ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2019
Т. 8, № 4

ISSN 2305-9052

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий, в которых должны быть опубликованы результаты диссертаций на соискание ученых степеней кандидата и доктора наук

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет (национальный исследовательский университет)»

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Искусственный интеллект и машинное обучение
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY
SERIES "COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2019
vol. 8, no. 4

ISSN 2305-9052

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

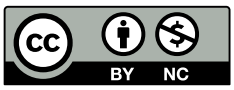
South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- Artificial intelligence and machine learning
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Contents



УДК 519.633.2 **DOI: 10.14529/cmse190401**

ИССЛЕДОВАНИЕ МЕТОДА ПИКАРА ПРИ РЕШЕНИИ ОБРАТНОЙ ЗАДАЧИ КОШИ ДЛЯ УРАВНЕНИЯ ТЕПЛОПРОВОДНОСТИ

© 2019 Х.К. Аль-Махдави

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: hssnkd@gmail.com

Поступила в редакцию: 08.01.2019

В данной работе поставлена и решена обратная задача Коши для уравнения теплопроводности. В этой задаче начальное распределение температуры неизвестно, а вместо него дано распределение температуры в момент времени $t = T > 0$. Среди математических задач выделяется класс задач, решения которых неустойчивы к малым изменениям исходных данных. Они характеризуются тем, что сколь угодно малые изменения исходных данных могут приводить к большим изменениям решений. Хорошо известно, что данная задача некорректно поставлена. Для решения прямой задачи используется метод разделения переменных. Заметим, что метод разделения переменных совершенно неприменим для решения обратной задачи Коши, так как приводит к достаточно большим погрешностям, а также к расходящимся рядам. В.К. Иванов заметил, что если обратную задачу решать методом разделения переменных, а затем полученный ряд заменять частичной суммой ряда, у которой число слагаемых зависит от δ, $N = N(\delta)$, то в результате получим устойчивое приближенное решение. Метод Пикара использует регуляризующее семейство операторов $\{R_N\}$, отображающих пространство $L_2[0,1]$ в себя. Приведены результаты вычислительных экспериментов и произведена оценка эффективности данного метода.

Ключевые слова: обратная задача теплопроводности, метод Пикара, некорректная задача, проблема Коши.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Аль-Махдави Х.К. Исследование метода Пикара при решении обратной задачи Коши для уравнения теплопроводности // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 4. С. 5–14. DOI: 10.14529/cmse190401.

Введение

Многие прикладные задачи, сформулированные как обратные задачи математической физики, относятся к классу некорректно поставленных задач. Обратная задача Коши для уравнения теплопроводности определяется как некорректная задача [3] в том смысле, что «небольшое» произвольное изменение данных может привести к «большим» ошибкам в решении.

Заметим, что исследуемая нами обратная задача для уравнения теплопроводности может быть решена и многими другими методами. Например, методом регуляризации А.Н. Тихонова [5], методом М.М. Лаврентьева [6], методом квазирешений В.К. Иванова [1, 2] и многими другими. Так, многие из методов, описанных для решения обратных задач математической физики, также описывали решение классической задачи прямой теплопроводности. Основы оптимальных методов были получены для решения некорректных задач, а также способов оценки точного решения и точности оценок ошибок порядка для этих методов [4].

Теоретические концепции и вычислительная реализация, связанные с обратной задачей Коши уравнения теплопроводности, обсуждались многими авторами, и было описано

много методов. В [7] используется метод регуляризации Тихонова для построения функции стоимости для исходной задачи обратной теплопроводности и преобразования ее в задачу оптимизации. Метод регуляризации Тихонова, используемый для решения теплового потока на передней поверхности толстой пластины, основан на измеренной истории температуры на задней поверхности пластины, которая изолирована. Метод полудискретного регулирования используется для определения распределения температуры в пластине [8]. Итеративная схема разбивает сложную задачу оптимизации на несколько более простые подзадачи и объединяет преимущества альтернативной методики оптимизации. Обратные задачи теплопроводности треугольной стенки, решаемые методом сопряженного градиента, были связаны с методом конечных элементов для определения двумерных изменений температур и тепловых потоков на поверхности стенки со временем [9]. Обратная пространственно-зависимая задача об источнике была исследована для уравнения теплопроводности. Метод регуляризации Тихонова использовался для решения задачи [10].

Основная идея этой работы состоит в том, чтобы восстановить функцию источника уравнения диффузии, используя метод Пикара, предложенный в [1, 2]. Метод разделения переменных использовался для решения прямой задачи для уравнения с частными производными для уравнения теплопроводности. Все эти шаги будут реализованы через разделы этой статьи. В разделе 1 была рассмотрена математическая формулировка этой проблемы, и метод разделения переменных использовался для ее решения в качестве прямой задачи. Раздел 2 посвящен обратной задаче Коши для уравнений теплопроводности и дает описание известных данных и оператора. Раздел 3 демонстрирует использование метода Пикара при помощи регуляризующего семейства операторов $\{\mathrm{R_N}\}$, отображающих пространство $\mathrm{L_2}[0,1]$ в себя. Пример для проверки точности нашего оценочного решения представлен в разделе 4. Наконец, объяснение предложенного метода было обобщено в заключительном разделе с предлагаемой будущей работой.

1. Постановка прямой задачи Коши для уравнения теплопроводности

Прямая задача состоит в теплопроводности, проходящей через бар с определенным граничным условием и начальным температурным условием. Математическая формулировка этой проблемы была описана следующим парциальным дифференциальным уравнением

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2};\ 0 < x < 1\ , t \in (0,T], \tag{1}$$

$$u(x,0) = u_0(x);\ x \in [0,1], \tag{2}$$

$$u(0,t) = u(1,t) = 0;\ t \in\ [0,T]. \tag{3}$$

Предложим, что $u_0(x) \in\ H_0^2[0,1]$.

Требуется определить функцию $u(x,t) \in C\ ([0,1]\ \times [0,T]) \cap C^{2,1}(0,1)\ \times\ (0,T])$, удовлетворяющую уравнению (1) на $x \in [0,1]$ и $t \in (0,T]$, а также начальному условию (2) и граничным условиям (3).

Решая задачу (1–3) методом разделения переменных, получим

$$u(x,t) = \sum_{n=1}^{\infty} a_n\, e^{-(n\pi)^2 t}\, sin(\,n\pi x), \tag{4}$$

где $x \in [0,1],\ t \in [0,T]$, а

$$u_0(x) = \sum_{n=1}^{\infty} a_n\ sin(n\pi x), \tag{5}$$

где

$$a_n = 2\int_0^1 u_0(x)\ sin(n\pi x)\, dx. \tag{6}$$

2. Постановка обратной задачи Коши для уравнения теплопроводности

В (1–3) можно рассмотреть обратную задачу Коши для уравнения теплопроводности, т.е. известно распределение температуры в момент времени $t = T > 0$, $u(x,T)$, и требуется найти начальное распределение $u_0(x)$, считая, что $u_0(x) \in H_0^2[0,1]$.

Предположим, что нам известна функция $g_0(x) \in C[0,1]$, являющаяся решением прямой задачи при $t = T$,

$$u(x,T) = g_0(x). \tag{7}$$

Точное значение функции $g_0(x)$ неизвестно, а вместо него дано пара ($g_\delta(x)$ и δ), где $g_\delta(x) \in C[0,1]$, $\delta > 0$, требуется определить функцию $u_0(x) \in M_r$,

$$M_r = \left\{u(x):\ u(x) \in H_0^2[0,1], \|u(x)\|_{H_0^2}^2 \le r^2, u_0'(0) = u_0'(1) = 0\right\}, \tag{8}$$

такую что при подстановке ее в условие (2), получим решение $u(x,t)$ задачи. (1–3), удовлетворяющее условию.

Дополнительно предположим, что точное значения функции $g_0(x)$ нам не известно, а вместо того дано $g_\delta(x) \in C[0,1]$ и $\delta > 0$ так, что

$$\| g_\delta(x) - g_0(x)\|^2 \le \delta^2. \tag{9}$$

Используя исходные данные задачи $g_\delta(x)$ и δ требуется определить приближенное значение $u_\delta(x)$, а также получить оценку погрешности $\|u_\delta(x) - u_0(x)\|_{L_2}$.

3. Решение обратной задачи Коши методом Пикара

Метод Пикара [1, 2] использует регуляризующее семейство операторов $\{R_N\}$, отображающих пространство $L_2[0,1]$ в себя и определяемых формулой. При равномерной регуляризиции $\forall N \Longrightarrow \|R_N\| \le 1$ и $\forall u \Longrightarrow R_N A u \to u$ при $N \to \infty$, это семейство операторов называется регуляризующими

$$R_N g(x)\ = \sum_{n=1}^{N} g_n\, e^{(n\pi)^2 T}\, sin(n\pi x), \tag{10}$$

где

$$g_n = 2\int_0^1 g(x)\, sin\, \pi n x\, dx. \tag{11}$$

Приведем ряд свойств семейства $\{R_N\}$, сформулированных в виде лемм.

Лемма 1. Для любого N оператор R_N, определенный формулой (10), является линейным ограниченным с нормой $\|R_N\| = e^{(\pi N)^2 T}$.

Доказательство. Линейность оператора R_N следует из формулы (10).

Теперь докажем, что

$$\|R_N\| = e^{(\pi N)^2 T}, \tag{12}$$

ограниченность оператора R_N будет следовать из формулы (12).

Так как

$$\|R_N\|^2 = sup\ \{\|R_N g\|^2 : g \in L_2[0,1], \|g\|^2 \le 1\}, \tag{13}$$

а

$$\|g\|^2 = \sum_{n=1}^{\infty} g_n^2, \tag{14}$$

то из (13) и (14) будет следовать, что при $g(x) \in L_2[0,1]$ и $\|g\| \le 1$

$$\|R_N\| \le e^{2(\pi N)^2 T}. \tag{15}$$

Рассмотрим элемент $g_N(x) = 2\sin \pi N x$.

Так как $g_N \in L_2[0,1]$ и $\|g_N\| = 1$, то подействуем на него оператор R_N.

Тогда

$$\|R_N g\|^2 = e^{2(\pi N)^2 T} \sin \pi N x, \tag{16}$$

из (16) следует, что

$$\|R_N g\|^2 = e^{2(\pi N)^2 T}, \tag{17}$$

а из (17), что

$$\|R_N g\|^2 \ge e^{2(\pi N)^2 T}, \tag{18}$$

из (15) и (18) следует, что $\|R_N\|^2 = e^{2(\pi N)^2 T}$. Тем самым лемма доказана.

Лемма 2. Семейство операторов $\{R_N\}$ регуляризует обратную задачу (1), (3) и (7) на M_r.

Доказательство. Из определения регуляризующего семейства операторов [1, 2] следует, что для любого элемента $u(x) \in M_r$ справедливо соотношение, $R_N = \sum_{n=1}^{N} \to u(x)$ при $N \to \infty$ в метрике $L_2[0,1]$,

$$R_N \left[\sum_{n=1}^{\infty} u_n\, e^{(n\pi)^2 T} \sin(n\pi x)\right] \to u(x), \tag{19}$$

при $N \to \infty$ в метрике $L_2[0,1]$, где

$$u_n = 2\int_0^1 u(x)\ \sin(\pi n x)\ dx. \tag{20}$$

Так как

$$R_N \left[\sum_{n=1}^{\infty} u_n\, e^{(n\pi)^2 T} \sin(n\pi x)\right] - u(x) = \sum_{n=N+1}^{\infty} u_n\ \sin(n\pi x), \tag{21}$$

то

$$\left\| R_N \left[\sum_{n=1}^{\infty} u_n\, e^{(n\pi)^2 T} \sin(n\pi x)\right] - u(x)\right\|^2 = \sum_{n=N+1}^{\infty} u_n^2, \tag{22}$$

виду того, что $u(x) \in L_2[0,1]$ следует сходимость ряда

$$\sum_{n=N+1}^{\infty} u_n^2. \tag{23}$$

Таким образом,

$$\sum_{n=N+1}^{\infty} u_n^2 \to 0 \text{ при } N \to \infty. \tag{24}$$

Тем самым лемма доказана.

Приближенное решение $u_\delta^N(x)$ обратной задачи определим формулой

$$u_\delta^N(x) = R_N g_\delta(x). \tag{25}$$

Теперь прейдем к определению зависимости $N(\delta)$. Для этого сделаем оценку

$$\|u_0(x) - u_\delta^N(x)\| \le \|u_0(x) - u_0^N(x)\| + \|u_0^N(x) - u_\delta^N(x)\|, \tag{26}$$

где

$$u_0^N(x) = R_N g_0(x). \tag{27}$$

Сначала оценим второе слагаемое в формуле (26)

$$\|u_0^N(x) - u_\delta^N(x)\| = \|R_N g_0(x) - g_\delta(x)\| \le \|R_N\|.\delta. \tag{28}$$

Из (28) и леммы 1 следует, что

$$\|u_0^N(x) - u_\delta^N(x)\| = e^{(\pi N)^2 T}.\delta, \tag{29}$$

перейдем к оценке первого слагаемого в формуле (26)

$$\|u_0^N(x) - u_\delta^N(x)\|^2 = \sum_{n=N+1}^{\infty} u_n^2, \tag{30}$$

где

$$u_n = 2\int_0^1 u_0(x)\, sin(\pi n x)\, dx. \tag{31}$$

Так как $u_0 \in M_r$, что $u_0''(x) \in L_2[0,1]$ и $\|u_0''(x)\|_{L_2}^2 \le r^2$.

Таким образом,

$$u_0''(x) = \sum_{n=1}^{\infty} v_n \sin \pi n x, \tag{32}$$

где

$$v_n = 2\int_0^1 u_0''(x) \sin \pi n x\, dx. \tag{33}$$

Проинтегрировав формулу (32), получим, что

$$v_n = -(\pi n)^2\, u_n, \tag{34}$$

Из (33) следует, что

$$\sum_{n=N+1}^{\infty} u_n^2 \le \frac{r^2}{(\pi N)^4}, \tag{35}$$

а из (30) и (33), что

$$\|u_0(x) - u_0^N(x)\| \le \frac{r}{(\pi N)^2} \tag{36}$$

окончательно

$$\|u_0(x) - u_0^N(x)\| \le \frac{r}{(\pi N)^2} + e^{(\pi N)^2 T}.\delta. \tag{37}$$

Таким образом, из (35) и (37) определим параметр регуляризации $\mathrm{N}(\delta)$ из условия

$$\min_N \left(\frac{r}{(\pi N)^2} + e^{(\pi N)^2 T}.\delta\right). \tag{38}$$

4. Численный пример

Учитывая задачу из (1–4), нам нужно определить неизвестную функцию для начальной температуры $u(x)$ по известной функции $u(x,T) = g(x)$.

Пример 1. Прямое решение для температуры, где время (T = 0,001 и T = 0,005) для проверки приближения мы имеем точную начальную температуру $u_0(x) = 4sin(3\pi x) + 7sin(8\pi x)$, как показано на рис. 1. Мы можем найти функцию $g(x)$, используя (4).

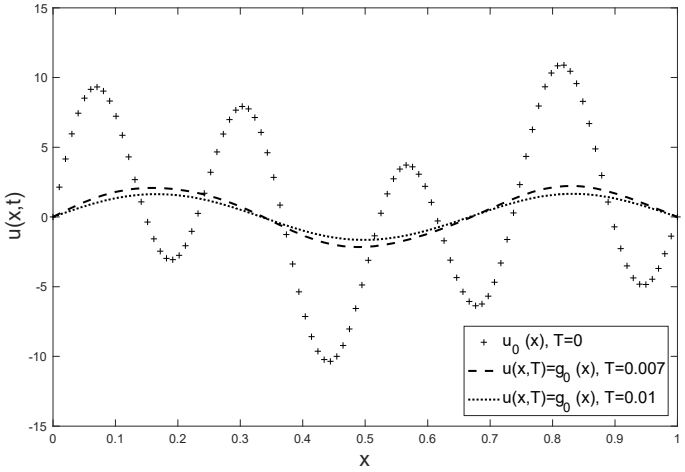
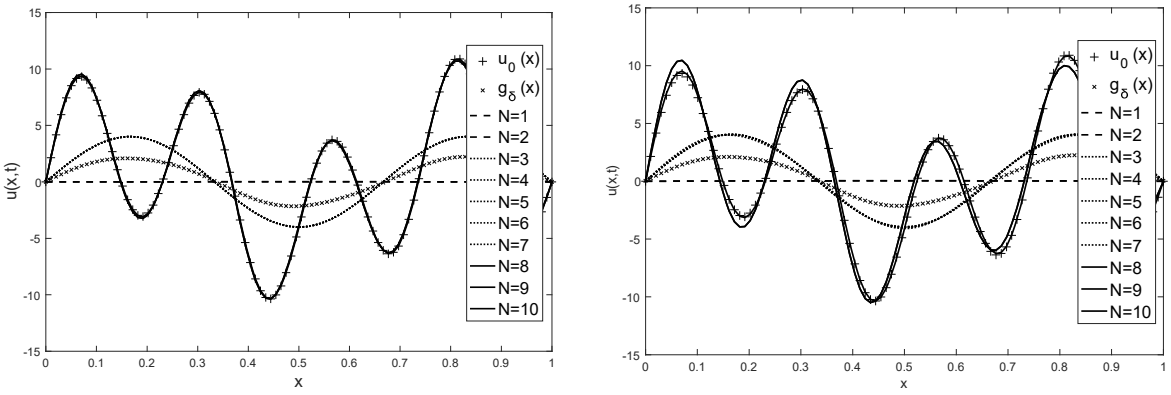


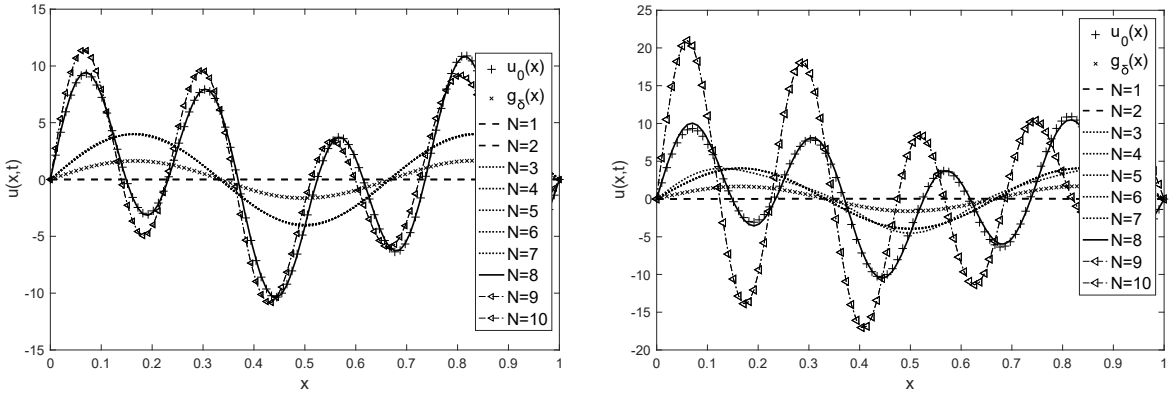
Рис. 1. Прямое решение для температуры измерения
$u(x,T) = g_0(x)$ и $u_0(x) = 4sin(3\pi x) + 7sin(8\pi x)$

Мы можем добавить шумовой сигнал к заданным данным $g_0(x)$ для его использования в анализе проблем. Используя алгоритм обратной задачи, который определен в (10) и (11), мы получим решения (см. рис. 2 и рис. 3).



а) $\delta = 0{,}05$ б) $\delta = 0{,}27$

Рис. 2. Обратное решение $u_0(x)$, $u_\delta^N(x)$, где T = 0,007



а) $\delta = 0{,}05$ б) $\delta = 0{,}27$

Рис. 3. Обратное решение $u_0(x)$, $u_\delta^N(x)$, где T = 0,01

Пример 2. Прямое решение для температуры, где время (T = 0,05 и T = 0,01) для проверки приближения мы имеем точную начальную температуру $u_0(x) = sin(\pi x)$, как показано на рис. 4. Мы можем найти функцию $g(x)$, используя (4).

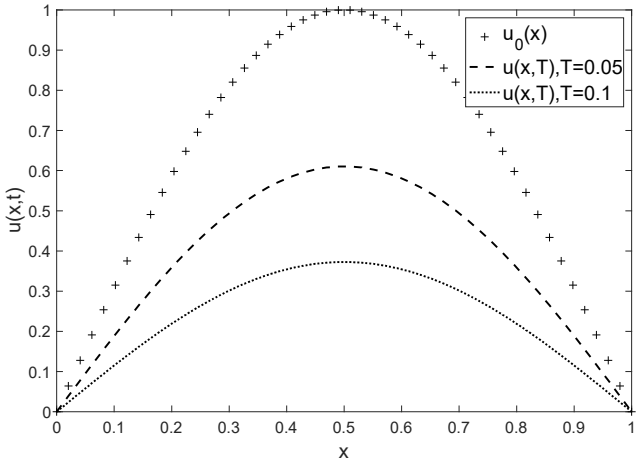
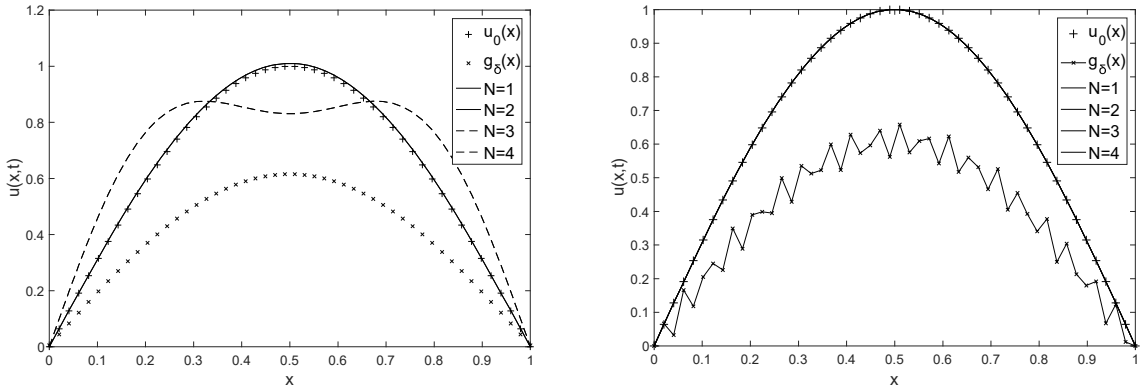


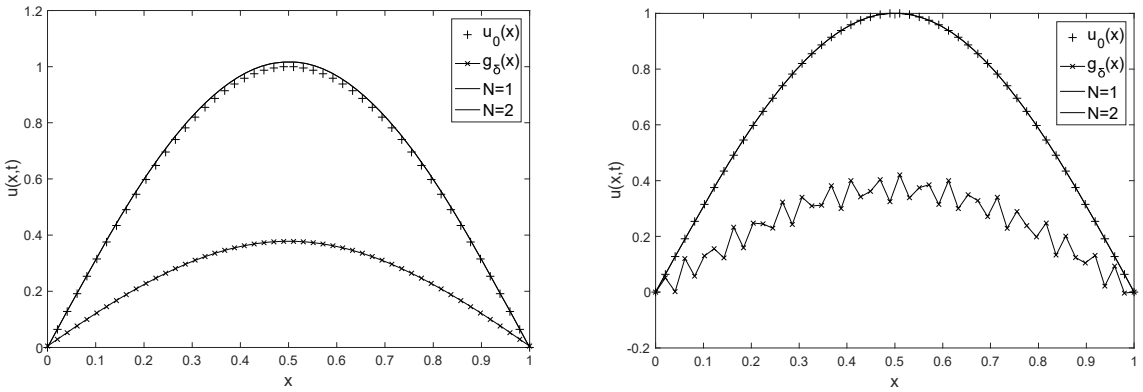
Рис. 4. Прямое решение для температуры измерения $u(x,T) = g_0(x)$ и $u_0(x) = sin(\pi x)$

Мы можем добавить шумовой сигнал к заданным данным $g_0(x)$ для его использования в анализе проблем. Используя алгоритм обратной задачи, который определен в (10) и (11), мы получим решения (см. рис. 5 и рис. 6).



а) $\delta = 0{,}0354$ б) $\delta = 0{,}247$

Рис. 5. Обратное решение $u_0(x)$, $u_\delta^N(x)$, где T = 0,05



а) $\delta = 0{,}0354$ б) $\delta = 0{,}247$

Рис. 6. Обратное решение $u_0(x)$, $u_\delta^N(x)$, где T = 0,1

Заключение

Эта работа посвящена алгоритму решения проблемы обратной теплопроводности. Данная задача должна решаться некорректной задачей Коши и специальным методом. Используется метод разделения переменных для решения уравнения обратной теплопроводности. Численные анализы успешно применяются для решения обратной задачи теплопроводности с использованием теоремы Пикара. Алгоритм выбирает соответствующий параметр для регуляризации оцененного решения. Можно отметить, что алгоритм был эффективен для оценки начальной температуры в зависимости от заданной температуры измерения с известным уровнем шума δ. Предложенная будущая работа решит обратную задачу Коши для уравнения теплопроводности для композиционных материалов.

Литература

1. Иванов В.К. О применении метода Пикара к решению интегральных уравнений первого рода // Bui. Inst. Politehn. Iasi. 1968. Т. 4, № 34. С. 71–78.
2. Иванов В.К., Васин В.В., Танана В.П. Теория линейных некорректных задач и ее приложения. Москва: Наука, 1978. 206 с.
3. Kabanikhin S.I. Inverse and Ill-Posed Problems: Theory and Applications. Inverse and Ill-Posed Problems, Ser. 55. De Gruyter, 2012. 458 p.
4. Tanana V.P., Sidikova A.I. Optimal Methods for Solving Ill-Posed Heat Conduction Problems. Inverse and ill-posed problems, Ser. 62. De Gruyter, 2018. 138 p.
5. Тихонов А.Н. О Регуляризации некорректно поставленных задач // Докл. АН СССР, 1963. Т. 153, № 1. С. 49–52.
6. Лаврентьев М.М. Некоторых некорректных задачах математической физики. Новосибирск: Изд-во Сиб. отд-ния АН СССР, 1962. 92 с.
7. Mu H., Li J., Wang X. Optimization Based Inversion Method for the Inverse Heat Conduction Problems // IOP Conference Series: Earth and Environmental Science. 2017. Vol. 64, no. 1. P. 1–9. DOI: 10.1088/1755-1315/64/1/012094.
8. Duda P. Solution of Inverse Heat Conduction Problem Using the Tikhonov Regularization Method // Journal of Thermal Science. 2017. Vol. 26, no. 1. P. 60–65. DOI: 10.1007/s11630-017-0910-2.
9. Frąckowiak A., Botkin N.D., Ciałkowski M. Iterative Algorithm for Solving the Inverse Heat Conduction Problems with the Unknown Source Function // Inverse Problems in Science and Engineering. 2015. Vol. 23, no. 6. P. 1056–1071. DOI: 10.1080/17415977.2014.986723.
10. Yang S., Xiong X. A. Tikhonov Regularization Method for Solving an Inverse Heat Source Problem // Bull. Malays. Math. Sci. Soc. 2018. Vol. 5, no. 19. P. 1–12. DOI: 10.1007/s40840-018-0693-y.

Аль-Махдави Хассан К. Ибрахим, аспирант, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse190401

STUDYING THE PICARD'S METHOD FOR SOLVING THE INVERSE CAUCHY PROBLEM FOR HEAT CONDUCTIVITY EQUATIONS

© 2019 H.K. Al-Mahdawi

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: hssnkd@gmail.com

Received: 08.01.2019

In this paper, the inverse Cauchy problem for the heat equation is posed and solved. In this problem, the initial temperature is unknown, and instead of it, the temperature at a specific time is given, $t = T > 0$. They are characterized by the fact that arbitrarily small changes in the source data can lead to large changes in the solution. It is well known that this problem is an ill-posed problem. In order to solve the direct problem, the method of separation of variables is used. We noticed that the method of separating variables is not applicable to solving the inverse Cauchy problem since it leads to large errors, as well as to divergent rows. V.K. Ivanov noted that if the inverse problem is solved by the method of separation of variables, the resulting series is replaced with a partial sum of a series, where the number of terms depends on δ, $N = N(\delta)$. The Picard's method uses the regularizing family of $\{R_N\}$, operators mapping the $L_2[0,1]$ space into itself. The results of computational experiments are presented and the effectiveness of this method is estimated.

Keywords: inverse heat conduction problem, Picart's method, ill-posed problem, Cauchy problem.

FOR CITATION

Al-Mahdawi H.K. Studying the Picard's Method for Solving the Inverse Cauchy Problem for Heat Conductivity Equations. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 4. pp. 5–14. DOI: 10.14529/cmse190401.

References

1. Ivanov V.K. About Application of Picard Method to the Solution of Integral Equations for the First Kind. Bui. Inst. Politehn. Iasi. 1968. vol. 4, no. 34. pp. 71–78. (in Russian)
2. Ivanov V.K., Vasin V.V., Tanana V.P. Theory of Linear Ill-Posed Problem and Application. Moscow, Nauok, 1978. 206 p. (in Russian)
3. Kabanikhin S.I. Inverse and Ill-posed Problems: Theory and Applications. Inverse and Ill-Posed Problems, Ser. 55. De Gruyter, 2012. 458 p.
4. Tanana V.P., Sidikova A.I. Optimal Methods for Solving Ill-Posed Heat Conduction Problems. Inverse and ill-posed problems, Ser. 62. De Gruyter, 2018. 138 p.
5. Tikhonov A.N. On the Regularization of Ill-Posed Problems. Proceedings of the USSR Academy of Sciences, 1963. vol. 153, no. 1. pp. 49–52. (in Russian)
6. Lavrent'ev M.M. On Some Ill-Posed Problems of Mathematical Physics. Novosibirsk, Siberian Branch of the Academy of Sciences of the USSR, 1962. 92 p. (in Russian)
7. Mu H., Li J., Wang X. Optimization Based Inversion Method for the Inverse Heat Conduction Problems. IOP Conference Series: Earth and Environmental Science. 2017. vol. 64, no. 1. pp. 1–9. DOI: 10.1088/1755-1315/64/1/012094.

8. Duda P. Solution of Inverse Heat Conduction Problem Using the Tikhonov Regularization Method. Journal of Thermal Science. 2017. vol. 26, no. 1. pp. 60–65. DOI: 10.1007/s11630-017-0910-2.
9. Frąckowiak A., Botkin N.D., Ciałkowski M. Iterative Algorithm for Solving the Inverse Heat Conduction Problems with the Unknown Source Function. Inverse Problems in Science and Engineering. 2015. vol. 23, no. 6. pp. 1056–1071. DOI: 10.1080/17415977.2014.986723.
10. Yang S., Xiong X. A. Tikhonov Regularization Method for Solving an Inverse Heat Source Problem. Bull. Malays. Math. Sci. Soc. 2018. vol. 5, no. 19. pp. 1–12. DOI: 10.1007/s40840-018-0693-y.

УДК 517.977, 519.688 DOI: 10.14529/cmse190402

АЛГОРИТМ ДИНАМИЧЕСКОЙ РЕКОНСТРУКЦИИ ВХОДОВ СТОХАСТИЧЕСКОГО ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ: НАСТРОЙКА ПАРАМЕТРОВ И ЧИСЛЕННЫЕ ЭКСПЕРИМЕНТЫ*

© 2019 Л.А. Мельникова, В.Л. Розенберг

Институт математики и механики им. Н.Н. Красовского УрО РАН

(620990 Екатеринбург, ул. С. Ковалевской, д. 16)

E-mail: meln@imm.uran.ru, rozen@imm.uran.ru

Поступила в редакцию: 16.04.2019

Задача реконструкции неизвестных входов стохастического дифференциального уравнения исследуется с позиций подхода теории динамического обращения. Рассматривается постановка, в которой одновременное восстановление возмущений в детерминированном и стохастическом членах уравнения проводится на основе дискретной информации о некотором количестве реализаций случайного процесса. Задача сводится к обратной задаче для системы обыкновенных дифференциальных уравнений, которым удовлетворяют математическое ожидание и ковариационная матрица исходного процесса. Разработан программно-ориентированный алгоритм решения, основанный на конструкциях теории позиционного управления с моделью; получена оценка его точности относительно количества доступных измерению реализаций. Предложена программная процедура настройки параметров алгоритма для получения наилучшего результата аппроксимации различных возмущений, удовлетворяющих априорным ограничениям, в конкретной динамической системе. Искомые зависимости параметров алгоритма от количества измеряемых реализаций определяются эмпирически через решение специальной экстремальной задачи, в которой минимизируется отклонение выхода алгоритма от тестовой функции. Для оптимизации времяемкого процесса адаптации алгоритма к системе, предполагающего моделирование большого числа независимых траекторий стохастического уравнения, используется распараллеливание вычислений. Приведен модельный пример, иллюстрирующий предложенные конструкции. Рассмотрена система, упрощенно описывающая популяционную динамику двух взаимодействующих видов. Представлены результаты расчетов и характеристики эффективности распараллеливания.

Ключевые слова: стохастическое дифференциальное уравнение, динамическая реконструкция, управляемая модель, настройка параметров, распараллеливание вычислений.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Мельникова Л.А., Розенберг В.Л. Алгоритм динамической реконструкции входов стохастического дифференциального уравнения: настройка параметров и численные эксперименты // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 4. С. 15–29. DOI: 10.14529/cmse190402.

Введение

Проблематика реконструкции неизвестных характеристик управляемых систем в режиме реального времени на основе неточной и/или неполной информации о фазовом состоянии возникает во многих теоретических и практических исследованиях и по этой причине привлекает пристальное внимание специалистов в различных областях. Задачи динамического восстановления входов могут рассматриваться как специальный случай задач теории идентификации; как правило, они являются некорректными (в силу

*Статья рекомендована к публикации программным комитетом научной конференции «Параллельные вычислительные технологии (ПаВТ) 2019».

неединственности искомого параметра, как следствия неточности измерений, и в силу разрывности обратного оператора) и требуют применения регуляризирующих процедур. В отличие от достаточно большого количества работ, в которых изучаются апостериорные постановки обратных задач, а разрешающие алгоритмы могут использовать всю историю измерения выхода системы, мы ориентируемся на так называемый метод динамического обращения, предложенный и развитый в работах А.В. Кряжимского, Ю.С. Осипова и их коллег [1–3]. Он основан на сочетании принципов теории позиционного управления [4] и идей теории некорректных задач [5]. Задача реконструкции заменяется задачей позиционного управления вспомогательной динамической системой (моделью) с использованием обратных связей. Фактически необходимо подобрать, на основании поступающей информации о фазовом состоянии, такой закон модельного управления, чтобы движение модели отслеживало динамику реальной системы. При выполнении определенных условий оказывается, что в этом случае модельное управление аппроксимирует (в смысле определенной метрики) неизвестное входное воздействие. Метод динамического обращения был реализован для различных систем, описываемых обыкновенными дифференциальными уравнениями (ОДУ) [1, 2], дифференциально-функциональными уравнениями [6], уравнениями и вариационными неравенствами с распределенными параметрами [3, 7] и др. Были также разработаны алгоритмы восстановления, функционирующие в условиях неполной информации о фазовом состоянии динамической системы [2, 8].

Настоящая работа продолжает исследования (в рамках указанной теории) задач реконструкции для линейного/квазилинейного стохастического дифференциального уравнения (СДУ) [9–11]. Предполагается, что в системе действуют два неизвестных неслучайных возмущения: одно входит в детерминированный член уравнения и влияет на математическое ожидание искомого процесса, второе входит в интеграл Ито и характеризует амплитуду случайных помех. Рассматриваются условия, при выполнении которых возможно одновременное восстановление возмущений на основе дискретной по времени информации (в том числе неполной) о некотором, достаточно большом, количестве реализаций фазового вектора. Показывается, что задача для СДУ сводится к обратной задаче для системы ОДУ, описывающей динамику математического ожидания и ковариационной матрицы исходного процесса. Разработана конечношаговая программно-ориентированная разрешающая процедура, базирующаяся на конструкциях теории позиционного управления с моделью, адаптированных ранее к обратным задачам в различных постановках для систем ОДУ [1, 2, 8]; получена оценка качества восстановления в зависимости от количества доступных измерению реализаций [11]. Существенную трудность в применении алгоритмов реконструкции указанного типа представляет их адаптация к конкретной динамической системе. Дело в том, что до сих пор не существует универсальной процедуры подбора модельных параметров даже в случае задач для ОДУ; сложности такого рода «всплывали» во многих численных примерах [2]. Новизна данной статьи состоит в рассмотрении этого прикладного аспекта решения задачи реконструкции и разработке эмпирической процедуры для автоматизации процесса калибровки модели. В рассматриваемой задаче настройка параметров алгоритма предполагает многочисленные прогонки вычислительного модуля и моделирование большого числа траекторий СДУ. Для оптимизации этого процесса используется распараллеливание времяемких вычислительных процедур.

Статья организована следующим образом. В разделе 1 приводится формальная постановка задачи динамической реконструкции входов квазилинейной системы СДУ. Раздел 2 посвящен описанию процедуры сведения задачи для системы СДУ к задаче для системы ОДУ. Конструктивный алгоритм решения последней и результаты, характеризующие его сходимость, представлены в разделе 3. Прикладные аспекты, связанные с настройкой параметров алгоритма на определенную систему и с распараллеливанием этого процесса, обсуждаются в разделе 4. В разделе 5 приводится иллюстративный пример с результатами реконструкции возмущений в модельной системе СДУ. В заключении резюмируются полученные результаты и намечаются направления дальнейших исследований.

1. Постановка задачи реконструкции

Простые линеаризованные модели могут быть полезны при изучении динамики рыночных цен при воздействии случайных факторов, изменения численности многовидовой биологической популяции в стохастической среде или процессов хаотического движения однотипных частиц. В этом контексте рассматривается квазилинейная система СДУ специального вида с диффузией, зависящей от фазового состояния:

$$dx(t,\omega) = (A(t)x(t,\omega) + B(t)u_1(t) + f(t))\,dt + U_2(t)x(t,\omega)\,d\xi(t,\omega), \quad x(0,\omega) = x_0. \qquad (1)$$

Здесь $t \in T = [0,\vartheta]$, $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ — вектор-столбец, $\xi \in \mathbb{R}$; x_0 — известный детерминированный или случайный (нормально распределенный) вектор начальных условий; $\omega \in \Omega$, (Ω, F, P) — вероятностное пространство; $\xi(t,\omega)$ — стандартный скалярный винеровский процесс (т. е. выходящий из нуля процесс с нулевым математическим ожиданием и дисперсией, равной t); $A(t) = \{a_{ij}(t)\}$, $B(t) = \{b_{ij}(t)\}$ и $f(t) = \{f_i(t)\}$ — непрерывные матричные функции размерности $n \times n$, $n \times r$ и $n \times 1$ соответственно. На систему действуют два внешних неслучайных возмущения: вектор $u_1(t) = (u_{11}(t), u_{12}(t), \ldots, u_{1r}(t)) \in \mathbb{R}^r$ и диагональная матрица $U_2(t) = \{u_{21}(t), u_{22}(t), \ldots, u_{2n}(t)\} \in \mathbb{R}^{n\times n}$. Воздействие u_1 входит в детерминированную компоненту и влияет на математическое ожидание искомого процесса. Поскольку $U_2 x d\xi = (u_{21}x_1 d\xi, u_{22}x_2 d\xi, \ldots, u_{2n}x_n d\xi)$, то можно считать, что вектор $u_2 = (u_{21}, u_{22}, \ldots, u_{2n})$ регулирует амплитуду случайных помех. Полагаем, что векторы u_1 и u_2 принимают значения из априори известных выпуклых компактов S_{u1} и S_{u2}, $u_1(\cdot) \in L_2(T;\mathbb{R}^r)$, $u_2(\cdot) \in L_2(T;\mathbb{R}^n)$.

Под решением системы (1) понимаем случайный процесс, удовлетворяющий при любом t с вероятностью 1 соответствующему интегральному тождеству, содержащему в правой части стохастический интеграл Ито. Известно, что сделанные предположения обеспечивают существование единственного решения, являющегося нормальным марковским процессом с непрерывными реализациями [12].

Рассматриваемая задача формулируется следующим образом. В дискретные, достаточно частые, моменты времени $\tau_i \in T$, $\tau_i = i\delta$, $\delta = \vartheta/l$, $i \in [0:l]$, поступает информация о некотором количестве N реализаций случайного процесса $x(\tau_i)$. Считаем, что $l = l(N)$ и существуют оценки m_i^N математического ожидания процесса $m(t) = Mx(t)$ и D_i^N ковариационной матрицы $D(t) = M(x(t) - m(t))(x(t) - m(t))'$ (штрих означает транспонирование) такие, что выполняется соотношение

$$P\Big(\max_{i\in[1:l(N)]} \left\{\left\|m_i^N - m(\tau_i)\right\|, \left\|D_i^N - D(\tau_i)\right\|\right\} \le h(N)\Big) = 1 - g(N), \qquad (2)$$

причем $h(N),\ g(N) \to 0$ при $N \to \infty$. Символ $\|\cdot\|$ обозначает евклидову норму в соответствующем пространстве. Статистические процедуры построения стандартных оценок m_i^N и D_i^N [13] допускают модификации, обеспечивающие выполнение (2).

