\partial S_2)}{\partial y^2} + \frac{\partial(k_{z2}\partial S_2)}{\partial z^2}\right) - \\
- F_{x2}\frac{\partial S_2}{\partial x}; \\
\frac{\partial S_3}{\partial t} = \frac{1}{\eta_3 *}\left(\frac{\partial(k_{x3}\partial S_3)}{\partial x^2} + \frac{\partial(k_{y3}\partial S_3)}{\partial y^2} + \frac{\partial(k_{z3}\partial S_3)}{\partial z^2}\right) - \\
-F_{x3}\frac{\partial S_3}{\partial x} - \hat{S}(t)\cdot\delta(x_i, y_i, z_i); \\
\frac{\partial S_4}{\partial t} = \frac{1}{\eta_4 *}\left(\frac{\partial(k_{x4}\cdot\partial S_4)}{\partial x^2} + \frac{\partial(k_{y4}\cdot\partial S_4)}{\partial y^2} + \frac{\partial(k_{z4}\cdot\partial S_4)}{\partial z^2}\right) - \\
- F_{x4}\frac{\partial S_4}{\partial x};
\end{cases} \quad (2)$$

$$0 < x < L_x,\ 0 < y < L_y,\ 0 < z < L_z.$$

Начальные условия для математической модели (2) заданы в следующем виде:

$$S_j(x,\ y,\ z,\ 0) = 0;\ j = 1...4,$$

где j — номер пласта.

Граничные условия внутри объекта заданы в следующем виде:

$$\frac{\partial S_j(L_x,\ y,\ z,\ t)}{\partial x} = 0,\ S_j(0,\ y,\ z,\ t) = 0;$$

$$\frac{\partial S_j(x,\ 0,\ z,\ t)}{\partial y} = \frac{\partial S_j(x,\ L_y, z,\ t)}{\partial y} = 0;$$

$$j = 1...4.$$

Граничные условия на границах пласта заданы в следующем виде:

$$k_{zj}\frac{\partial S_j(x,\ y,\ L_{z_j},\ t)}{\partial z} = k_{zj+1}\frac{\partial S_{j+1}(x,\ y,\ L_{z_{j+1}},\ t)}{\partial z};$$

$$\frac{\partial S_1(x,\ y,\ z = L_{z_1},\ t)}{\partial z} = 0,\ \frac{\partial S_3(x,\ y,\ L_{z_4},\ t)}{\partial z} = 0;$$

$$0 < x < L_x,\ 0 < y < L_y.$$

Граничные условия на нижней границе четвертого пласта заданы в следующем виде:

$$\partial S_4(x, y, L_{z_4}, t)/\partial z = 0.$$

Боковые грани также должны быть описаны в системе уравнений. Боковые грани являются условными границами пластов.

Для формирования граничных условий по координате y полагаем, что мощность пластов такова, что возмущения от заборных скважин не влияют на состояние пласта в граничных точках, т. е.

$$S_j(x, 0, z, t) = S_j(x, L_y, z, t) = S_j(0, y, z, t), \quad j = 1...4.$$

Граничные условия между пластами задаются в виде уравнений Дарси. Связь между первым и вторым пластами имеет следующий вид:

$$S_1(x, y, L_{z_1}, t) = S_1(x, y, L_{z_1}, t) + b_1(x, y)\left(S_2(x, y, 0, t) - S_1(x, y, L_{z_1}, t)\right),$$

$$S_2(x, y, 0, t) = S_2(x, y, 0, t) - b_1(x, y)\left(S_2(x, y, 0, t) - S_1(x, y, L_{z_1}, t)\right).$$

Связь между вторым и третьим пластами имеет следующий вид:

$$S_2(x, y, L_{z_2}, t) = S_1(x, y, L_{z_2}, t) + b_2(x, y)\left(S_3(x, y, 0, t) - S_2(x, y, L_{z_2}, t)\right),$$

$$S_3(x, y, 0, t) = S_3(x, y, 0, t) - b_2(x, y)\left(S_3(x, y, 0, t) - S_2(x, y, L_{z_3}, t)\right).$$