Задача состоит в построении алгоритма динамической реконструкции неизвестных возмущений $u_1(\cdot)$ и $u_2(\cdot)$, определяющих случайный процесс $x(\cdot)$, по дискретной информации о его реализациях, причем вероятность сколь угодно малого отклонения приближений от искомых входов в метрике соответственно пространств $L_2(T;\mathbb{R}^r)$ и $L_2(T;\mathbb{R}^n)$ должна быть близка к 1 при достаточно большом N и специальным образом согласованном с N шаге временной дискретизации $\delta = \delta(N) = \vartheta/l(N)$.

Квазилинейность системы (1) позволяет, в соответствии с методом моментов [14], заменить задачу для СДУ задачей для системы ОДУ, которой удовлетворяют математическое ожидание и ковариационная матрица искомого процесса. После чего возможно организовать процедуру динамической реконструкции возмущений в детерминированном и стохастическом членах правой части в ситуации, когда доступны одновременные измерения достаточно большого количества траекторий (например, движения однотипных частиц).

2. Редукция задачи

Следуя [9–11], заменим задачу восстановления для квазилинейного СДУ задачей для системы ОДУ. Нам понадобится

Лемма. *Стандартные оценки m_i^N математического ожидания $m(\tau_i)$ и D_i^N ковариационной матрицы $D(\tau_i)$, построенные по N $(N > 1)$ реализациям $x^1(\tau_i), x^2(\tau_i), \ldots, x^N(\tau_i)$ случайных величин $x(\tau_i)$ по следующим правилам* [13]:

$$m_i^N = \frac{1}{N}\sum_{r=1}^{N} x^r(\tau_i), \tag{3}$$

$$D_i^N = \frac{1}{N-1}\sum_{r=1}^{N}(x^r(\tau_i) - m_i^N)(x^r(\tau_i) - m_i^N)', \tag{4}$$

обеспечивают выполнение свойства (2).

Отметим, что оценку (4) с целью оптимизации вычислений (в частности, для их распараллеливания) удобно переписать в виде:

$$D_i^N = \frac{1}{N-1}\sum_{r=1}^{N} x^r(\tau_i)(x^r(\tau_i))' - \frac{N}{N-1} m_i^N (m_i^N)'. \tag{5}$$

Доказательство Леммы приведено в [11], где получены следующие параметры оценки (2):

$$\delta(N) = C_1/N^{\min\{\alpha,\alpha(1/2+3\epsilon)\}}, \quad l(N) = \vartheta/\delta(N),$$

$$h(N) = C_2/N^{1/2-\epsilon}, \quad g(N) = C_3/N^{\min\{1-\alpha,(1-\alpha)(1/2+3\epsilon)\}}. \tag{6}$$

Здесь $0 < \epsilon < 1/2$, $0 < \alpha < 1$ — параметры, позволяющие регулировать порядки малости в аппроксимационной и вероятностной частях оценки (2). Очевидно, $\delta(N) \to 0$, а также $h(N)$, $g(N) \to 0$, как требуется в соотношении (2), при любых допустимых значениях ϵ и α. Через C_i обозначаем вспомогательные константы, которые зависят от структуры системы и от заданных ограничений на возмущения (но не от самих функций, подлежащих реконструкции) и могут быть выписаны явно.

Кратко опишем процедуру получения системы ОДУ, которой удовлетворяют математическое ожидание и ковариационная матрица исходного процесса (подробное изложение приведено в [10, 11]). Введем обозначения: $m_0 = Mx_0$, $D_0 = M(x_0 - m_0)(x_0 - m_0)'$. В силу квазилинейности исходной системы и равенства нулю математического ожидания интеграла Ито величина $m(t)$ не зависит от $u_2(t)$; ее динамика задается уравнением

$$\dot{m}(t) = A(t)m(t) + B(t)u_1(t) + f(t),\ m \in \mathbb{R}^n,\ m(0) = m_0.$$

Ковариационная матрица $D(t)$ явно не зависит от $u_1(t)$; в соответствии со схемой метода моментов [14], получаем следующее матричное уравнение:

$$\dot{D}(t) = A(t)D(t) + D(t)A'(t) + U_2(t)(D(t) + m(t)m'(t))U_2'(t),\ D \in \mathbb{R}^{n\times n},\ D(0) = D_0.$$

Его удобно переписать в виде более традиционного для рассматриваемых задач векторного уравнения, размерность которого, с учетом симметричности матрицы $D(t)$, равна $n_d = (n^2 + n)/2$. Вводится вектор $d(t) = \{d_s(t)\}$, $s \in [1 : n_d]$, состоящий из последовательно записанных и пронумерованных элементов матрицы $D(t)$, взятых построчно, начиная с элемента, расположенного на главной диагонали; его координаты находятся по элементам матрицы $D(t) = \{d_{ij}(t)\}$, $i, j \in [1 : n]$:

$$d_s(t) = d_{ij}(t), \quad i \le j, \quad s = (n - i/2)(i - 1) + j.$$

Затем матричное уравнение записывается в виде

$$\dot{d}(t) = \bar{A}(t)d(t) + \bar{B}(d(t), m(t))u_3(t),\ d(t_0) = d_0.$$

Здесь матрица $\bar{A}(t) : T \to R^{n_d\times n_d}$ и диагональная матрица $\bar{B}(d(t), m(t)) : T \to R^{n_d\times n_d}$ могут быть выписаны явно [11], вектор $u_3(t) = \{u_{3s}(t)\}$, $s \in [1 : n_d]$ получается из $u_2(t)$ по формулам $u_{3s} = u_{2i}u_{2j}$, $i \le j$, $s = (n - i/2)(i - 1) + j$, он принимает значения из некоторого выпуклого компакта $S_{u3} \in \mathbb{R}^{n_d}$ и принадлежит пространству $L_2(T; \mathbb{R}^{n_d})$. Начальное состояние d_0 получается из D_0, а измерения d_i^N — из D_i^N.

Таким образом, имеем систему уравнений размерности $n + n_d$:

$$\dot{m}(t) = A(t)m(t) + B(t)u_1(t) + f(t), \quad m(0) = m_0, \tag{7}$$

$$\dot{d}(t) = \bar{A}(t)d(t) + \bar{B}(d(t), m(t))u_3(t), \quad d(0) = d_0. \tag{8}$$

Переформулируем исходную задачу восстановления. Исходя из (2), считаем, что по ходу развития процесса в дискретные моменты времени $\tau_i \in T$ поступает неточная информация m_i^N, d_i^N о фазовом состоянии системы (7), (8) такая, что выполняется соотношение

$$P\Big(\max_{i\in[1:l(N)]}\left\{\left\|m_i^N - m(\tau_i)\right\|, \left\|d_i^N - d(\tau_i)\right\|\right\} \le h(N)\Big) = 1 - g(N), \tag{9}$$

где $h(N)$, $g(N) \to 0$ при $N \to \infty$. Следует указать алгоритм динамической реконструкции неизвестных возмущений $u_1(\cdot)$ и $u_3(\cdot)$ по информации (9), причем вероятность сколь угодно малого отклонения приближений от искомых входов в метрике соответственно пространств $L_2(T; \mathbb{R}^r)$ и $L_2(T; \mathbb{R}^{n_d})$ должна быть близка к 1 при достаточно большом N и специальным образом согласованном с N шаге выполнения измерений $\delta = \delta(N) = \vartheta/l(N)$. Отметим, что при дополнительных предположениях восстанавливается и функция $u_2(\cdot)$ с помощью координат вектора $u_3(\cdot)$, равных u_{2i}^2, $i \in [1 : n]$.

Система (7), (8) нелинейна по фазовым переменным, но линейна по управлению. Сформулированная для нее задача соответствует рассмотренной в [1, 2]. В настоящей работе модифицируется конечношаговый разрешающий алгоритм, предложенный ранее для ОДУ, и показывается, что имеет место конструктивное согласование его параметров с количеством доступных измерению реализаций исходного случайного процесса.

3. Алгоритм реконструкции возмущений

Опишем поэтапно искомый алгоритм для системы (7), (8). В начальный момент $\tau_0 = 0$ фиксируется значение N, определяются величины $l^N = l(N)$, $h^N = h(N)$ и $g^N = g(N)$ (см. (6)) и строится равномерное разбиение промежутка T с шагом $\delta^N = \vartheta/l^N$: $\tau_i \in T$, $\tau_i = i\delta^N$, $i \in [0 : l^N]$. Вводится дискретная модель, динамика и начальное состояние которой определяются соотношениями

$$w_m(\tau_{i+1}) = w_m(\tau_i) + (A(\tau_i)m_i^N + B(\tau_i)v_{1i}^N + f(\tau_i))\delta^N,\ w_m(0) = m_0, \tag{10}$$

$$w_d(\tau_{i+1}) = w_d(\tau_i) + (\bar{A}(\tau_i)d_i^N + \bar{B}(d_i^N, m_i^N)v_{3i}^N)\delta^N,\ w_d(0) = d_0. \tag{11}$$

Здесь $i \in [0 : l^N - 1]$, w_m, w_d — модельные переменные, аппроксимирующие фазовое состояние системы (7), (8), v_{1i}^N, v_{3i}^N — управляющие воздействия, вычисляемые в момент τ_i по правилам, которые конкретизируются ниже.

Работа алгоритма разбивается на l^N однотипных шагов. На i-м шаге, который выполняется на интервале $(\tau_i, \tau_{i+1}]$, исходными данными для вычислений служат оценки m_i^N, d_i^N и сформированное к этому моменту состояние модели $w_m(\tau_i)$, $w_d(\tau_i)$. Модельные управления определяются следующим образом: v_{1i}^N и v_{3i}^N представляют собой единственные решения экстремальных задач

$$v_{1i}^N = \arg\min\left\{2\langle w_m(\tau_i) - m_i^N,\, B(\tau_i)v_1\rangle + \alpha_m^N \|v_1\|^2 :\, v_1 \in S_{u1}\right\}, \tag{12}$$

$$v_{3i}^N = \arg\min\left\{2\langle w_d(\tau_i) - d_i^N,\, \bar{B}(d_i^N, m_i^N)v_3\rangle + \alpha_d^N \|v_3\|^2 :\, v_3 \in S_{u3}\right\}, \tag{13}$$

где $\langle\cdot,\,\cdot\rangle$ — скалярное произведение в соответствующем евклидовом пространстве, α_m^N, α_d^N — параметры регуляризации. После вычисления управлений (12) и (13) пересчитывается согласно (10) и (11) состояние модели $w_m(\tau_{i+1})$, $w_d(\tau_{i+1})$. Процесс заканчивается в конечный момент времени ϑ.

Пусть $U_* = U_*(m(\cdot), d(\cdot))$ — множество всех возмущений $u(\cdot) = (u_1(\cdot), u_3(\cdot)) \in L_2(T; \mathbb{R}^{r+n_d})$, порождающих пару $(m(\cdot), d(\cdot))$. Минимальный по норме пространства $L_2(T; \mathbb{R}^{r+n_d})$ элемент множества U_* обозначим через $u_*(\cdot)$, его существование и единственность обусловлены выпуклостью U_* и строгой выпуклостью нормы в $L_2(T; \mathbb{R}^{r+n_d})$, см. [1, 2]. Функция $u^N(t) = (v_1^N(t), v_3^N(t))$, $v_1^N(t) = v_{1i}^N$, $v_3^N(t) = v_{3i}^N$, $t \in [\tau_i, \tau_{i+1})$, $i \in [0 : l^N - 1]$ определяется из соотношений (12), (13). Приведем основной результат [11] о сходимости алгоритма.

Теорема. *Пусть выполняются условия согласования параметров*

$$h^N,\, \delta^N,\, \alpha_m^N,\, \alpha_d^N \to 0,\ \frac{\delta^N + h^N}{\alpha_m^N},\ \frac{\delta^N + h^N}{\alpha_d^N} \to 0 \quad \textit{при } N \to \infty. \tag{14}$$

Тогда последовательность $\{u^N(\cdot)\}$ *компактна в* $L_2(T; \mathbb{R}^{r+n_d})$ *и имеет место сходимость*

$$P\left(\|u^N(\cdot) - u_*(\cdot)\|_{L_2(T;\mathbb{R}^{r+n_d})} \to 0\right) \to 1 \quad \textit{при} \quad N \to \infty. \tag{15}$$

В предположении об ограниченности вариации реальных возмущений справедлива следующая оценка точности алгоритма относительно количества реализаций процесса, доступных измерению:

$$P\left(\|u^N(\cdot)-u_*(\cdot)\|_{L_2(T;\mathbb{R}^{r+n_d})}\le D_1\left(\frac{1}{N}\right)^{2/13}\right)=1-D_2\left(\frac{1}{N}\right)^{2/13}, \tag{16}$$

где D_1 и D_2 — некоторые положительные константы.

Как показано в [11], для получения оценки (16) следует положить

$$h^N=1/N^{6/13},\quad \delta^N=K_1/N^{6/13},\quad \alpha_m^N=K_2/N^{4/13},\quad \alpha_d^N=K_3/N^{4/13}, \tag{17}$$

где K_1, K_2 и K_3 — некоторые положительные константы.

Заметим, что константы в (16), (17) зависят от параметров исходной системы, от априорных структурных ограничений, в частности, на вариацию реальных возмущений, но не зависят от самих восстанавливаемых функций. Выбор показателей степени величины $1/N$ в аппроксимационной и вероятностной частях оценки типа (16) допускает определенный произвол в диапазоне изменения ϵ и α из соотношений (6). В теореме критерием является равенство показателей, его обеспечивают соотношения (17). Очевидно, возможны другие варианты. В случае, когда вектор $u_2=(u_{21},u_{22},\ldots,u_{2n})$ единственен, а его координаты неотрицательны (ограничение представляется естественным для характеристики помехи с нулевым средним, ниже считаем эти условия выполненными), алгоритм (10)–(14) восстанавливает и саму функцию $u_2(\cdot)$ в смысле метрики пространства $L_2(T;\mathbb{R}^n)$ с оценкой точности вида (16).

4. Настройка параметров алгоритма

Количество N доступных измерению траекторий исходного СДУ (1) определяет как погрешность h^N входных оценок (2), (9), так и параметры алгоритма δ^N, α_m^N и α_d^N, обеспечивающие сходимость (15), (16). Соотношения не прописаны явно, как это необходимо для расчетов, а задаются в виде порядка сходимости при $1/N\to 0$ с точностью до констант, для которых оценки сверху могут быть выписаны явно, но сами значения нуждаются в уточнении. Отметим подтвержденную экспериментально чувствительность алгоритма к ограничениям на функции, формирующие исходную систему (1), на множества допустимых значений возмущений S_{u1} и S_{u2}, на вариацию возмущений и др. Необходимо получить соотношения (17), обеспечивающие успешную работу алгоритма по восстановлению в конкретной системе вида (1) любых возмущений, удовлетворяющих априорным ограничениям (в этом суть калибровки алгоритма). На данном этапе полагаем, что «оптимальные» зависимости параметров определяются эмпирически через решение следующей экстремальной задачи:

$$(K_1^*,\,K_2^*,\,K_3^*)=\arg\min\left\{\sum_{N\in I^N}\beta^N\|u^N_{K_1,K_2,K_3}(\cdot)-u_*(\cdot)\|_{L_2(T)}:\,(K_1,\,K_2,\,K_3)\in S_K\right\}. \tag{18}$$

Здесь $L_2(T)=L_2(T;\mathbb{R}^{r+n_d})$, $u^N_{K_1,K_2,K_3}(\cdot)$ — выход алгоритма, зависящий от констант из (17); I^N — множество рассматриваемых значений N, β^N — весовые коэффициенты, $\sum_{N\in I^N}\beta^N=1$; S_K — некоторое множество допустимых значений вектора $(K_1,\,K_2,\,K_3)$, как правило, параллелепипед в положительном октанте. Фиксируется некоторая реальная функция $u_*(\cdot)$, вводится равномерная сетка по K_1, K_2, K_3 и полным перебором по ней

решается задача (18), что требует многочисленных прогонок вычислительного модуля с моделированием большого числа независимых траекторий СДУ (1) для формирования оценок (2) для различных N. Эксперименты подтверждают применимость решения (18) для восстановления других реальных возмущений при сохранении заданных ограничений. Для получения динамики уравнения (1) используется метод Эйлера с заменой винеровского процесса последовательностью случайных импульсов (аппроксимационная схема детально описана в [15]); его среднеквадратический порядок точности для квазилинейного СДУ равен $O(\delta_s)$, где δ_s — шаг моделирования. Этот шаг должен быть мал относительно шага δ^N, с которым поступает информация о траекториях и, соответственно, работает процедура восстановления неизвестных функций $u_1(t)$ и $u_2(t)$. Таким образом, в процедуре подбора параметров алгоритма для конкретной системы задействованы две временные сетки, причем моделирование с более мелким шагом имеет место только на стадии настройки алгоритма, при решении реальной задачи входная информация должна поступать из «внешнего мира». Отметим, что вероятностный характер оценки сходимости (16) предполагает усреднение выхода алгоритма при фиксированном наборе параметров по серии запусков.

Вычислительные эксперименты показали, что описанная процедура, подразумевая множество прогонов на сетке по нескольким параметрам алгоритма, требует значительных ресурсов при расчете на последовательной машине, однако допускает естественное (и эффективное) распараллеливание. Оно базируется на стандартной схеме «мастер-рабочий» [16] с единым загрузочным MPI-модулем, почти равномерным распределением вычислительной нагрузки по всем ядрам и отсутствием обменов между рабочими модулями. Моделирование большого числа независимых траекторий СДУ с мелким временным шагом является единственной времяемкой частью процедуры, поэтому оно распределяется поровну между всеми ядрами. В момент поступления входной информации об оценках (2), (3), (5) все модули проводят расчет одинаковых по размеру порций $x^r(\tau_i)$ и усредняют как сами величины, так и их квадраты, в соответствии с формулами (3), (5). Затем рабочие отправляют свой вклад в оценки мастеру, который добавляет свою порцию и, усредняя полученные суммы по количеству задействованных ядер (легитимность операции обеспечивается одинаковостью порций), окончательно формирует (3) и (5). Все вычисления в соответствии с алгоритмом реконструкции (10)–(13) выполняются мастером. Блок-схема основного фрагмента параллельной процедуры приведена на рис. 1.

5. Модельный пример

Рассмотрим квазилинейную систему, описывающую случайный процесс, близкий к известному средне-возвратному процессу Орнштейна—Уленбека [12]:

$$dx_1(t) = (-2x_1(t) + 0.1x_2(t) + u_{11}(t) + u_{12}(t) + 1)\,dt + u_{21}(t)x_1(t)\,d\xi(t),$$

$$dx_2(t) = (0.1x_1(t) - x_2(t) + 2u_{12}(t) + 2)\,dt + u_{22}(t)x_2(t)\,d\xi(t),$$

$$t \in T = [0,1],\ x_1(0) = 1, x_2(0) = 1. \tag{19}$$

Здесь $u(t) = (u_1(t),\, u_2(t))$, $u_1(t) = (u_{11}(t),\, u_{12}(t))$ и $u_2(t) = (u_{21}(t),\, u_{22}(t))$ — неизвестные возмущения, подлежащие восстановлению, $u_{1i}(t) \in [-2,2]$, $u_{2i}(t) \in [0,2]$, $i = 1,2$; $\xi(t)$ — стандартный скалярный винеровский процесс. Уравнения типа (19) используются, в частности, в некоторых простейших моделях, описывающих динамику численности

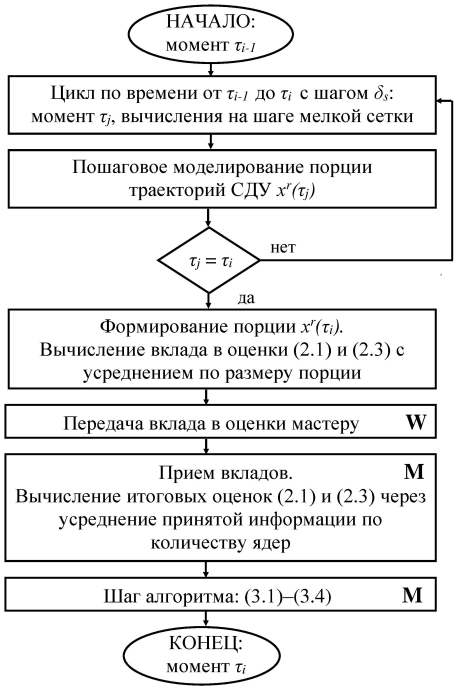


Рис. 1. Схема основного фрагмента параллельной процедуры. Операции, выполняемые только мастером, помечены символом «M», только рабочими — символом «W» (остальные операции выполняются на всех ядрах)

относительно устойчивых популяций биологических особей [12]. В таком случае величина $x(t) = (x_1(t), x_2(t))$ представляет текущую численность (в условных единицах) двух взаимодействующих видов; структура детерминированной части уравнения определяет некоторые численности (например, средние величины за прошедший интервал), стремление вернуться к которым «подсознательно» присутствует у популяций. Такому возврату могут препятствовать соседи и воздействие внешних факторов через возмущение $u_1(t)$, влияющее на математическое ожидание процесса, и возмущение $u_2(t)$, характеризующее амплитуду случайных шумов. Измерению (в дискретные моменты времени) доступны реализации $x(t)$, соответствующие различным колониям. Запишем для (19) систему ОДУ, соответствующую (7), (8):

$$\dot{m}_1(t) = -2m_1(t) + 0.1m_2(t) + u_{11}(t) + u_{12}(t) + 1,$$

$$\dot{m}_2(t) = 0.1m_1(t) - m_2(t) + 2u_{12}(t) + 2,$$

$$\dot{d}_1(t) = -4d_1(t) + 0.2d_2(t) + (d_1(t) + m_1^2(t))u_{31}(t),$$

$$\dot{d}_2(t) = 0.1d_1(t) - 3d_2(t) + 0.1d_3(t) + (d_2(t) + m_1(t)m_2(t))u_{32}(t),$$

$$\dot{d}_3(t) = 0.2d_2(t) - 2d_3(t) + (d_3(t) + m_2^2(t))u_{33}(t),$$

$$x_1(0) = 1, x_2(0) = 1, d_1(0) = 0, d_2(0) = 0, d_3(0) = 0. \tag{20}$$

Здесь $u_{31} = u_{21}^2$, $u_{32} = u_{21}u_{22}$, $u_{33} = u_{22}^2$.

Целью вычислительных экспериментов являлась адаптация алгоритма к конкретной динамической системе (19)–(20), т.е. получение универсального набора зависимостей его параметров δ^N, α_m^N и α_d^N от N через K_1, K_2, K_3 (см. (17)) для восстановления любых

возмущений, удовлетворяющих заданным ограничениям. В первой серии для решения (18) были выбраны функции

$$u_1(t) = (\sin 10t, 1),\ u_2(t) = (\sqrt{t}, 10\sqrt{t^3/3 - t^2/2 + 3t/16})$$

и параметры $K_i \in [0.1, 5]$, $i = 1, 2, 3$, шаг сетки по K_i равный 0.05, $I^N = \{10^3, 10^6\}$, $\beta^N = \{0.5, 0.5\}$. Были получены оптимальные значения констант (в смысле (18)) в соотношениях (17) для системы (19) и принятых ограничений на возмущения: $K_1^* = 0.6$, $K_2^* = 0.35$, $K_3^* = 3.5$. Результаты восстановления функций $u_1(t)$ и $u_2(t)$, полученные для найденных K_i и различных N, в принципе согласуются с основным утверждением статьи, подтверждая сходимость вида (15): чем больше N, тем меньше погрешность реконструкции, см. рис. 2, 3. Затем полученные соотношения параметров были протестированы на

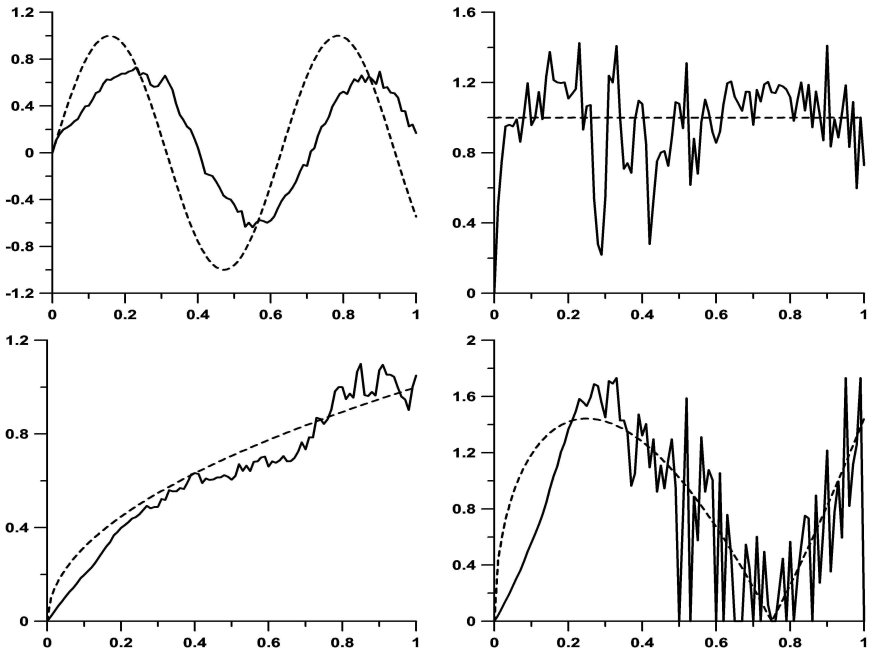


Рис. 2. Восстановление возмущений u_{11}, u_{12} (верхний ряд) и u_{21}, u_{22} (нижний ряд): реальные функции изображены пунктирной линией, результаты их восстановления — сплошной; горизонтальная ось соответствует времени t, вертикальная — значениям возмущений. Параметры: $N = 10^3$, $\delta^N = 0.025$, $\delta_s = \delta^N/10^2$, $\alpha_m^N = 0.04$, $\alpha_d^N = 0.4$, погрешность $\|u^N(\cdot) - u_*(\cdot)\|_{L_2(T)} = 0.868$

других возмущениях, удовлетворяющих заданным геометрическим ограничениям. На рис. 4 приведен пример восстановления $u_2(t) = (\sqrt{2}(1 - 2|t - 0.5|), 1 + 0.5\sin 30t)$ со значениями параметров алгоритма из варианта, представленного на рис. 3. Точность реконструкции сравнима с результатами, полученными для тестового (в смысле (18)) возмущения. Отметим, что вторая координата возмущения u_2 восстанавливается хуже первой, что объясняется большей амплитудой шума во втором уравнении системы (19).

Для иллюстрации показателей, характеризующих качество распараллеливания, был выбран вариант из рис. 3 с симуляцией 128000 траекторий уравнения (19). Моделирование проводилось в Институте математики и механики им. Н.Н.Красовского УрО РАН на гибридном суперкомпьютере «Уран», который состоит из 196 вычислительных узлов, оснащенных процессорами Intel Xeon и графическими ускорителями NVIDIA Tesla, и имеет пиковую производительность более 215 Тфлопс. Отметим, что в общей сложности

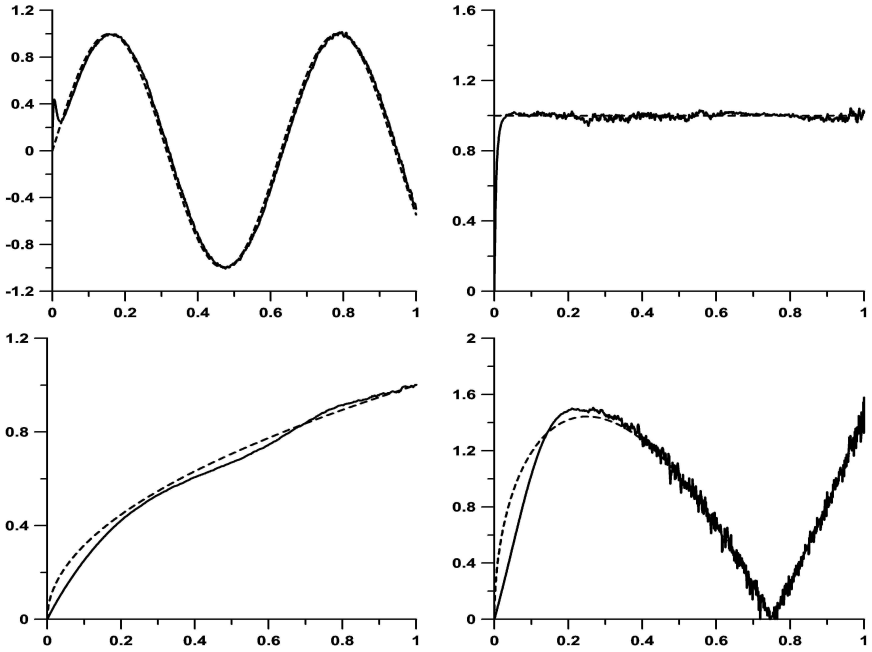


Рис. 3. Восстановление возмущений u_{11}, u_{12} (верхний ряд) и u_{21}, u_{22} (нижний ряд): реальные функции изображены пунктирной линией, результаты их восстановления — сплошной; горизонтальная ось соответствует времени t, вертикальная — значениям возмущений. Параметры: $N = 10^6$, $\delta^N = 0.001$, $\delta_s = \delta^N/10^2$, $\alpha_m^N = 0.005$, $\alpha_d^N = 0.05$, погрешность $\|u^N(\cdot) - u_*(\cdot)\|_{L_2(T)} = 0.118$

пользователям доступно 1940 вычислительных ядер CPU, 314 платы GPU и 4 Тб оперативной памяти. В наших экспериментах использовался компактный раздел кластера, называемый *Apollo* (16 узлов по два 18-и ядерных процессора Intel Xeon CPU E5-2697 с частотой 2.30 ГГц, оперативная память 256 Гб, локальный жесткий диск 1 Тб); его узлы объединены высокоскоростной сетью Infiniband нового поколения со скоростью передачи данных 100 Гбит/с.

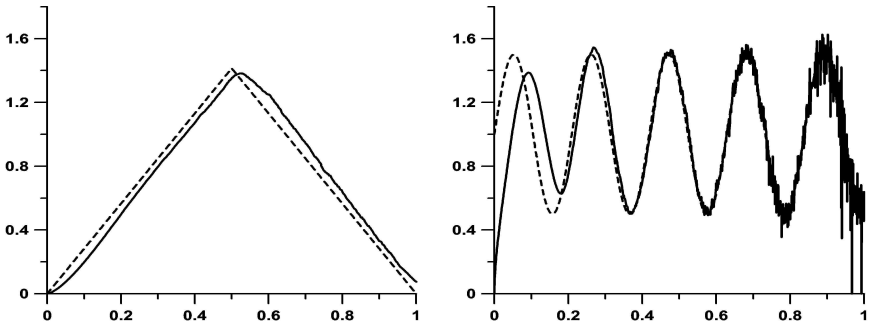


Рис. 4. Восстановление альтернативных возмущений u_{21} и u_{22}: реальные функции изображены пунктирной линией, результаты их восстановления — сплошной; горизонтальная ось соответствует времени t, вертикальная — значениям возмущений. Параметры: $N = 10^6$, $\delta^N = 0.001$, $\delta_s = \delta^N/10^2$, $\alpha_m^N = 0.005$, $\alpha_d^N = 0.05$, погрешность $\|u^N(\cdot) - u_*(\cdot)\|_{L_2(T)} = 0.169$

Результаты тестирования представлены на рис. 5, где приведены графики зависимостей ускорения и эффективности параллельного алгоритма от количества ядер. Напомним, что ускорение S_p и эффективность E_p вычисляются, соответственно, по формулам: $S_p = T_1/T_p$, $E_p = S_p/p$, T_p — время выполнения программы (для одного набора параметров) на

кластере для p ядер, T_1 — время работы последовательного алгоритма. Для «идеального» параллельного алгоритма ускорение равно числу ядер; в этом случае имеем единичную эффективность.

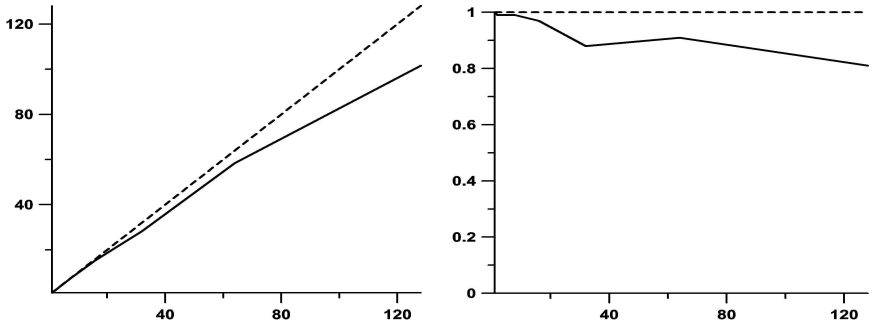


Рис. 5. Ускорение S_p (слева) и эффективность E_p (справа) в зависимости от количества ядер: реальные характеристики изображены сплошной линией, идеальные величины — пунктирной; горизонтальная ось соответствует количеству ядер p, вертикальная — значениям S_p и E_p

Из рис. 5 следует, что эффективность распараллеливания достаточно высока, причем с ростом числа задействованных процессоров она не падает ниже разумного уровня (например, при $p = 128$ имеем $S_p = 103.5$ и $E_p = 0.81$). Это важно, поскольку для настройки алгоритма на конкретную систему и его тестирования (в вероятностном смысле) требуется достаточно большое количество запусков программы.

Заключение

В работе рассмотрена обратная задача для квазилинейной стохастической системы с диффузией, зависящей от фазового состояния. Постановка предполагает динамическую реконструкцию двух неизвестных неслучайных возмущений, входящих в детерминированную и стохастическую части системы, на основе точных дискретных измерений некоторого количества реализаций искомого случайного процесса. Задача для СДУ заменяется обратной задачей для системы двух ОДУ, которым удовлетворяют математическое ожидание и ковариационная матрица исходного процесса. Для конструктивного алгоритма восстановления получена оценка точности относительно количества доступных измерению реализаций. Ключевым результатом статьи следует считать разработку процедуры адаптации алгоритма к конкретной динамической системе и ограничениям на ее структурные элементы; это обусловило необходимость прогонки вычислительного модуля на сетке параметров и моделирования большого числа траекторий СДУ для получения статистических оценок моментов. С целью экономии вычислительных ресурсов введено естественное и достаточно эффективное распараллеливание основных времяемких процедур.

В качестве перспективных направлений исследований отметим создание теоретического обоснования процедуры автоматизации настройки алгоритма через поиск экстремума некоторого критерия качества реконструкции на множестве допустимых параметров и апробацию методики на других алгоритмах восстановления входов динамических систем.

Литература

1. Кряжимский А.В., Осипов Ю.С. О моделировании управления в динамической системе // Изв. АН СССР. Техн. кибернетика. 1983. № 2. С. 51–60.
2. Осипов Ю.С., Кряжимский А.В., Максимов В.И. Методы динамического восстановления входов управляемых систем. Екатеринбург: Изд-во УрО РАН, 2011. 291 с.
3. Максимов В.И. Задачи динамического восстановления входов бесконечномерных систем. Екатеринбург: Изд-во УрО РАН, 2000. 305 с.
4. Красовский Н.Н., Субботин А.И. Позиционные дифференциальные игры. М.: Наука, 1984. 456 с.
5. Тихонов А.Н., Арсенин В.Я. Методы решения некорректных задач. М.: Наука, 1978. 142 с.
6. Близорукова М.С., Максимов В.И. Об одном алгоритме реконструкции траектории и управления в системе с запаздыванием // Труды Института математики и механики УрО РАН. 2012. Т. 18, № 1. С. 109–122. DOI: 10.1134/S0081543813020065.
7. Кадиев А.М., Максимов В.И. О реконструкции управлений в параболическом уравнении // Дифференц. уравнения. 2007. Т. 43, № 11. С. 1545–1552.
8. Кряжимский А.В., Осипов Ю.С. Об устойчивом позиционном восстановлении управления по измерениям части координат // Некоторые задачи управления и устойчивости. 1989. С. 33–47.
9. Розенберг В.Л. Задача динамического восстановления неизвестной функции в линейном стохастическом дифференциальном уравнении // Автом. и телемех. 2007. № 11. С. 76–87. DOI: 10.1134/S0005117907110069.
10. Розенберг В.Л. Восстановление амплитуды случайной помехи в линейном стохастическом уравнении по измерениям части координат // Журнал вычисл. математики и мат. физики. 2016. Т. 56, № 3. С. 377–386. DOI: 10.7868/S0044466916030169.
11. Rozenberg V.L. Dynamical Input Reconstruction Problem for a Quasi-linear Stochastic System // IFAC-PapersOnLine. 2018. Vol. 51, no. 32. P. 727–732. DOI: 10.1016/j.ifacol.2018.11.460.
12. Оксендаль Б. Стохастические дифференциальные уравнения. Введение в теорию и приложения. М.: Мир, 2003. 408 с.
13. Королюк В.С., Портенко Н.И., Скороход А.В., Турбин А.Ф. Справочник по теории вероятностей и математической статистике. М.: Наука, 1985. 640 с.
14. Черноусько Ф.Л., Колмановский В.Б. Оптимальное управление при случайных возмущениях. М.: Наука, 1978. 272 с.
15. Мильштейн Г.Н. Численное интегрирование стохастических дифференциальных уравнений. Свердловск: Изд-во УрГУ, 1988. 224 с.
16. Гергель В.П. Теория и практика параллельных вычислений. М.: БИНОМ, 2007. 424 с.

Мельникова Лидия Антоновна, математик 1-й кат., Институт математики и механики им. Н.Н. Красовского УрО РАН (Екатеринбург, Российская Федерация)

Розенберг Валерий Львович, к.ф.-м.н., с.н.с., Институт математики и механики им. Н.Н. Красовского УрО РАН (Екатеринбург, Российская Федерация)

DOI: 10.14529/cmse190402

ALGORITHM OF DYNAMICAL INPUT RECONSTRUCTION FOR A STOCHASTIC DIFFERENTIAL EQUATION: TUNING OF PARAMETERS AND NUMERICAL EXPERIMENTS

© 2019 L.A. Melnikova, V.L. Rozenberg

Krasovskii Institute of Mathematics and Mechanics,
Ural Branch of the Russian Academy of Sciences
(S. Kovalevskoi str. 16, Yekaterinburg, 620990 Russia)
E-mail: meln@imm.uran.ru, rozen@imm.uran.ru

Received: 16.04.2019

The problem of reconstructing unknown inputs in a stochastic differential equation is investigated by means of the approach of the theory of dynamic inversion. The statement when the simultaneous reconstruction of disturbances in the deterministic and stochastic terms of the equation is performed from the discrete information on a number of realizations of the stochastic process is considered. The problem is reduced to an inverse problem for ordinary differential equations describing the mathematical expectation and covariance matrix of the process. A software-oriented solving algorithm based on constructions of the theory of positional control with a model is designed; an estimate for its convergence rate with respect to the number of measurable realizations is obtained. A program procedure for the automatic tuning of the algorithm's parameters in order to have the best approximation results for different disturbances satisfying a priori constraints in a specific dynamical system is proposed. Desired dependencies of the algorithm's parameters on the number of measured realizations are determined empirically via solving a specific extremal problem, where the deviation of the algorithm's output from some test function is minimized. To optimize the time-taking adaptation process assuming the simulation of a large number of independent trajectories of the stochastic process, the parallelization of calculations is applied. A model example illustrating the method proposed is given. A system approximately describing the population dynamics of two interacting biological species is considered. Calculation results and parallelization efficiency characteristics are presented.

Keywords: stochastic differential equation, dynamical reconstruction, controlled model, parameter tuning, parallelization of calculations.

FOR CITATION

Melnikova L.A., Rozenberg V.L. Algorithm of Dynamical Input Reconstruction for a Stochastic Differential Equation: Tuning of Parameters and Numerical Experiments. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 4. pp. 15–29. (in Russian) DOI: 10.14529/cmse190402.

References

1. Kryazhimskii A.V., Osipov Yu.S. Modelling of a Control in a Dynamic System. Engrg. Cybernetics. 1984. vol. 21, no. 2, pp. 38–47.

2. Osipov Yu.S., Kryazhimskii A.V., Maksimov V.I. Dynamic Recovery Methods for Inputs of Control Systems. Yekaterinburg, Publishing of the UB of RAS, 2011. 291 p. (in Russian)

3. Maksimov V.I. Dynamical Inverse Problems of Distributed Systems. VSP, Utrecht–Boston, 2002. 269 p.

4. Krasovskii N.N., Subbotin A.I. Game-Theoretical Control Problems. Springer, New York, 1988. 517 p.

5. Tikhonov A.N., Arsenin V.Ya. Solutions of Ill-Posed Problems. Wiley, New York, 1981. 258 p.

6. Blizorukova M.S., Maksimov V.I. On a Reconstruction Algorithm for the Trajectory and Control in a Delay System. Proc. Steklov Inst. Math. 2013. vol. 280, no. 1, pp. S66–S79. DOI: 10.1134/S0081543813020065.

7. Kadiyev A. M., Maksimov V.I. On the Reconstruction of Controls in a Parabolic Equation. Differential Equations. 2007. vol. 43, no. 11, pp. 1585–1593. DOI: 10.1134/S0012266107110134.

8. Kryazhimskii A.V., Osipov Yu.S. On a Stable Positional Recovery of Control from Measurements of a Part of Coordinates. Some Problems of Control and Stability. Sverdlovsk, Publishing of the UB of Academy of Sciences of the USSR, 1989. pp. 33–47. (in Russian)

9. Rozenberg V.L. Dynamic Restoration of the Unknown Function in the Linear Stochastic Differential Equation. Autom. Remote Control. 2007. vol. 68, no. 11, pp. 1959–1969. DOI: 10.1134/S0005117907110069.

10. Rozenberg V.L. Reconstruction of Random-Disturbance Amplitude in Linear Stochastic Equations from Measurements of Some of the Coordinates. Comput. Math. Math. Phys. 2016. vol. 56, no. 3, pp. 367–375. DOI: 10.1134/S0965542516030143.

11. Rozenberg V.L. Dynamical Input Reconstruction Problem for a Quasi-linear Stochastic System. IFAC-PapersOnLine. 2018. vol. 51, no. 32. pp. 727–732. DOI: 10.1016/j.ifacol.2018.11.460.

12. Øksendal B. Stochastic Differential Equations: an Introduction with Applications. Springer, Berlin, 1985. 408 p.

13. Korolyuk V.S., Portenko N.I., Skorokhod A.V., Turbin A.F. Handbook on Probability Theory and Mathematical Statistics. Moscow, Nauka, 1985. 640 p. (in Russian)

14. Chernous'ko F.L., Kolmanovskii V.B. Optimal Control under Random Perturbation. Moscow, Nauka, 1978. 272 p. (in Russian)

15. Milshtein G.N. Numerical Integration of Stochastic Differential Equations. Sverdlovsk, Publishing of the Ural State University, 1988. 224 p. (in Russian)

16. Gergel' V.P. Theory and Practice of Parallel Computing. Moscow, Binom, 2007. 424 p. (in Russian)

УДК 512.5, 519.1(075.8) DOI: 10.14529/cmse190403

ПОСТРОЕНИЕ САМОНЕПЕРЕСЕКАЮЩИХСЯ *OE*-МАРШРУТОВ В ПЛОСКОМ ЭЙЛЕРОВОМ ГРАФЕ

© 2019 Т.А. Макаровских

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: Makarovskikh.T.A@susu.ru

Поступила в редакцию: 24.09.2019

В статье предложен полиномиальный алгоритм построения самонепересекающегося маршрута с упорядоченным охватыванием в плоском эйлеровом графе. Предложенный подход состоит в расщеплении всех вершин исходного графа степени выше 4 и введении фиктивных вершин и ребер, сводя, таким образом, исходную задачу к решенной ранее автором задаче построения A-цепи с упорядоченным охватыванием в плоском связном 4-регулярном графе. Приведенный алгоритм сведения решает поставленную задачу за полиномиальное время. Рассмотрен тестовый пример построения самонепересекающейся цепи с упорядоченным охватыванием. Данная задача возникает при технологической подготовке процесса раскроя, когда требуется определить маршрут движения режущего инструмента, при котором отсутствуют самопересечения траектории резки и отрезанная от листа часть не требует разрезаний. Раскройный план представлен в виде плоского графа, являющегося его гомеоморфным образом. Предложенный в статье алгоритм решает проблему маршрутизации при вырезании деталей, когда на маршрут движения режущего инструмента одновременно наложены такие технологические ограничения.

Ключевые слова: плоский граф, маршрут, раскройный план, полиномиальный алгоритм, процесс раскроя.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Макаровских Т.А. Построение самонепересекающихся *OE*-маршрутов в плоском эйлеровом графе // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 4. С. 30–42. DOI: 10.14529/cmse190403.

Введение

В работе [1] поставлена задача CPDP (Cutting Path Determination Problem), заключающаяся в определении оптимального маршрута вырезания деталей по заданному раскройному плану одним или несколькими режущими инструментами. При этом предполагается наличие двух очевидных ограничений: 1) все детали должны быть вырезаны; 2) ни один из вырезанных фрагментов не должен требовать дальнейших разрезаний, т.е. выполнено OE (Odered Eclosing) ограничение [2]. Для решения проблемы CPDP известны более детальные постановки: GTSP (General Travelling Salesman Problem) [3–11], CCP (Continuous Cutting Problem), ECP (Endpoint Cutting Problem) [12] и ICP (Intermittent Cutting Problem), [2]. Отметим, что ECP и ICP допускают совмещение границ вырезаемых деталей, что позволяет сократить расход материала, длину резки и длину холостых проходов [4]. Проблемы уменьшения отходов материала и максимального совмещения фрагментов контуров вырезаемых деталей решаются на этапе составления раскройного плана.

Несмотря на отмеченные преимущества компьютерных технологий ECP и ICP, в настоящее время большинство публикаций посвящено развитию технологий GTSP и CCP, которые используют очевидные алгоритмы маршрутизации режущего инструмента, состоящие в поконтурном вырезании деталей.

Развитию компьютерных технологий ECP и ICP посвящены работы [2, 12] и [13]. В них даны полиномиальные алгоритмы OE-маршрутизации, когда отрезанная от листа часть не требует дальнейших разрезаний.

Большую как теоретическую, так и практическую ценность представляет построение самонепересекающихся OE-маршрутов (NOE-маршрутов) в плоских эйлеровых графах. Под самонепересекающимся маршрутом имеется в виду циклический граф, представляющий плоскую жорданову кривую без самопересечений [16], полученный в результате расщепления вершин исходного графа. Под расщеплением понимается определенная в [18] операция, которая заключается в следующем. Пусть G — связный граф, $v \in V(G)$ — вершина степени $\deg(v) \geq 3$. Если $x = \{v, v_1\}$ и $y = \{v, v_2\}$ — пара ребер, инцидентных v, тогда под отщеплением пары ребер $\{x, y\}$ от вершины v понимается получение нового графа $G_{x,y}$, полученного из G удалением ребер x и y, введением новой вершины v_{xy} и пары ребер $\{v_{xy}v_1\}$ и $\{v_{xy}v_2\}$ (см. [14, рис. 1]).

В частном случае, когда раскройный план оказывается гомеоморфен плоскому связному 4-регулярному графу, в работе [15] предложен полиномиальный алгоритм построения AOE-цепи, являющейся самонепересекающейся цепью. Данный алгоритм позволяет решать задачи маршрутизации в плоских связных графах со степенями вершин, не превосходящими 4.

Отметим приведенное в [17] «доказательство» $\mathcal{NP}$-полноты задачи построения самонепересекающейся цепи в плоском эйлеровом графе. Рассуждения базируются на определении непересекающейся цепи, являющемся определением A-цепи [18] (т. е.цепи, в которой очередным в маршруте ребром является следующее ребро в определенном для текущей вершины циклическом порядке). Очевидно, что A-цепь представляет лишь частный случай самонепересекающейся цепи.

В данной статье предложен полиномиальный алгоритм для построения самонепересекающейся цепи в плоском связном эйлеровом графе. Предложенный в статье алгоритм решает задачу маршрутизации при вырезании деталей, когда выполняются два ограничения: отрезанная от листа часть не требует дополнительных разрезаний [2, 19, 20] и в траектории резки отсутствуют пересечения [21].

Статья организована следующим образом. В разделе 1 приведены необходимые определения, описаны используемые обозначения для представления данных. В разделе 2 рассматривается класс самонепересекающихся OE-цепей (называемых далее NOE-цепями). Цепи указанного класса соответствуют траектории движения режущего инструмента, избегающей пересечения траектории резания. Показано, что задача построения NOE-цепи в плоском связном эйлеровом графе может быть за полиномиальное время сведена к задаче построения AOE-цепи в плоском связном 4-регулярном графе. Приведен алгоритм такого сведения. В разделе 3 рассмотрен тестовый пример построения NOE-цепи. В заключении перечислены полученные в работе результаты, отмечены направления дальнейших исследований.

1. Определения и обозначения, используемые для представления данных

В данной работе будем использовать представление графа, используемое автором в предыдущих работах [2, 15, 19–21]. В работах автора предложено вместо раскройного плана использовать его гомеоморфный образ, представляющий плоский граф G с внешней

гранью f_0 на плоскости S. Для любой части J графа G (т.е. $J \subseteq G$) обозначим через $\mathsf{Int}\,(J)$ теоретико-множественное объединение его внутренних граней (объединение всех связных компонент $S \setminus J$, не содержащих внешней грани). Тогда если J — начальная часть маршрута, то $\mathrm{Int}(J)$ можно интерпретировать как отрезанную от листа часть. Топологическое представление плоского графа G на плоскости S с точностью до гомеоморфизма определяется заданием для каждого ребра $e \in E(G)$ следующих функций [2, 20]:

- $v_k(e)$, $k = 1, 2$ — вершины, инцидентные ребру e;
- $l_k(e)$, $k = 1, 2$ — ребра, полученные вращением ребра e против часовой стрелки вокруг вершины $v_k(e)$;
- $r_k(e)$, $k = 1, 2$ — ребра, полученные вращением ребра e по часовой стрелке вокруг вершины $v_k(e)$;
- $f_k(e)$ — грань, находящаяся справа при движении по ребру e от вершины $v_k(e)$ к вершине $v_{3-k}(e)$, $k = 1, 2$.

Пример представления графа подробно рассмотрен в [2].

Представление графа фактически задает ориентацию его ребер. Далее предполагается, что движение по ребру для определенности осуществляется от вершины $v_1(e)$ к вершине $v_2(e)$. Поскольку при задании графа G неизвестно, какое из ребер в каком направлении будет пройдено, то при выполнении алгоритма производится перестановка значений полей $v_k(e)$, $r_k(e)$ и $l_k(e)$, $f_k(e)$, $k = 1, 2$ некоторых ребер. В алгоритме данную процедуру выполняет функция `REPLACE`, функциональным назначением которой является замена индексов функций $v_k(e)$, $l_k(e)$, $r_k(e)$ и $f_k(e)$ на $3 - k$, $k = 1, 2$ [20].

Далее будем считать, что все рассматриваемые плоские графы представлены указанными функциями. Пространственная сложность такого представления будет $O(|E(G)| \cdot \log_2 |V(G)|)$ [15]. В дальнейшем будем использовать ряд понятий, определения которых имеются в работах [13, 18, 22]. Приведем основные из них для удобства читателя.

Определение 1. Будем говорить, что цикл $C = v_1e_1v_2e_2 \dots v_k$ в эйлеровом графе G имеет **упорядоченное охватывание** (называется OE-цепью), если для любой его начальной части $C_i = v_1e_1v_2e_2 \dots e_i$, $i \leq |E(G)|$ выполнено условие $\mathrm{Int}\,(C_i) \cap G = \emptyset$ [13].

Определение 2. Эйлерову цепь T будем называть **A-цепью** [18], если она является A_G-совместимой цепью. Таким образом, последовательные ребра в цепи T (инцидентные вершине v) являются соседями в циклическом порядке $O^{\pm}(v)$ [18].

Определение 3. Рангом ребра $e \in E$ графа $G = (V, E)$ будем называть значение функции $\mathrm{rank}(e) : E \to \mathbb{N}$, определяемое рекурсивно:

- пусть $E_1 = \{e \in E : e \subset f_0\}$ — множество ребер, ограничивающих внешнюю грань f_0 графа $G(V, E)$, тогда $(\forall e \in E_1)\,(\mathrm{rank}(e) = 1)$;
- пусть E_k — множество ребер ранга 1 графа

$$G_k\left(V, E \setminus \left(\bigcup_{l=1}^{k-1} E_l\right)\right),$$

тогда $(\forall e \in E_k)\,(\mathrm{rank}(e) = k)$ [13].

Алгоритм 1 `NOE-CHAIN (G)`

Require: плоский эйлеров граф G, заданный функциями $v_k(e)$, $l_k(e)$, $r_k(e)$, $f_k(e)$, $k = 1, 2$ и $\mathrm{rank}(e)$;
Ensure: NOE-цепь в графе G;
1: $\widehat{G} := \mathsf{NonIntersecting}(G)$; ▷ Расщепить все вершины степени выше 4
2: $\widetilde{G} := \mathsf{CutPointSplitting}(\widehat{G})$; ▷ Расщепить все точки сочленения всех рангов
3: $C^*{:=}\mathsf{AOE\_TRAIL}(\widetilde{G})$; ▷ Построить AOE-цепь в графе $\tilde{G}$
4: $C{:=}\mathsf{Absorb}(C^*)$; ▷ Стянуть все расщепленные вершины

Ранг ребра определяет его удаленность от внешней грани и показывает, какое минимальное число граней необходимо пересечь, чтобы добраться от внешней грани f_0 до этого ребра.

Определение 4. Рангом грани $f \in F(G)$ будем называть значение функции $\mathrm{rank} : F(G) \to \mathrm{Z}^{\geq 0}$:

$$\mathrm{rank}(f) = \begin{cases} 0, & \text{при } f = f_0, \\ \min_{e \in E(f)} \mathrm{rank}(e), & \text{в противном случае,} \end{cases}$$

где $E(f)$ — множество ребер инцидентных грани $f \in F$ [2].

2. Алгоритм построения NOE-цепи

Рассмотренный в [15] класс AOE-цепей достаточно узок. К тому же в общем случае не известны эффективные алгоритмы построения таких цепей. Для практических задач оказывается достаточным построение не AOE-цепи, а самонепересекающейся OE-цепи.

Определение 5. Эйлеров цикл в C плоском эйлеровом графе G называется самонепересекающимся если он гомеоморфен плоской замкнутой жордановой кривой [16] без самопересечений, который может быть получен из графа $G = (V, E)$ с помощью применения $|E|$ операций расщепления вершин.

Для построения самонепересекающейся эйлеровой OE-цепи (или цикла) в плоском эйлеровом графе (в дальнейшем эту цепь будем называть NOE-цепью (non-intersecting OE-trail)) можно воспользоваться алгоритмом 1.

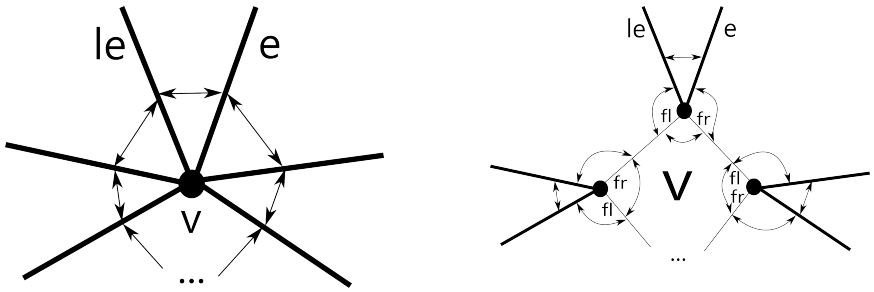
Процедура `Non-intersecting (G)` (алгоритм 2) строит 4-регулярный граф $\widehat{G}$ (с точностью до гомеоморфизма) расщепляя в графе G все вершины $v \in V(G)$ степени $2l$ ($l \geq 3$) на l фиктивных вершин степени 4 и вводит l фиктивных ребер, инцидентных полученным после расщепления вершинам и образующим цикл (см. рис. 1а и 1б). Для выполнения указанных преобразований необходимо просмотреть функции $v_k(e)$, $k = 1, 2$ для всех ребер $e \in E(G)$, и внести требуемые модификации в систему кодирования графа. С этой целью на множестве вершин графа $V(G)$ определена булева функция

$$\mathrm{Checked}(v) = \begin{cases} \texttt{true}, & \text{если вершина просмотрена;} \\ \texttt{false}, & \text{в противном случае.} \end{cases}$$

При выполнении инициализации (строки 1–3 в описании алгоритма 2) все вершины объявляют непросмотренными, т. е. $\mathrm{Checked}(v) =$ `false` для всех $v \in V(G)$. Просмотр вершины $v = v_1(e)$, такой что $\mathrm{Checked}(v) =$ `false` состоит в выполнении процедуры `Handle (e)` (алгоритм 3). которая производит обработку данной вершины, заключающуюся в ее расщеплении в соответствии с рис. 1а и 1б.

Алгоритм 2 Процедура `Non-intersecting (`G`)`

```
 1: procedure NON-INTERSECTING(G)
Require: плоский эйлеров граф G, заданный функциями v_k(e), l_k(e), r_k(e), f_k(e), k = 1,2 и rank(e);
Ensure: плоский связный 4-регулярный граф G*, определяемый аналогичным образом;
 2:     for all v ∈ V(G) do                          ▷ Инициализация функции Checked(v)
 3:         Checked(v) := false;
 4:     end for
 5:     for all (e ∈ E(G) ) do                       ▷ Поиск вершин степени больше 4 и их расщепление
 6:         k := 1;                                  ▷ Просмотреть вершину с индексом 1, затем – 2
 7:         while (k ≤ 2) do
 8:             if (not Checked(v_k(e))) then        ▷ Обработать только не обработанную ранее вершину
 9:                 if (k = 2) then                  ▷ Скорректировать индексы
10:                     REPLACE(e);                  ▷  обрабатываются вершины v_1(e)
11:                 end if
12:                  Handle ( e);                    ▷ Вызвать функцию для обработки вершины v_1(e)
13:                 Checked(v_1(e)) := true;         ▷ Пометить вершину как просмотренную
14:             end if
15:             k := k + 1;
16:         end while
17:     end for
18: end procedure
```



а) Исходные указатели на соседние ребра в расщепляемой вершине

б) Расщепление вершины (жирными линиями показаны ребра графа G, тонкими линиями – дополнительные (фиктивные) ребра) и модификация указателей в соответствии с расщеплением

Рис. 1. Расщепление вершины степени выше 4 и модификация указателей на ребра

Алгоритм 3 в результате цикла `repeat -until` (строки 6–11) подсчитывает степень d текущей вершины v. Если $d > 4$, выполняется второй цикл `repeat -until` (строки 12–23), в котором обрабатываемая вершина расщепляется на $d/2$ фиктивных вершин, вводятся d фиктивных ребер, инцидентных этим вершинам и образующим цикл.

Отметим, что строки 18–23 затрагивают не только изменение указателей на ребра, но и вводят новую (фиктивную) грань F, инцидентную всем фиктивным вершинам и ребрам, а также определяют ранги фиктивных ребер.

Алгоритм 3 Процедура Handle (e)

```
1: procedure HANDLE(e)
2:     v := v_1(e);                                          ▷ Расщепляемая вершина
3:     e_first := e;                                         ▷ Сохранить первое рассматриваемое ребро
4:     d := 0;                                               ▷ Инициализация счетчика для степени вершины d
5:     F := FaceNum() + 1;                                   ▷ Определить номер для новой грани
6:     repeat                                                ▷ Проход 1: Определение степени вершины v
7:         le := l_1(e);
8:         if (v_1(le) ≠ v) then REPLACE(le);
9:         end if                                            ▷ При необходимости поменять индексацию функций
10:        e := le; d := d + 1;                              ▷ Учесть ребро при подсчете степени и перейти к следующему
11:    until (e = e_first);                                  ▷ Повторять, пока не будут просмотрены все ребра, инцидентные v
12:    if (d > 4) then                                       ▷ Если степень текущей вершины больше 4
13:        e := e_first;                                     ▷ Начать с первого рассматриваемого ребра
14:        le := l_k (e);                                    ▷ Определить номер его левого соседа
15:        e_next := l_k (le);                               ▷ Сохранить ребро, для следующей итерации
16:        fl := new EDGE; fle := fl; e_first := e;          ▷ Ввести фиктивное ребро, смежное le
17:        repeat                                            ▷ Расставить указатели для ребер
18:            e := e_next; le := l_k (e); fr := fl;
19:            f_1(fl) := F; f_2(fl) := f_2(e);              ▷ Определить грани, смежные фиктивному ребру
20:            rank(fl) := facerank(f_2(fl));                ▷ Определить «ранг» фиктивного ребра
21:                    ▷ Функция facerank() вычисляет ранг грани в соответствии с определением
22:            fl := new EDGE; e_next := l_k (le);
23:        until (l_k(le) = e_first);
24:    end if
25: end procedure
```

Определение 6. Ранг фиктивного ребра (строка 20) равен рангу инцидентной фиктивному ребру грани исходного графа.

Для 4-регулярного графа $\widehat{G}$ с определенными для него рангами фиктивных ребер и введенными в его представление фиктивными гранями можно применить последовательно алгоритм `CUT-POINT-SPLITTING`($\widehat{G}$) построения графа $\widetilde{G}$ с расщепленными точками сочленения (понятие точки сочленения ранга k см. в [15, Определение 12]), и алгоритм `AOE-TRAIL`($\widetilde{G}$) [15] построения AOE-цепи C^* в графе $\widetilde{G}$. При построении цепи C^* алгоритм `AOE-TRAIL`($\widetilde{G}$) при наличии двух смежных непройденных ребер одного ранга для гарантированного выполнения условия упорядоченного охватывания в первую очередь выбирает фиктивное ребро.

Процедура `Absorb`(*) заменяет в * все фиктивные ребра и инцидентные им вершины, полученные при расщеплении вершины v (выполняет операцию стягивания фиктивных вершин). В результате выполнения процедуры получим NOE-цепь C в исходном графе G. Цепь C, полученная после удаления фиктивных ребер за счет стягивания вершин, будет принадлежать классу OE, т.к. процедура удаления ребер не нарушает порядка следования оставшихся ребер в цепи, что исключает появление цикла, охватывающего еще непройденные ребра.

Так как процедура `Handle` состоит из двух последовательных просмотров ребер, инцидентных текущей вершине v, то вычислительная сложность процедуры равна $O(|E(G)|)$. Процедура `Non-Intersecting` заключается в однократном просмотре всех

ребер, то есть ее вычислительная сложность также составляет величину $O(|E(G)|)$. Следовательно, алгоритм сведения плоского связного эйлерова графа к плоскому связному 4-регулярному графу решает поставленную задачу за время $O(|E(G)|^2)$. Поскольку алгоритм `AOE-TRAIL` и предшествующая его вызову функция `CUT-POINT-SPLITTING` [15] решают задачу построения AOE-цепи C за время $O(|E(G^*)| \cdot \log_2 |V(G^*)|)$, то и задача построения NOE-цепи в графе G решается за полиномиальное время $O(|E(G)|^2)$.

Из сказанного выше в данном разделе следует справедливость следующей теоремы.

Теорема 1. Алгоритм `NOE-CHAIN` решает задачу построения NOE-цепи в плоском эйлеровом графе за время $O(|E(G)|^2)$.

3. Иллюстрация работы алгоритма

Рассмотрим работу алгоритма на примере графа, приведенного на рис. 2а. Пример затрагивает общий случай: в графе имеется вершина степени выше 6, не смежная внешней грани, а также присутствуют (и возникают в процессе расщепления) точки сочленения разных рангов. Рассматриваемый граф является эйлеровым, следовательно, построение NOE-цепи можно начать из любой вершины, смежной внешней грани. Пусть это будет вершина v_2.

После применения процедуры `Handle()` получим граф, представленный на рис. 2б. В полученном графе все вершины имеют степень 4. В графе на рис. 2б помимо расщепленной вершины v_3 имеются и точки сочленения v_4 и v_6 рангов 3 и 2 соответственно, а также точка сочленения ранга 2 в расщепленной вершине v_3, инцидентная ребрам e_{13}, e_4 и двум фиктивным ребрам того же ранга. Эти вершины необходимо расщепить с помощью алгоритма `CUT-POINT-SPLITTING`. В результате выполнения указанного алгоритма получим граф, представленный на рис. 2в. С помощью алгоритма `AOE-Trail` в полученном графе определим AOE-маршрут, в котором символом «*» обозначен переход по фиктивным ребрам

$$\begin{aligned} T^* \;=\; & v_2e_5v_1e_4v_3 * * * e_{19}v_4e_{17}v_5e_{18}v_4e_{20}v_3e_{16}v_5e_{15}v_3 * \\ & e_3v_8e_{12}v_6e_9v_7e_{10}v_6e_{11}v_9e_{14}v_3e_6v_9e_7v_7e_8v_8e_2v_3 * e_{13}v_1e_1v_2, \end{aligned}$$

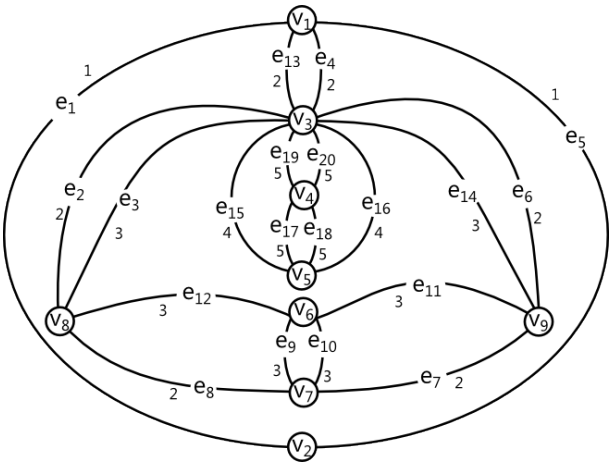
которому после стягивания расщепленных вершин соответствует NOE-маршрут

$$\begin{aligned} T^* \;=\; & v_2e_5v_1e_4v_3e_{19}v_4e_{17}v_5e_{18}v_4e_{20}v_3e_{16}v_5e_{15}v_3 \\ & e_3v_8e_{12}v_6e_9v_7e_{10}v_6e_{11}v_9e_{14}v_3e_6v_9e_7v_7e_8v_8e_2v_3e_{13}v_1e_1v_2, \end{aligned}$$

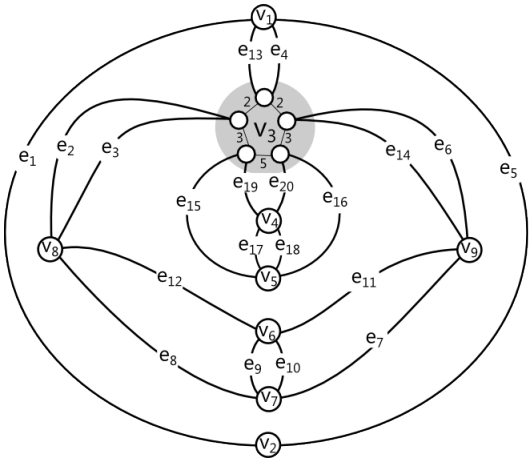
в исходном графе.

Заключение

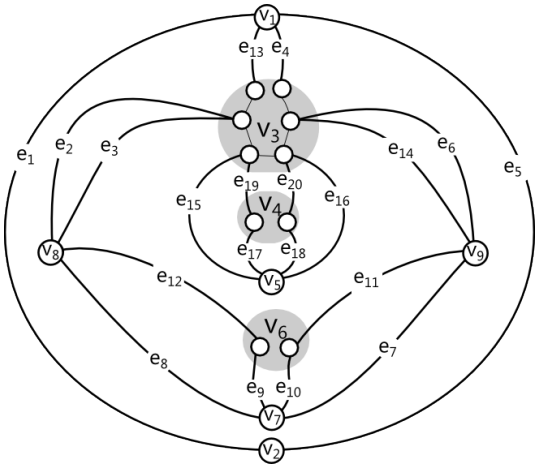
Предложенный в работе алгоритм строит NOE-цепь в плоском эйлеровом графе. В случае плоского неэйлерова (в общем случае несвязного) графа G без висячих вершин необходимо расщепить все вершины степени выше 4 в соответствии с алгоритмом 3. В результате получим граф, степени вершин которого равны 3 или 4 (не уменьшая общности рассуждений, вершины степени 2 не рассматриваются). Для этого графа применима та же последовательность действий, что описана в [15] для построения AOE-покрытия. В цепях полученного покрытия удалим все искусственные ребра, стягивая все расщепленные вершины. В результате получим NOE-покрытие.



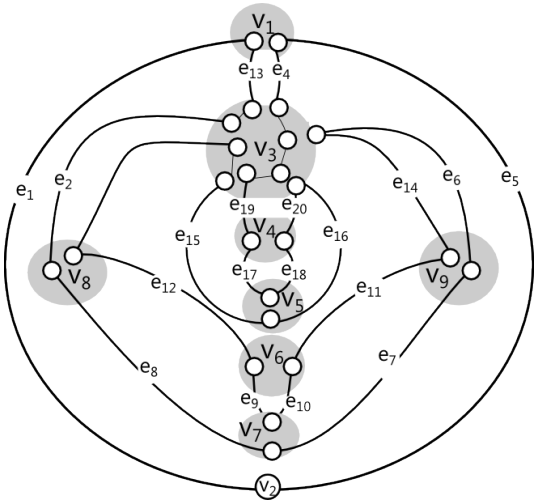
а) Граф G



б) Менее, чем 4-регулярный граф $\widehat{G}$



в) граф $\widetilde{G}$ без точек сочленения



г) результирующий самонепересекающийся OE-маршрут C

Рис. 2. Пример построения NOE-цепи в плоском эйлеровом графе, имеющем вершину степени выше 6, не смежную внешней грани

Предложенный в статье алгоритм решает проблему маршрутизации при вырезании деталей, когда на маршрут движения режущего инструмента одновременно наложены такие технологические ограничения, как (1) отрезанная от листа часть не требует дополнительных разрезаний, (2) отсутствуют самопересечения траектории резки.

В качестве направлений дальнейших исследований можно выделить создание библиотеки классов для решения задачи маршрутизации в плоских графах.

Литература

1. Silva E.F., Oliveira L.T., Oliveira J.F., Toledo F.M.B. Exact approaches for the cutting path determination problem // Computers & Operations Research. 2019. Vol. 112, art. 104772. DOI: 10.1016/j.cor.2019.104772.

2. Makarovskikh T.A., Panyukov A.V., Savitsky E.A. Mathematical Models and Routing Algorithms for CAD Technological Preparation of Cutting Processes // Automation and Remote Control. 2017. Vol. 78, no. 4. P. 868–882. DOI: 10.1016/10.1134/S0005117917050095.

3. Dewil R., Vansteenwegen P., Cattrysse D. A Review of Cutting Path Algorithms for Laser Cutters // International Journal Adv Manuf. Technol. 2016. Vol. 87. P. 1865–1884. DOI: 10.1007/s00170-016-8609-1.

4. Dewil R., Vansteenwegen P., Cattrysse D., Laguna M., Vossen T. An Improvement Heuristic Framework for the Laser Cutting Tool Path Problem // International Journal of Production Research. 2015. Vol. 53, no. 6. P. 1761–1776. DOI: 10.1080/00207543.2014.959268.

5. Hoeft J., Palekar U. Heuristics for the Plate-cutting Travelling Salesman Problem // IIE Transactions. 1997. Vol. 29(9). P. 719–731. DOI: 10.1023/A:1018582320737.

6. Dewil R., Vansteenwegen P., Cattrysse D. Construction Heuristics for Generating Tool Paths for Laser Cutters // International Journal of Production Research. 2014. Vol. 52(20). P. 5965–5984. DOI: 10.1080/00207543.2014.895064.

7. Петунин А.А., Ченцов А.Г., Ченцов П.А. К вопросу о маршрутизации перемещений при листовой резке деталей // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2017. Т. 10, № 3. С. 25–39. DOI: 10.14529/mmp170303.

8. Chentsov A.G., Grigoryev A.M., Chentsov A.A. Solving a Routing Problem with the Aid of an Independent Computations Scheme // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2018. Т. 11, № 1. С. 60–74. DOI: 10.14529/mmp180106.

9. Petunin A., Stylios C. Optimization Models of Tool Path Problem for CNC Sheet Metal Cutting Machines // 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016 (Troyes, France, June, 28–30, 2016). IFAC-PapersOnLine. 2016. Vol. 49. P. 23–28. DOI: 10.1016/j.ifacol.2016.07.544.

10. Chentsov A., Khachay M., Khachay D. Linear Time Algorithm for Precedence Constrained Asymmetric Generalized Traveling Salesman Problem // 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016 (Troyes, France, June, 28–30, 2016). IFAC-PapersOnLine. 2016. Vol. 49. P. 651–655. DOI: 10.1016/j.ifacol.2016.07.767.

11. Khachay M., Neznakhina K. Towards Tractability of the Euclidean Generalized Travelling Salesman Problem in Grid Clusters Defined by a Grid of Bounded Height // Communications in Computer and Information Science. 2018. Vol. 871. P. 68–77. DOI: 10.1007/978-3-319-93800-4_6.

12. Manber U., Israni S. Pierce Point Minimization and Optimal Torch Path Determination in Flame Cutting // J. Manuf. Syst. Vol. 3(1). 1984. P. 81–89. DOI: 10.1016/0278-6125(84)90024-4.

13. Panyukova T. Chain Sequences with Ordered Enclosing // Journal of Computer and System Sciences International. 2007. Vol. 46, no. 1(10). P. 83–92. DOI: 10.1134/S1064230707010108.

14. Borse Y.M. Splitting Lemma for 2-Connected Graphs // International Scholarly Research Network, ISRN Discrete Mathematics. 2012. Vol. 2012, art. 850538. DOI: 10.5402/2012/850538.

15. Макаровских Т.А. Программное обеспечение для построения А-цепей с упорядоченным охватыванием в плоском связном 4-регулярном графе // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 1. С. 36–53. DOI: 10.14529/cmse190103.

16. Филиппов А.Ф. Элементарное доказательство теоремы Жордана // Успехи математических наук. 1950. Т. 5:5(39). С. 173–176.

17. Manber U., Bent S.W. On Non-intersecting Eulerian Circuits // Discrete Applied Mathematics. 1987. Vol. 18. P. 87–94. DOI: 10.1016/0166-218X(87)90045-X.

18. Fleischner H. Eulerian Graphs and Related Topics. Part 1, Vol. 1.: Ann. Discrete Mathematics, 1990. no. 45.

19. Panyukova T.A. Constructing of OE-postman Path for a Planar Graph // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2014. Т. 7, № 4. С. 90–101. DOI: 10.14529/mmp140407.

20. Makarovskikh T.A., Panyukov A.V., Savitsky E.A. Mathematical Models and Routing Algorithms for CAM of Technological Support of Cutting Processes // ScienceDirect IFAC-PapersOnLine 49–12. 2016. P. 821–826. DOI: 10.1016/j.ifacol.2016.07.876.

21. Makarovskikh T., Panyukov A. The Cutter Trajectory Avoiding Intersections of Cuts // IFAC-PapersOnLine. 2017. Vol. 50, no. 1. P. 2284–2289. DOI: 10.1016/j.ifacol.2017.08.226.

22. Szeider S. Finding Paths in Graphs Avoiding Forbidden Transitions // Discrete Applied Mathematics. 2003. no. 126. P. 261–273. DOI: 10.1016/S0166-218X(02)00251-2.

Макаровских Татьяна Анатольевна, к.ф.-м.н., доцент, кафедра математического и компьютерного моделирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse190403

CONSTRUCTING SELF-NON-INTERSECTING OE-CHAINS IN A PLANE EULERIAN GRAPH

© 2019 T.A. Makarovskikh

South Ural University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: Makarovskikh.T.A@susu.ru

Received: 24.09.2019

The paper is devoted to a polynomial-time algorithm for constructing a self-non-intersecting ordered enclosing chain for a plane Eulerian graph. The proposed approach consists in splitting all the vertices of the original graph of degree higher than 4 and introducing fictive vertices and edges and, thus, reducing the considered earlier problem to the problem of finding an A-chain with ordered enclosing in a plane connected 4-regular graph. The presented reduction algorithm solves the problem in polynomial time. A test example of constructing a self-non-intersecting chain with ordered enclosing is considered. This problem arises during the technological preparation of the cutting process, when it is necessary to determine the path of the cutter when there are no self-intersections of the cutting path and the part cut off from the sheet does not require any cuts. The cutting plan may be presented as a planar graph which is the homeomorphic image of the cutting plan. The algorithm proposed in the article solves the problem of routing when cutting parts when such technological constraints are simultaneously imposed on the path of the cutter.

Keywords: plane graph, path, cutting plan, polynomial-time algorithm, cutting process.

FOR CITATION

Makarovskikh T.A. Constructing Self-non-intersecting OE-chains in a Plane Eulerian Graph. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 4. pp. 30–42. (in Russian) DOI: 10.14529/cmse190403.

References

1. Silva E.F., Oliveira L.T., Oliveira J.F., Toledo F.M.B. Exact approaches for the cutting path determination problem. Computers & Operations Research. 2019. vol. 112, art. 104772. DOI: 10.1016/j.cor.2019.104772.

2. Makarovskikh T.A., Panyukov A.V., Savitsky E.A. Mathematical Models and Routing Algorithms for CAD Technological Preparation of Cutting Processes. Automation and Remote Control. 2017. vol. 78, no. 4. pp. 868–882. DOI: 10.1016/10.1134/S0005117917050095.

3. Dewil R., Vansteenwegen P., Cattrysse D. A Review of Cutting Path Algorithms for Laser Cutters. International Journal Adv Manuf. Technol. 2016. vol. 87. pp. 1865–1884. DOI: 10.1007/s00170-016-8609-1.

4. Dewil R., Vansteenwegen P., Cattrysse D., Laguna M., Vossen T. An Improvement Heuristic Framework for the Laser Cutting Tool Path Problem. International Journal of Production Research. 2015. vol. 53, no. 6. pp. 1761–1776. DOI: 10.1080/00207543.2014.959268.

5. Hoeft J., Palekar U. Heuristics for the Plate-cutting Travelling Salesman Problem. IIE Transactions. 1997. vol. 29(9). pp. 719–731. DOI: 10.1023/A:1018582320737.

6. Dewil R., Vansteenwegen P., Cattrysse D. Construction Heuristics for Generating Tool Paths for Laser Cutters. International Journal of Production Research. 2014. vol. 52(20). pp. 5965–5984. DOI: 10.1080/00207543.2014.895064.

7. Petunin A.A., Chentsov A.G., Chentsov P.A. On the Issue of Routing Movements in Sheet Cutting of Parts. Bulletin of the South Ural State University. Series: Mathematical Modelling and Programming. 2017. vol. 10, no. 3. pp. 25–39. (in Russian) DOI: 10.14529/mmp170303.

8. Chentsov A.G., Grigoryev A.M., Chentsov A.A. Solving a Routing Problem with the Aid of an Independent Computations Scheme. Bulletin of the South Ural State University. Series: Mathematical Modelling and Programming. 2018. vol. 11, no. 1. pp. 60–74. (in Russian) DOI: 10.14529/mmp180106.

9. Petunin A., Stylios C. Optimization Models of Tool Path Problem for CNC Sheet Metal Cutting Machines. 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016 (Troyes, France, June, 28–30, 2016). IFAC-PapersOnLine. 2016. vol. 49. pp. 23–28. DOI: 10.1016/j.ifacol.2016.07.544.

10. Chentsov A., Khachay M., Khachay D. Linear Time Algorithm for Precedence Constrained Asymmetric Generalized Traveling Salesman Problem. 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016 (Troyes, France, June, 28–30, 2016). IFAC-PapersOnLine. 2016. vol. 49. pp. 651–655. DOI: 10.1016/j.ifacol.2016.07.767.

11. Khachay M., Neznakhina K. Towards Tractability of the Euclidean Generalized Travelling Salesman Problem in Grid Clusters Defined by a Grid of Bounded Height. Communications in Computer and Information Science. 2018. vol. 871. pp. 68–77. DOI: 10.1007/978-3-319-93800-4_6.

12. Manber U., Israni S. Pierce Point Minimization and Optimal Torch Path Determination in Flame Cutting. J. Manuf. Syst. 1984. vol. 3(1). pp. 81–89. DOI: 10.1016/0278-6125(84)90024-4.

13. Panyukova T. Chain Sequences with Ordered Enclosing. Journal of Computer and System Sciences International. 2007. vol. 46, no. 1(10). pp. 83–92. DOI: 10.1134/S1064230707010108.

14. Borse Y.M. Splitting Lemma for 2-Connected Graphs. International Scholarly Research Network, ISRN Discrete Mathematics. 2012. vol. 2012. Art. 850538. DOI: 10.5402/2012/850538.

15. Makarovskikh T.A. Software for Constructing of A-chains with Ordered Enclosing for a Plane Connected 4-regular Graph. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2019. vol. 8, no. 1. pp. 36–53. (in Russian) DOI: 10.14529/cmse190103.

16. Filippov A.F. The Elementary Proof of Jordan Theorem. Advances in Mathematical Sciences. 1950. vol. 5:5(39). pp. 173–176. (in Russian)

17. Manber U., Bent S.W. On Non-intersecting Eulerian Circuits. Discrete Applied Mathematics. 1987. vol. 18. pp. 87–94. DOI: 10.1016/0166-218X(87)90045-X.