Связь между третьим и четвертым пластами имеет следующий вид:

$$S_3(x, y, L_{z_3}, t) = S_3(x, y, L_{z_3}, t) + b_3(x, y)\left(S_4(x, y, 0, t) - S_3(x, y, L_{z_3}, t)\right),$$

$$S_4(x, y, 0, t) = S_4(x, y, 0, t) - b_3(x, y)\left(S_4(x, y, 0, t) - S_3(x, y, L_{z_3}, t)\right).$$

При использовании стандартного метода задачи граничных условий коэффициент перетекания b задан отдельно для каждого пласта как единичное значение. В данной работе предлагается впервые представить коэффициент перетекания в каждой точке дискретизации, т. е. в виде массива чисел. Данное преобразование позволяет более точно отразить гидрогеологическую структуру каждого пласта, что необходимо для повышения точности моделирования.

Программная реализация

Распределенная система управления была разработана с использованием объектно-ориентированного языка программирования Delphi программного комплекса Embarcadero RAD Studio компании Embarcadero Technologies. Программный продукт

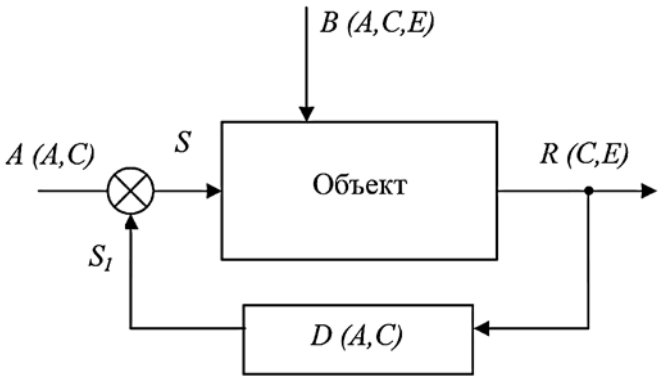


Рис. 2. Общая модель разработанной системы

предназначен для управления дебитом месторождения минеральных вод путем корректировки параметров процесса добычи в зависимости от изменения состояния месторождения.

Общая модель системы представлена на рис. 2, где:

A — исходные геофизические данные по скважине: расположение скважины, геофизические параметры пласта, прогнозируемая добыча;

C — дебит скважины;

D — управление на основе распределенного регулятора;

B — управление инженерным объектом: расчетные параметры добычи, полученные специалистами по эксплуатации месторождений;

E — экономический эффект эксплуатации скважины;

R — оперативные данные по эксплуатации (дебит скважины, экономический эффект добычи минеральной воды);

S — входное воздействие;

S_1 — входное воздействие с учетом управления.

Алгоритм работы системы организован следующим образом: на начальном этапе система получает управление от двух компонентов. Первый из них реализует управление методом распределенного регулятора (компонент *D*), а другой — управление эксплуатационными параметрами месторождения (компонент *B*). Компонент *B* определяется предприятием (пользователем), которое проводит добычу ресурса. При эксплуатации месторождения предприятие старается получить максимальную прибыль, что приводит к превышению допустимых параметров водозабора. Вместе с тем сделать добычу экономически невыгодной нецелесообразно. Следовательно, режим эксплуатации должен быть таким, чтобы выполнялись два основных условия:

- обеспечение размера прибыли, превышающего расчетные показатели (группа параметров, определяемых компонентом *B*);
- эксплуатация месторождения в режиме, позволяющем поддерживать сохранность источника, путем определения параметров водозабора, не превышающих расчетные (группа параметров, определяемых компонентом *D*).

Результат выполнения данных условий представлен группой параметров R (выходные данные). Полученные данные позволяют также внести необходимые коррективы и организовать управляющее воздействие $S_1(x, t)$, которое формируется на основе анализа результатов эксплуатации месторождения. На следующем шаге итерации указанная схема повторяется, позволяя достоверно установить оптимальный эксплуатационный режим.