18. Fleischner H. Eulerian Graphs and Related Topics. Part 1, vol. 1. Ann. Discrete Mathematics. 1990. no. 45.

19. Panyukova T.A. Constructing of OE-postman Path for a Planar Graph. Bulletin of the South Ural State University. Series: Mathematical Modelling, Programming and Computer Software. 2014. vol. 7, no. 4. pp. 90–101. DOI: 10.14529/mmp140407.

20. Makarovskikh T.A., Panyukov A.V., Savitsky E.A. Mathematical Models and Routing Algorithms for CAM of Technological Support of Cutting Processes. ScienceDirect IFAC-PapersOnLine 49–12. 2016. pp. 821–826. DOI: 10.1016/j.ifacol.2016.07.876.

21. Makarovskikh T., Panyukov A. The Cutter Trajectory Avoiding Intersections of Cuts. IFAC-PapersOnLine. 2017. vol. 50, no. 1. pp. 2284–2289. DOI: 10.1016/j.ifacol.2017.08.226.

22. Szeider S. Finding Paths in Graphs Avoiding Forbidden Transitions. Discrete Applied Mathematics. 2003. no. 126. pp. 261–273. DOI: 10.1016/S0166-218X(02)00251-2.

УДК 004.932, 528.854 DOI: 10.14529/cmse190404

ПОМЕХОУСТОЙЧИВЫЙ АЛГОРИТМ ПОСТРОЕНИЯ ПОЛЯ ПОТОКОВ ИЗОБРАЖЕНИЙ ОТПЕЧАТКОВ ПАЛЬЦЕВ

© 2019 А.В. Агафонов[1], Д.С. Рожина[1], Х.И. Ваххаб[1,2], А.Н. Аль Анссари[1,3]

[1]*Южно-Уральский государственный университет (454080 Челябинск, пр. им. В.И. Ленина, д. 76),*

[2]*Кербельский университет (56001, а/я 1125, Республика Ирак, Кербела),*

[3]*Куфинский университет (54003, а/я 21, Республика Ирак, Наджаф)*

E-mail: asp17aav494@susu.ru, asp17rds604@susu.ru, haider.wahhab@uokerbala.edu.iq, alaan.azeez@uokufa.edu.iq

Поступила в редакцию: 19.08.2019

Поле потоков отпечатков пальцев является одной из наиболее важных характеристик узора и оказывает большое влияние на всю процедуру дактилоскопической идентификации. Методы для построения поля потоков, основанные на градиенте, весьма популярны, однако, слишком чувствительны к различным шумам и дефектам, которые тем или иным образом проявляются на многих изображениях отпечатков. В данной статье предлагается новый метод построения поля потоков для цифровых изображений отпечатков пальцев. Он позволяет улучшить решение ряда ключевых задач обработки изображений. Среди таких задач прогноз направлений линий в области складок кожи, шрамов и других дефектов поверхности пальца. Метод опирается на такие подходы, как обработка изображения на субпиксельном уровне, на кластерный анализ поля градиентов изображения и заключатся в последовательном применении нескольких алгоритмов. Это интерполяция изображения и оценка значений градиента на нем, свертка поля градиента с заданным шаблоном для борьбы с шумами на субпиксельном уровне, выделение опорных областей на основе построения локальных оценок качества узора, прогнозирование поля направлений от опорных областей на все изображение с адаптацией прогнозируемых значений под результаты измерений. Верификация результатов работы предлагаемого метода выполнена с помощью веб-фреймворка, созданного на базе Болонского университета в Италии. Новые результаты верификации сравниваются с результатами верификации предшествующего метода, развитого в данной статье, и с другими опубликованными алгоритмами на том же веб-фреймворке.

Ключевые слова: биометрическая идентификация, поле потоков, распознавание образов, отпечатки пальцев, верификация.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Агафонов А.В., Рожина Д.С., Ваххаб Х.И., Аль Анссари А.Н. Помехоустойчивый алгоритм построения поля потоков изображений отпечатков пальцев // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 4. С. 43–55. DOI: 10.14529/cmse190404.

Введение

Широкое распространение автоматизированных систем идентификации по отпечаткам пальцев свидетельствует о высокой надежности их функционирования и рентабельности. Под рентабельностью понимается более выгодное использование таких систем, по сравнение с ручной работой специалиста. Для надежной работы этих систем необходимо решать ключевые задачи идентификации. Одна из таких задач — построение поля потоков. В настоящее время не известен алгоритм, безошибочно восстанавливающий изображение, свободное от дефектов и шумов, поэтому данное направление исследований является актуальным.

В реальных условиях изображения отпечатков пальцев обычно содержат шумы и другие дефекты. Они затрудняют работу алгоритмов распознавания и повышают общую ошибку идентификации.

В данной статье предложен метод, способный значительно снизить влияние различных дефектов на результат построения поля потоков. С этой целью был разработан и программно реализован алгоритм построения поля потоков и выполнено его тестирование фреймворком, предложенным на сайте FVC Ongoing. Результаты тестирования демонстрируют тринадцатипроцентное снижение ошибки идентификации на наборе изображений плохого качества по сравнению с другим опубликованным алгоритмом.

В данной статье раздел 1 посвящен обзору подходов к построению поля потоков в других работах. Раздел 2 посвящен формальной постановке проблемы построения алгоритма измерения, анализа и синтеза поля потоков. В разделе 3 описана реализация предложенного алгоритма. Раздел 4 содержит анализ результатов разработанного алгоритма. В заключении дана сводка результатов, полученных в работе, и определены направления дальнейших исследований.

1. Предшествующие работы

Поставленная задача широко исследуется авторами из разных стран. Рассмотрим существующие известные методы, подходы и алгоритмы, опубликованные в других работах.

Публикация [1] предлагает улучшенный метод построения поля потоков на основе квадратурных градиентов и шкалы качества. Для борьбы с шумами направление линий узора корректируется при помощи дискретных ортогональных многочленов. В качестве критерия оценки качества используется когерентность. Однако одной когерентности недостаточно, так как она не позволяет отличить жировой след отпечатка пальца от прикосновения кожи к поверхности сенсора.

Статья [2] посвящена применению фильтра Габора к дактилоскопическим изображениям. В нем одним из входных параметров для процедуры фильтрации является информация о поле потоков, подсчитанная на основе метода квадратурных градиентов. Предложенный в статье алгоритм опирается лишь на фильтрацию изображения, чего может быть недостаточно для построения информативного поля потоков для изображений плохого качества.

Работа [3] направлена на улучшение алгоритма построения поля потоков. В ней предлагается усовершенствованный метод квадратурных градиентов на основе минимальной дисперсии. Главная идея метода заключается в том, чтобы найти минимальную дисперсию между четырьмя направлениями квадратурных градиентов. В центральный блок помещается среднее значение, для которого дисперсия достигает минимального значения. Недостатком метода является анализ малого количества заранее заданных направлений с большим шагом, что негативно сказывается на точности вычислений.

Публикация [4] предлагает улучшенный алгоритм построения поля потоков на основе метода градиентов для работы с зашумленными изображениями. Основная идея метода заключается в разбивке изображения на блоки и подсчете для каждого блока величин градиента и когерентности — показателей качества. Значение в целевом блоке рассчитывается как средневзвешенное между его соседними блоками с учетом когерентности. К недостаткам данной работы можно отнести то, что качество области определя-

ется лишь на основе когерентности, а само качество используется только для оценки степени влияния локальных областей на результат. Таким образом, этот метод является, по сути, усовершенствованным сглаживанием.

Интересный подход к построению поля потоков предложен в статье [5]. В описанном методе поле потоков строится на основе сочетания глобальной ориентационной модели и локального подхода к определению направления линий узора. Как недостаток можно отметить то, что фрагменты узоров, которые часто отпечатываются при прикосновении пальцев к поверхности сенсора, плохо вписываются в глобальную модель узора.

В работе [6] предложено решение для построения поля потоков в несколько этапов. На первом этапе происходит нормализация яркостей изображения, на втором этапе — улучшение четкости линий узора на основе подсчета колебаний яркости в восьми направлениях, на третьем — собственно построение поля потоков модифицированным методом наименьших средних квадратов. В качестве недостатка здесь можно отметить наличие восьми фиксированных направлений линий узора, что оказывает негативное влияние на точность работы алгоритма.

Статья [7] предлагает многоэтапный подход к улучшению качества дактилоскопических изображений, включающий построение поля потоков. Первый этап использует оператор Собеля для начального определения направления линий узора. Лучшая оценка дается при помощи индексов Пуанкаре с использованием информации об особых точках. После этого поле потоков уточняется с применением полиномов Лежандра. Подход является любопытным, однако для реализации его основной части требуется информация об особых точках узора, поиск которых является отдельной, нетривиальной задачей и выходит за границы текущей работы. Кроме того, начальное направление линий подсчитывается простым оператором Собеля и не раскрыта тема качества областей узора.

В работе [8] для анализа поля потоков применяют градиентный метод с двумя последовательными процедурами сглаживания — по окрестности и вдоль направления линий узора. Однако работа алгоритма выполняется без учета качественных характеристик локальной области узора. Из-за этого метод успешно работает на изображениях хорошего качества, но на зашумленных изображениях велика ошибка распознавания.

Алгоритм, предлагаемый в данной статье, позволяет устранить подобные недостатки, реализуя многоэтапное улучшение построения поле потоков. Основными этапами являются: оценка производной в точке с применением интерполяции и разностных производных; применение метода квадратурных градиентов со сглаживанием; выделение опорной области внутри области интереса на основе оценки качества фрагментов узора; расширение опорной области до всей области интереса. Такие изменения алгоритма позволяют улучшить результаты распознавания направлений линий.

2. Постановка задачи

Поле потоков — это матрица, элементы которой представляют собой значения направлений линий узора, усредненные для каждого сегмента. Такая матрица обычно визуализируется как сегментированное изображение, состоящее из неперекрывающихся областей, в каждой из которых отображается усредненное направление.

Большинство существующих алгоритмов дактилоскопической идентификации используют построение поля потоков как один из ключевых этапов, который оказывает влияние на последующие этапы обработки изображения, так как является для них

входной информацией. Кроме того, качественное построение поля потоков улучшает детектирование истинных контрольных точек (разветвлений и окончаний линий), что непременно уменьшает ошибки идентификации.

Обычно методы определения поля потоков основаны на методе градиента. К недостаткам этого метода можно отнести чувствительность к шумам и к крутым изгибам линий узора. Но хорошее описание в опубликованных трудах и относительно простая идея обеспечивают его популярность. Различные авторы предлагают богатый набор методов для преодоления указанных недостатков.

Для верификации полезности нового алгоритма авторами было выполнено его сравнение с предыдущим алгоритмом, описанным в статье [8], с использованием инструментов, предоставляемых на ресурсе FVC Ongoing [9]. На нем выполняется верификация результатов работы алгоритма на закрытой базе изображений и вычисляется ряд показателей, которые можно использовать для оценки качественного развития алгоритма.

Кроме того, там же опубликованы и доступны для анализа и сравнения результаты верификации других алгоритмов. Всего на сайте оценено свыше 6000 алгоритмов. Изображения, показанные на рис. 1 и рис. 2, доступны в открытой базе на сайте. База разделена экспертами FVC на изображения хорошего и плохого качества.

Рис. 1. Примеры изображений хорошего качества из открытой базы FVC

Рис. 2. Примеры изображений плохого качества из открытой базы FVC

Требуется предложить такое решение задачи построения поля потоков, которое позволило бы повысить ранг показателей качества поля потоков в таблице опубликованных алгоритмов сайта FVC Ongoing.

3. Описание алгоритма

3.1. Интерполяция и измерение градиента

Градиент — вектор, показывающий направление наискорейшего подъема двумерной функции яркости изображения. Градиент изображения в точке (x, y) — это вектор:

$$\nabla f = \left(\mathrm{G}_x,\ G_y\right) = \left(\frac{df}{dx}, \frac{df}{dy}\right), \tag{1}$$

где $\frac{df}{dx}$ и $\frac{df}{dy}$ — скорость изменения яркости соседних пикселей по осям X и Y.

Направление вектора градиента в точке (x, y) совпадает с направлением наибольшего роста функции, а модуль равен значению этого приращения. Вектор градиента, как и любой другой вектор, имеет две основные характеристики: модуль и направление. Модуль вектора находят в виде [10]:

$$|\nabla f(x, y)| = \sqrt{\mathrm{G}_\mathrm{x}^2 + \mathrm{G}_\mathrm{y}^2}, \tag{2}$$

а направление — угол $\alpha(x, y)$ между направлением ∇f в точке (x, y) и осью абсцисс:

$$\alpha(x, y) = \ arctg\left(\frac{G_y}{G_x}\right). \tag{3}$$

Направление линий узора в окрестности точки (x, y), очевидно, перпендикулярно направлению вектора градиента в этой точке, а вычислить градиент изображения можно, вычислив величины частных производных $\frac{\partial f}{\partial x}$ и $\frac{\partial f}{\partial \mathrm{y}}$ в каждой точке.

Из-за дискретности изображения его производная не определена, поэтому находят ее приближенное значения, например, оператором Собеля [11]. В работе предложен собственный метод оценки значений производных на основе интерполяции.

В нем исходное изображение с разрешением 500 dpi приводится к 1000 dpi с помощью билинейной интерполяции [10]. После чего на нем применяется метод разностной оценки производной (4) на сегментах 2×2 и сворачивание к исходному разрешению:

$$f'(\mathrm{x}) = \frac{f(x + h) - f(x)}{h}. \tag{4}$$

3.2. Градиент в окрестности

Для повышения точности построения поля потоков градиент усредняют в некоторой окрестности. Однако простого усреднения направлений векторов будет недостаточно, так как противоположные по направлению векторы на разных сторонах одной линии взаимно компенсируются, хотя и соответствуют одним и тем же линиям узора.

Обозначим через S окрестность точки (x, y) размером $I * J$. Значения модуля и аргумента градиента в окрестности находят в виде [12]:

$$X(x, y) = \sum_{i=1}^{I}\sum_{j=1}^{J}|\nabla f(i, j)| \cdot \cos(2\alpha(i, j)), \tag{5}$$

$$Y(x, y) = \sum_{i=1}^{I}\sum_{j=1}^{J}|\nabla f(i, j)| \cdot sin(2\alpha(i, j)), \tag{6}$$

$$|\nabla F|\ (x, y) = \sqrt{X \cdot X + Y \cdot Y}, \tag{7}$$

$$A\ (x, y) = \ Atan(\frac{Y}{X})/2. \tag{8}$$

Следует обратить внимание, что в формулах 5 и 6 используется удвоенный угол. Это сделано для того, чтобы векторы градиента на противоположных сторонах линии узора не компенсировали друг друга, так как они относятся к краям одной и той же линии. Фактически, это удвоение полуплоскости $[0, \pi)$ до плоскости $[0, 2\pi)$. В формуле (8) присутствует деление на два, чтобы восстановить аргумент градиента.

3.3. Выделение опорной области

Для выделения опорной области выполняется оценка качества поля потоков, которая строится на основе когерентности и модуля градиента.

В окрестности S мощностью N с центром в точке (x, y) на основе формул (5)–(8) рассчитывают когерентность [13]:

$$Coh(x,y) = \frac{\left|\sum_{n=1}^{N} \nabla f_n\right|}{\sum_{n=1}^{N} |\nabla f_n|} \tag{9}$$

и среднее значение модуля градиента:

$$\overline{|\nabla F|\,(x,y)} = \frac{\sum_{n=1}^{N} |\nabla F(x,y)|_n}{N}. \tag{10}$$

Когерентность является отношением модуля суммы векторов в окрестности к сумме модулей и не может превышать единицы. Чем ближе значение когерентности к единице, тем качественнее окрестность точки (x, y). Тогда качество окрестности в точке (x, y) можно оценить по формуле

$$R(x,y) = |\nabla F(x,y)| \ * Sin(Coh(x,y) * 90^o). \tag{11}$$

В работе для выделения опорной области отбирают верхний квартиль качества по гистограмме качества окрестностей.

3.4. Расширение опорной области

Расширение опорной области происходит в два этапа. На первом этапе элементы, примыкающие к опорной области, включаются в нее, если их направление отличается от среднего направления смежных элементов из опорной области не более, чем на заданное значение δ. Первый этап итерационно повторяется пока существуют такие элементы. На втором этапе элементы, примыкающие к опорной области, принимают значение, которое является усредненным среди соседей уже включенных в опорную область.

Для примера если рассматриваемый элемент примыкает к границе опорной области, то среднее значение его соседних элементов, входящих в опорную область, рассчитывают по формулам:

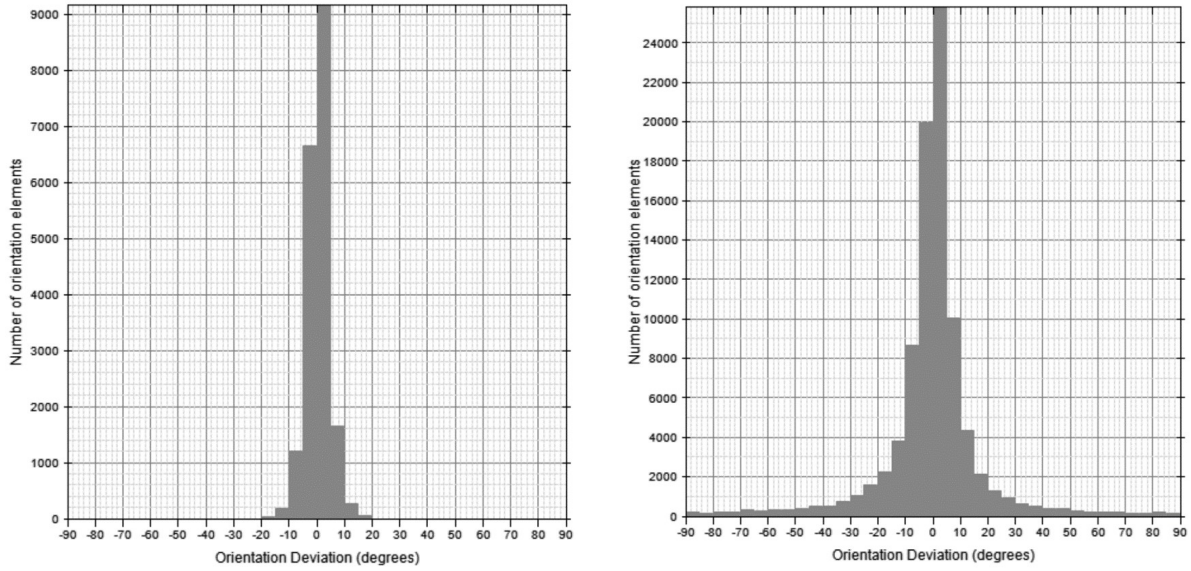
$$X(x,y) = \sum_{i=1}^{I} \sum_{j=1}^{J} \cos\big(2\alpha(i,j)\big) * t(i,j), \tag{12}$$

$$Y(x,y) = \sum_{i=1}^{x} \sum_{j=1}^{y} \sin\big(2\alpha(i,j)\big) * t(i,j), \tag{13}$$

где функция $t(i, j)$ принимает значение 1, если элемент входит в опорную область и 0 в противном случае. Собственно, среднее направление находится по формуле 8.

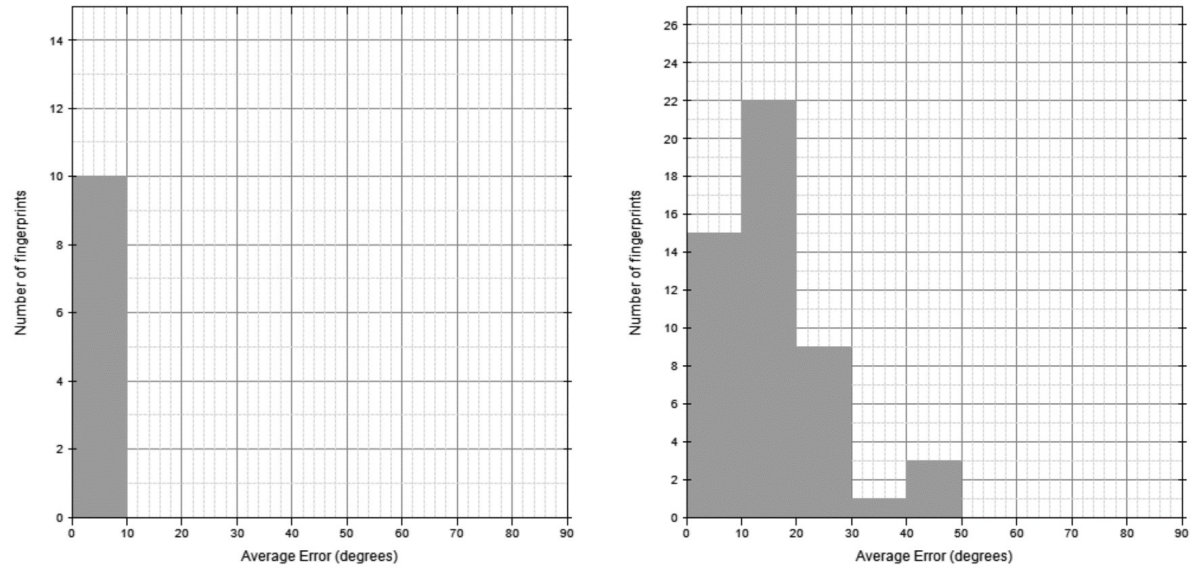
4. Анализ результатов

Программа с реализацией предлагаемого метода протестирована на сайте FVC Ongoing и результат доступен для сравнения с другими участниками [9]. Сами результаты приведены в 12 строке таблицы и на рис. 3, рис. 4. Также таблица 12 отражает сравнение с другими участниками по параметрам СКО на хорошем и плохом наборе изображений. Разделение изображений по качеству проведено экспертами FVC.



а) набор изображений хорошего качества б) набор изображений плохого качества

Рис. 3. Гистограммы распределения отклонений ориентации для наборов изображений



а) набор изображений хорошего качества б) набор изображений плохого качества

Рис. 4. Гистограммы распределения СКО для наборов изображений

В одиннадцатой строке таблицы приведены результаты тестирования предыдущей версии алгоритма. Сравнительный анализ показывает, что ошибка распознавания на плохом наборе изображений уменьшилась на 13 %. Кроме того, время работы алгоритма снизилось на 46 % (с 4324 мс до 2342 мс).

Таблица

Сравнение опубликованных алгоритмов

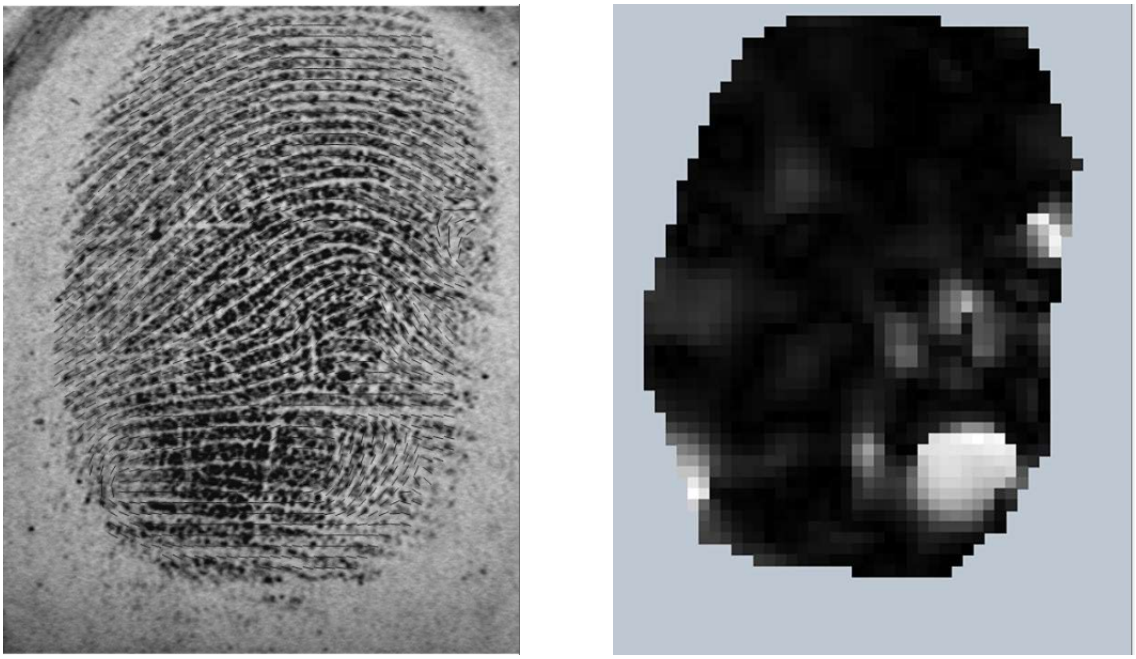
№	Участник тестирования	Название алгоритма	СКО EG	СКО EB
1	Dermalog Identification Systems GmbH	DEX-OF	4,89°	7,52°
2	Dept. of Information Science & Electronic Engineering, Zhejiang University	OriNet	6,94°	8,44°
3	Dermalog Identification Systems GmbH	ConvNetOF	5,80°	8,53°
4	Department of Automation, Tsinghua University	LocalDict	6,08°	9,66°
5	Institute of Automation, Chinese Academy of Sciences	ROF	5,24°	11,20°
6	Antheus Technology, Inc.	AntheusOriEx	5,46°	17,06°
7	Zengbo Xu	MXR	5,59°	11,36°
8	Biometric System Laboratory	Adaptive-3 (Baseline)	5,93°	13,27°
9	School of Engineering and Information Technology, UNSW@ADFA	FOMFE	6,70°	21,44°
10	Biometric System Laboratory	Gradient (baseline)	5,86°	21,83°
11	**Предыдущая версия алгоритма**	AVG+	5,10°	17,90°
12	**Алгоритм из данной работы**	DPA	5,03°	15,58°

На основе ресурса FVC Ongoing можно проанализировать влияние модификаций и нововведений алгоритма на снижение ошибки распознавания на примерах набора изображений плохого качества, показанных на рис. 5.

Рис. 5. Примеры изображений для демонстрации работы алгоритма

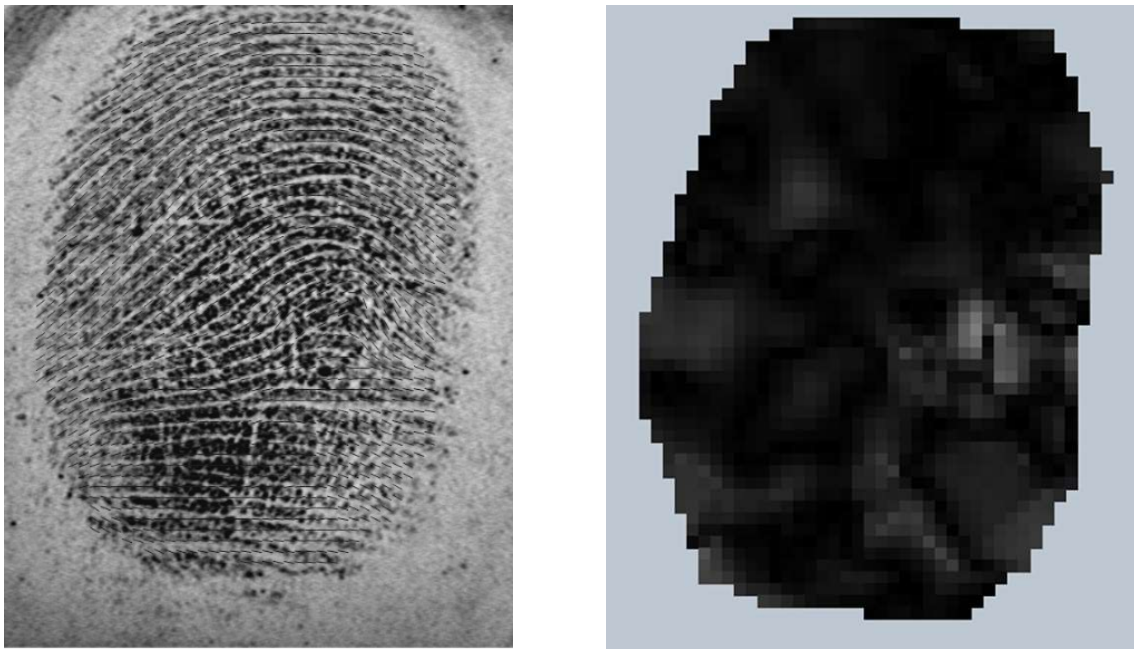
На рис. 6–9 показаны результаты построения поля потоков для нового и предшествующего алгоритмов для одних и тех же изображений. На них под литерой *а* показано построение поля потоков для дактилоскопического узора, а под литерой *б* демонстрируется сравнение с модельным полем потоков, составленным экспертами FVC, в цветовой интерпретации. Чем светлее области, тем больше ошибка оценки поля потоков алгоритма.

Изображения на рис. 6–9 наглядно демонстрируют преимущество построения поля потоков нового алгоритма в области шумов и дефектов кожи, которое достигается за счет предсказания направлений линий на основе опорной области.



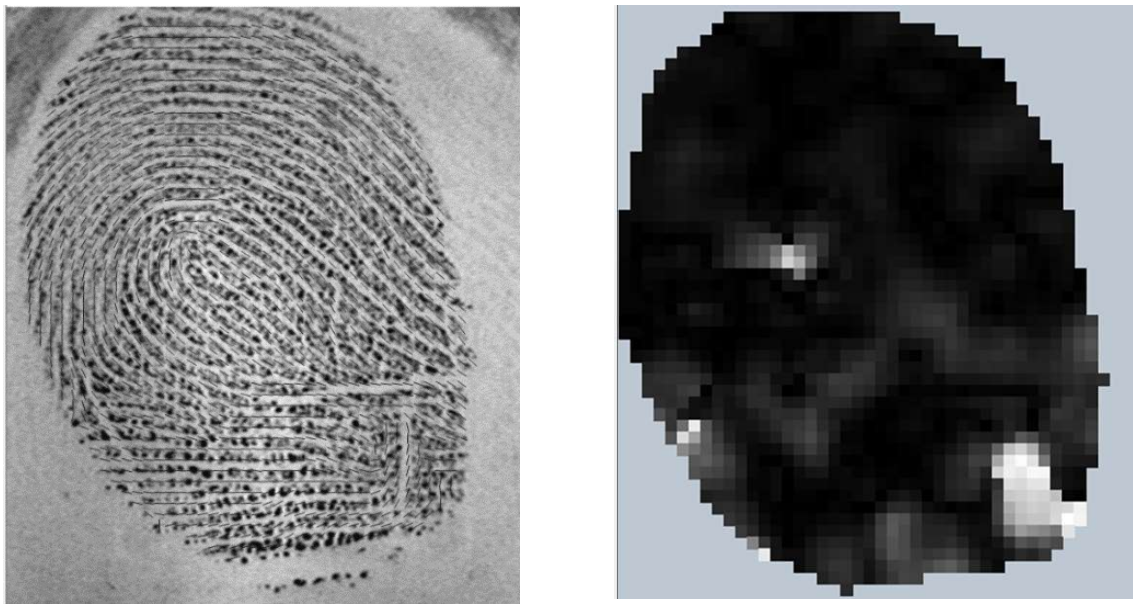
а) поле направлений б) сравнение с модельным полем

Рис. 6. Результаты работы предшествующего алгоритма



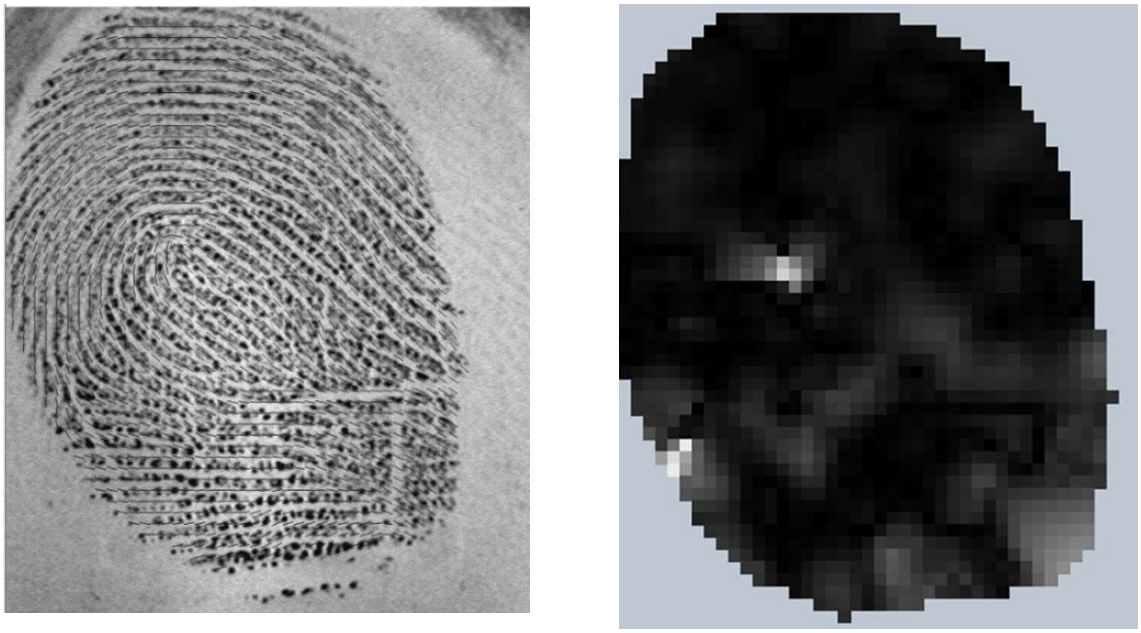
а) поле направлений б) сравнение с модельным полем

Рис. 7. Результаты работы нового алгоритма



а) поле направлений б) сравнение с модельным полем

Рис. 8. Результаты работы предшествующего алгоритма



а) поле направлений б) сравнение с модельным полем

Рис. 9. Результаты работы нового алгоритма

Заключение

В ходе исследования разработана улучшенная версия алгоритма построения поля потоков для изображений отпечатков пальцев.

Предложенный метод позволяет значительно повысить качество построения поля потоков. Продемонстрирована работа метода на независимых международных тестах FVC Ongoing. Показано улучшение ранговых оценок среди других участников тестов. По результатам теста удалось подняться с восьмого места на седьмое по набору изображений плохого качества.

Анализ и сравнение результатов тестирования показывают, что предложенные идеи привели к значительному повышению качества построения поля потоков по сравнению с

предыдущим алгоритмом. Это способствует снижению ошибки идентификации отпечатков пальцев.

Кроме того, удалось преодолеть недостатки некоторых из ранее опубликованных работ [2, 4, 7], обзор которых представлен в разделе 1.

Дальнейшее направление исследований видится в применении подходов и методов машинного обучения для построения поля потоков.

Литература

1. Weixin B., Shifei D., Yu X. An Improved Fingerprint Orientation Field Extraction Method Based on Quality Grading Scheme. // International Journal of Machine Learning and Cybernetics. 2018. Vol. 9, no. 8. P. 1249–1260. DOI: 10.1007/s13042-016-0627-7.
2. Гудков В.Ю., Бойцов А.В. Улучшение изображений отпечатков пальцев с помощью фильтра Габора // Вестник ЮУрГУ. Серия: Компьютерные технологии, управление, радиоэлектроника. 2015. Т. 15, № 1. С. 128–132.
3. Saparudin M., Ghazali S. A Technique to Improve Ridge Flows of Fingerprint Orientation Fields Estimation // Telkonomika. 2016. Vol. 14. P. 987–998. DOI: 10.12928/telkomnika.v14i2.3112.
4. Wang Y., Jiankun H., Heiko S. A Gradient Based Weighted Averaging Method for Estimation of Fingerprint Orientation Fields // Digital Image Computing: Techniques and Applications. 2005. DOI: 10.1109/DICTA.2005.4.
5. Carsten G., Benjamin T., Stephan H. Perfect Fingerprint Orientation Fields by Locally Adaptive Global Models // IET Biometrics. 2017. Vol. 6, no. 3. P. 183–190. DOI: 10.1049/iet-bmt.2016.0087.
6. Wieclaw L. Fingerprint Orientation Field Enhancement // Computer Recognition Systems 4. 2011. Vol. 95. P. 33–40. DOI: 10.1007/978-3-642-20320-6_4.
7. Khachay M., Pasynkov M. Theoretical approach to developing efficient algorithms of fingerprint enhancement // Analysis of Images, Social Networks and Texts. Communications in Computer and Information Science (CCIS), Springer. 2015. Vol. 542. P. 83–95. DOI: 10.1007/978-3-319-26123-2_8.
8. Агафонов А.В., Рожина Д.С. Верификация алгоритма построения поля потоков цифровых изображений отпечатков пальцев // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2018. Т. 7, № 4. С. 67–82. DOI: 10.14529/cmse180405.
9. Biometric System Laboratory. FVC-onGoing: on-line evaluation of fingerprint recognition algorithms. URL: https://biolab.csr.unibo.it/FvcOnGoing/UI/Form/Home.aspx (дата обращения: 20.03.2019).
10. Gonsalez. R., Woods E. Digital Image Processing. Prentice Hall, 2001. 794 p.
11. Jane B. Digital Image Processing: Concepts, Algorithms and Scientific applications. Springer Verlag, 2005. 589 p.
12. Maltoni D., Maio D., Jain A.K. Handbook of Fingerprint Recognition. New York: Springer-Verlag, 2003. 348 p. DOI: 10.1007/978-1-84882-254-2.
13. Bazen A. Fingerprint Identification: Feature Extraction, Matching, and Database Search. The Netherlands: Univ. of Twente, 2002. 187 p.

Агафонов Андрей Валерьевич, аспирант, кафедра электронных вычислительных машин, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Рожина Дарья Сергеевна, аспирант, кафедра электронных вычислительных машин, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Ваххаб Хадер Ибас АбдулВаххаб, аспирант, кафедра электронных вычислительных машин, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Аль Анссари Алаа Неамах, аспирант, кафедра электронных вычислительных машин, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse190404

ROBUST FINGERPRINT FLOW CHART ALGORITHM

© 2019 A.V. Agafonov[1], D.S. Rozhina[1], H.I. Wahhab[1,2], A.N. Alanssari[1,3]

[1]South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia),

[2]University of Karbala (P.O. Box 1125, Karbala, 56001 Iraq),

[3]University of Kufa (P.O. Box 21, Al-Najaf, Kufa, 54003 Iraq)

E-mail: asp17aav494@susu.ru, asp17rds604@susu.ru, haider.wahhab@uokerbala.edu.iq, alaan.azeez@uokufa.edu.iq

Received: 19.08.2019

The orientation field is an important characteristic of human skin patterns and has a significant impact on the results of fingerprint identification. The methods of constructing the orientation field based on the gradient are diverse, but they are united by a high sensitivity to noise and defects that appear on the images during the formation of tracks. The article proposes a new method for constructing a stream field for digital images of fingerprints. The method allows to improve the solution of a number of key tasks of image processing, including the task of predicting the direction of lines in the area of skin folds, scars and other finger surface defects. The method relies on such approaches as image processing at the subpixel level, on cluster analysis of the field of image gradients and consists in the sequential application of several algorithms. These are image interpolation and estimation of gradient values on it, convolution of the gradient field with a given pattern for noise control at the subpixel level, selection of reference areas based on the construction of local quality estimates of the pattern, prediction of the direction field from reference areas over the entire image with the adaptation of the predicted values for the measurement results. The results of the proposed method were verified using the web framework created on the basis of the University of Bologna in Italy. The new verification results are compared with the verification results of the earlier method developed by the authors, and with other published algorithms on the same web framework.

Keywords: biometrical identification, fingerprint orientation field, pattern recognition, fingerprints, verification.

FOR CITATION

Agafonov A.V., Rozhina D.S., Wahhab H.I., Alanssari A.N. Robust Fingerprint Flow Chart Algorithm. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 4. pp. 43–55. (in Russian) DOI: 10.14529/cmse190404.

References

1. Weixin B., Shifei D., Yu X. An Improved Fingerprint Orientation Field Extraction Method Based on Quality Grading Scheme. International Journal of Machine Learning and Cybernetics. 2018. vol. 9, no. 8. pp. 1249–1260. DOI: 10.1007/s13042-016-0627-7.
2. Gudkov V.Ju., Bojcov A.V. Enhancement of Fingerprint Images with the Gabor Filter. Bulletin of South Ural State University. Series: Mathematical Modeling, Programming & Computer Software. 2015. vol. 15, no. 1. pp. 128–132. (in Russian)
3. Saparudin M., Ghazali S. A Technique to Improve Ridge Flows of Fingerprint Orientation Fields Estimation. Telkonomika. 2016. vol. 14, no. 3. pp. 987–998. DOI: 10.12928/telkomnika.v14i2.3112.
4. Yi W., Jiankun H., Heiko S. A Gradient Based Weighted Averaging Method for Estimation of Fingerprint Orientation Fields. Digital Image Computing: Techniques and Applications. 2005. DOI: 10.1109/DICTA.2005.4.
5. Carsten G., Benjamin T., Stephan H. Perfect Fingerprint Orientation Fields by Locally Adaptive Global Models. IET Biometrics. 2017. vol. 6, no. 3. pp. 183–190. DOI: 10.1049/iet-bmt.2016.0087.
6. Wieclaw L. Fingerprint Orientation Field Enhancement. Computer Recognition Systems 4. 2011. vol. 95. pp. 33–40. DOI: 10.1007/978-3-642-20320-6_4.
7. Khachay M., Pasynkov M. Theoretical approach to developing efficient algorithms of fingerprint enhancement. Analysis of Images, Social Networks and Texts. Communications in Computer and Information Science (CCIS), Springer. 2015. vol. 542. pp. 83-95. DOI: 10.1007/978-3-319-26123-2_8.
8. Agafonov A.V., Rozhina D.S. Verification of the Algorithm for Estimating the Flow Chart of Fingerprint Images. Bulletin of South Ural State University. Series: Computational Mathematics and Computer Science. 2018. vol. 7, no. 4. pp. 67–82. (in Russian) DOI: 10.14529/cmse180405.
9. Biometric System Laboratory. FVC-Ongoing: On-line Evaluation of Fingerprint Recognition Algorithms. Available at: https://biolab.csr.unibo.it/FvcOnGoing/UI/Form/Home.aspx (accessed 20.03.2019).
10. Gonsalez R., Woods E. Digital Image Processing. Prentice Hall, 2001. 794 p.
11. Jane B. Digital Image Processing: Concepts, Algorithms and Scientific Applications. Springer Verlag, 2005. 589 p.
12. Maltoni D., Maio D., Jain A.K. Handbook of Fingerprint Recognition. New York, Springer-Verlag, 2003. 348 p. DOI: 10.1007/978-1-84882-254-2.
13. Bazen A. Fingerprint Identification: Feature Extraction, Matching, and Database Search. The Netherlands: Univ. of Twente, 2002. 187 p.

УДК 004.588 **DOI: 10.14529/cmse190405**

СТРУКТУРНО-ИЕРАРХИЧЕСКАЯ ДИДАКТИЧЕСКАЯ МОДЕЛЬ ЭЛЕКТРОННОГО ОБУЧЕНИЯ

© 2019 Н.С. Силкина, Л.Б. Соколинский

Южно-Уральский государственный университет

(454080 Челябинск, пр. им. В.И. Ленина, д. 76)

E-mail: SilkinaNS@susu.ru, Leonid.Sokolinsky@susu.ru

Поступила в редакцию: 22.10.2019

В данной статье описывается оригинальная структурно-иерархическая дидактическая (СИД) модель электронного обучения. В основе модели лежит четырехуровневая методическая база знаний. Первый уровень включает в себя комплекс электронных учебных энциклопедий по различным областям знаний. Второй уровень включает в себя электронные учебные курсы. Модель поддерживает структурирование электронного учебного курса по дидактическим компонентам (вертикальное слоение) и уровням детализации (горизонтальное слоение). Третий уровень включает в себя комплекс рабочих учебных программ. Четвертый уровень включает в себя комплекс ФГОС ВО. Отличительной особенностью СИД модели является деление образовательных объектов на дидактические компоненты. Это позволит, во-первых, производить автоматическую верификацию дидактической полноты электронного учебного курса. Во-вторых, включать части одного курса в другой без потери дидактической структуры. В-третьих, выделять из электронного учебного курса отдельный дидактический слой и на его основе автоматически формировать специализированные учебно-методические материалы: конспекты лекций, сборники задач, экзаменационные тесты и др. Описаны основные операции СИД модели, на основе которых предложены алгоритмы по анализу образовательных программ и электронных учебных курсов. В заключении дается краткая сводка результатов и направления дальнейших исследований.

Ключевые слова: электронное обучение, e-learning, электронный учебный курс, модель электронного обучения, образовательный объект, дидактическая структура, СИД модель.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Силкина Н.С., Соколинский Л.Б. Структурно-иерархическая дидактическая модель электронного обучения // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 4. С. 56–83. DOI: 10.14529/cmse190405.

Введение

Идея использования компьютеров в учебном процессе нашла свое выражение еще в 60-е годы в форме концепции программированного обучения [1], которая была предложена американским психологом Б.Ф. Скиннером. В соответствие с этой концепцией весь учебный материал делился на небольшие порции, а процесс обучения — на шаги. В ходе одного шага учащийся осваивал одну порцию материала. Программа, управляющая обучением, поддерживала только линейную последовательность шагов. Позднее Н. Краудером был предложен алгоритм разветвленного программированного обучения [2]. Основным отличием данного подхода является введение индивидуальных путей прохождения учебного материала. Путь для каждого учащегося определяет сама программа в процессе обучения, основываясь на ответах учащихся. Следующий этап развития программированного обучения связан с использованием гипертекстовых и мультимедийных технологий [3], которые, по существу, стали базовой технологической платформой образовательного контента. Это дало возможность использовать в качестве образовательных объектов

не только текстовую информацию, но также графику, аудио и видеоинформацию. Усложнение внутренней структуры образовательного контента породило проблему повторного использования элементов электронных учебных курсов в других электронных обучающих системах. Необходимость решения этой проблемы стимулировала развитие универсальных моделей электронного обучения и стандартов, разработанных на их основе.

Множество моделей электронного обучения можно представить в виде иерархии классов, изображенной на рис. 1. Первый уровень представлен *моделями данных*, предназначенными для обмена данными между образовательными объектами (Learning Objects, LO) и системой управления обучением (Learning Management System, LMS). LO использует модель данных для того, чтобы получить от LMS информацию, которая позволит ему выполнить требуемые обучающие функции. LMS использует модель данных для того, чтобы получать от LO информацию, которая позволит ей правильно управлять объектами LO. Таким образом, модель данных должна описывать структуру информации, которая может быть передана в и получена от LO. Однако, модель данных не должна специфицировать как, когда и в каком направлении информация может передаваться. К моделям данного уровня относится *Модель данных для взаимодействия с образовательными объектами* (*Data Model for Content Object Communication*) [4].

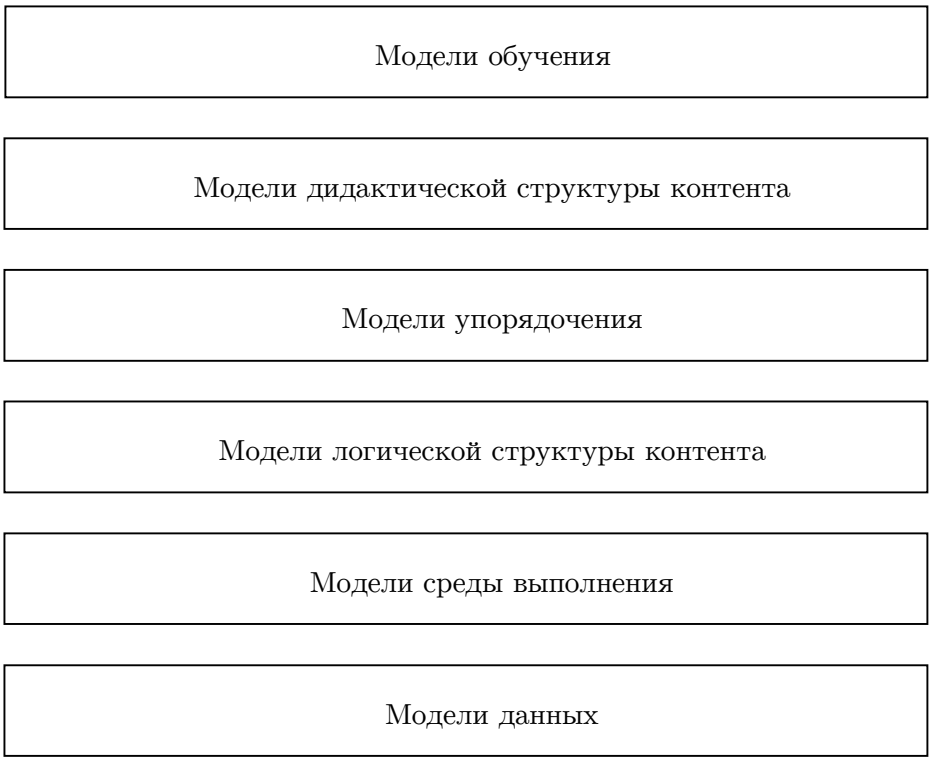


Рис. 1. Иерархия классов моделей электронного обучения

Второй уровень представлен *моделями среды выполнения*, которые описывают взаимодействие LO и LMS через прикладной программный интерфейс (Application Program Interface, API). Модель среды выполнения позволяет обеспечить совместимость LO и LMS, чтобы каждая система электронного обучения могла взаимодействовать с LO таким же образом, как и любая другая, поддерживающая ту же модель среды выполнения. Данная модель должна обеспечивать доставку требуемых ресурсов учащемуся, за-

пуск LO, отслеживание и обработку информации о действиях учащегося. Примером моделей данного уровня может служить *Модель среды выполнения* (*Run-Time Environment, RTE*) стандарта SCORM [5].

Следующий уровень представлен *моделями логической структуры контента*, согласно которым образовательный контент представляет собой некую структурно-иерархическую организацию (как правило, в виде графа или дерева). К ним относится *Модель структуры контента стандарта* (*Content Aggregation Model, CAM*) SCORM [6], *карта знаний* (*Concept Map*) [7], *граф содержания* [8].

Следующий уровень представлен моделями, которые позволяют определить последовательность изучения LO, в том числе повторное изучение. К таким моделям относится *Модель упорядочения и навигации* (*Sequencing and Navigation, SN*) стандарта SCORM [9], а также упорядочивание образовательного контента в *модели CDCGM* [10].

На пятом уровне иерархии классов находятся *модели дидактической структуры контента*. Примером может служить *Модель компетенций* (*Data Model for Reusable Competency Definitions*), описываемая стандартом IEEE 1484.20.1-2007 [11]. Также к этому классу может быть отнесена *модель динамического контента* (*Dynamic Content Model, DCM*) [12].

На верхнем уровне иерархии находятся модели обучения, которые можно разделить на три группы: модели адаптивного обучения, модели коллаборативного обучения и модель электронной учебной энциклопедии [13, 14, 15].

Адаптивное обучение — это способ организации учебного процесса с учетом индивидуального уровня подготовки учащегося до начала обучения или в процессе обучения, при котором направление дальнейшего обучения (график и интенсивность) определяется по результатам завершения предыдущих курсов или их частей. К этому классу моделей можно отнести *модель KFS* [16–18], *модель содержания учебного материала Соловова* [19, 20] и *модель CDCGM* [21].

Коллаборативное обучение — это такой способ организации обучения, при котором обучение происходит в процессе совместного решения учебных задач, осуществления взаимообмена знаниями. Одной из самых популярных систем, реализующих подобную модель, является LMS с открытым кодом Moodle (http://moodle.org), которая широко известна в мире и используется более чем в 100 странах, в том числе и в России.

Электронные энциклопедии [22] представляют собой базу знаний по некоторой предметной области (или сразу нескольким областям). Учебный материал в электронной энциклопедии представляется в виде описания понятий (статей), каждое понятие может «опираться» на другие понятия. Примерами реализации такого подхода являются электронные энциклопедии по линейной алгебре Линеал (http://lineal.guru.ru), по параллельному программированию Параллель (http://parallel.guru.ru), а также свободная энциклопедия по языкам программирования Прогопедия (http://progopedia.ru) и свободная энциклопедия Wikipedia (http://www.wikipedia.org). Статьи в электронной энциклопедии во многих случаях имеют линейный (алфавитный) порядок, однако, на ряду с этим встречаются энциклопедии, в которых словарные статьи организованы по иерархическому принципу [23, 24]. Подробнее все рассмотренные модели проанализированы авторами в работах [25, 26].

Целью данной работы является построение структурно-иерархической дидактической модели электронного обучения (или СИД модели). Данная модель относится к пятому

уровню иерархии, представленной на рис. 1. В основе модели лежит четырехуровневая методическая база знаний. Первый уровень включает в себя комплекс электронных учебных энциклопедий по различным областям знаний. Верхний уровень включает в себя комплекс ФГОС ВО. Третий уровень включает в себя комплекс рабочих учебных программ. Второй уровень включает в себя электронные учебные курсы. Модель поддерживает структурирование электронного учебного курса по дидактическим компонентам (вертикальное слоение) и уровням детализации (горизонтальное слоение). Отличительной особенностью этой модели является поддержка деления образовательных объектов на дидактические компоненты. Это позволит, во-первых, производить автоматическую верификацию дидактической полноты электронного учебного курса. Во-вторых, включать части одного курса в другой без потери дидактической полноты. В-третьих, выделять из электронного учебного курса отдельный дидактический слой и на его основе автоматически формировать самостоятельные учебно-методические материалы: конспекты лекций, сборники задач, экзаменационные тесты и др.

Статья организована следующим образом. В разделе 1 описываются требования к структурно-иерархической дидактической модели. Раздел 2 посвящен формальному описанию СИД модели. В разделе 3 дается описание основных операций СИД модели. В разделе 4 описываются основные алгоритмы по анализу образовательных программ и электронных учебных курсов на базе СИД модели. Заключение содержит выводы и направления дальнейших исследований.

1. Требования к модели дидактической структуры контента

Модель дидактической структуры контента должна удовлетворять следующим основным требованиям:

1) учитывать специфику образовательных стандартов «ФГОС 3++»;
2) определять дидактические типы для образовательных объектов;
3) поддерживать модульный подход при формировании образовательного контента;
4) предусматривать возможность создания дидактически структурированных хранилищ образовательных объектов в определенной предметной области (электронных энциклопедий);
5) предусматривать возможность создания дидактически структурированных электронных курсов, а также обеспечивать возможность их автоматизированного создания на основе электронных энциклопедий и других электронных курсов;
6) обеспечивать совместимость с существующими стандартами в области электронного обучения.

Рассмотрим указанные требования более подробно.

1.1. Образовательные стандарты «ФГОС 3++»

Федеральные государственные образовательные стандарты высшего профессионального образования «ФГОС 3++» представляют собой совокупность требований, обязательных при реализации основных образовательных программ высшего профессионального образования образовательными учреждениями, имеющими государственную аккредитацию. Федеральным законом «Об образовании в Российской Федерации» от 29 декабря 2012 года № 273-ФЗ утверждена структура федерального государственного образовательного стандарта (ФГОС). Каждый ФГОС включает три группы требований:

1) требования к структуре основных образовательных программ (в том числе соотношению обязательной части основной образовательной программы и части, формируемой участниками образовательных отношений) и их объему;
2) требования к условиям реализации основных образовательных программ, в том числе кадровым, финансовым, материально-техническим и иным условиям;
3) требования к результатам освоения основных образовательных программ.

Согласно Федеральному закону образовательная программа — это комплекс основных характеристик образования (объем, содержание, планируемые результаты), организационно-педагогических условий и в случаях, предусмотренных настоящим Федеральным законом, форм аттестации, который представлен в виде учебного плана, календарного учебного графика, рабочих программ учебных предметов, курсов, дисциплин (модулей), иных компонентов, а также оценочных и методических материалов. Основная образовательная программа (ООП) разрабатывается образовательным учреждением на основе ФГОС и определяет цели, задачи, планируемые результаты, содержание и организацию образовательного процесса. Требования к результатам освоения основных образовательных программ подготовки бакалавров (магистров, специалистов) в целом и их разделов формулируется в виде компетенций как в области профессиональной, так и социально-личностной деятельности [27].

1.2. Дидактические типы образовательных объектов

Дидактическим элементом в педагогике называют логически самостоятельную часть учебного материала, по своему объему и структуре соответствующую таким компонентам содержания как понятие, теория, закон, явление, факт, объект и др. [28, 29]. Наряду с этим понятием в педагогике существует также понятие *фонда оценочных средств* (*ФОС*), который представляет собой совокупность *контрольно-измерительных материалов* (*КИМ*) и методов их использования [29]. Примерами КИМ являются типовые задачи (задания), контрольные работы, тесты и др. С точки зрения обучения в высшей школе фонд оценочных средств позволяет оценить достижение запланированных в образовательной программе результатов обучения и уровень сформированности компетенций, заявленных в образовательной программе дисциплины. Таким образом, можно выделить следующие основные дидактические типы:

1) теоретическое (гипертекстовое) описание понятия;
2) вопрос с открытым ответом, проверяющий знание (понимание) теоретического описания понятия;
3) пример использования понятия при решении задач;
4) упражнение на умение применять изученное понятие при решении задач;
5) вопрос с закрытым ответом, проверяющий знание (понимание) теоретического описания понятия;
6) практическое задание, задание для самостоятельной работы, формирующие навыки использования изученного понятия при решении задач;
7) библиографическая ссылка.

Модель дидактической структуры контента в обязательном порядке должна поддерживать указанные типы.

1.3. Модульный подход при формировании образовательного контента

Модель дидактической структуры контента должна поддерживать модульный подход при формировании образовательного контента. С точки зрения дидактики *модуль* представляет собой наименьшую логическую единицу обучения, объединяющую в себе образовательный контент для формирования определенных знаний, умений и навыков. С методической точки зрения модуль соответствует определенному понятию. Каждый модуль имеет определенную *дидактическую схему*. Эта схема однозначно определяется набором атрибутов. Каждый атрибут представляет собой пару (идентификатор, дидактический тип). Значением атрибута может являться образовательный объект соответствующего типа. Далее значения атрибутов модуля будем называть *компонентами*. Некоторые компоненты модуля могут быть пустыми. С точки зрения стандарта SCORM, модулю соответствует агрегация [6].

1.4. Структурированные электронные энциклопедии

Модель дидактической структуры контента должна предусматривать возможность формирования дидактически структурированных *электронных энциклопедий* (хранилищ образовательных объектов) по различным предметным областям. *Дидактическая схема* (структура) энциклопедии однозначно определяется набором атрибутов, представляющих собой пару: (идентификатор, дидактический тип). Энциклопедия может включать в себя модули, которые имеют ту же дидактическую схему, что и энциклопедия. Между родственными модулями энциклопедии возможно установление связей «смотри также». Под родственными модулями понимаются модули, связанные отношениями «должен знать». В отличие от связей «должен знать» связи «смотри также» могут идти в любом направлении (от отца к сыну и от сына к отцу), а также могут идти через поколения (от отца к внуку и далее). Кроме этого, некоторые (возможно все) родственные модули могут в энциклопедии не иметь взаимных связей. Модули энциклопедии не обязаны образовывать иерархическую структуру.

1.5. Электронные учебные курсы

Модель дидактической структуры контента должна предусматривать возможность формирования дидактически структурированных *электронных учебных курсов*. *Дидактическая схема* электронного учебного курса однозначно определяется набором атрибутов, представляющих собой пару (идентификатор, дидактический тип). Электронный учебный курс может включать в себя модули, которые имеют ту же дидактическую схему, что и курс. Все модули курса должны образовывать упорядоченную древовидную структуру. В качестве корня дерева фигурирует модуль, включающий в себя аннотацию курса. Связи в направлении от корня к листьям задают отношение «следует изучить». Линейный порядок узлов одного уровня задает порядок их изучения. По умолчанию порядок изучения модулей курса определяется прямым обходом дерева: корень, узлы первого поддерева, узлы второго поддерева и так далее. Указанный порядок соответствует модели простого упорядочивания стандарта SCORM [9].

Модель дидактической структуры контента должна обеспечивать возможность переноса (копирования) модулей из электронной энциклопедии в электронный курс и из одного электронного курса в другой.

1.6. Совместимость с существующими стандартами в области электронного обучения

Модель дидактической структуры контента должна быть совместима со стандартом SCORM. Это предполагает совместимость с моделями, которые были описаны в работах [5, 6, 9]. Совместимость модели дидактической структуры контента со стандартом SCORM позволит осуществлять импорт и экспорт образовательного контента из одной системы управления обучением в другую.

2. Формальное описание СИД модели

Образовательный контент в СИД модели хранится в электронных учебных энциклопедиях (далее — просто энциклопедии). Основной структурной единицей энциклопедии является модуль. Модуль содержит образовательный контент, связанный с определенным понятием. С формальной точки зрения модуль представляет собой сложный образовательный объект, состоящий из следующих компонент:

1) `concept`: теоретическое (текстовое) описание понятия;
2) `open_question`: вопрос с открытым ответом для самопроверки понимания теоретического описания понятия;
3) `example`: пример решения задачи с использованием изучаемого понятия;
4) `exercise`: задание, проверяющее умение применять изученное понятие при решении задач;
5) `close_question`: вопрос с закрытым ответом для проверки освоения теоретического описания понятия;
6) `problem`: практическое задание, формирующее навыки использования изученного понятия при решении задач;
7) `bibitem`: библиографическая ссылка.

Отметим, что модулю в стандарте SCORM соответствует разделяемый объект контента SCO. Компоненты модуля соответствуют дидактическим типам, описанным в разделе 1.2. Значением компоненты модуля является коллекция, представляющая собой список (упорядоченную последовательность) элементов, имеющих соответствующий дидактический тип. Коллекция может не содержать ни одного элемента, то есть быть пустой. Основными операциями над коллекциями являются следующие:

1) `CreateCursor(collection)` — создание курсора;
2) `Fetch(cursor)` — возвращает элемент коллекции, на который указывает курсор;
3) `Set(cursor, value)` — присваивает указанное значение элементу коллекции, на который указывает курсор;
4) `Next(cursor)` — передвигает курсор на следующий элемент коллекции;
5) `Insert(cursor)` — добавляет новый элемент непосредственно перед элементом, на который указывает курсор;
6) `Delete(cursor)` — удаляет элемент коллекции, на который указывает курсор.

Тип компоненты определяет дидактический класс ассоциированного с ним объекта SCO стандарта SCORM. Каждый такой объект SCO включает в себя ассет в виде html-фрагмента и необязательный ассет в виде функции JavaScript. Первый ассет включает в себя образовательный контент в виде гипертекста. Второй ассет содержит программный

код на языке JavaScript, реализующий интерактивные действия по взаимодействию с учащимся. В соответствии со стандартом SCORM система управления обучением LMS запускает SCO в дочернем окне или фрейме окна LMS. Описание SCO осуществляется в соответствии с моделью метаданных LOM [33]. Таким образом, объекты СИД модели имеют возможность взаимодействия с учащимся, а также могут запрашивать и передавать данные от/в LMS [4], используя стандартный API [34].

Источником образовательного контента для наполнения курсов служат энциклопедии. Энциклопедии включают в себя образовательные модули, которые, в свою очередь, состоят из дидактических компонент. Каждая компонента характеризуется своим дидактическим типом. Пусть $\mathbb{T}$ — множество всех дидактических типов. Для каждого дидактического типа $T \in \mathbb{T}$ будем обозначать через $\mathfrak{D}_T$ множество всех конечных множеств (включая пустое), состоящих из значений этого типа.

Компонента w представляет собой четверку

$$w = \left\langle id_{component}, T, \xi, \lambda \right\rangle, \tag{1}$$

где $id_{component}$ — уникальный идентификатор компоненты, T — дидактический тип компоненты, $\xi \in \mathfrak{D}_T$ — коллекция, представляющая компоненту, λ — трудоемкость компоненты (в зачетных единицах).

Энциклопедия R представляет собой пару $\left\langle id_{encyclopedia}, M \right\rangle$, где $id_{encyclopedia}$ — уникальный идентификатор энциклопедии, M — коллекция (упорядоченный список) модулей энциклопедии.

Модуль μ представляет собой тройку:

$$\mu = \left\langle id_{module}, id_{encyclopedia}, \vec{w} \right\rangle, \tag{2}$$

где id_{module} — уникальный идентификатор модуля, $id_{encyclopedia}$ — энциклопедия модуля, $\vec{w} = (w_1, \ldots, w_7)$ — вектор компонент модуля в порядке, определенном выше.

В основе СИД модели лежит понятие компетенции, которая имеет следующее формальное определение.

Компетенция представляет собой четверку $\left\langle source, type, number, description \right\rangle$. Здесь *source* — уникальный идентификатор источника, содержащий перечень компетенций (в качестве такого идентификатора может выступать URL [32]); *type* — тип компетенции с возможными значениями из множества {УК, ОПК, ПК}, где УК обозначает универсальные компетенции, ОПК — общепрофессиональные компетенции, ПК — профессиональные компетенции [30, 31]; *number* — порядковый номер компетенции; *description* — наименование компетенции. В качестве примера можно привести следующую компетенцию:

<http://fgosvo.ru/uploadfiles/ProjFGOSVO3++/Bak3++/020302_B_3plus_21062017 .pdf, УК, 3, "Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде">.

Отметим, что роль первичного ключа в общей базе данных компетенций играют первые три атрибута: $(source, type, number)$.

Обозначим $\mathbb{K}$ — множество всех возможных компетенций, $\mathbb{K}_{univer}$ — подмножество универсальных компетенций, $\mathbb{K}_{genprof}$ — подмножество общепрофессиональных компетенций и $\mathbb{K}_{prof}$ — подмножество профессиональных компетенций. Множество компетенций обладает следующими свойствами:

$$\mathbb{K} = \mathbb{K}_{univer} \cup \mathbb{K}_{genprof} \cup \mathbb{K}_{prof}, \tag{3}$$

$$\mathbb{K}_{univer} \cap \mathbb{K}_{genprof} = \varnothing, \mathbb{K}_{univer} \cap \mathbb{K}_{prof} = \varnothing, \mathbb{K}_{genprof} \cap \mathbb{K}_{prof} = \varnothing. \tag{4}$$

На основе понятия компетенции вводится следующее формальное определение образовательного стандарта.

Образовательный стандарт S представляет собой семерку

$$S = \left\langle id_{standart}, K_{ungen}, \beta_{stud}, \beta_{pract}, \beta_{crt-l}, \beta_{crt-h}, \beta_{total} \right\rangle, \tag{5}$$

где $id_{standart}$ — уникальный идентификатор стандарта; $K_{ungen} \subset \left(\mathbb{K}_{univer} \cup \mathbb{K}_{genprof} \right)$ — множество универсальных и общепрофессиональных компетенций, входящих в стандарт; β_{stud} — минимальное количество зачетных единиц, отводимых на освоение учебных дисциплин; β_{pract} — минимальное количество зачетных единиц, отводимых на практики; β_{crt-l} и β_{crt-h} — соответственно нижняя и верхняя граница для зачетных единиц, отводимых на государственную итоговую аттестацию; β_{total} — общее количество зачетных единиц образовательной программы.

На основе образовательного стандарта формируется *образовательная программа*, включающая в себя совокупность учебных дисциплин.

Образовательная программа Q представляет собой четверку

$$Q = \left\langle id_{syllabus}, id_{standart}, K_{prof}, C \right\rangle, \tag{6}$$

где $id_{syllabus}$ — уникальный идентификатор образовательной программы, $id_{standart}$ — идентификатор образовательного стандарта, $K_{prof} \subset \mathbb{K}_{prof}$ — множество профессиональных компетенций, предусмотренных образовательной программой, C — множество курсов, входящих в образовательную программу. Определение курса будет дано ниже.

Электронный учебный курс γ представляет собой восьмерку вида:

$$\gamma = \left\langle id_{course}, id_{syllabus}, \beta_{lec}, \beta_{lab}, \beta_{out}, R, G, f \right\rangle, \tag{7}$$

где id_{course} — уникальный идентификатор курса; $id_{syllabus}$ — идентификатор образовательной программы, в рамках которой создается курс; β_{lec}, β_{lab}, β_{out} — трудоемкость лекций, лабораторных (практических) занятий и самостоятельной работы студента соответственно; R — энциклопедия курса, $G = \left\langle V, E \right\rangle$ — граф-план курса (см. определение ниже); $f : V \to R$ — однозначное инъективное отображение, сопоставляющее каждой вершине $\nu \in V$ граф-плана G некоторый модуль $\mu \in R$: $f(\nu) = \mu$.

Для задания иерархической структуры курса используется граф-план. *Граф-план* представляет собой связный ациклический граф $G = \left\langle V, E \right\rangle$ с выделенной вершиной $\overline{\nu}$, где $V = \left\{ v_l \mid l = 1,...,b \right\}$ — множество всех вершин граф-плана, $E = \left\{ e_p \mid p = 1,...,d \right\}$ — множество всех ребер граф-плана. Граф-план можно рассматривать как ориентированное дерево с корнем в вершине $\overline{\nu}$ в соответствии со следующим правилом:

Ребро, инцидентное вершинам ν *и* ν', *заменяем дугой от* ν *к* ν' *тогда и только тогда, когда простой путь от* ν *к* $\overline{\nu}$ *проходит через* ν', *то есть когда он имеет вид* $(\nu_0,\nu_1,\ldots,\nu_k)$, *где* $k>0,\ \nu_0=\nu,\ \nu_1=\nu',\ \nu_k=\overline{\nu}$.

Это означает, что направления дуг в ориентированном дереве полностью определяется положением корневой вершины $\overline{\nu}$, то есть их можно не указывать, если корень дерева обозначен явно.

Уровнем вершины $v\in V$ граф-плана $G=\langle V,E\rangle$ назовем длину простого пути от корня $\overline{\nu}$ до узла ν. Уровень корня $\overline{\nu}$ равен нулю. Множество вершин граф-плана, расположенных на уровне i от корня $\overline{\nu}$, назовем *i-тым ярусом* граф-плана и обозначим V_i.

Средняя трудоемкость λ_{avg} *модулей* курса γ вычисляется по следующей формуле

$$\lambda_{avg}=\frac{\sum\limits_{\mu\in f(V)}\sum\limits_{i=1}^{7}\mu.w_i.\lambda}{\left|V\right|}. \tag{8}$$

Дисбаланс курса определяется по формуле:

$$\delta=\frac{\sum\limits_{\mu\in f(V)}\left|\lambda_{avg}-\sum\limits_{i=1}^{7}\mu.w_i.\lambda\right|}{\left|V\right|}. \tag{9}$$

3. Основные операции СИД модели

Операции, предусмотренные в СИД модели, можно разделить на следующие группы:

1) операции над энциклопедиями;
2) операции над стандартами;
3) операции над образовательными программами;
4) операции над курсами и граф-планами.

Рассмотрим каждую из этих групп подробно.

3.1. Операции над энциклопедиями, модулями и компонентами

Для создания новой (пустой) энциклопедии используется операция:

```
encyclopedia = CREATE_ENCYCLOPEDIA().
```

Для работы с модулями энциклопедии используется внутренний курсор, указывающий на текущий модуль. Доступ к текущему модулю энциклопедии осуществляется следующим образом:

```
module = FETCH_MODULE(encyclopedia).
```

Перемещение внутреннего курсора на первый модуль энциклопедии осуществляется с помощью операции:

```
RESET_MODULE(encyclopedia).
```

Перемещение внутреннего курсора на следующий модуль энциклопедии осуществляется с помощью операции:

```
NEXT_MODULE(encyclopedia).
```

Перемещение внутреннего курсора на предыдущий модуль энциклопедии осуществляется с помощью операции:

```
PRIOR_MODULE(encyclopedia).
```

Для добавления в энциклопедию нового пустого модуля используется операция:

```
INSERT_MODULE(encyclopedia).
```

При этом текущим становится вновь добавленный модуль.

Удаление из энциклопедии текущего модуля выполняется с помощью операции:

DELETE_MODULE(encyclopedia).

После удаления текущим становится модуль, следующий за удаленным.

Для копирования содержимого модуля module1 в модуль module2 используется операция:

COPY_MODULE(module1, module2),

где module1 — модуль-донор, module2 — модуль-реципиент.

Доступ к коллекции элементов компоненты <COMPONENT> модуля module осуществляется следующим образом:

collection = <COMPONENT>_COLLECTION(module),

где в качестве <COMPONENT> фигурируют префиксы: CONCEPT, OPEN_QUESTION, EXAMPLE, EXERCISE, CLOSE_QUESTION, PROBLEM или BIBITEM. Работа с элементами коллекции, представляющей указанную компоненту, осуществляется с помощью операций над коллекциями, приведенными ранее.

Трудоемкость компоненты <COMPONENT> модуля module может быть получена аналогичным образом:

credit = <COMPONENT>_CREDIT(module).

3.2. Операции над стандартами

Для создания образовательного стандарта используется операция:

standart = CREATE_STANDART(description, min_stud, min_pract, min_crt, max_crt, total),

где min_stud — минимальное количество зачетных единиц, отводимых на освоение учебных дисциплин; min_pract — минимальное количество зачетных единиц, отводимых на практики; min_crt и max_crt — соответственно нижняя и верхняя граница для зачетных единиц, отводимых на государственную итоговую аттестацию; total — общее количество зачетных единиц образовательной программы. При создании стандарта неявно создаются курсоры для работы с коллекциями универсальных и общепрофессиональных компетенций. Доступ к атрибутам стандарта осуществляется с помощью операции GET_<atr>(standart), где в качестве <atr> указывается одно из следующих значений: DESCRIPTION, MIN_STUD, MIN_PRACT, MIN_CRT, MAX_CRT, TOTAL — названия соответствующих атрибутов стандарта.

Доступ к атрибуту универсальной компетенции осуществляется с помощью операции UC_GET_<atr>(standart), где в качестве <atr> указывается одно из следующих значений: NUMBER — порядковый номер компетенции; DESCRIPTION — наименование компетенции.

Изменение значения атрибута универсальной компетенции осуществляется с помощью операции UC_SET_<atr>(standart, value).

Перемещение внутреннего курсора на первую универсальную компетенцию стандарта осуществляется с помощью операции:

RESET_UC(standart).

Перемещение курсора на следующую универсальную компетенцию осуществляется с помощью операции:

NEXT_UC(standart).

Перемещение курсора на предыдущую универсальную компетенцию осуществляется с помощью операции:

```
PRIOR_UC(standart).
```

Для добавления в стандарт новой универсальной компетенции используется операция:

```
INSERT_UC_INTO_STANDART(standart, number, description),
```

где `number` — номер компетенции, `description` — наименование компетенции.

Удаление универсальной компетенции из стандарта выполняется с помощью операции:

```
DELETE_UC_FROM_STANDART(standart).
```

Аналогичным образом определяются операции для работы с общепрофессиональными компетенциями, только вместо суффикса `UC` используется суффикс `GPC`.

3.3. Операции над образовательными программами

Образовательная программа включает в себя (см. раздел 2) следующие атрибуты:

1) ссылка на образовательный стандарт;
2) перечень профессиональных компетенций, покрываемых образовательной программой;
3) перечень курсов, предусмотренных образовательной программой.

Для создания образовательной программы используется операция:

```
edu_prog = CREATE_EDU_PROG(standart),
```

где `standart` — указатель на стандарт, на основе которого разрабатывается образовательная программа. При создании образовательной программы неявно создаются курсоры для просмотра коллекции профессиональных компетенций и коллекции курсов.

Для получения указателя на стандарт, которые является базовым для образовательной программы `edu_prog` используется операция:

```
standart = GET_STANDART(edu_prog).
```

Операции для работы с профессиональными компетенциями образовательной программы определяются аналогично набору операций для работы с универсальными компетенциями стандарта, у которых вместо суффикса `UC` используется суффикс `PC`, а вместо указателя на стандарт `standart` — указатель на образовательную программу `edu_prog`.

Для работы с курсами образовательной программы используется внутренний курсор, указывающий на текущий курс.

Перемещение внутреннего курсора на начало списка учебных курсов образовательной программы осуществляется с помощью операции:

```
RESET_COURSE(edu_prog).
```

Перемещение курсора на следующий курс осуществляется с помощью операции:

```
NEXT_COURSE(edu_prog).
```

Перемещение курсора на предыдущий курс осуществляется с помощью операции:

```
PRIOR_COURSE(edu_prog).
```

Удаление текущего курса из образовательной программы выполняется с помощью операции:

```
DELETE_COURSE(edu_prog).
```

Для добавления в образовательную программу нового (пустого) курса используется операция:

```
INSERT_COURSE(edu_prog).
```

Получение указателя на текущий курс осуществляется с помощью операции:

```
course = FETCH_COURSE(edu_prog).
```

Доступ к атрибуту текущего курса осуществляется с помощью операции `COURSE_GET_<atr>(course)`, где в качестве `<atr>` указывается одно из следующих постфиксов: `TITLE` — название дисциплины; `LECTURE` — трудоемкость лекций; `LABORATORY` — трудоемкость лабораторных (практических) занятий; `OUTWORK` — трудоемкость самостоятельной работы студента.

Изменение значения атрибута текущего курса осуществляется с помощью операции `COURSE_SET_<atr>(course, value)`.

3.4. Операции над курсами и граф-планами

При создании нового курса автоматически создаются пустая *энциклопедия курса* и граф-план, состоящий из одной вершины (корня).

Доступ к энциклопедии курса осуществляется с помощью операции:

```
encyclopedia = GET_ENCYCLOPEDIA(course),
```

где `course` — курс соответствующей образовательной программы.

Доступ к корневой вершине граф-плана курса осуществляется с помощью операции:

```
graph_plan_root = GRAPH_PLAN(course),
```

где `course` — курс соответствующей образовательной программы. С каждой вершиной граф-плана ассоциируется коллекция дочерних вершин (возможно пустая), указатель на модуль из энциклопедии курса (возможно пустой) и коллекции номеров универсальных, общепрофессиональных и профессиональных компетенций. Для работы с дочерними вершинами для каждого узла граф-плана создается внутренний курсор, указывающий на текущую дочернюю вершину. Получение указателя на текущую дочернюю вершину узла `node` осуществляется с помощью операции:

```
child = FETCH_CHILD(node).
```

Перемещение внутреннего курсора на первую дочернюю вершину узла `node` осуществляется с помощью операции:

```
RESET_CHILD(node).
```

Перемещение внутреннего курсора на следующую дочернюю вершину узла `node` осуществляется с помощью операции:

```
NEXT_CHILD(node).
```

Перемещение курсора на предыдущую дочернюю вершину узла осуществляется с помощью операции:

```
PRIOR_CHILD(node).
```

Для добавления новой дочерней вершины узла `node` используется операция:

```
INSERT_CHILD(node).
```

Удаление дочерней вершины узла, на которую указывает курсор, выполняется с помощью операции:

```
DELETE_CHILD(node).
```

После удаления внутренний курсор устанавливается на дочернюю вершину, следующую за удаленной. Отметим, что при удалении узла граф-плана удаляется все поддерево, рекурсивно ассоциированное с удаляемым узлом.

Связывание узла граф-плана `node` с модулем `module` осуществляется следующим образом:

```
SET_LINK(node, module).
```

Для получения доступа к модулю энциклопедии курса, который ассоциирован с узлом граф-плана node, осуществляется следующим образом:

```
module = GET_LINK(node).
```

Добавление номера number в коллекцию номеров универсальных компетенций узла граф-плана node осуществляется следующим образом:

```
INSERT_UC_INTO_GP(node, number).
```

Удаление номера number из коллекции номеров универсальных компетенций узла граф-плана node осуществляется следующим образом:

```
DELETE_UC_FROM_GP(node, number).
```

Определение наличия универсальной компетенции с номером number в узле node граф-плана осуществляется следующим образом:

```
UC_ISEXIST(node, number).
```

Операции для работы с коллекциями номеров общепрофессиональных и профессиональных компетенций узла node граф-плана определяются аналогично (вместо суффикса UC используются суффиксы GPC и PC соответственно).

4. Анализ образовательных программ и электронных учебных курсов в СИД модели

В данном разделе рассматриваются основные алгоритмы по анализу образовательных программ и электронных учебных курсов на базе СИД модели.

4.1. Вычисление целостности курса

Перед использованием курса необходимо проверить его целостность. Под *целостностью* курса в СИД модели понимается тот факт, что каждый узел граф-плана *связан* с некоторым модулем. Под *коэффициентом целостности* курса понимается отношение количества *связанных* узлов к общему количеству узлов граф-плана курса. На рис. 2 представлена реализация рекурсивной функции вычисления количества связанных узлов поддерева с корнем root граф-плана курса.

```
function Amount_of_linked_nodes(root)
1)  amount = 0;
2)  if (GET_LINK(root) ≠ Null) then
3)       amount += 1;
4)  end if;
5)  RESET_CHILD(root);
6)  child = FETCH_CHILD(root);
7)  while (child ≠ Null)
8)       amount += Amount_of_linked_nodes(child);
9)       NEXT_CHILD (root);
10)      child = FETCH_CHILD(root);
11) end while;
12) return amount;
end function;
```

Рис. 2. Количество связанных узлов поддерева с корнем root

Количество связанных узлов граф-плана вычисляется в переменной amount. В строке 1 переменной amount присваивается начальное значение. В строках 2–4 значение переменной amount увеличивается на единицу, если узел root является связанным. В строке 5 выполняется операция RESET_CHILD, устанавливающая внутренний курсор на

начало списка дочерних узлов узла root. Операторы 6–11 добавляют в переменную amount количество связанных узлов в поддеревьях, соответствующих дочерним узлам. При этом используется рекурсивный вызов функции Amount_of_linked_nodes.