Актуальность разработки обусловлена растущей потребностью работающих в этой сфере деятельности предприятий в автоматизации процесса эксплуатации и контроля состояния месторождений минеральных вод. Применение таких систем повысит скорость и точность расчета параметров режима эксплуатации, что, в свою очередь, позволит увеличить прибыль предприятия и обеспечит более высокий уровень безопасности процессов эксплуатации месторождений.

Анализ результатов моделирования

Вычислим частотные характеристики объекта управления. Для этого представим входное воздействие $S(x, y, z, t)$ (понижение уровня в зоне расположения заборных скважин (2)) в следующем виде:

$$S(x, t) = \sum_{n}^{\infty} A_n \cos(\psi_n x), \tag{3}$$

где n — номер гармоники ряда Фурье; A_n — максимальное изменение уровня воды в пласте, м; ψ_n — пространственная частота.

В ходе моделирования построены графики переходных процессов результата воздействия первой и четвертой гармоник ряда (3), соответственно (рис. 3, 4).

Для гидролитосферных объектов управления передаточные функции по каждой моде входного воздействия могут быть аппроксимированы передаточными функциями следующего вида:

$$W = \frac{K}{Tp + 1} \exp(-p\tau),$$

где K — коэффициент передачи системы; T — постоянная времени; τ — время запаздывания, сут; p — переменная преобразования Лапласа.

С использованием графоаналитического анализа [1] были определены значения τ, T и K, что позволило перейти к синтезу регулятора.

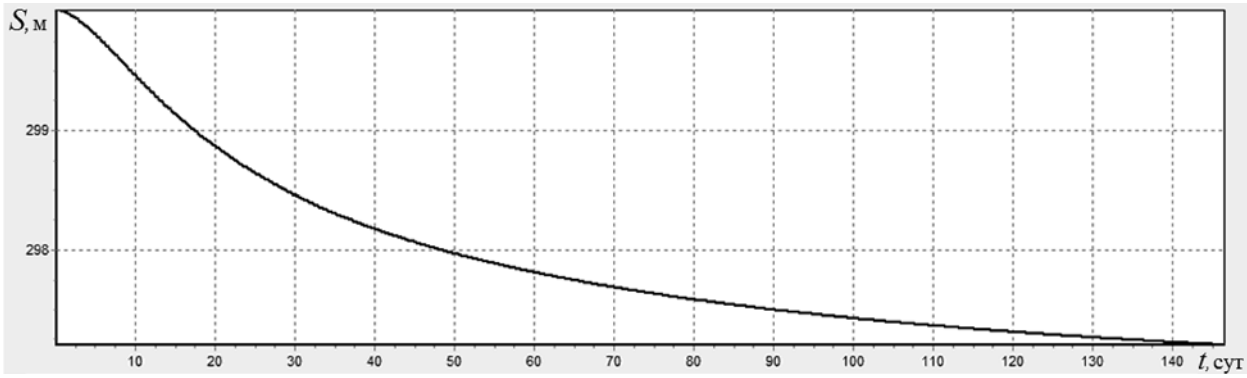


Рис. 3. Временная характеристика понижения уровня $S(x, t)$ (воздействие первой гармоники)