На рис. 3 представлена рекурсивная реализация функции вычисления общего количества узлов поддерева граф-плана с корнем root.

```
function Amount_of_nodes(root)
1)  amount = 1;
2)  RESET_CHILD(root);
3)  child = FETCH_CHILD(root);
4)  while (child ≠ Null))
5)       amount += Amount_of_nodes(child);
6)       NEXT_CHILD (root);
7)       child = FETCH_CHILD(root);
8)  end while;
9)  return amount;
end function;
```

Рис. 3. Количество узлов поддерева с корнем root

На рис. 4 представлена реализация функции вычисления коэффициента целостности курса.

```
function Consistency_rate(course)
1)  root = GRAPH_PLAN(course);
2)  return
    Amount_of_linked_nodes(root) / Amount_of_nodes(root);
end function;
```

Рис. 4. Коэффициент целостности курса

4.2. Оценка трудоемкости курса

При оценке трудоемкости курса используется понятие дидактического слоя. Под *дидактическим слоем* понимается совокупность всех компонент курса, имеющих одинаковый дидактический тип. Оценка трудоемкости электронного учебного курса в зачетных единицах осуществляется на основе оценки трудоемкости отдельных дидактических слоев:

1) Course_concept_credit(course) — трудоемкость слоя «Теория»;
2) Course_open_question_credit(course) — трудоемкость слоя «Вопросы с открытым ответом»;
3) Course_example_credit(course) — трудоемкость слоя «Примеры»;
4) Course_exercise_credit(course) — трудоемкость слоя «Упражнения»;
5) Course_close_question_credit(course) — трудоемкость слоя «Вопросы с закрытым ответом»;
6) Course_problem_credit(course) — трудоемкость слоя «Практические задания»;
7) Course_bibitem_credit(course) — трудоемкость слоя «Библиография».

Функция вычисления трудоемкости дидактического слоя в свою очередь реализуется с помощью соответствующей рекурсивной функции, вычисляющей трудоемкость слоя для поддерева граф-плана курса. На рис. 5 представлена реализация рекурсивной функции Subtree_concept_credit, вычисляющей трудоемкость слоя «Теория» для поддерева граф-плана курса.

```
function Subtree_concept_credit(root)
1)  module = GET_LINK(root);
2)  credit = CONCEPT_CREDIT(module);
3)  RESET_CHILD(root);
4)  child = FETCH_CHILD(root);
5)  while (child ≠ Null))
6)       credit += Subtree_concept_credit(child);
7)       NEXT_CHILD (root);
8)       child = FETCH_CHILD(root);
9)  end while;
10) return credit;
end function;
```

Рис. 5. Трудоемкость слоя «Теория» для поддерева с корнем `root`

Трудоемкость в зачетных единицах вычисляется в переменной `credit`. В строке 1 вычисляется ссылка на модуль, ассоциированный с текущим узлом граф-плана. Здесь предполагается, что все узлы граф-плана связаны с соответствующими модулями (это необходимо заранее проверить с помощью функции `Consistency_rate`, рассмотренной в разделе 4.1). В строке 2 переменной `credit` присваивается начальное значение. В строке 3 выполняется операция `RESET_CHILD`, устанавливающая внутренний курсор на начало списка узлов, являющихся дочерними по отношению к текущему. Операторы 4–9 добавляют в переменную `credit` трудоемкость «теории» дочерних узлов, используя рекурсивный вызов функции `Subtree_concept_credit`.

На рис. 6 представлен алгоритм вычисления трудоемкости слоя «теория» для всего курса в целом.

```
function Course_concept_credit(course)
1)  root = GRAPH_PLAN(course);
2)  credit = Subtree_concept_credit(root);
3)  return credit;
end function;
```

Рис. 6. Трудоемкость слоя «теория» для курса в целом

В строке 1 вычисляется ссылка на корень дерева граф-плана указанного курса. В строке 2 вычисляется общая трудоемкость слоя «теория». Как уже указывалось, для корректной работы функции `Course_concept_credit` предварительно необходимо проверить целостность курса с помощью функции `Consistency_rate`. Для остальных дидактических слоев алгоритмы вычисления трудоемкости строятся аналогично.

На рис. 7 представлен алгоритм вычисления трудоемкости курса в целом.

```
function Course_credit(course)
1)  credit = Course_concept_credit(course) +
         Course_open_question_credit(course) +
         Course_example_credit(course) +
         Course_exercise_credit(course) +
         Course_close_question_credit(course) +
         Course_problem_credit(course) +
         Course_bibitem_credit(course);
2)  return credit;
end function;
```

Рис. 7. Общая трудоемкость курса

4.3. Оценка сбалансированности курса

Величина дисбаланса курса, вычисляемая по формуле (9), определяет меру разброса трудоемкостей модулей и является одним из критериев оценки качества разработанного курса. Если все модули курса имеют примерно одинаковую трудоемкость, то величина дисбаланса близка к нулю. Данная величина вычисляется на основе средней трудоемкости модулей курса, для вычисления которой используется функция вычисления трудоемкости модуля, представленная на рис. 8.

```
function Module_credit(module)
1) credit = CONCEPT_CREDIT(module) +
         OPEN_QUESTION_CREDIT(module) +
         EXAMPLE_CREDIT(module) +
         EXERCISE_CREDIT(module) +
         CLOSE_QUESTION_CREDIT(module) +
         PROBLEM_CREDIT(module) +
         BIBITEM_CREDIT(module);
2) return credit;
end function;
```

Рис. 8. Трудоемкость модуля

На рис. 9 представлен алгоритм вычисления средней трудоемкости модулей для поддерева с корнем `root` граф-плана курса.

```
function Average_credit(root)
1)  amount = Amount_of_nodes(root);
2)  credit = Subtree_concept_credit(root) +
         Subtree_open_question_credit(root) +
         Subtree_example_credit(root) +
         Subtree_exercise_credit(root) +
         Subtree_close_question_credit(root) +
         Subtree_problem_credit(root) +
         Subtree_bibitem_credit(root);
3)  return credit/amount;
end function;
```

Рис. 9. Средняя трудоемкость модулей для поддерева с корнем `root`

Для корректной работы функции `Average_credit` предварительно необходимо проверить целостность курса с помощью функции `Consistency_rate`.

На рис. 10 представлена реализация функции, вычисляющей для поддерева с корнем `root` суммарное отклонение значений трудоемкости модулей от указанного значения. Данная функция используется в реализации функции вычисления дисбаланса курса, представленной на рис. 11.

В строке 1 вычисляется ссылка на модуль, ассоциированный с текущим узлом граф-плана. В строке 2 переменной `total_deviation` присваивается начальное значение — отклонение трудоемкости текущего модуля от указанного значения. В строках 3–9 рекурсивно вычисляется сумма отклонений трудоемкости модулей, ассоциированных с дочерними вершинами текущего узла `root`.

```
function Sum_of_deviations(root, value)
1)  root_module = GET_LINK(root);
2)  total_deviation = abs(value – Module_credit(root_module));
3)  RESET_CHILD(root);
4)  child = FETCH_CHILD(root);
5)  while (child ≠ Null))
6)       total_deviation += Sum_of_deviations(child, value);
7)       NEXT_CHILD(root);
8)       child = FETCH_CHILD(root);
9)  end while;
10) return total_deviation;
end function;
```

Рис. 10. Суммарное отклонение трудоемкостей модулей от значения value для поддерева с корнем root

```
function Course_imbalance(course)
1)  root = GRAPH_PLAN(course);
2)  average = Average_credit(root);
3)  amount = Amount_of_nodes(root)
4)  return Sum_of_deviations(root, average) / amount;
end function;
```

Рис. 11. Оценка дисбаланса курса

4.4. Соответствие образовательной программы стандарту

Образовательная программа *соответствует* стандарту, если выполняются следующие два условия:

1) *ограничения на трудоемкость*: трудоемкость освоения учебных дисциплин и трудоемкость практик образовательной программы должны быть не меньше соответствующих минимальных значений, указанных в стандарте, а сумма этих значений (с учетом трудоемкости итоговой аттестации) должна быть равна общему значению трудоемкости образовательной программы, указанной в стандарте;
2) *покрытие компетенций*: все универсальные и общепрофессиональные компетенции, указанные в стандарте, покрываются курсами образовательной программы.

Алгоритм проверки ограничений на трудоемкость представлен на рис. 12.

В строке 1 вычисляется ссылка на стандарт, на основе которого создана текущая образовательная программа. В строках 2 и 3 переменным study и practice присваиваются начальные значения. В строках 4–15 вычисляются трудоемкости учебных дисциплин курса и практики соответственно. В строках 16–18 осуществляется проверка ограничений на трудоемкость.

Проверка покрытия компетенций распадается на две подзадачи: 1) проверка покрытия универсальных компетенций; 2) проверка покрытия общепрофессиональных компетенций. Для решения первой подзадачи используется рекурсивная функция Course_UC_credit(root, uc_number), реализация которой приведена на рис. 13. Эта функция вычисляет в поддереве с корнем root суммарную трудоемкость модулей, связанных с узлами, в которых присутствует номер универсальной компетенции uc_number.

```
function Credit_constraints(edu_prog)
1)  standart = GET_STANDART(edu_prog);
2)  study = 0;
3)  practice = 0;
4)  RESET_COURSE(edu_prog);
5)  course = FETCH_COURSE(edu_prog);
6)  while (course ≠ Null))
7)       practice += Course_problem_credit(course);
8)       study += Course_concept_credit(course);
9)       study += Course_open_question_credit(course);
10)      study += Course_example_credit(course);
11)      study += Course_exercise_credit(course);
12)      study += Course_close_question_credit(course);
13)      NEXT_COURSE(edu_prog);
14)      course = FETCH_COURSE(edu_prog);
15) end while;
16) if ((study ≥ GET_MIN_STUD(standart))
    and (practice ≥ GET_MIN_PRACT(standart))
    and (study + practice ≤ GET_TOTAL(standart) -
    GET_MIN_CRT(standart))
    and (study + practice ≥ GET_TOTAL(standart) -
    GET_MAX_CRT(standart))) then
17)      return true;
18) end if;
19) return false;
end function;
```

Рис. 12. Проверка ограничений на трудоемкость

```
function Course_UC_credit(root, uc_number)
1)  module = GET_LINK(root);
2)  if (UC_ISEXIST(root, uc_number)) then
3)       credit = Module_credit(module);
4)  else
5)       credit = 0;
6)  end if;
7)  RESET_CHILD(root);
8)  child = FETCH_CHILD(root);
9)  while (child ≠ Null)
10)      credit += Course_UC_credit(child, uc_number);
11)      NEXT_CHILD (root);
12)      child = FETCH_CHILD(root);
13) end while;
14) return credit;
end function;
```

Рис. 13. Покрытие универсальной компетенции uc_number в поддереве с корнем root

В строке 1 вычисляется ссылка на модуль, ассоциированный с текущим узлом графплана. В строке 2 с помощью операции UC_ISEXIST осуществляется проверка присутствия универсальной компетенции uc_number в узле root. В зависимости от результата этой проверки, переменной credit присваивается общая трудоемкость соответствующего модуля либо ноль. В строках 7–13 рекурсивно вычисляется суммарная трудоемкость модулей, связанных с узлами, в которых присутствует универсальная компетенция uc_number.

Аналогичным образом реализуется функция Course_GPC_credit, вычисляющая трудоемкость для общепрофессиональной компетенции.

На рис. 14 представлена реализация функции проверки покрытия универсальных компетенций образовательного стандарта курсами образовательной программы.

```
function UC_coverage(edu_prog)
1)  standart = GET_STANDART(edu-prog);
2)  RESET_UC(standart);
3)  number = UC_GET_NUMBER(standart);
4)  while (number ≠ Null)
5)       credit = 0;
6)       RESET_COURSE(edu_prog);
7)       course = FETCH_COURSE(edu_prog);
8)       while (course ≠ Null))
9)            root = GRAPH_PLAN(course);
10)           credit += Course_UC_credit(root, number);
11)           NEXT_COURSE(edu_prog);
12)           course = FETCH_COURSE(edu_prog);
13)      end while;
14)      if (credit == 0) then
15)           return false;
16)      end if;
17)      NEXT_UC(standart);
18) end while;
19) return true;
end function;
```

Рис. 14. Проверка покрытия универсальных компетенций

В строке 1 вычисляется ссылка на стандарт, на основе которого реализована образовательная программа. В строках 2–18 для каждой универсальной компетенции стандарта вычисляется покрытие компетенции курсами образовательной программы. В строке 5 переменной credit присваивается начальное значение. В строках 6–13 вычисляется суммарное покрытие универсальной компетенции с номером number всеми курсами образовательной программы. Если суммарное покрытие универсальной компетенции с номером number равно нулю, тогда функция UC_coverage завершает свою работу с результатом false (строки 14–16). В строке 19 функция UC_coverage завершает свою работу с результатом true.

Реализация функции проверки покрытия общепрофессиональных компетенций реализуется аналогичным образом.

На рис. 15 представлена реализация функции оценки соответствия образовательной программы стандарту.

```
function Conformance_to_standart(edu_prog)
1)  if (Credit_constraints(edu_prog) == true)
and (UC_coverage(edu_prog) == true)
and (GPC_coverage(edu_prog) == true) then
2)       return true;
3)  else
4)       return true;
5)  end if;
end function;
```

Рис. 15. Соответствие образовательной программы стандарту

В строке 1 осуществляется проверка ограничений на трудоемкость и покрытия компетенций стандарта. При успешной проверке функция Conformance_to_standart завершает свою работу с результатом true (строка 2), иначе функция Conformance_to_standart завершает свою работу с результатом false (строка 4).

4.5. Целостность образовательной программы

Образовательная программа называется *целостностной*, если выполняются следующие условия:

1) все курсы образовательной программы являются целостными;
2) образовательная программа соответствует стандарту;
3) профессиональные компетенции образовательной программы покрываются курсами образовательной программы.

Функция вычисления целостности курса Consistency_rate(course) представлена в п. 4,1, функция вычисления соответствия образовательной программы стандарту Conformance_to_standart(edu_prog) представлена в п. 4,4. Алгоритм вычисления покрытия профессиональных компетенций курсами образовательной программы основан на рекурсивном алгоритме, реализация которого представлена на рис. 16.

```
function Course_PC_credit(root, pc_number)
1)  module = GET_LINK(root);
2)  if (PC_ISEXIST(root, pc_number)) then
3)       credit = Module_credit(module);
4)  else
5)       credit = 0;
6)  end if;
7)  RESET_CHILD(root);
8)  child = FETCH_CHILD(root);
9)  while (child ≠ Null)
10)      credit += Course_PC_credit(child, pc_number);
11)      NEXT_CHILD (root);
12)      child = FETCH_CHILD(root);
13) end while;
14) return credit;
end function;
```

Рис. 16. Покрытие профессиональной компетенции pc_number в поддереве с корнем root

В строке 1 вычисляется ссылка на модуль, ассоциированный с текущим узлом графплана. В строке 2 с помощью операции PC_ISEXIST осуществляется проверка присутствия профессиональной компетенции pc_number в узле root. В зависимости от результата этой проверки, переменной credit присваивается общая трудоемкость соответствующего модуля либо ноль. В строках 7–13 рекурсивно вычисляется суммарная трудоемкость модулей, связанных с узлами, в которых присутствует профессиональная компетенция pc_number.

На рис. 17 представлена реализация функции проверки покрытия профессиональных компетенций курсами образовательной программы.

В строках 1–17 для каждой профессиональной компетенции вычисляется покрытие компетенции курсами образовательной программы. В строке 4 переменной credit присваивается начальное значение. В строках 5–12 вычисляется суммарное покрытие профессиональной компетенции с номером number всеми курсами образовательной программы. Если суммарное покрытие профессиональной компетенции с номером number равно нулю,

```
function PC_coverage(edu_prog)
1)  RESET_PC(edu_prog);
2)  number = PC_GET_NUMBER(edu_prog);
3)  while (number ≠ Null)
4)       credit = 0;
5)       RESET_COURSE(edu_prog);
6)       course = FETCH_COURSE(edu_prog);
7)       while (course ≠ Null))
8)            root = GRAPH_PLAN(course);
9)            credit += Course_PC_credit(root, number);
10)           NEXT_COURSE(edu_prog);
11)           course = FETCH_COURSE(edu_prog);
12)      end while;
13)      if (credit == 0) then
14)           return false;
15)      end if;
16)      NEXT_PC(edu_prog);
17) end while;
18) return true;
end function;
```

Рис. 17. Покрытие профессиональных компетенций курсами образовательной программы

тогда функция PC_coverage завершает свою работу с результатом false (строки 13–15). В строке 18 функция PC_coverage завершает свою работу с результатом true.

Реализация функции вычисления целостности образовательной программы представлена на рис. 18.

```
function Consistency(edu_prog)
1)  RESET_COURSE(edu_prog);
2)  course = FETCH_COURSE(edu_prog);
3)  while (course ≠ Null))
4)       if (Consistency_rate(course) ≠ 1) then
5)            return false;
6)       end if;
7)       NEXT_COURSE(edu_prog);
8)       course = FETCH_COURSE(edu_prog);
9)  end while;
10) if (Correctness_eduprog(edu_prog) == true))
and (PC_coverage(edu_prog) == true) then
11)      return true;
12) else
13)      return false;
14) end if;
end function;
```

Рис. 18. Целостность образовательной программы

В строках 1–9 производится оценка целостности всех курсов образовательной программы. Если коэффициент целостности хотя бы одного курса отличается от 1, функция Consistency завершает свою работу с результатом `false`. В строках 10–14 осуществляется проверка соответствия образовательной программы стандарту и покрытия профессиональных компетенций образовательной программы курсами образовательной программы. В зависимости от этой проверки функция Consistency завершает свою работу с результатом `false` или `true`.

Заключение

В статье определены требования к модели дидактической структуры контента. Представлено формальное описание структурно-иерархической дидактической модели электронного обучения, отличительной особенностью которой является поддержка деления образовательных объектов на дидактические компоненты. Определены основные операции СИД модели, на основе которых построены алгоритмы анализа качественных характеристик образовательных программ и электронных учебных курсов. В рамках дальнейших исследований авторы предполагают разработать прототип программной системы для создания электронных учебных курсов на базе СИД модели, с помощью которой выполнить верификацию СИД модели.

Исследование выполнено при финансовой поддержке Правительства РФ в соответствии с Постановлением №211 от 16.03.2013 г. (соглашение № 02.A03.21.0011) и Министерства науки и высшего образования РФ (государственное задание 2.7905.2017/8.9).

Литература

1. Скиннер Б.Ф. Наука об учении и искусство обучения // Программированное обучение за рубежом: Сб. статей / Под ред. И.И. Тихонова. М.: Высшая школа. 1968. С. 32–46
2. Краудер Н.А. О различиях между линейным и разветвленным программированием // Программированное обучение за рубежом: Сб. статей / Под ред. И.И. Тихонова. М.: Высшая школа. 1968. С. 58–67.
3. Engelbart D.C. Toward Augmenting the Human Intellect and Boosting our Collective IQ // Communications of the ACM. 1995. Vol. 38, no. 8. P. 30–33. DOI: 10.1145/208344.208352.
4. IEEE 1484.11.1. Standard for Learning Technology — Data Model for Content Object Communication. 2004.
5. SCORM 2004 4th Edition. Run-Time Environment (RTE). Advanced Distributed Learning (ADL) Initiative. 2009.
6. SCORM 2004 4th Edition. Content Aggregation Model (CAM). Advanced Distributed Learning (ADL) Initiative. 2009.
7. Novak J.D., Canas A.J. The theory underlying concept maps and how to construct them. Technical Report IHMC CmapTools 2006–01 Rev 2008–01. Florida Institute for Human and Machine Cognition, USA. 2008. URL: http://cmap.ihmc.us/docs/theory-of-concept-maps (дата обращения: 10.07.2019).
8. Соловов А.В. Математические модели содержания и процессов электронного обучения // Телекоммуникации и информатизация образования. 2006. № 4. С. 20–37.
9. SCORM 2004 4th Edition. Sequencing and Navigation (SN).Advanced Distributed Learning (ADL) Initiative. 2009.
10. De-Marcos L., Pages C., Martinez J.J., Gutierrez J.A. Competency-Based Learning Object Sequencing Using Particle Swarms // Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2007 (Oct., 29–31, 2007). Vol. 2. P. 111–116. DOI: 10.1109/ICTAI.2007.14.
11. IEEE 1484.20.1. Standard for Learning Technology — Data Model for Reusable Competency Definitions. 2007.

12. Kristensen T., Lamo Y., Hinna K.R., Hole G.O. Dynamic Content Manager — A New Conceptual Model for E-Learning // Proceedings of the International Conference on Web Information Systems and Mining, WISM '09. Springer-Verlag, Berlin, Heidelberg, 2009. P. 499–507. DOI: 10.1007/978-3-642-05250-7_52.
13. Брызгалов П.А. Система «Ареола» — программная оболочка для создания электронных энциклопедий // Вычислительные методы и программирование. 2005. Т. 6, № 2. С. 22–26.
14. Брызгалов П.А,. Воеводин В.В., Воеводин Вл.В. О некоторых проблемах компьютеризации знаний // Научный сервис в сети Интернет: Тезисы докладов Всероссийск. науч. конф. (Новороссийск, 18–23 сентября 2000 г.). М.: Изд-во МГУ, 2000.
15. Cunningham W., Leuf B. The Wiki Way: Quick Collaboration on the Web. Addison-Wesley Professional, 2001. 464 p.
16. Курганская Г.С. Модель представления знаний и система дифференцированного обучения через Интернет на его основе // Известия Челябинского Научного Центра. 2000. Вып. 2. С. 84–88.
17. Курганская Г.С. Облачные технологии интернет-образования на основе KFS модели представления знаний // Вестник Бурятского государственного университета. 2013. № 9. С. 69–75.
18. Курганская Г.С. Развитие методик адаптивного обучения: опыт применения системы «ГЕКАДЕМ» // Интернет и современное общество: Труды XI Всероссийской объединенной конференции IMS–2008 (Санкт-Петербург, 28–30 октября 2008 г.). СПб.: Изд-во С.-Петерб. ун-та. 2008. С. 65–67.
19. Соловов А.В. Математическое моделирование содержания, навигации и процессов электронного обучения в контексте международных стандартов и спецификаций. Лекция-доклад // Труды Всероссийской научно-практической конференции с международным участием «Информационные технологии в обеспечении нового качества высшего образования» (Москва, НИТУ «МИСиС», 14–15 апреля 2010 г.). М.: Исследовательский центр проблем качества подготовки специалистов, 2010. 52 с.
20. Соловов А.В. Проектирование компьютерных систем учебного назначения. Гриф «Рекомендовано Госкомитетом по высшему образованию РФ к изданию». Самара: СГАУ, 1995. 138 с.
21. De-Marcos L., Pages C., Martinez J.J., Gutierrez J.A. Competency-Based Learning Object Sequencing Using Particle Swarms // Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence ICTAI 2007 (Oct., 29–31, 2007). Vol. 2. P. 111–116. DOI: 10.1109/ICTAI.2007.14.
22. Калитина В.В. Электронная энциклопедия как средство повышения уровня запоминания учебного материала // Вестник Красноярского государственного педагогического университета им. В.П. Астафьева. 2013. № 1 (23). С. 111–114.
23. Воеводин В.В. Открытая энциклопедия свойств алгоритмов AlgoWiki: от мобильных платформ до экзафлопсных суперкомпьютерных систем // Вычислительные методы и программирование: новые вычислительные технологии. 2015. Т. 16, № 1. С. 99–111. DOI: 10.26089/NumMet.v16r111.
24. Voevodin V., Antonov A., Dongarra Ja. AlgoWiki: an open encyclopedia of parallel algorithmic features // Supercomputing Frontiers and Innovations. 2015. Vol. 2, no. 1. P. 4–18. DOI: 10.14529/jsfi150101.

25. Силкина Н.С., Соколинский Л.Б. Обзор адаптивных моделей электронного обучения // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 4. С. 61–76. DOI: 10.14529/cmse160405.
26. Силкина Н.С., Соколинский Л.Б. Модели и стандарты электронного обучения // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2014. Т. 3, № 4. С. 5–35. DOI: 10.14529/cmse140401.
27. Приказ Минобрнауки России «Об утверждении порядка организации и осуществления образовательной деятельности по образовательным программам высшего образования — программам бакалавриата, программам специалитета, программам магистратуры» от 19 декабря 2013 года № 1367. URL: http://минобрнауки.рф/документы/5242 (дата обращения: 10.07.2019).
28. Новгородцева И.В. Педагогика с методикой преподавания специальных дисциплин. М. : ФЛИНТА, 2011. 378 с. URL: http://e.lanbook.com/book/2440.
29. Национальная педагогическая энциклопедия. URL: http://didacts.ru/.
30. Рудской А.И., Боровков А.И., Романов П.И., Колосова О.В. Общепрофессиональные компетенции современного российского инженера // Высшее образование в России. 2018. № 2. С. 5–18. URL: https://vovr.elpub.ru/jour/article/download/1267/1072 (дата обращения: 10.07.2019).
31. Сергеев А.Г. Компетентность и компетенции: монография. Владимир: Изд-во Владим. гос. ун-та, 2010. 107 с.
32. Berners-Lee T. Uniform Resource Locators “URL”: A Syntax for the Expression of Access Information of Objects on the Network. World Wide Web Consortium. 1994. URL: https://www.w3.org/Addressing/URL/url-spec.txt (дата обращения: 22.05.2019).
33. IEEE 1484.12.1. Draft Standard for Learning Object Metadata. 2002. URL: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf (дата обращения: 10.07.2019).
34. IEEE 1484.11.2. Standard for Learning Technology — ECMA Script Application Programming Interface for Content to Runtime Services Communication. 2003.

Силкина Надежда Сергеевна, старший преподаватель, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Соколинский Леонид Борисович, д.ф.-м.н., профессор, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse190405

STRUCTURAL-HIERARCHICAL DIDACTIC MODEL OF E-LEARNING

© 2019 N.S. Silkina, L.B. Sokolinsky

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: SilkinaNS@susu.ru, Leonid.Sokolinsky@susu.ru

Received: 22.10.2019

This article describes the original Structural-Hierarchical Didactic (SHD) model of e-learning. The model is based on a four-level methodological knowledge base. The first level includes a set of electronic educational encyclopedias in various fields of knowledge. The second level includes e-learning courses. The model supports the structuring of e-learning course on didactic components (vertical structuring) and levels of detail (horizontal structuring). The third level includes a set of syllabuses. The fourth level includes a set of a federal state educational standard of higher education (FSES HE). A distinctive feature of the SHD model is the division of educational objects into didactic components. This will allow, firstly, to automatically verify the didactic completeness of the electronic training course. Secondly, to include parts of one course in another without losing the didactic structure. Thirdly, to select a separate didactic layer from the electronic training course and, based on it, automatically generate specialized training materials: lecture notes, task collections, exam tests, etc. The basic operations of the SHD model are described, based on which analysis algorithms are proposed syllabuses and e-learning courses. In conclusion, a brief summary of the results and directions for further research is given.

Keywords: e-learning, course, e-learning model, learning object, didactic structure, SHD model.

FOR CITATION

Silkina N.S., Sokolinsky L.B. Structural-Hierarchical Didactic Model of E-Learning. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 4. pp. 56–83. (in Russian) DOI: 10.14529/cmse190405.

References

1. Skinner B.F. The science of learning and the art of teaching. Programmed study abroad: Col. of Articles / Ed. by I.I. Tikhonov. M.: Higher school. 1968. pp. 32–46. (in Russian)
2. Crowder N.A. On the differences between linear and branched programming. Programmed study abroad: Col. of Articles / Ed. by I.I. Tikhonov. M.: Higher school. 1968. pp. 58–67. (in Russian)
3. Engelbart D.C. Toward Augmenting the Human Intellect and Boosting our Collective IQ. Communications of the ACM. 1995. vol. 38, no. 8. pp. 30–33. DOI: 10.1145/208344.208352.
4. IEEE 1484.11.1. Standard for Learning Technology — Data Model for Content Object Communication. 2004.
5. SCORM 2004 4th Edition. Run-Time Environment (RTE). Advanced Distributed Learning (ADL) Initiative. 2009.
6. SCORM 2004 4th Edition. Content Aggregation Model (CAM). Advanced Distributed Learning (ADL) Initiative. 2009.
7. Novak J.D., Canas A.J. The theory underlying concept maps and how to construct them. Technical Report IHMC CmapTools 2006–01 Rev 2008–01. Florida Institute for Human

and Machine Cognition, USA. 2008. Available at: http://cmap.ihmc.us/docs/theory-of-concept-maps (accessed: 10.07.2019).
8. Solovov A.V. Mathematical models of the content and processes of e-learning. Telecommunications and informatization of education. 2006. no. 4. pp. 20–37. (in Russian)
9. SCORM 2004 4th Edition. Sequencing and Navigation (SN).Advanced Distributed Learning (ADL) Initiative. 2009.
10. De-Marcos L., Pages C., Martinez J.J., Gutierrez J.A. Competency-Based Learning Object Sequencing Using Particle Swarms. Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence ICTAI 2007 (Oct., 29–31, 2007). vol. 2. pp. 111–116. DOI: 10.1109/ICTAI.2007.14.
11. IEEE 1484.20.1. Standard for Learning Technology — Data Model for Reusable Competency Definitions. 2007.
12. Kristensen T., Lamo Y., Hinna K.R., Hole G.O. Dynamic Content Manager — A New Conceptual Model for E-Learning. Proceedings of the International Conference on Web Information Systems and Mining, WISM '09. Springer-Verlag, Berlin, Heidelberg, 2009. pp. 499–507. DOI: 10.1007/978-3-642-05250-7_52.
13. Bryzgalov P.A. System "Areola" — a software shell for creating electronic encyclopedias. Computational methods and programming. 2005. vol. 6, no. 2. pp. 22–26. (in Russian)
14. Bryzgalov P.A. Voevodin V.V., Voevodin Vl.V. On some problems of computerization of knowledge. Scientific service on the Internet: Proceedings of the the All-Russian Scientific Conference (Novorossiysk, September, 18, 2000). M.: Publishing House of Moscow State University, 2000. (in Russian)
15. Cunningham W., Leuf B. The Wiki Way: Quick Collaboration on the Web. Addison-Wesley Professional, 2001. 464 p.
16. Kurganskaya G.S. A knowledge representation model and a system of differentiated learning through the Internet. Bulletin of the Chelyabinsk Scientific Center. 2000. no. 2. pp. 84–88. (in Russian)
17. Kurganskaya G.S. Cloud technologies of Internet education based on the KFS knowledge representation model. Bulletin of the Buryat State University. 2013. no. 9. pp. 69–75. (in Russian)
18. Kurganskaya G.S. The development of adaptive learning methods: the experience of using HECADEM. Internet and modern society: Proceedings of the XI All-Russian Joint Conference IMS–2008 (St. Petersburg, October, 28–30, 2008). SPb.: Publishing House of Sankt Petersburg University. 2008. pp. 65–67. (in Russian)
19. Solovov A.V. Mathematical modeling of the content, navigation and e-learning processes in the context of international standards and specifications. Lecture report. Proceedings of the All-Russian Scientific and Practical Conference with International Participation "Information Technologies in Ensuring a New Quality of Higher Education" (Moscow, NUST "MISiS", April, 14–15, 2010). M.: Research Center for the Problems of Quality of Training of Specialists, 2010. 52 p. (in Russian)
20. Solovov A.V. Design of computer systems for educational purposes. Fingerboard "Recommended by the State Committee for Higher Education of the Russian Federation for publication". Samara: SSAU, 1995. 138 p. (in Russian)
21. De-Marcos L., Pages C., Martinez J.J., Gutierrez J.A. Competency-Based Learning Object Sequencing Using Particle Swarms. Proceedings of 19th IEEE International Conference on

Tools with Artificial Intelligence, ICTAI 2007 (Oct., 29–31, 2007). vol. 2. pp. 111–116. DOI: 10.1109/ICTAI.2007.14.

22. Kalitina V.V. Electronic encyclopedia as a tool of increasing the level of memorization of educational material. Bulletin of the Krasnoyarsk State Pedagogical University named after V.P. Astafieva. 2013. no. 1 (23). pp. 111–114. (in Russian)
23. Voevodin V.V. An open encyclopedia of the properties of algorithms AlgoWiki: from mobile platforms to exaflops supercomputer systems. Computational methods and programming: new computing technologies. 2015. vol. 16, no. 1. pp. 99–111. (in Russian) DOI: 10.26089/NumMet.v16r111.
24. Voevodin V., Antonov A., Dongarra Ja. AlgoWiki: an open encyclopedia of parallel algorithmic features. Supercomputing Frontiers and Innovations. 2015. vol. 2, no 1. pp. 4–18. DOI: 10.14529/jsfi150101.
25. Silkina N.S., Sokolinsky L.B. Survey of Adaptive E-learning Models. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 4. pp. 61–76. (in Russian) DOI: 10.14529/cmse160405.
26. Silkina N.S., Sokolinsky L.B. E-learning models and standards. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2014. vol. 3, no. 4. pp. 5–35. (in Russian) DOI: 10.14529/cmse140401.
27. Order of the Ministry of Education and Science of Russia "On the approval of the organization and implementation of educational activities for educational programs of higher education — undergraduate programs, specialty programs, master's programs" dated December 19, 2013. no. 1367. (in Russian)
28. Novgorodtseva I.V. Pedagogy with the methodology of teaching special disciplines. M.: FLINT, 2011. 378 p. (in Russian)
29. National pedagogical encyclopedia. Available at: http://didacts.ru/ (accessed: 10.07.2019). (in Russian)
30. Rudskoy A.I., Borovkov A.I., Romanov P.I., Kolosova O.V. General professional competences of a modern Russian engineer. Higher education in Russia, 2018. no. 2. pp. 5–18. Available at: https://vovr.elpub.ru/jour/article/download/1267/1072 (accessed: 10.07.2019). (in Russian)
31. Sergeev A.G. Competence and competencies: monograph. Vladimir: Publishing house Vladimir State University, 2010. 107 p. (in Russian)
32. Berners-Lee T. Uniform Resource Locators "URL": A Syntax for the Expression of Access Information of Objects on the Network. World Wide Web Consortium. 1994. Available at: https://www.w3.org/Addressing/URL/url-spec.txt (accessed: 10.07.2019).
33. IEEE 1484.12.1. Draft Standard for Learning Object Metadata. 2002. Available at: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf (accessed: 10.07.2019).
34. IEEE 1484.11.2. Standard for Learning Technology — ECMA Script Application Programming Interface for Content to Runtime Services Communication. 2003.

УДК 004.051 **DOI: 10.14529/cmse190406**

ПРОГРАММНАЯ ПОДДЕРЖКА МОДЕЛИ BSF

© 2019 Н.А. Ежова, Л.Б. Соколинский

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: EzhovaNA@susu.ru, Leonid.Sokolinsky@susu.ru

Поступила в редакцию: 08.11.2019

В статье описана программная поддержка модели параллельных вычислений BSF (Bulk Synchronous Farm), ориентированной на итерационные алгоритмы с высокой вычислительной сложностью, разрабатываемые для многопроцессорных систем с распределенной памятью экзафлопсного уровня производительности. Программная поддержка включает в себя параллельный BSF-каркас и веб-приложение BSF-Studio. Приведены определение и классификация параллельных программных каркасов. Описан новый BSF-каркас, разработанный согласно BSF-модели: его файловая структура и логика работы. BSF-каркас представляет собой совокупность файлов исходного кода на языке C++, используемых для быстрого создания BSF-программ. Приведено подробное описание проектирования и реализации веб-приложения BSF-Studio, которое представляет собой визуальный конструктор BSF-программ. BSF-Studio обеспечивает поэтапное заполнение проблемно-зависимых частей BSF-каркаса, а также компиляцию и запуск программного кода.

Ключевые слова: BSF, модель параллельных вычислений, параллельный каркас, BSF-Studio, веб-приложение.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Ежова Н.А., Соколинский Л.Б. Программная поддержка модели BSF // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 4. С. 84–99. DOI: 10.14529/cmse190406.

Введение

Полноценная модель параллельных вычислений [1] должна включать в себя спецификационный компонент определяющий, что есть корректная программа в контексте данной модели. Параллельная программа может выдавать правильный результат, однако быть некорректной по отношению к требованиям модели параллельных вычислений. Такого рода некорректность приводит к тому, что применение стоимостных метрик используемой модели будет выдавать неверные оценки. Для предотвращения таких ситуаций и для ускорения создания программных кодов на практике часто используют параллельные каркасы, разрабатываемые на основе требований соответствующей модели параллельных вычислений.

Параллельный каркас (*parallel skeleton*) в общем виде представляет собой программную конструкцию (или библиотеку функций), которая инкапсулирует некоторый шаблон параллельных вычислений и межпроцессорных коммуникаций [2]. В идеальном случае параллельный каркас представляет собой компилируемый, но не исполняемый код. Чтобы использовать параллельный каркас, программист должен добавить проблемно-зависимые типы и структуры данных, а также реализовать проблемно-зависимые функции. При этом дополнения программного кода могут делаться инкрементально с сохранением свойства компилируемости. Каркас берет на себя функцию объединения проблемно-зависимых операций с программным кодом, обеспечивающим их параллельное выполнение и коммуникации. Таким образом, абстрагирование от аспектов, связанных с параллелизмом, может значительно упростить и систематизировать разработку параллельных программ и помочь в эффективном использовании стоимостных метрик, преобразовании и оптимизации

исходного кода. В силу высокого уровня абстракции параллельные каркасы имеют концептуальную общность с функциями высшего порядка, применяемыми в функциональном программировании, а также с шаблонами объектно-ориентированного программирования, и многие конкретные каркасы используют эти механизмы.

В традиционных инструментах параллельного программирования используется подход, в соответствии с которым программист должен реализовывать низкоуровневое управление взаимодействием между параллельными процессами, используя механизмы, соответствующие используемой модели. Например, ядро MPI основано на обмене данными между процессами типа точка-точка с различными режимами синхронизации. Подобно этому, библиотеки потоковой обработки основаны на примитивах блокировки, условной синхронизации, атомарности и так далее. Этот подход является очень гибким, позволяя организовать взаимодействия любой сложности. Однако на практике многие параллельные алгоритмы основываются на хорошо понятных и хорошо известных шаблонах. Например, алгоритмы обработки изображений часто используют конвейерный параллелизм. Аналогичным образом, алгоритм для подбора параметра требует планирования параллельных процессов, которое не зависит от параметризованной задачи и диапазона значений исследуемого параметра.

Термин *параллельный каркас (algorithmic skeleton)* был введен Мюрреем Коулом (Murray Cole) в работе [3]. Появление этого термина явилось результатом осознания того факта, что многие параллельные приложения используют одни и те же внутренние коммуникационные шаблоны. Каркасный подход предполагает, что подобные шаблоны должны быть обобщены в виде программной конструкции или библиотеки функций, которые отвечают за реализацию скрытого управляющего «каркаса», оставляя программисту реализацию проблемно-зависимых функций, обеспечивающих решение конкретной задачи. Например, *конвейерный каркас* потребует от программиста реализацию операций, выполняемых на каждом шаге конвейера, в то время как каркас будет отвечать за распределение конвейерных операций между различными процессорами, межпроцессорные коммуникации, непрерывность работы конвейера и так далее. Аналогичным образом, в *каркасе для подбора параметра* программист должен будет предоставить код для одиночного прогона параметризованной задачи и требуемый диапазон значений параметра. Каркас будет определять (и, возможно, динамически корректировать) количество используемых рабочих, гранулярность распределенных вычислительных заданий, коммуникационные механизмы и отказоустойчивость. На более высоком уровне абстракции использование *каркаса «разделяй-и-властвуй»* может потребовать от программиста описания способов разделения задачи на независимые подзадачи и последующего объединения результатов их работы, решения вопроса, является ли исходная задача делимой соответствующим образом, и реализации алгоритмов решения неделимых подзадач. Каркас в этом случае будет отвечать за все остальное, начиная с вопроса, стоит ли вообще использовать параллелизм, и кончая такими деталями, как динамическая диспетчеризация, гранулярность и коммуникации.

Подобный высокоуровневый подход меняет процесс разработки программы в нескольких отношениях. Во-первых, он освобождает пользователя от нетривиальной задачи принятия правильных проектных решений на основе многочисленных, взаимосвязанных низкоуровневых особенностей конкретного приложения и конкретной вычислительной плат-

формы. Во-вторых, использование отлаженного и апробированного параллельного каркаса существенно уменьшает вероятность появления ошибок в результирующем коде, что особенно важно для сложных задач, решаемых на вычислительных системах с массовым параллелизмом, где традиционная отладка является слишком сложной. В-третьих, использование параллельного каркаса дает гарантию параллельной эффективности вместо апостериорной оценки производительности, в результате которой, возможно, придется отказаться от трудоемкой параллельной программы по причине ее фатальной неэффективности. В-четвертых, каркасы предоставляют семантически обоснованные методы для сборки и оптимизации программного кода, открывающие новые перспективы в разработке программного обеспечения (в том числе, в повторном использовании программных кодов). И последнее, но не менее важное: повышение уровня абстракции, заключающееся в переходя от частного к общему, дает новое понимание фундаментальных принципов параллельного программирования [2].

В зависимости от функциональности параллельные каркасы могут быть разделены на следующие три типа [4]:

– *каркасы с параллелизмом данных* (*data-parallel*), применяемые для выполнения одинаковых операций над элементами больших однородных структур данных;
– *каркасы с параллелизмом задач* (*task-parallel*), оперирующие задачами и обеспечивающие их синхронизацию в соответствии со связями по данным;
– *резолюционные* (*resolution*) *каркасы*, определяющие общий метод распараллеливания для семейства схожих задач.

Таксономия базовых параллельных каркасов приведена в табл. 1.

– *Map* является наиболее важным базовым каркасом с параллелизмом данных. Его природа тесно связана с функциональными языками программирования. Семантика каркаса *map* заключается в том, что некоторая функция может быть одновременно применена ко всем элементам списка в целях распараллеливания вычислений. Параллелизм данных организуется следующим образом: список разбивается на подсписки равной длины по числу процессорных узлов, и каждый процессорный узел обрабатывает свой подсписок параллельно с другими узлами. Обменов данными между процессорными уздами при этом не происходит. По завершению обработки полученные результаты объединяются в общий результат. Таким образом, каркас *map* соответствует схеме параллельной обработки SIMD (single instruction, multiple data) [5].

Таблица 1

Таксономия базовых параллельных каркасов

Тип каркаса	**Область применения**	**Примеры элементарных каркасов**
Параллелизм данных	Структуры данных	map, fork, reduce
Параллелизм задач	Задачи	farm, pipe

Fork работает подобно *map*. Разница в том, что вместо применения одной и той же функции ко всем элементам списка, к каждому элементу применяется своя функция. Таким образом, каркас *fork* соответствует схеме параллельной обработки MIMD (multiple instruction, multiple data) [5].

Reduce используется для вычисления инфиксных операций над списком путем деления элементов списка на пары и применения к каждой паре указанной инфиксной операции. Полученные результаты образуют список в два раза меньшей длины, к которому снова применяется *reduce*, и так да тех пор, пока не будет получено итоговое интегральное значение. Таким образом, в отличие от *map*, каркас *reduce* требует обменов данными между процессорными узлами для агрегирования частичных результатов.

Farm реализует параллельное выполнение задач по схеме мастер-рабочие. Этот каркас также называют *bag of tasks* (пакет задач). *Farm* обеспечивает параллельное выполнение независимых подзадач на узлах-рабочих и слияние полученных ими результатов в общий результат на узле-мастере, который финализирует решение задачи.

Pipe реализует конвейерный параллелизм. Этот каркас целесообразно использовать, когда одна и та же задача должна решаться для многих наборов входных данных. В этом случае решение задачи разбивается на последовательные стадии, количество которых называется длиной конвейера. Каждая стадия выполняется отдельным процессорным узлом. Параллелизм достигается за счет того, что одновременно с выполнением i-той стадии для набора данных $x^{(j)}$ выполняется стадия $(i-1)$ для набора $x^{(j+1)}$. Каркас *pipe* обеспечивает синхронизацию выполнения стадий и передачу промежуточных результатов от одной стадии другой. Заметим, что в общем случае достаточно иметь каркас *pipe* с фиксированной длиной, так как каркасы могут вкладываться друг в друга.

Статья имеет следующую структуру. В разделе 1 описывается структура параллельного BSF-каркаса на языке C++, используемого для быстрого создания BSF-программ. В разделе 2 описывается визуальный конструктор BSF-программ.

1. Параллельный каркас для модели BSF

Модель параллельных вычислений BSF (Bulk Synchronous Farm) была предложена в работах [6, 7]. Модель BSF является расширением модели BSP и ориентирована на итерационные алгоритмы с высокой вычислительной сложностью, разрабатываемые для многопроцессорных систем с распределенной памятью экзафлопсного уровня производительности. Отличительной особенностью модели BSF от других известных моделей параллельных вычислений является возможность оценки границы масштабируемости алгоритма на ранних этапах его разработки.

В данном разделе описывается параллельный BSF-каркас на языке C++, предназначенный для быстрого создания параллельных BSF-программ для кластерных вычислительных систем. Для организации обменов сообщениями между процессорными узлами BSF-каркас использует программный интерфейс MPI [8–10]. BSF-каркас спроектирован таким образом, что он допускает поэтапное заполнение проблемно-зависимых частей, и при этом компилируется на всех этапах разработки. Исходные тексты BSF-каркаса свободно доступны в сети Интернет по адресу https://github.com/nadezhda-ezhova/BSF-MR.

1.1. Файловая структура каркаса

BSF-каркас состоит из двух групп файлов:

1) файлы с префиксом «BSF» содержат проблемно-независимый код и не подлежат изменениям со стороны пользователя;
2) файлы с префиксом «Problem» предназначены для заполнения пользователем проблемно-зависимых частей программы.

Описания всех файлов исходного кода приведены в табл. 2. Граф зависимостей файлов BSF-каркаса по включению с помощью директивы #include приведен на рис. 1. Серым закрашены файлы, в которых пользовательские изменения не допускаются; узорной штриховкой обозначены файлы с предопределенными заголовками определений, тело которых должен заполнить пользователь; белое заполнение соответствует файлам, полностью заполняемым пользователем. Порядок заполнения BSF-каркаса приведен в файле ReadMe, доступном по адресу https://github.com/nadezhda-ezhova/BSF-MR.

Таблица 2

Файлы исходного кода BSF-каркаса

Файл	**Описание**
Проблемно-независимый код	
BSF-Code.cpp	Реализации головной функции main и всех проблемно-независимых функций
BSF-Data.h	Проблемно-независимые переменные и структуры данных
BSF-Forwards.h	Предописания проблемно-независимых функций
BSF-Include.h	Включение проблемно-независимых библиотек (iostream, mpi.h, omp.h)
BSF-ProblemFunctions.h	Предописания предопределенных функций с проблемно-зависимой реализацией
BSF-Types.h	Определения проблемно-независимых типов
Проблемно-зависимый код	
Problem-bsfCode.cpp	Реализация предопределенных проблемно-зависимых функций
Problem-bsfParameters.h	Предопределенные проблемно-зависимые параметры
Problem-bsfTypes.h	Предопределенные проблемно-зависимые типы
Problem-Data.h	Проблемно-зависимые переменные и структуры данных
Problem-Forwards.h	Предописания проблемно-зависимых функций
Problem-Include.h	Включение проблемно-зависимых библиотек
Problem-Parameters.h	Проблемно-зависимые параметры
Problem-Types.h	Проблемно-зависимые типы

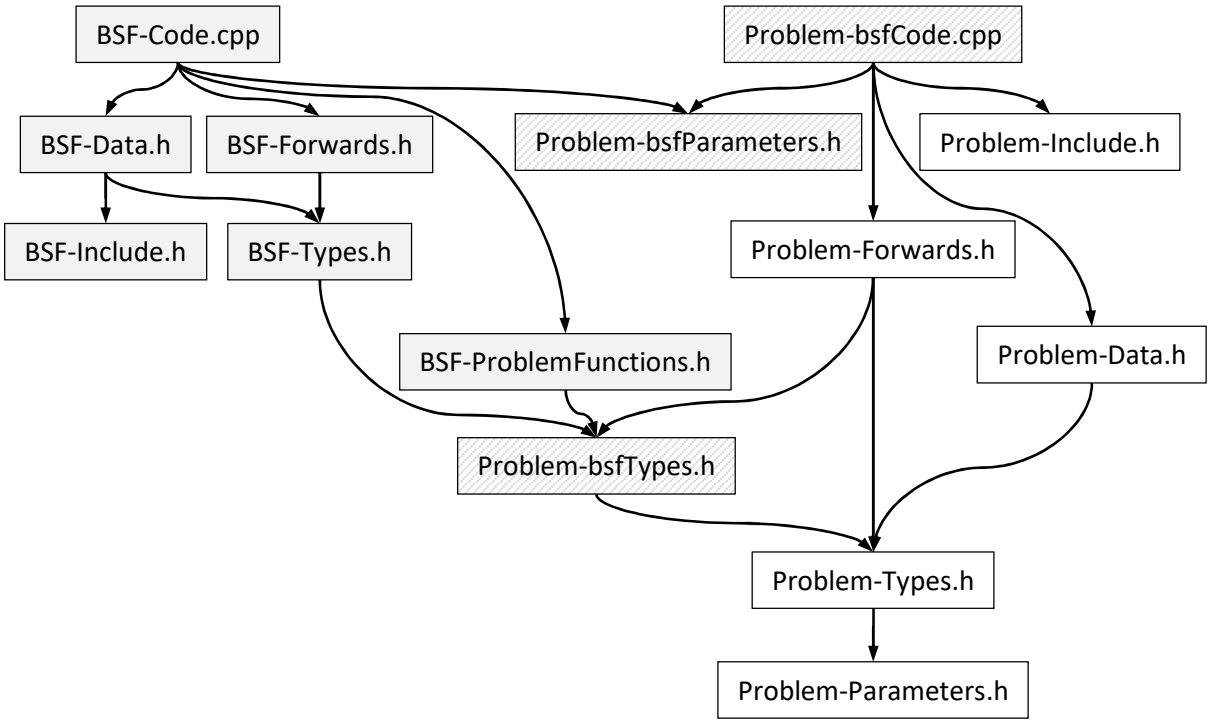


Рис. 1. Граф зависимостей файлов BSF-каркаса по включению #include

Файл ***Problem-bsfParameters.h*** содержит следующие предопределенные константы (#define), используемые в BSF-Code.cpp, значения которых устанавливает пользователь:

- PP_BSF_PRECISION: задает количество значащих цифр (ширину поля вывода) для чисел с плавающей точкой;
- PP_BSF_ITER_OUTPUT: если указанный #define определен, то будет выполняться вывод результатов итераций с определенным шагом, задаваемым константой PP_BSF_TRACE_COUNT;
- PP_BSF_TRACE_COUNT: задает частоту вывода результатов итераций, например, если указано значение 10, то будет выводиться результат каждой десятой итерации (если определен #define PP_BSF_ITER_OUTPUT);
- PP_BSF_OMP: если указанный #define определен, то будет включена #pragma omp parallel for перед циклом, реализующим функцию Map;

PP_BSF_NUM_THREADS: указывает количество потоков управления, в директиве #pragma omp parallel (если константа *PP_BSF_NUM_THREADS* не определена, то запускается количество потоков, определенное по умолчанию).

Файл ***Problem-bsfTypes.h*** включает в себя заголовки описаний (описания с пустым телом) трех предопределенных структурных типов:

- *PT*_bsf_data_T: описывает структуру данных, передаваемых каждому рабочему для выполнения очередной итерации (может содержать текущее приближение и другие параметры);
- PT_bsf_mapElem_T: описывает структуру элемента списка «Map»;
- PT_*bsf_reduceElem_T*: описывает структуру элемента списка «Reduce».

Файл ***BSF-ProblemFunctions.h*** включает в себя предописания (форварды) следующих предопределенных пользовательских функций, которые должны быть реализованы в файле *Problem-bsfCode.cpp*:

- PC_bsf_AssignListSize(int* listSize): присваивает переменной *listSize длину списка «Map» (совпадает с длиной списка «Reduce»);
- PC_bsf_CopyData(PT_bsf_data_T* dataIn, PT_bsf_data_T* dataOut): копирует данные из структуры *dataOut в структуру *dataIn;
- PC_bsf_IterOutput(PT_bsf_reduceElem_T* reduceResult, int count, PT_bsf_data_T data, int iterCount, double elapsedTime): выводит результаты итерации, используя в качестве параметров reduceResult — результат выполнения функции Reduce над всем списком; count — количество элементов, участвовавших в Reduce; data — задание, выполненное на данной итерации; iterCount — количество выполненных итераций; elapsedTime — общее время, затраченное на решение задачи.
- PC_bsf_Init(bool* success): выделяет память для проблемно-зависимых структур данных и выполняет их инициализацию; если необходимую память выделить не удалось, переменной *success должно быть присвоено значение false;
- PC_bsf_MapF(PT_bsf_mapElem_T* mapElem, PT_bsf_reduceElem_T* reduceElem, PT_bsf_data_T* data, int* success): вычисляет функцию F, являющуюся параметром оператора Map, используя аргументы *mapElem — элемент списка «Map», над которым выполняется функция F; *reduceElem — элемент списка «Reduce», которому должно быть присвоено значение функции F; *data — задание, содержащее текущее приближение; параметру *success должно быть присвоено значение 0, если полученный элемент списка «Reduce» должен игнорироваться при выполнении операции Reduce.
- PC_bsf_ParametersOutput(int numOfWorkers, PT_bsf_data_T data): выводит начальные параметры задачи, используя аргументы numOfWorkers — количество процессов-рабочих и data — начальное задание, содержащее начальное приближение;
- PC_bsf_ProblemOutput(PT_bsf_reduceElem_T* reduceResult, int count, PT_bsf_data_T data, int iterCount, double t): выводит конечные результаты работы программы, используя аргументы *reduceResult — результат выполнения оператора Reduce, count количество элементов «суммированных» при выполнении оператора Reduce, data — последнее задание (последнее приближение), t — общее время работы программы;
- PC_bsf_ProcessResults(bool* exit, PT_bsf_reduceElem_T* reduceResult, int count, PT_bsf_data_T* data): обрабатывает результаты, полученные в результате выполнения очередной итерации, используя аргументы *reduceResult — результат выполнения оператора Reduce, count количество элементов «суммированных» при выполнении оператора Reduce, data — последнее задание (последнее приближение); если вычисления необходимо продолжить, переменной *exit присваивается значение true, в противном случае — false;
- PC_bsf_ReduceF(PT_bsf_reduceElem_T* x, PT_bsf_reduceElem_T* y, PT_bsf_reduceElem_T* z): реализует функцию, являющуюся аргументом оператора Reduce, по формуле $z := x \oplus y$.
- PC_bsf_SetInitApproximation(PT_bsf_data_T* data): записывает в *data начальное задание (начальное приближение);
- PC_bsf_SetMapSubList(PT_bsf_mapElem_T* subList, int count, int offset, bool* success): заполняет подсписок списка «Map», используя аргументы *subList — указа-

тель на первый элемент подсписка, count — длина подсписка, offset — сдвиг, относительно начала списка; если у элемента списка «Map» имеются поля типа указатель, то необходимо выделить память под вектор, матрицу или другую структуру; если обнаружена нехватка памяти, переменной **success* необходимо присвоить значение *false*.

Файл ***BSF-Types.h*** содержит определения двух проблемно-независимых структурных типов: *BT_order_T* и *BT_extendedReduceElem_T*. Структурный тип *BT_order_T* определяет структуру задания, пересылаемого мастером рабочим, и включает в себя два поля: *data* типа *PT_bsf_data_T*, определяемого в файле *Problem-bsfTypes.h* (см. стр. 89), и *exit* типа *char*. Если поле *exit* содержит двоичное значение 1, то процессы-рабочие должны завершить свою работу. Структурный тип *BT_extendedReduceElem_T* определяет структуру элемента расширенного списка «*Reduce*» и включает в себя два поля: *elem* типа *PT_bsf_reduceElem_T*, определяемого в файле *Problem-bsfTypes.h* (см. выше), и *counter* типа *int*. До выполнения оператора *Reduce* в поле *counter* может находиться значение 1, либо 0. Значение 0 означает, что данный элемент должен быть проигнорирован при выполнении оператора *Reduce*. Значение 1 означает, что данный элемент должен быть учтен при выполнении оператора *Reduce*. После выполнения оператора *Reduce* над списком (подсписком) в поле *counter* результирующего элемента заносится количество учтенных исходных элементов.

Файл ***BSF-Data.h*** содержит определения проблемно-независимых переменных и структур данных. В число переменных входят следующие:

- *BD_*listSize: длина списка «Map» (совпадает с длиной списка «Reduce»);
- BD_size: количество MPI-процессов;
- BD_rank: номер текущего MPI-процесса;
- BD_masterRank: номер процесса-мастера (равен $BD\_size - 1$);
- BD_numOfWorkers: количество процессов-рабочих (равно $BD\_size - 1$);
- BD_elemsPerWorker: количество элементов списка, приходящихся на одного рабочего (равно $\lfloor BD\_listSize \,/\, BD\_numOfWorkers \rfloor$;
- BD_tailLength: длина остатка списка после деления на число рабочих (равно $BD\_listSize - BD\_elemsPerWorker \cdot BD\_numOfWorkers$);
- BD_exit: индикатор завершения вычислений;
- BD_success: индикатор успешного выполнения инициализации;
- BD_t: время, затраченное на вычисления (не учитываются ввод/вывод данных и инициализация);
- BD_iterCount: количество выполненных итераций.

В число структур данных входят следующие:

- *BD_data*: структура, в которой формируется данные, необходимые рабочим для выполнения очередной итерации;
- BD_mapSubList: указатель на подсписок, обрабатываемый рабочим (используется только процессами-рабочими);
- BD_extendedReduceList: указатель на расширенный список «Reduce»;
- BD_extendedReduceResult_P: указатель на структуру, в которой вычисляется результирующий элемент при выполнении оператора Reduce над расширенным списком «Reduce»;
- BD_order: указатель на структуру, в которой формируется задание для рабочего;

- BD_status: указатель на массив системных MPI-структур «MPI_Status», содержащих параметры сообщений для каждого рабочего;
- BD_request: указатель на массив системных MPI-переменных, содержащий идентификаторы асинхронных обменов по числу процессов-рабочих (используется только процессом-мастером);
- BD_subListSize: указатель на массив целых чисел, содержащий размеры подсписков для всех рабочих;
- BD_offset: указатель на массив целых чисел, содержащий для каждого рабочего сдвиг его подсписка относительно начала общего списка.

Файл ***BSF-Code.cpp*** включает в себя реализацию головной функции main и реализации следующих проблемно-независимых функций:

- BC_MpiRun выполняет инициализацию MPI и соответствующих переменных;
- BC_Init выделяет необходимую память для структур данных, инициализирует переменные и заполняет список «Map» исходными данными;
- BC_Master реализует процесс, выполняемый на узле-мастере;
- BC_Worker реализует процесс, выполняемый на узлах-рабочих;
- BC_MasterMap реализует шаг «Map» на узле-мастере;
- BC_MasterReduce реализует шаг «Reduce» на узле-мастере;
- BC_MpiRun осуществляет инициализацию MPI;
- BC_WorkerMap реализует шаг «Map» на узлах-рабочих;
- BC_WorkerReduce реализует шаг «Reduce» на узлах-рабочих;
- BC_ProcessExtendedReduceList обрабатывает указанную часть списка «Reduce».

Файл ***BSF-Include.h*** подключает с помощью директивы #include необходимые системные библиотеки. Файл ***BSF-Forwards.h*** содержит предописания функций, реализуемых в файле *BSF-Code.cpp.* Файл ***Problem-bsfCode.cpp*** содержит реализации проблемно-зависимых функций, предописания которых находятся в файле *BSF-ProblemFunctions.h* (см. стр. 89). Файл ***Problem-Include.h*** подключает с помощью директивы #include системные библиотеки, необходимые для реализаций проблемно-зависимых функций. Файл ***Problem-Forwards.h*** содержит предописания пользовательских функций, используемых для реализации предопределенных проблемно-зависимых функций. Файл ***Problem-Types.h*** содержит определения пользовательских типов данных. Файл ***BSF-Types Problem-Data.h*** содержит определения пользовательских глобальных переменных и структур данных. Файл ***Problem-Parameters.h*** содержит определения проблемно-зависимых констант.

1.2. Логика работы каркаса

Кратко опишем общую логику работы каркаса. ***Головная функция main*** вызывает функцию *BC_MpiRun*, которая выполняет инициализацию MPI и определяет значения переменных:

- *BD_size*: количество запущенных MPI-процессов (не может быть меньше двух);
- BD_rank: номер собственного MPI-процесса.

Затем вызывается функция *BC_Init*, которая выделяет необходимую память и определяет значения проблемно-независимых глобальных переменных и структур данных, определенных в файле *BSF-Data.h* (см. стр. 91). Если хотя бы в одном MPI-процессе не

удалось выделить необходимую память, процесс-мастер выводит в глобальный поток *cout* диагностическое сообщение «*Error: PC_bsf_Init failed (not enough memory)!*», и все MPI-процессы заканчивают свою работу. Если выделение памяти и инициализация прошли успешно, то процесс с номером *BD_masterRank* выполняет функцию *BC_Master*, а остальные процессы выполняют функцию *BC_Worker*.

Функция BC_Master реализует алгоритм работы мастера. Сначала выводятся параметры задачи с помощью функции *PC_bsf_ParametersOutput*. Затем организуется основной итерационный цикл. В ходе каждой итерации выполняются следующие действия:

1) вызывается функция *BC_MasterMap*, пересылающая всем рабочим в асинхронном режиме (с помощью функции *MPI_Isend*) задание для очередной итерации;
2) вызывается функция *BC_MasterReduce*, получающая от всех рабочих в асинхронном режиме (с помощью функции *MPI_Irecv*) результат выполнения оператора Reduce на подсписках;
3) вызывается предопределенная проблемно-зависимая функция *PC_bsf_ProcessResults*, проверяющая критерий завершения, вычисляющая очередное приближение и формирующая задание для следующей итерации;
4) если определен *#define PP_BSF_ITER_OUTPUT*, выводятся результаты итерации.

Основной итерационный цикл завершается, когда в переменную *BD_exit* функция *PC_bsf_ProcessResults* поместит значение *false*. В этом случае выполняется функция *BC_MasterMap* с параметром *false*, приводящая к завершению процессов-рабочих. После этого происходит вывод результатов с помощью предопределенной проблемно-зависимой функции *PC_bsf_ProblemOutput*, после чего функция *BC_Master* завершает свою работу.

Функция BC_Worker, реализует алгоритм работы рабочего, выполняя в цикле последовательно две функции: *BC_WorkerMap* и *BC_WorkerReduce*. Выход из этого цикла происходит, когда функция *BC_WorkerMap* вернет значение *false*. Функция *BC_WorkerMap* выполняет следующие действия:

1) в синхронном режиме (с помощью функции *MPI_Recv*) получает от мастера задание для выполнения очередной итерации;
2) если поле *exit* в полученном задании содержит значение *true*, происходит выход из функции;
3) применяет предопределенную проблемно-зависимую функцию *PC_bsf_MapF* ко всем элементам своего подсписка «Map», помещая результаты в расширенный список «Reduce».

Функция *BC_WorkerReduce* выполняет следующие действия:

1) выполняет оператор Reduce для своей части расширенного списка «Reduce» с помощью функции *BC_ProcessExtendedReduceList*, которая помещает результат в структуру с указателем *extendedReduceResult_P*;
2) пересылает в синхронном режиме (с помощью функции *MPI_Send*) полученный результат мастеру.

2. Визуальный конструктор BSF-программ

В данном разделе описывается процесс проектирования и реализации программной системы BSF-Studio для быстрого создания корректных BSF-программ на языке C++ с использованием описанного в разделе 1 параллельного каркаса. Основными функциональными возможностями BSF-Studio являются следующие:

– пошаговое заполнение проблемно-зависимых частей BSF-каркаса;
– инкрементная (после каждого шага) компиляция программного кода в облачной среде, поддерживающей технологии параллельного программирования OpenMP [11] и MPI [9], с выводом диагностических сообщений компилятора;
– запуск скомпилированного исполняемого кода в облачной среде, поддерживающей технологии параллельного программирования OpenMP и MPI, с выводом результатов выполнения;
– загрузка отлаженного исходного кода.

Программная система BSF-Studio представляет собой веб-приложение с клиент-серверной архитектурой. Клиентская часть представляет собой одностраничное приложение (Single Page Application) [12], серверная часть — Docker-контейнер [13], в котором запускается приложение, выполняющее по запросу компиляцию и запуск программы, и возвращающее результат. Связь между клиентом и севером осуществляется посредством REST API [14].

2.1. Проектирование и реализация клиентской части

Интерфейс клиента BSF-Studio представляет собой мастер (wizard) для заполнения BSF-каркаса. Процесс работы мастера разбивается на следующие *шаги*:

1) определение константных параметров модели BSF и параметров задачи (размерность, число уравнений и др.);
2) определение типов данных текущего приближения и элементов списков «Map» и «Reduce»;
3) определение пользовательских переменных и структур данных;
4) реализация пользовательских функций.

Указанные шаги реализуются путем последовательного заполнения HTML-форм, содержащих текстовые поля для заполнения соответствующих фрагментов программного кода.

Реализация интерфейса выполнена с помощью JavaScript-библиотеки React с открытым исходным кодом [15]. React позволяет создавать интерактивные пользовательские интерфейсы на основе компонентного подхода: фрагменты веб-приложения (компоненты) инкапсулируются и используются независимо друг от друга. А за счет большого количества библиотек с открытым исходным кодом, предоставляющих готовые React-компоненты, разработка приложений сводится к подбору, подключению и использованию сторонних модулей.

Для работы с данными использовалась библиотека Redux [15]. Redux берет на себя функции записи, хранения, организации доступа к данным из любого React-компонента. Для реализации HTML-форм использовалась библиотека Redux Form [16]. Redux Form предоставляет готовые компоненты и программную обвязку для реализации HTML-форм и хранения их данных с использованием Redux. Для реализации текстовых полей, предназначенных для работы с программным кодом, использовалась библиотека React Ace [17]. Она предоставляет компонент, в котором реализовано оформление текста в зависимости от языка программирования, выбранного из предоставляемого списка, нумерация строк, настраиваемые панели инструментов.

По нажатию на кнопки «Compile» и «Run» программный код компонуется в файлы и отправляется посредством HTTP-запроса на сервер, где происходит объединение пользовательских файлов и файлов BSF-каркаса и выполняется их компиляция и/или запуск.

Результаты компиляции и запуска отображаются в интерфейсе веб-приложения. По нажатию на кнопку «Download» формируются и загружаются файлы BSF-каркаса.

2.2. Проектирование и реализация серверной части

Для разработки серверной части приложения BSF-Studio использована технология Docker [18], которая обеспечивает возможность развертывания приложения на базе контейнерной виртуализированной среды. Это позволило создать виртуальное окружение, максимально приближенное к окружению кластерной вычислительной системы. Данный подход позволяет развернуть серверную часть приложения BSF-Studio в рамках любой операционной системы и аппаратного обеспечения, а также упростить процесс развертывания.

Docker предполагает написание сценария, который содержит последовательные инструкции по настройке окружения. В результате выполнения этого сценария строится виртуальный образ, эмулирующий описанное окружение. В рамках этого Docker-сценария необходимо описать на основе какой операционной системы будет функционировать образ, какие подсистемы и библиотеки должны быть установлены, а также произведена настройка и подготовка образа к функционированию приложения внутри него.

Для решения поставленных задач было взято наиболее компактное ядро Linux Alpine [18], содержащее минимальный необходимый набор программ, который позволяет операционной системе функционировать. В качестве компилятора был выбран GNU C Compiler (GCC) [19]. Помимо этого, были установлены библиотеки параллельного программирования OpenMP и MPICH [20]. Для проверки корректности установки и функционирования всех необходимых элементов при сборке образа производится тестовая компиляция нескольких проектов.

Далее необходимо выбрать технологии и средства реализации непосредственно серверного приложения, которое будет осуществлять сборку предоставленного пользователем исходного кода и возвращать результат компиляции посредством REST API.

Серверное приложение разработано на базе программной платформы Node.js, предоставляющей JavaScript-окружение и веб-фреймворка Express.js, который занимается обработкой входящих запросов, и выполняет функции веб-сервера [21]. Также использовались следующие библиотеки:

- Express-request-id — модуль для Express, который позволяет получать параметры из строки запроса [22].
- ShellJS — библиотека, которая позволяет вызывать команды операционной системы [23].
- Multer — библиотека, которая позволяет фреймворку Express принимать и отправлять файлы [24].

Для автоматизации развертывания контейнеризованных приложений клиентской и серверной частей BSF-Studio создан кластер на основе системы Kubernetes [17].

BSF-Studio-API работает согласно следующему алгоритму. Импортируются необходимые библиотеки и модули: Express, express-request-id, ShellJS, Multer. Далее формируются пути до рабочих директорий для текущего запроса и для приложения. Происходит инициализация экземпляра веб-сервера Express и хранилища, в котором будут находиться полученные в текущем запросе файлы и их именование, и инициализация экземпляра Multer. Модуль Express-Request-Id подключается к веб-серверу Express и происходит его запуск. Далее веб-сервер ожидает HTTP-запрос на компиляцию и запуск программного

кода. Такой запрос должен содержать файлы проблемно-зависимой части кода для BSF-каркаса в качестве параметров. При получении запроса происходит его обработка и загрузка присланных файлов и файлов каркаса в рабочую директорию. Далее выполняется компиляция, результаты отправляются клиенту в формате JSON, рабочая директория удаляется.

Исходные тексты BSF-Studio свободно доступны в сети Интернет:

– серверная часть: https://github.com/ezhova-nadezhda/BSF-Studio-API;
– клиентская часть: https://github.com/ezhova-nadezhda/BSF-Studio.

Заключение

В статье была описана программная поддержка модели BSF. Она включает в себя параллельный BSF-каркас и веб-приложение BSF-Studio. Параллельный BSF-каркас представляет собой совокупность файлов исходного кода на языке C++, используемых для быстрого создания BSF-программ. BSF-каркас содержит реализацию всех проблемно-независимых функций, организующих параллельное выполнение программы с использованием библиотек MPI и OpenMP. Он также содержит определения (заголовки) проблемно-зависимых функций, реализуемых пользователем. Отличительной особенностью BSF-каркаса является то, что он предполагает поэтапное заполнение пользовательской части исходных кодов и при этом компилируется на всех этапах. Веб-приложение BSF-Studio представляет собой визуальный конструктор BSF-программ. BSF-Studio обеспечивает поэтапное заполнение проблемно-зависимых частей программного кода, компиляцию и запуск приложения в облачной среде.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 17-07-00352 а, Правительства РФ в соответствии с Постановлением №211 от 16.03.2013 г. (соглашение № 02.A03.21.0011) и Министерства образования и науки РФ (государственное задание 2.7905.2017/8.9).

Литература

1. Ежова Н.А., Соколинский Л.Б. Обзор моделей параллельных вычислений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 3. С. 58–91. DOI: 10.14529/cmse190304.
2. Leasure B. et al. Parallel Skeletons // Encyclopedia of Parallel Computing. Springer US, 2011. P. 1417–1422. DOI: 10.1007/978-0-387-09766-4_24.
3. Cole M.I. Algorithmic Skeletons: Structured Management of Parallel Computation. MIT Press, 1991.
4. Gonzalez-Velez H., Leyton M. A Survey of Algorithmic Skeleton Frameworks: High-Level Structured Parallel Programming Enablers // Softw. Pract. Exp. John Wiley & Sons, Ltd. 2010. Vol. 40, no. 12. P. 1135–1160. DOI: 10.1002/spe.1026.
5. Dongarra J. et al. Sourcebook of Parallel Computing. Morgan Kaufmann, 2003. 842 p.
6. Ежова Н.А., Соколинский Л.Б. BSF: модель параллельных вычислений для многопроцессорных систем с распределенной памятью // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2018. Т. 7, № 2. С. 32–49. DOI: 10.14529/cmse180204.

7. Ежова Н.А., Соколинский Л.Б. Исследование масштабируемости итерационных алгоритмов при суперкомпьютерном моделировании физических процессов // Вычислительные методы и программирование. 2018. Т. 19, № 4. С. 416–430. DOI: 10.26089/NumMet.v19r437.
8. Gropp W. et al. A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard // Parallel Comput. 1996. Vol. 22, no. 6. P. 789–828. DOI: 10.1016/0167-8191(96)00024-5.
9. Gropp W. MPI3 and Beyond: Why MPI Is Successful and What Challenges It Faces // Recent Advances in the Message Passing Interface. EuroMPI 2012. Lecture Notes in Computer Science, Vol. 7490 / Ed. by J.L. Träff, S. Benkner, J.J. Dongarra. Springer, 2012. P. 1–9. DOI: 10.1007/978-3-642-33518-1_1.
10. Gropp W., Lusk E., Skjellum A. Using MPI: Portable Parallel Programming with the Message-Passing Interface. Second Edition. MIT Press, 1999. 395 p.
11. Pas R. van der, Stotzer E., Terboven C. Using OpenMP — The Next Step: Affinity, Accelerators, Tasking, and SIMD (Scientific and Engineering Computation). 1st ed. MIT Press, 2017. 392 p.
12. Patel Y. White Paper On Single Page Application. Knowarth. 2015. URL: https://www.knowarth.com/wp-content/uploads/2015/02/Single-Page-Application-White-Paper.pdf (дата обращения: 19.09.2019)
13. Docker Open Source Engine Guide. SUSE Linux Enterprise Server 15 SP1. SUSE LLC. 2019. URL: https://documentation.suse.com/sles/15-SP1/pdf/book-sles-docker_color_en.pdf (дата обращения: 19.09.2019)
14. Allamaraju S. RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity. 1st edition. Yahoo Press, 2010. 316 p.
15. Banks A., Porcello E. Learning React Functional Web Development with React and Redux. O'Reilly Media, 2017. 350 p.
16. Rasmussen E. Redux Form: The Best Way to Manage Your Form State in Redux. URL: https://redux-form.com/8.2.2/docs/api/ (дата обращения: 25.02.2018).
17. Hrisho J. React-Ace (GitHub Repository). URL: https://github.com/securingsincity/react-ace (дата обращения: 25.02.2019).
18. Matthias K., Kane S.P. Docker: Up & Running: Shipping Reliable Containers in Production. 2nd Edition. O'Reilly Media, 2018. 352 p.
19. Rothwell T., Youngman J. The GNU C Reference Manual. Free Software Foundation, Inc, 2007. P. 86. URL: https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.pdf (дата обращения: 20.02.2018).
20. MPICH Documentation and Guides. URL: https://www.mpich.org/documentation/guides/ (дата обращения: 20.02.2018).
21. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. 1st ed. O'Reilly Media, 2014. 332 p.
22. Strukchinsky V. Express-Request-Id (GitHub Repository). URL: https://github.com/floatdrop/express-request-id (дата обращения: 17.03.2018).
23. Fischer N., Freitag B. ShellJS - Unix Shell Commands for Node.js (GitHub Repository). URL: https://github.com/shelljs/shelljs (дата обращения: 13.03.2018).
24. Unnebäck L. Multer (GitHub Repository). URL: https://github.com/expressjs/multer (дата обращения: 17.03.2018).

Ежова Надежда Александровна, аспирант, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Соколинский Леонид Борисович, д.ф.-м.н., профессор, проректор по информатизации, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse190406

SOFTWARE SUPPORT OF THE PARALLEL COMPUTATION MODEL BSF

© 2019 N.A. Ezhova, L.B. Sokolinsky

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: EzhovaNA@susu.ru, Leonid.Sokolinsky@susu.ru

Received: 08.11.2019

The paper is devoted to the software support of the parallel computation model BSF (Bulk Synchronous Farm) intended to iterative algorithms with high computational complexity, which developed for multiprocessor systems with distributed memory of exaflops performance level. Software support includes parallel BSF-skeleton and BSF-Studio web-application. The definition and classification of parallel skeletons are given. The logic and file structure of new BSF-skeleton are described. The BSF-skeleton is a collection of C++ source code files used for BSF-programs development. A detailed description of the design and implementation of the BSF-Studio web-application, which is a visual constructor of BSF-programs, is given. BSF-Studio provides compilation and execution of program code.

Keywords: parallel computation model, BSF, parallel skeleton, BSF-Studio, web-application.

FOR CITATION

Ezhova N.A., Sokolinsky L.B. Software Support of the Parallel Computation Model BSF. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 4. pp. 84–99. (in Russian) DOI: 10.14529/cmse190406.

References

1. Ezhova N.A., Sokolinsky L.B. Survey of Parallel Computation Models. Bulletin of South Ural State University. Series: Computational Mathematics and Software Engineering. 2019. vol. 8, no. 3. pp. 58–91. (in Russian) DOI: 10.14529/cmse190304.
2. Leasure B. et al. Parallel Skeletons. Encyclopedia of Parallel Computing. Springer US, 2011. pp. 1417–1422. DOI: 10.1007/978-0-387-09766-4_24.
3. Cole M.I. Algorithmic Skeletons : Structured Management of Parallel Computation. MIT Press, 1991.
4. Gonzalez-Velez H., Leyton M. A Survey of Algorithmic Skeleton Frameworks: High-Level Structured Parallel Programming Enablers. Softw. Pract. Exp. John Wiley & Sons, Ltd. 2010. vol. 40, no. 12. pp. 1135–1160. DOI: 10.1002/spe.1026.
5. Dongarra J. et al. Sourcebook of Parallel Computing. Morgan Kaufmann, 2003. 842 p.
6. Ezhova N.A., Sokolinsky L.B. BSF: Parallel Computation Model for Multiprocessor Systems with Distributed Memory. Bulletin of South Ural State University. Series: Computational Mathematics and Software Engineering. 2018. vol. 7, no. 2. pp. 32–49. (in Russian) DOI: 10.14529/cmse180204.

7. Ezhova N.A., Sokolinsky L.B. Scalability Evaluation of Iterative Algorithms Used for Supercomputer Simulation of Physical Processes. Proceedings — 2018 Global Smart Industry Conference, GloSIC 2018. Article number 8570107. IEEE, 2018. 10 p. DOI: 10.1109/GloSIC.2018.8570131.
8. Gropp W. et al. A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard. Parallel Comput. 1996. vol. 22, no. 6. pp. 789–828. DOI: 10.1016/0167-8191(96)00024-5.
9. Gropp W. MPI3 and Beyond: Why MPI Is Successful and What Challenges It Faces. Recent Advances in the Message Passing Interface. EuroMPI 2012. Lecture Notes in Computer Science, vol. 7490. / Ed. by J.L. Träff, S. Benkner, J.J. Dongarra. Springer, 2012. pp. 1–9. DOI: 10.1007/978-3-642-33518-1_1.
10. Gropp W., Lusk E., Skjellum A. Using MPI: Portable Parallel Programming with the Message-Passing Interface. Second Edition. MIT Press, 1999. 395 p.
11. Pas R. van der, Stotzer E., Terboven C. Using OpenMP — The Next Step: Affinity, Accelerators, Tasking, and SIMD (Scientific and Engineering Computation). 1st ed. MIT Press, 2017. 392 p.
12. Patel Y. White Paper On Single Page Application. Knowarth. 2015. Available at: https://www.knowarth.com/wp-content/uploads/2015/02/Single-Page-Application-White-Paper.pdf (accessed: 19.09.2019)
13. Docker Open Source Engine Guide. SUSE Linux Enterprise Server 15 SP1. SUSE LLC. 2019. Available at: https://documentation.suse.com/sles/15-SP1/pdf/book-sles-docker_color_en.pdf (accessed: 19.09.2019)
14. Allamaraju S. RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity. 1st ed. Yahoo Press, 2010. 316 p.
15. Banks A., Porcello E. Learning React Functional Web Development with React and Redux. O'Reilly Media, 2017. 350 p.
16. Rasmussen E. Redux Form: The Best Way to Manage Your Form State in Redux. Available at: https://redux-form.com/8.2.2/docs/api/ (accessed: 25.02.2018).
17. Hrisho J. React-Ace (GitHub Repository). Available at: https://github.com/securingsincity/react-ace (accessed: 25.02.2019).
18. Matthias K., Kane S.P. Docker: Up & Running: Shipping Reliable Containers in Production. 2nd ed. O'Reilly Media, 2018. 352 p.
19. Rothwell T., Youngman J. The GNU C Reference Manual. Free Software Foundation, Inc, 2007. 86 p. Available at: https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.pdf (accessed: 20.02.2018).
20. MPICH Documentation and Guides. Available at: https://www.mpich.org/documentation/guides/ (accessed: 20.02.2018).
21. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. 1st ed. O'Reilly Media, 2014. 332 p.
22. Strukchinsky V. Express-Request-Id (GitHub Repository). Available at: https://github.com/floatdrop/express-request-id (accessed: 17.03.2018).
23. Fischer N., Freitag B. ShellJS — Unix Shell Commands for Node.js (GitHub Repository). Available at: https://github.com/shelljs/shelljs (accessed: 13.03.2018).
24. Unnebäck L. Multer (GitHub Repository). Available at: https://github.com/expressjs/multer (accessed: 17.03.2018).

УДК 123.456, 789.012 DOI: 10.14529/cmse190407

МИКРО-ПОТОКИ РАБОТ: СОЧЕТАНИЕ ПОТОКОВ РАБОТ И ПОТОКОВОЙ ОБРАБОТКИ ДАННЫХ ДЛЯ ПОДДЕРЖКИ ЦИФРОВЫХ ДВОЙНИКОВ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

© 2019 А.Б.А. Алаасам[1], Г.И. Радченко[1], А.Н. Черных[1,2]

[1]*Южно-Уральский государственный университет*
(454080 Челябинск, пр. им. В.И. Ленина, д. 76),
[2]*Научно-исследовательский центр Энсенады*
(22860 Энсенада, Мексика, Carretera Ensenada - Tijuana No. 3918)
E-mail: alaasamab@susu.ru, gleb.radchenko@susu.ru, chernykh@cicese.mx

Поступила в редакцию: 12.11.2019

В последнее время наблюдается взрывной рост в развитии концепции цифровой индустрии. Одним из важнейших элементов этой концепции является применение методов математического моделирования и интеллектуального анализа данных для создания моделей производственных процессов и конечной продукции, базирующихся на обработке сигналов, поступающих с интеллектуальных сенсоров. Совокупность таких моделей, представляющих собой виртуальное представление промышленных процессов, систем и оборудования называют цифровыми двойниками. Цифровые двойники используют данные, получаемые от сенсоров, установленных на производственных линиях или на базе конечной продукции, для прогнозирования сбоев в работе оборудования, оптимизации качества продукции и сокращения негативного воздействия производственных процессов на окружающую среду. Комплексы моделей, лежащие в основе цифровых двойников, могут быть описаны в виде вычислительных потоков работ (Workflow), состоящих из набора вычислительных сервисов, каждый из которых представляет собой модель одного из этапов технологического процесса. Для организации гибкой поддержки облачных вычислений для выполнения цифровых двойников, мы предлагаем концепцию микро-потоков работ (Micro-Workflows), которая сочетает в себе мощность концепции научных потоков работ (Scientific Workflows), гибкость контейнерных технологий и устойчивость подхода потоковой обработки данных (Stream Processing) в распределенных вычислительных системах.