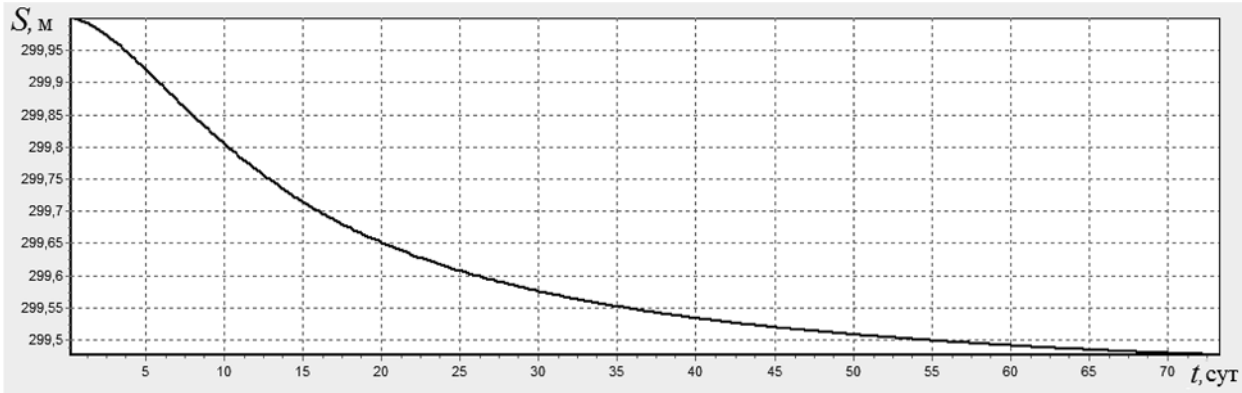


Рис. 4. Временная характеристика понижения уровня $S(x, t)$ (воздействие четвертой гармоники)

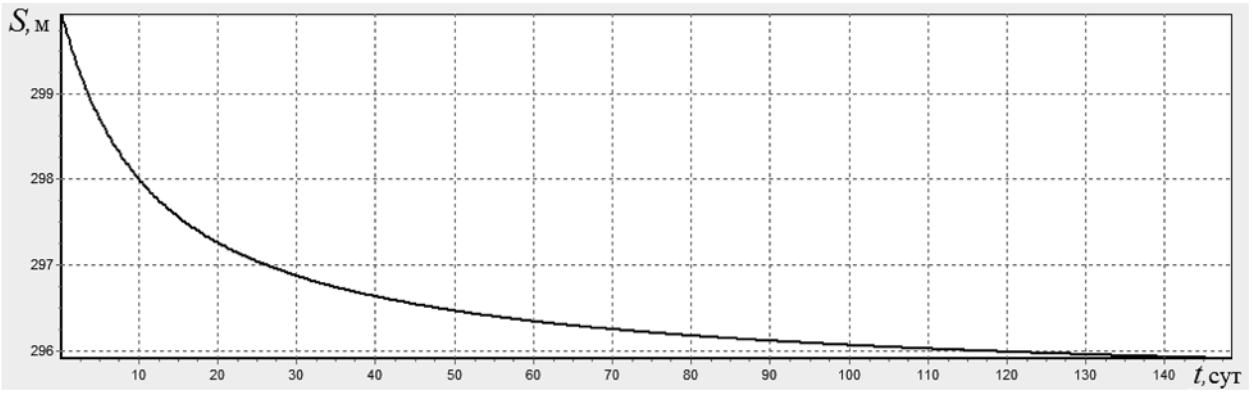


Рис. 5. Временная характеристика понижения уровня $S(x, t)$. Замкнутая система управления

Синтез регулятора распределенной системы

Требуется разработать систему управления дебитом месторождения минеральных вод для обеспечения стабильного режима добычи ресурсов [3]. Ставится задача синтеза системы управления, способной поддерживать стабильный режим эксплуатации [8]. Передаточная функция распределенного высокоточного регулятора представлена следующей формулой [1]:

$$W(y, p) = E_1\left[\frac{n_1 - 1}{n_1} - \frac{1}{n_1}\nabla^2\right] + \\ + E_4\left[\frac{n_4 - 1}{n_4} - \frac{1}{n_4}\nabla^2\right]\frac{1}{p} + E_2\left[\frac{n_2 - 1}{n_2} - \frac{1}{n_2}\nabla^2\right]p,$$

где E_1, E_2, E_4 — общие коэффициенты усиления (заданное число); ∇^2 — лапласиан; n_1, n_2, n_4 — весовые коэффициенты ($n \geqslant 1$).

Проведенные расчеты позволяют определить параметры передаточной функции:

$$W(y,\ p) = 1657,419\left[-\nabla^2\right] + 159,332\left[\frac{0,000073}{1,000073} - \nabla^2\right]\frac{1}{p} + 10,308p.$$

С использованием полученных численных моделей объекта управления и регулятора было выполнено моделирование функционирования замкнутой системы управления. По итогам моделирования получены графики переходных процессов. Результат синтеза представлен на рис. 5.

Проверка устойчивости системы, выполненная с использованием критерия Найквиста, показала, что система устойчива.

Результаты испытания модели замкнутой системы показали, что полученная система управления достаточно динамична, а значение ошибки регулирования не более 8 %. В практических реализациях в системах управления с использованием распределенных высокоточных регуляторов, ошибка регулирования сопоставима с ошибкой наблюдения.

Полученные результаты подтверждают целесообразность представления коэффициента перетекания $b(x, y)$ в виде массива значений для каждой точки дискретизированной области.

Заключение

Представлено описание процесса синтеза регулятора для системы управления реальным гидролитосферным объектом. Задача решена для трехслойной пространственной модели. Моделирование гидродинамических процессов проводится для граничных условий второго рода, что определено особенностями месторождения. Постановка задачи выполнена для Нагутского месторождения минеральных вод с учетом пространственной неоднородности объекта. В модели учитывается зависимость коэффициента перетекания от пространственно-неоднородной структуры пластов месторождения, что ранее не применялось для решения задач данной предметной области.

Список литературы

1. **Малков А. В., Першин И. М**. Синтез распределенных регуляторов для систем управления гидролитосферными процессами. М.: Науч. мир, 2007. С. 235—240.

2. **Мартиросян А. В., Мартиросян К. В.** Модели обеспечения устойчивых режимов эксплуатации месторождений минеральных вод на примере Нагутского месторождения // Недропользование — XXI век. 2014. № 6а (44). С. 88—93.

3. **Мартиросян А. В.** Повышение эффективности управления гидродинамическими процессами путем учета неоднородности коэффициента перетекания // Современная наука и инновации. 2015. № 4. С. 17—25.

4. **Мартиросян К. В., Мартиросян А. В.** Анализ способов построения математических моделей месторождений минеральных вод // Фундаментальные исследования. 2014. № 11—12. С. 2599—2603.

5. **Chernyshev A. B., Martirosyan K. V., Martirosyan A. V.** Analysis of the nonlinear distributed control system's sustainability // Journal of Mathematics and Statistics. 2014. Vol. 10, N 3. P. 316—321.

6. **Finkbeiner B., Schewe S.** Bounded synthesis // International Journal on Software Tools for Technology Transfer, 2013. Vol. 15. N 5. P. 519—539.

7. **Martirosyan A. V., Martirosyan A. V., Yanukyan E. G.** Methods of complex object's transfer function calculation for distributed control system // Journal of Mathematics and Statistics. 2014. Vol. 10. N 3. P. 408—413.

8. **Martirosyan A. V., Martirosyan K. V., Chernyshev A. B.** Methods of distributed systems' structured modeling // 2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (February 2—3). St. Petersburg, 2016. P. 283—289.

9. **Pershin I. M., Martirosyan A. V., Martirosyan K. V.** Analysis of the Caucasus mineral waters' field's modeling // Modern Applied Science. 2015. Vol. 9. N 1. P. 204—210.

Distributed Control System Synthesis for Spatially Nonuniform Hydrogeological Objects

K. V. Martirosyan, kv1961@live.ru, **A. V. Martirosyan**, martalex11@mail.ru, North-Caucasus Federal University, Stavropol, 355009, Russian Federation

Corresponding author:

Martirosyan Alexander V., Postgraduate Student, North-Caucasus Federal University, 355009, Stavropol, Russian Federation,
e-mail: martalex11@mail.ru

Received on June 15, 2016
Accepted on August 25, 2016

This article presents an algorithm for simulation of hydrodynamic processes of a complex hydrogeological object. The formulation of the problem of modeling of hydrodynamic processes as the steps of the synthesis of distributed control systems is presented. The control object is described. Mathematical model of an object is also presented. Nagutskoe mineral water field was selected as a control object, and also as the most advanced field of Caucasus Mineral Waters region. Taking possible spatial heterogeneity factors into account will significantly complicate the calculation. However, it is proposed to amend the classical approach of controller synthesis for a hydrogeological object problem formulation. It is proposed to explore the possibility of obtaining more reliable simulation results by providing spillover factor as an array of data that reflects the spatial variation of the value of this coefficient. In the standard modeling of a hydrological object problem formulation the migration coefficient is set separately for each stratum. Thus, earlier the average gain value of migration was used. This means that the spatial variation of migration coefficient was neglected to simplify the model. This study shows that the application of the proposed presentation factor increases the accuracy of simulation and stability of the system, which confirms the feasibility of this solution.

Keywords: *control systems, distributed systems, system synthesis, mathematical modeling, models of hydrogeological objects*

For citation:

Martirosyan K. V., Martirosyan A. V. Distributed Control System Synthesis for Spatially Nonuniform Hydrogeological Objects, *Programmnaya Ingeneria,* 2016, vol. 7, no. 11, pp. 522—528.

DOI: 10.17587/prin.7.522-528

References

1. **Malkov A., Pershin I.** *Sintez raspredelennyh reguljatorov dlja sistem upravlenija gidrolitosfernymi processami* (Distributed controller synthesis for the hydrolitosphere processes management systems), Moscow, Nauch. Mir, 2007, 256 p. (in Russian).
2. **Martirosyan A., Martirosyan K.** Modeli obespechenija ustojchivyh rezhimov jekspluatacii mestorozhdenij mineral'nyh vod na pri mere Nagutskogo mestorozhdenija (Ensuring sustainable model of mineral water deposits operating modes on the example of the deposit Nagutskaya), *Nedropol'zovanie — XXI vek*, 2014, vol. 44, no. 6a, pp. 5—17 (in Russian).
3. **Martirosyan A. V.** Povyshenie jeffektivnosti upravlenija gidrodinamicheskimi processami putem ucheta neodnorodnosti kojefficienta peretekanija (The increasing of hydromineral processes' control effectiveness with the accounting discount of migration coefficient's nonuniformity), *Sovremennaja Nauka i Innovacii*, 2015, no. 4, pp. 17—25 (in Russian).
4. **Martirosyan K. V., Martirosyan A. V.** Analiz sposobov postroenija matematicheskih modelej mestorozhdenij mineral'nyh vod (Analysis of the mineral water field's mathematical models' development methods), *Fundamental'nye issledovanija*, 2014, no. 11—12, pp. 2599—2603 (in Russian).
5. **Chernyshev A. B., Martirosyan K. V., Martirosyan A. V.** Analysis of the nonlinear distributed control system's sustainability, *Journal of Mathematics and Statistics*, 2014, vol. 10, no 3, pp. 316—321.
6. **Finkbeiner B., Schewe S.** Bounded synthesis, *International Journal on Software Tools for Technology Transfer*, 2013, vol. 15, no. 5, pp. 519—539.
7. **Martirosyan A. V., Martirosyan A. V., Yanukyan E. G.** Methods of complex object's transfer function calculation for distributed control system, *Journal of Mathematics and Statistics,* 2014, vol. 10, no. 3, pp. 408—413.
8. **Martirosyan A. V., Martirosyan K. V., Chernyshev A. B.** Methods of distributed systems' structured modeling, *2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference,* St.Petersburg, Russian Federation, Feb. 2—3, 2016, pp. 283—289.
9. **Pershin I. M., Martirosyan, A. V., Martirosyan K. V.** Analysis of the Caucasus mineral waters' field's modeling, *Modern Applied Science*, 2015, vol. 9, no. 1, pp. 204—210.

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева.* Корректор *З. В. Наумова*

Сдано в набор 02.09.2016 г. Подписано в печать 20.10.2016 г.Формат 60×88 1/8.Заказ PI1116
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз".Отпечатано в ООО "Адвансед солюшнз".
119071, г.Москва, Ленинский пр-т, д. 19, стр.1. Сайт: **www.aov.ru**