Ключевые слова: цифровые двойники, микросервисы, контейнеризация, микро-потоки работ, Kafka, Kepler.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Алаасам А.Б.А., Радченко Г.И., Черных А.Н. Микро-потоки работ: сочетание потоков работ и потоковой обработки данных для поддержки цифровых двойников технологических процессов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 4. С. 100–116. DOI: 10.14529/cmse190407.

Введение

Концепция цифровой индустрии подразумевает организацию производственных процессов с использованием интеллектуальных программных платформ, обеспечивающих сбор, хранение и обработку данных с сенсорных сетей, с использованием методов интеллектуального анализа данных и машинного обучения, для повышения эффективности производства [8]. Среди инициатив по развитию этой концепции можно выделить программу «Индустрия 4.0» [20], разработанную в Германии, «Коалицию лидеров умного производства» [6] в США и концепцию промышленного Интернета вещей, предложенную корпорацией General Electric [22].

Одним из ключевых подходов, объединяющих эти программы, является концепция цифровых двойников (ЦД), которая обеспечивает создание и поддержку виртуальных моделей реального оборудования, промышленных процессов и конечной продукции. Согласно [14], ЦД — это интегрированная мульти-физическая, мульти-масштабная вероятностная симуляция сложного изделия, которая использует наиболее подходящие физические модели, актуальные данные сенсоров и др. для того чтобы получить как можно более достоверное представление соответствующего реального объекта.

ЦД используют различные подходы для моделирования реальных объектов и технологических процессов, включая методы статистического и интеллектуального анализа данных, методы вычислительного моделирования, такие как метод конечных элементов и т.д. [18]. Каждый из этих методов предъявляет особые требования к необходимым вычислительным ресурсам. Например, методы интеллектуального анализа данных требуют хранилищ большого объема с высокой пропускной способностью для сбора и получения доступа к аналитическим данным, а также высокой масштабируемости вычислительной системы для их обработки; для применения методов машинного обучения требуются узлы с установленными ускорителями а для моделей, использующей метод конечных элементов, требуется высокая производительность центрального процессора и большие объемы оперативной памяти [1].

Предоставление комплекса вычислительных систем, обеспечивающего возможности высокого масштабирования, а также соответствующего столь разнообразным требованиям к вычислительным ресурсам может быть реализовано посредством применения технологий облачных вычислений [13]. В настоящее время, облачные вычислительные системы стали повсеместно применяться для выполнения задач компьютерного моделирования, предоставляя доступ к большому разнообразию вычислительных ресурсов, используя, в основном, технологию виртуальных машин. Но применение виртуальных машин связано с большими накладными расходами, которые могут значительно ограничить производительность систем ввода/вывода и эффективность использования вычислительных ресурсов [17, 32]. Технология контейнеризации, получившая значительное развитие в последнее время, основана на ограничении количества ресурсов вычислительного узла, отдаваемых приложению. Контейнер предоставляет среду выполнения приложения на уровне операционной системы, сокращая накладные расходы по сравнению с виртуальной машиной [4].

Для описания технологических процессов часто применяется концепция потоков работ (Workflows) [30]. Потоки работ применяются в промышленности, коммерческих компаниях и науке как способ формализации и структурирования процессов, состоящих из совокупности независимых действий, связанных между собой отношениями зависимости. Вычислительный поток работ предоставляет эффективное решение для моделирования сложных технологических процессов путем управления сервисами и оптимизации порядка их выполнения [26]. В связи с этим, цифровой двойник промышленного процесса или оборудования, может быть представлен в виде вычислительного потока работ, состоящего из совокупности вычислительных сервисов, которые представляют собой модели этапов технологического процесса.

Для того, чтобы цифровой двойник мог обеспечить синхронизацию между состоянием процесса в реальном мире и его виртуальной копией [5], мы должны обеспечить ему воз-

можность получать, передавать и анализировать поток данных от оборудования промышленного Интернета вещей (IIoT) в режиме реального времени. Для решения этой задачи могут использоваться платформы потоковой обработки данных (Stream Processing), обеспечивающие слабосвязанность программной архитектуры, гибкость и контроль над сложными процессами, завязанными на асинхронной обработке данных [7].

Еще одной важной концепцией, сформировавшейся в последнее время, является *микросервисный подход* проектирования приложений, реализуемых на базе облачных вычислительных платформ. В рамках микросервисного подхода, серверная часть приложения разбивается на отдельные, изолированные компоненты — микросервисы, обеспечивающие прозрачный доступ к своим функциональным возможностям и взаимодействующим друг с другом посредством кроссплатформенного API. Эти компоненты могут быть развернуты и масштабироваться независимо друг от друга в автоматическом режиме [21]. Такой подход к проектированию и реализации распределенных систем, части которых подвергаются разной нагрузке, либо должны обеспечивать различные характеристики предоставляемого качества обслуживания, позволяет обеспечить независимое развертывание, масштабирование и управление временем жизни их компонентов [33].

В данной работе нами будет дано описание концепции цифровых двойников для моделирования промышленных процессов. Также, нами будет представлена модель событийного-ориентированного подхода, совмещающего возможности научных потоков работ, гибкость контейнерных технологий и устойчивость подхода потоковой обработки данных в распределенной вычислительной системе. Нами будет представлен пример реализации данного подхода, посредством разработки акторов, обеспечивающих потребление и отправку потоков данных в платформу Apache Kafka из системы управления научными потоками работ Kepler. Данный подход обеспечивает возможность формирования потока работ как совокупности слабосвязанных сервисов (микро-потоков работ или Micro-Workflows, по аналогии с микросервисами). Такие сервисы могут быть развернуты внутри вычислительных контейнеров и обеспечивать возможность коммуникации через платформу потоковой обработки данных.

Дальнейшая статья имеет следующую структуру. В разделе 2 нами дается обзор существующих подходов к облачным системам, поддерживающим концепцию цифровых двойников. В разделе 3 описывается концепция цифровых двойников, позволяющая осуществлять моделирование реальных производственных процессов. В разделе 4 представлен обзор концепции облачной платформы цифровых двойников. В разделе 5 мы даем описание подхода микро-потоков работ для проектирования цифровых двойников. В разделе 6 представлен процесс разработки микро-потоков работ и реализация эксперимента. В разделе 7 приводятся результаты оценки эффективности микро-потоков работ. В заключении представлены обобщенные результаты работы и описаны возможные направления дальнейших исследований.

1. Обзор смежных работ

На сегодняшний день одной из наиболее распространенных концепций, связанных с платформами цифровых двойников, являются облачные системы Интернета вещей (IoT). Они обеспечивают сбор данных с различных датчиков и проведение их анализа. Авторы статьи [23] дают описание модели Sensing as a service (SenaaS, «Детектирование как сервис»), состоящей из четырех концептуальных слоев: 1) датчики и владельцы датчиков, 2)

публикаторы датчиков, 3) поставщики расширенных сервисов и 4) потребители данных от датчиков. В статье приводится описание нескольких облачных систем для сбора данных от датчиков. Например, Xively [27] — это общедоступная облачная среда для Интернета вещей, которая упрощает и ускоряет создание, развертывание и масштабирование систем датчиков. В рамках проекта OpenIoT [28] разрабатывается платформа с открытым исходным кодом, обеспечивающая динамическое формирование самоуправляемых облачных сред для Интернета вещей. Авторы платформы IoT-Hub [11] предлагают решение для сбора, проверки качества, хранения и визуализации данных от датчиков Интернета вещей на основе платформы Apache Spark[1] в качестве механизма обработки данных.

Цифровой двойник может быть представлен как последовательность заданий по обработке данных, связанных между собой набором ребер, представляющих зависимости по данным между этими заданиями. Такой подход может быть реализован в виде научного потока работ (Scientific Workflow или SWF), который обычно управляется, выполняется и отслеживается системами управления научными потоками работ (Scientific Workflows Management Systems — SWfMS), такими как DiVTB [25], Pegasus [9], Kepler [3], ASKALON [10], или tGSF [15, 16]. SWfMS позволяет ученым сконцентрироваться на решении исследовательских задач, обеспечивая автоматизацию процесса проведения сложных вычислительных экспериментов. Авторы статьи [12] предлагают среду для отображения вычислительных потоков работ, написанных на языке Dispel, на другие платформы такие как Apache Storm[2] и MPI.

Однако выполнение набора SWF, характеризующихся сильно-связанностью вычислительных сервисов, последовательным выполнением и ограниченной поддержкой потоковой обработки данных не соответствует событийному подходу, необходимому для работы с цифровыми двойниками. В нашей работе мы предлагаем решение для интеграции вычислительных потоков работ с концепцией микросервисов с целью создания слабосвязанной событийно-ориентированной инфраструктуры для разработки и выполнения цифровых двойников.

2. Цифровой двойник

Концепция цифровых двойников (см. рис. 1) была представлена в начале 2000-х годов. Она основывается на подходе синхронизации реально существующих объектов с их виртуальными представлениями в виде вычислительных моделей [29]. Цифровой двойник — это иерархическая система математических моделей, вычислительных методов и сервисов программного обеспечения, которая предоставляет близкую к реальному времени синхронизацию между состоянием реального процесса или системы и его/ее виртуальной копией [5].

Синхронизация в цифровом двойнике математических моделей (мультифизических, многомасштабных и вероятностных) с реальными данными позволяет оценивать эффективность работы оборудования, получать информацию о состоянии процесса или конечной продукции и прогнозировать их характеристики, используя информацию из архивной базы данных.

Можно выделить следующие характеристики цифровых двойников:

[1] https://spark.apache.org/
[2] https://storm.apache.org/

– *Отображение в режиме реального времени.* Цифровой двойник отображает физический объект (оборудование, процесс и систему) в режиме реального времени, обеспечивая и соответствие виртуальной копии реально существующему объекту.
– *Иерархия.* Цифровой двойник промышленного процесса может состоять из цифровых двойников этапов процесса, которые, в свою очередь, могут содержать цифровые двойники оборудования, на котором осуществляются данные этапы.
– *Саморазвитие.* Так как цифровой двойник является отображением объекта или процесса, то он должен постоянно совершенствовать модель путем сравнения процесса моделирования одновременно с физическим объектом и процессом [31].

Цифровые двойники могут использовать различные типы моделей для моделирования текущего состояния реального объекта. Полное и своевременное отображение состояния можно получить только путем использования моделей с необходимой и достаточной степенью точности для соответствия требованию «отображения в режиме реального времени». Чем большая точность требуется, тем больше необходимо ресурсов и времени для выполнения моделирования. В связи с этим можно выделить следующие типы моделей, используемые в цифровых двойниках:

– *Сценарные модели* применяются для уведомления конечных пользователей о событиях, в случае если параметры технологического процесса превышают установленные пределы. Низкая вычислительная сложность таких моделей позволяет внедрять их непосредственно на оборудовании, установленном на промышленном участке. В то же время необходимо выделять вычислительные ресурсы для фильтрации поступающих сообщений и направления уведомлений заинтересованным исполнителям.
– *Первопринципные модели*, основанные на основных принципах термодинамических, физических и химических моделей, на которых базируется работа конкретного оборудования. Эти модели позволяют прогнозировать ухудшение характеристик технологических процессов путем учета текущей динамики, а также предсказывать дату и время для проведения очередного технического обслуживания оборудования по фактическому состоянию его эффективности [18].
– *Конечно-элементные вычислительные модели* дают возможность моделирования сложных многофакторных процессов путем решения систем дифференциальных

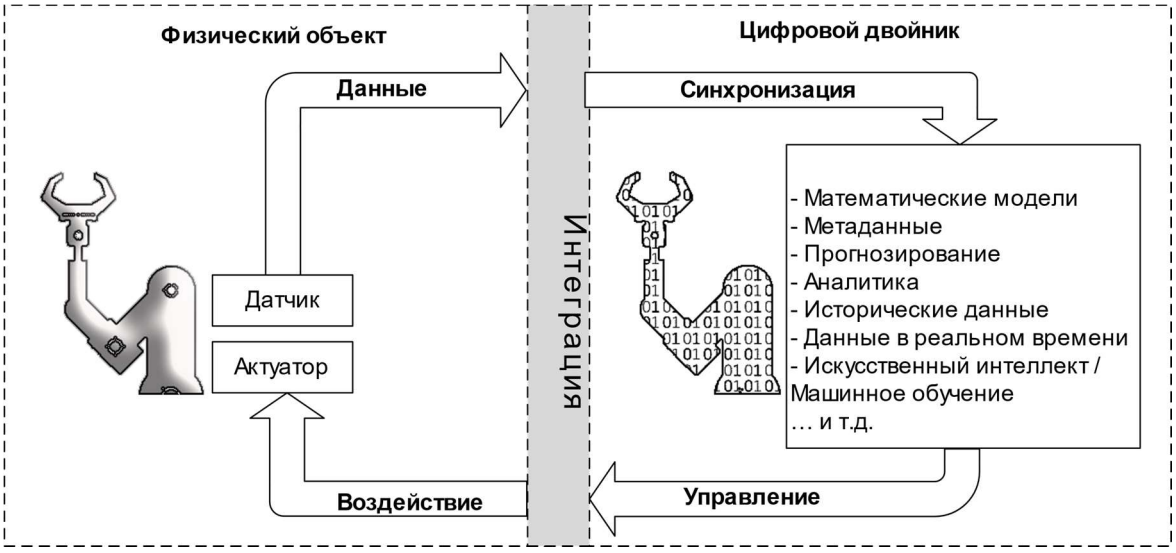


Рис. 1. Концепция цифрового двойника

уравнений с заданными граничными условиями. Они используются для проектирования новых продуктов, моделирования новых технологических циклов и моделирования чрезвычайных ситуаций. Решение задач этого класса требует специального вычислительного оборудования.

– *Методы интеллектуального анализа данных* используются для решения задач, для которых не подходит ни один из вышеперечисленных методов ввиду сложности, многофакторности или вероятностных особенностей происходящих процессов. Эти методы требуют больших объемов данных хорошего качества (предварительно очищенных) различной природы происхождения.

3. Облачная платформа для поддержки цифровых двойников

Для поддержки работы цифровых двойников нами предлагается концепция облачной платформы для поддержки цифровых двойников, которая позволяет осуществить динамическое распределение вычислительных ресурсов и предоставляет прикладной программный интерфейс для представления цифровых двойников на базе микросервисного подхода. Таким образом, облачная платформа цифровых двойников предоставляет облачную модель Digital Twin-as-a-Service (DTaaS, «Цифровой двойник как услуга»). В рамках модели DTaaS, цифровой двойник представляется как совокупность облачных сервисов, обеспечивающих хранение и анализ данных, получаемых от датчиков, для моделирования реально существующих объектов и визуализации их виртуальных представлений.

Облачная платформа цифровых двойников предоставляет единую точку входа для следующих ключевых категорий пользователей (см. рис. 2):

– *Производители промышленного оборудования* могут создавать цифровые двойники поставляемого оборудования для потребителей, осуществлять удаленный мониторинг и анализ использования поставленного оборудования в целях повышения качества продукции.

Рис. 2. Облачная платформа цифровых двойников

– *Промышленные предприятия* могут создавать цифровые двойники промышленных процессов, обеспечивающих анализ и принятие решений для развития предприятия.
– *Исследовательские центры* могут разрабатывать и продвигать свои аналитические модели для цифровых двойников, а также предоставлять услуги по поддержке и повышению эффективности промышленных процессов и систем.

Облачная платформа цифровых двойников обеспечивает функционирование на следующих уровнях:

– *Уровень пользователя цифрового двойника.* На этом уровне пользователь может получить доступ к имеющимся цифровым двойникам в виде облачных приложений на основе модели Software-as-a-Service («Программное обеспечение как услуга»). Пользовательский интерфейс цифрового двойника может быть представлен конечному пользователю в различном виде: от трехмерных моделей и мнемонических схем до табличных данных и текстовых сообщений, описывающих выбранные аспекты моделируемого реального объекта для конкретных типов пользователей. Например, инженер может использовать цифровой двойник газового котла в виде трехмерного объекта в дополненной реальности для обнаружения закупорки в печи. Тот же цифровой двойник может быть представлен в виде графика снижения эффективности работы котла и информации о прогнозируемой оптимальной дате проведения ТО для экономиста.
– *Уровень разработчика цифрового двойника.* На этом уровне облачная платформа предоставляет ресурсы для разработки цифрового двойника на основе модели Platform-as-a-Service («Платформа как услуга»). Цифровой двойник может быть описан как совокупность вычислительных сервисов, осуществляющих связь друг с другом посредством передачи сообщений.
– *Уровень разработчика вычислительного сервиса.* На этом уровне облачная платформа предоставляет прикладной программный интерфейс для разработки вычислительного сервиса на основе модели Backend-as-a-Service («Серверное приложение как услуга»). Вычислительный сервис — это микросервис, отвечающий за конкретные операции обработки данных или за выполнение конкретного набора вычислительных методов, обеспечивающих моделирование определенного аспекта цифрового двойника.
– *Уровень поставщика облачной инфраструктуры.* На этом уровне экземпляры вычислительных сервисов сопоставляются с ресурсами облачной вычислительной платформы, предоставляемыми на основе модели Container-as-a-Service («Контейнер как услуга»).

4. Подход микро-потоков работ для проектирования цифровых двойников

Существующие в настоящий момент системы, поддерживающие выполнение научных потоков работ, не обладают механизмами поддержки масштабирования под-потоков и интеграцию с системами потоковой обработки данных. Такие возможности играют важную роль для поддержки работы цифровых двойников.

Для решения этой проблемы мы предлагаем перепроектировать монолитный подход, распространенный в настоящий момент при проектировании научных потоков работ в меньшие слабо-связанные совокупности микро-потоков работ (Micro-Workflows или

MWF) [2, 24], которые можно было бы реализовать в виде независимых вычислительных сервисов, развернутых в отдельных контейнерах и поддерживающих связь друг с другом посредством системы потоковой обработки данных.

Данный подход позволит обеспечить низкие значения задержки сетевого обмена для обработки данных Интернета вещей и позволит реализовать независимую разработку и развертывание микро-потоков работ, составляющих цифровые двойники.

Он также обеспечивает значительное расширение возможностей горизонтального масштабирования и вертикального распределения микро-потоков работ между публичными и частными облаками. Микро-потоки работ, которые должны обеспечивать ответ с низким значением задержки, могут быть развернуты в частных облаках, расположенных рядом с оборудованием Интернета вещей, в то время как микро-потоки работ, обеспечивающие пакетную обработку больших данных, могут быть расположены в узлах вблизи систем хранения.

Для преобразования изначального научного потока работ в совокупность микро-потоков работ, связанных между собой посредством системы потоковой обработки данных, мы ввели «Узел-потребитель» и «Узел-производитель» в точках разделения изначального потока работ. Эти узлы позволяют осуществлять обмен данными между микро-потоками работ и системой потоковой обработки данных (см. рис. 3).

5. Внедрение микро-потоков работ и развертывание эксперимента

Для проверки нашего подхода мы применили предложенную архитектуру с использованием системы Kepler[3] в качестве системы управления научными потоками работ, среды Apache Kafka[4] в качестве платформы потоковой обработки данных, которая под-

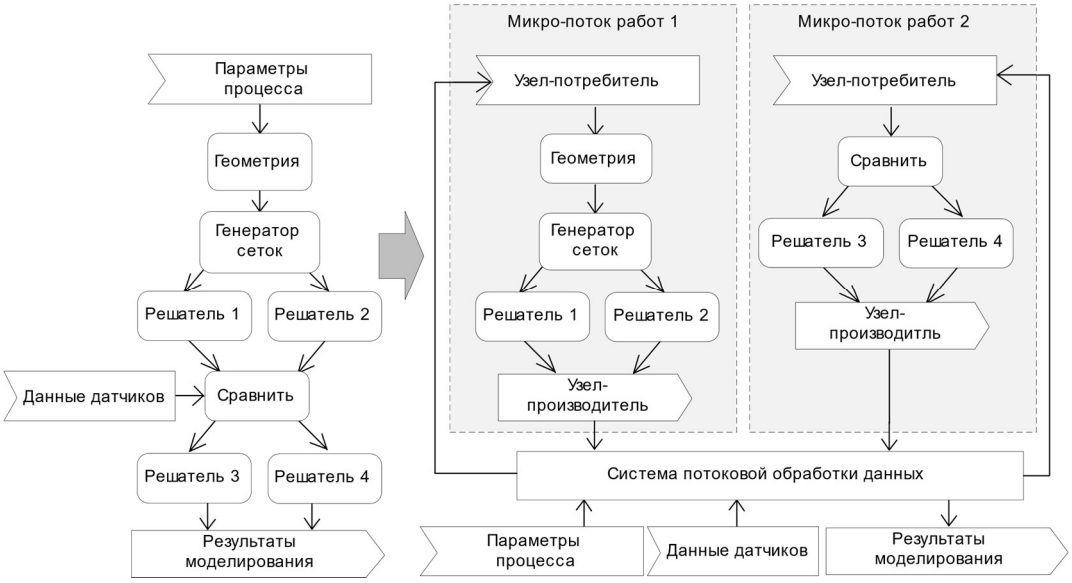


Рис. 3. Подход микро-потоков работ

[3] https://kepler-project.org/

[4] https://kafka.apache.org/

держивает горизонтальную масштабируемость, и Docker в качестве платформы контейнеризации. Когда гибко связанные сервисы взаимодействуют между собой посредством сообщений, они должны предоставлять универсальный формат сообщений (схему). Платформа Confluent[5] предоставляет реестр схем на основе системы сериализации данных Avro[6]. Для автоматизации развертывания среды Kafka мы используем образ и реестр схем на базе контейнера, созданного Landoop[7].

Проведем оценку нашего подхода, используя данные от датчиков, представленных в материалах Конкурса «DEBS[8] 2012 Grand Challenge: Manufacturing equipment». Представленные данные представляют собой запись реальной информации, передаваемой датчиками системы мониторинга технологического процесса в химической промышленности. Задержка между двумя последовательно расположенными точками в обрабатываемых нами исходных данных составляет около 10 мкс. Каждое обрабатываемое сообщение включает в себя 66 полей. Первый блок операторов регистрирует изменение состояния полей во входных данных и передает их вместе с временными отметками, отмечающими, когда произошло изменение состояния. Второй блок операторов соотносит изменение состояния датчиков и изменение состояния клапанов путем расчета временной разницы между возникновением изменений состояния и передает их далее вместе с временными отметками. Задачей последнего блока операторов является постоянный мониторинг тренда рассчитанных значений от предыдущих выходных данных на период, равный 24 часам.

5.1. Новые акторы для Kepler

Для оценки нашего подхода был разработан набор акторов для системы Kepler. Они расширяют возможности Kepler для работы в среде Apache Kafka и обеспечивают реализацию первого запроса для обработки данных конкурса DEBS 2012, включая:

- **KafkaConsumer** — это актор, обеспечивающий чтение сообщения из исходной темы (topic) Kafka, десериализацию полученных сообщений в формате Avro и пересылку этих сообщений в качестве записей на выходной порт;
- **KafkaProducer** — это актор, обеспечивающий получение записей с входного порта, сериализацию полученных записей путем преобразования в формат Avro и их пересылку в конечную тему Kafka;
- **DetectStateChange** — это актор, регистрирующий изменение в двух последовательных записях данных (с 0 на 1 и наоборот);
- **CorrelateStateChange** — это актор, который выполняет корреляционный анализ состояния изменения во временных последовательностях «0» и «1» между парами датчиков. Он рассчитывает временную разницу между повышенным (1) и пониженным (0) состояниями каждой из входных пар датчиков;
- **XXYY State** — это актор, который выделяет связанные величины временной разницы и подготавливает их для отображения в виде графического представления данных.

[5] https://www.confluent.io/
[6] https://avro.apache.org/
[7] https://www.landoop.com/
[8] http://debs.org/grand-challenges/

5.2. Развертывание

Развертывание нашего решения осуществлено с использованием суперкомпьютера «Торнадо» на базе ресурсов суперкомпьютерного центра Южно-Уральского государственного университета [19] (см. рис. 4). Виртуальная машина ВМ1 («Эмулятор датчиков») моделирует процесс генерации данных датчиками Интернета вещей. Она потребляет данные из базы данных DEBS 2012 и последовательно публикует их в среде Kafka, развернутой в контейнере Landoop (ВМ2). Мы разделяем поток работ по первому запросу DEBS 2012 на два микро-потока работ и упаковываем каждый микро-поток работ в отдельный контейнер. Контейнеры разворачиваются на виртуальных машинах ВМ2 и ВМ3. Прямого соединения между виртуальными машинами или узлами нет. Все взаимодействие осуществляется через сетевой сервер. Виртуальные машины ВМ1 и ВМ3 работают на одном и том же физическом узле и снабжены 4ГБ ОЗУ и четырьмя узлами процессора Intel Xeon X5680. Машина ВМ2 работает на отдельном физическом узле и снабжена 12 ГБ ОЗУ и восьмиядерным процессором Intel Xeon X5680. Соединение между виртуальными машинами организовано посредством внешнего физического узла, действующего как виртуальный маршрутизатор, подсоединенный через сеть Gigabit Ethernet.

Микро-поток работ MWF-1 потребляет изначальные данные от датчиков из исходной темы в системе Kafka и обрабатывает их, используя актор DetectStateChange (см. рис. 5а). Результаты публикуются в промежуточной теме в системе Kafka. Микро-поток работ MWF-2 потребляет данные из промежуточной темы Kafka и обрабатывает все оставшиеся блоки операторов, включая построение графика конечных результатов (см. рис. 5б).

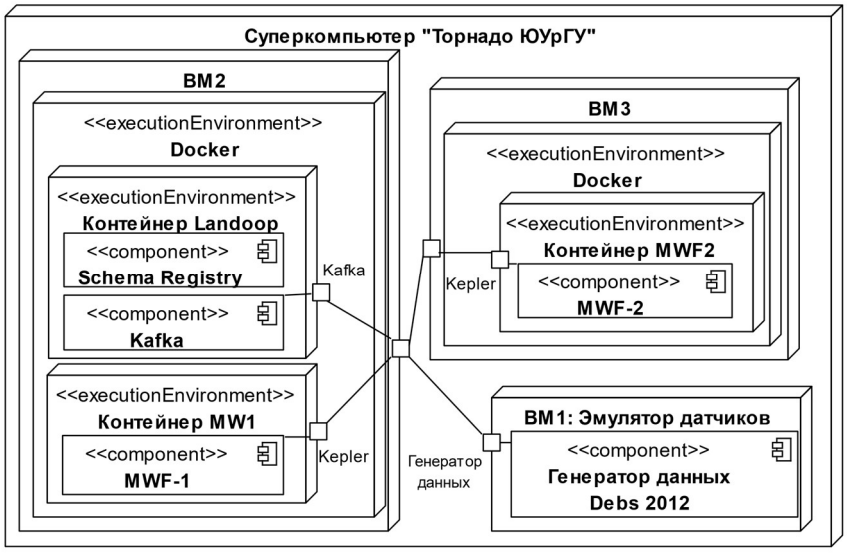


Рис. 4. Развертывание эксперимента на узлах суперкомпьютера «Торнадо ЮУрГУ»

6. Оценка эффективности микро-потоков работ

Для оценки эффективности предлагаемой системы мы добавили следующие дополнительные поля в создаваемые сообщения:

1. $\boldsymbol{mX_{orgts}}$: время, когда датчик отправляет исходное сообщение номер X.
2. $\boldsymbol{mX_{rcvts}}$: время, когда Kepler получает исходное сообщение X из исходной темы Kafka.
3. $\boldsymbol{K_{sentts}}$: время, когда актор KafkaProducer отправляет итоговое сообщение в конечную тему Kafka.

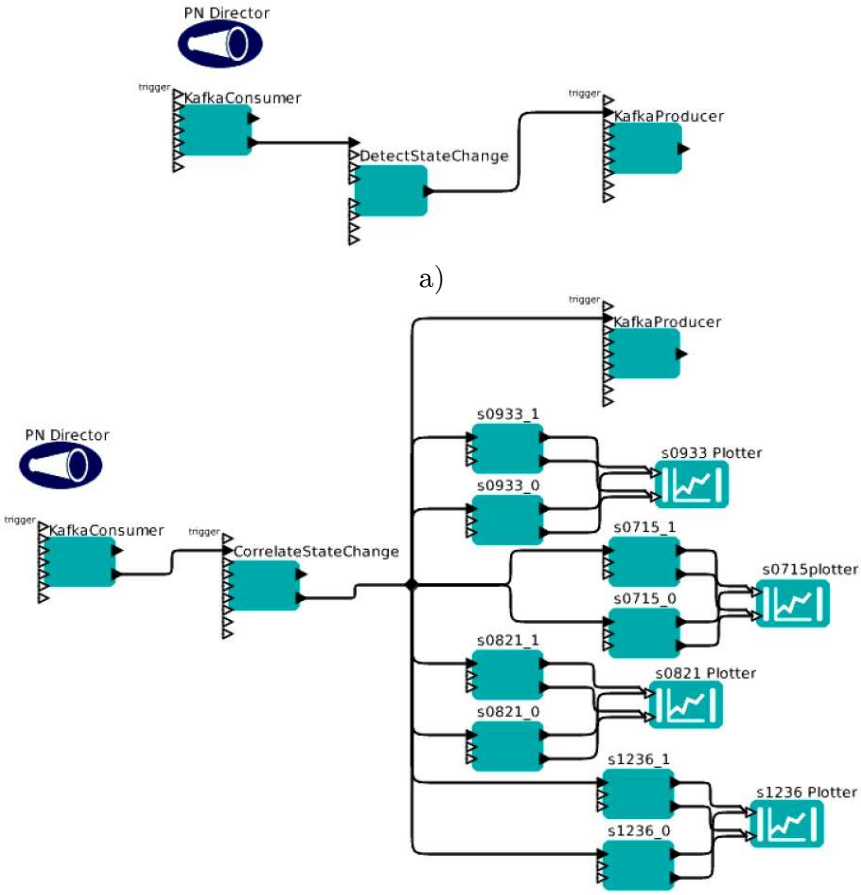


Рис. 5. Развертываемые микро-потоки работ

Для проверки эффективности обмена данными и их обработки, предлагаются следующие критерии оценивания:

1. $\boldsymbol{Av_{SM}}$ (средний интервал между исходными сообщениями): средняя задержка по времени между двумя последовательными исходными сообщениями с данными, переданными датчиком в Kafka.
2. $\boldsymbol{Av_{TAT}}$ (среднее время обработки): средний временной интервал, требуемый для создания одного итогового сообщения микро-потоком работ. При этом интервал отсчитывается, начиная от времени получения второго требуемого исходного сообщения ($\boldsymbol{m2_{rcvts}}$) из исходной темы Kafka, и заканчивая временем передачи итогового сообщения ($\boldsymbol{K_{sentts}}$) в Kafka.
3. $\boldsymbol{Av_{DV}}$ (среднее время доставки): средний временной интервал между $\boldsymbol{mX_{orgts}}$ и $\boldsymbol{mX_{rcvts}}$, включая сериализацию данных, передачу сообщения от VM1 к VM2, подтверждение получения сообщения от Kafka.
4. $\boldsymbol{Av_{L12}}$ (средняя задержка): средняя сетевая задержка между машиной VM1 и машиной VM2.

Результаты наших экспериментов представлены в таблице. За 24 часа тестирования наша система обработала более 7,3 миллионов сообщений со средним временем выполнения задания около 3,2 мс. Среднее ресурсопотребление нашей системы с использованием потока данных межплатформного программного обеспечения среды Kafka можно оценить следующим образом:

$$Av_{DV} - Av_{L12} = 4{,}4 - 3{,}5 = 0{,}9 \text{ (мс)}.$$

Таблица

Результаты оценки эффективности

Время теста	24 часа
Количество сообщений	7 328 844
$\boldsymbol{Av_{SM}}$	11,8 мс
$\boldsymbol{Av_{TAT}}$	3,2 мс
$\boldsymbol{Av_{DV}}$	4,4 мс
$\boldsymbol{Av_{L12}}$	3,5 мс

Заключение

В данной работе была представлена концепция облачной платформы цифровых двойников и подход микро-потоков работ, обеспечивающий поддержку цифровых двойников. Наше решение расширяет возможности существующих подходов благодаря сочетанию мощности научных потоков работ, гибкости контейнерных технологий и устойчивости подхода потоковой передачи данных. Оно поддерживает возможность обработки событий от различных источников (таких как датчики Интернета вещей) внутри вычислительных потоков работ. Внедрение и тестирование данной архитектуры на базе системы управления научными потоками работ Kepler и платформы потоковой обработки данных Apache Kafka показывают, что предлагаемая архитектура успешно решает задачи обработки данных интернета вещей, обеспечивая достаточно небольшие размеры сетевой задержки. Основным направлением для будущих работ является оценка масштабируемости и применимости данного подхода для поддержки значительных объемов данных Интернета вещей в режиме реального времени.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта No 18-07-01224 а.

Литература

1. Al-Jarrah O.Y. et al. Efficient Machine Learning for Big Data: A Review // Big Data Research. 2015. Vol. 2, no. 3. P. 87–93. DOI: 10.1016/j.bdr.2015.04.001.
2. Alaasam A.B.A. et al. Scientific Micro-Workflows: Where Event-Driven Approach Meets Workflows to Support Digital Twins // Proceedings of the international conference Russian Supercomputing Days, RuSCDays'18 (Moscow, Russia, September, 24–25, 2018). MSU. 2018. Vol. 1. P. 489–495.
3. Altintas I. et al. Kepler: an extensible system for design and execution of scientific workflows // Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 2004. Vol. I. P. 423–424. DOI: 10.1109/SSDM.2004.1311241.
4. Boettiger C. An introduction to Docker for reproducible research // ACM SIGOPS Operating Systems Review. 2015. Vol. 49, no. 1. P. 71–79. DOI: 10.1145/2723872.2723882.
5. Borodulin K. et al. Towards Digital Twins Cloud Platform: Microservices and Computational Workflows to Rule a Smart Factory // Proceedings of the 10th International Conference on Utility and Cloud Computing, UCC '17 (Austin, Texas, USA, December, 5–8, 2017). P. 209–210. DOI: 10.1145/3147213.3149234.
6. Bryner M. Smart manufacturing: The next revolution // Chemical Engineering Progress. 2012. Vol. 108, no. 10. P. 4–12.

7. Carvalho O., Roloff E., Navaux P.O.A. A Distributed Stream Processing based Architecture for IoT Smart Grids Monitoring // Companion Proceedings of the 10th International Conference on Utility and Cloud Computing, 2017. New York, NY, USA. ACM Press, 2017. P. 9–14. DOI: 10.1145/3147234.3148105.
8. Davis J. et al. Smart manufacturing, manufacturing intelligence and demand-dynamic performance // Computers and Chemical Engineering. 2012. Vol. 47. P. 145–156. DOI: 10.1016/j.compchemeng.2012.06.037.
9. Deelman E. et al. Pegasus, a workflow management system for science automation // Future Generation Computer Systems. 2015. Vol. 46. P. 17–35. DOI: 10.1016/j.future.2014.10.008.
10. Fahringer T. et al. ASKALON: a Grid application development and computing environment // The 6th IEEE/ACM International Workshop on Grid Computing, 2005. IEEE, 2005. P. 10. DOI: 10.1109/GRID.2005.1542733.
11. Filgueira R. et al. IoT-Hub: New IoT data-platform for Virtual Research Environments // 10th International Workshop on Science Gateways, IWSG 2018, 2018. P. 13–15.
12. Filguiera R. et al. Dispel4py: A Python framework for data-intensive scientific computing // International Journal of High Performance Computing Applications. 2017. Vol. 31, no. 4. P. 316–334. DOI: 10.1177/1094342016649766.
13. Foster I.T., Gannon D.B. Cloud Computing for Science and Engineering / Ed. by I.T. Foster, D.B. Gannon. The MIT Press, 2017. 392 p.
14. Glaessgen E., Stargel D. The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles // 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference (Reston, Virigina, 2012). American Institute of Aeronautics and Astronautics, 2012. P. 1–14. DOI: 10.2514/6.2012-1818.
15. Hirales-Carbajal A. et al. A Grid simulation framework to study advance scheduling strategies for complex workflow applications // 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, IPDPSW, 2010. IEEE, 2010. P. 1–8. DOI: 10.1109/IPDPSW.2010.5470918.
16. Hirales-Carbajal A. et al. Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid // Journal of Grid Computing. 2012. Vol. 10, no. 2. P. 325–346. DOI: 10.1007/s10723-012-9215-6.
17. Huber N. et al. Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments // Proceedings of the 1st International Conference on Cloud Computing and Services Science (Noordwijkerhout, The Netherlands, May, 7–9, 2011). SciTePress - Science and and Technology Publications, 2011. P. 563–573.
18. Korambath P. et al. A smart manufacturing use case: Furnace temperature balancing in steam methane reforming process via kepler workflows // Procedia Computer Science. 2016. Vol. 80. P. 680–689. DOI: 10.1016/j.procs.2016.05.357.
19. Kostenetskiy P.S., Safonov A.Y. SUSU Supercomputer Resources // Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies, PCT 2016 (Arkhangelsk, Russia, March, 29–31, 2016). CEUR Workshop Proceedings, 2016. P. 561–573.
20. Lee J., Bagheri B., Kao H.-A. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems // Manufacturing Letters. 2015. Vol. 3. P. 18–23. DOI: 10.1016/j.mfglet.2014.12.001.

21. Lewis J., Fowler M. Microservices. 2014. URL: http://martinfowler.com/articles/microservices.html (дата обращения: 28.01.2016).
22. Li S., Xu L. Da, Zhao S. The internet of things: a survey // Information Systems Frontiers. 2015. Vol. 17, no. 2. P. 243–259. DOI: 10.1007/s10796-014-9492-7.
23. Perera C. et al. Sensing as a service model for smart cities supported by Internet of Things // Transactions on Emerging Telecommunications Technologies. 2014. Vol. 25, no. 1. P. 81–93. DOI: 10.1002/ett.2704.
24. Radchenko G., Alaasam A., Tchernykh A. Micro-Workflows: Kafka and Kepler Fusion to Support Digital Twins of Industrial Processes // 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC Companion, 2018. P. 83–88. DOI: 10.1109/UCC-Companion.2018.00039.
25. Radchenko G., Hudyakova E. A service-oriented approach of integration of computer-aided engineering systems in distributed computing environments // UNICORE Summit 2012, Proceedings, 2012. P. 57–66.
26. Rahman M. et al. Adaptive workflow scheduling for dynamic grid and cloud computing environment // Concurrency Computation Practice and Experience. 2013. P. 1816–1842. DOI: 10.1002/cpe.3003.
27. Sinha N., Pujitha K.E., Alex J.S.R. Xively based sensing and monitoring system for IoT // 2015 International Conference on Computer Communication and Informatics, ICCCI, 2015. IEEE, 2015. P. 1–6. DOI: 10.1109/ICCCI.2015.7218144.
28. Soldatos J. et al. OpenIoT: Open Source Internet-of-Things in the Cloud // Lecture Notes in Computer Science. 2015. Vol. 9001. P. 13–25. DOI: 10.1007/978-3-319-16546-2_3.
29. Tao F. et al. Digital twin-driven product design, manufacturing and service with big data // International Journal of Advanced Manufacturing Technology. 2017. Vol. 94, no. 9–12, P. 3563–3576. DOI: 10.1007/s00170-017-0233-1.
30. Taylor I. et al. Workflows for e-Science. Springer London, 2007. 523 p.
31. Tuegel E.J. et al. Reengineering Aircraft Structural Life Prediction Using a Digital Twin // International Journal of Aerospace Engineering. 2011. Vol. 2011. P. 1–14. DOI: 10.1155/2011/154798.
32. Xavier M.G. et al. Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments // 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (Belfast, UK, February, 27–March, 1, 2013). IEEE, 2013. P. 233–240. DOI: 10.1109/PDP.2013.41.
33. Савченко Д.И., Радченко Г.И. Методология тестирования микросервисных облачных приложений // Суперкомпьютерные дни в России: Труды международной конференции. 2015. P. 245–256.

Алаасам Амир Басим Абдуламир, аспирант, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Радченко Глеб Игоревич, к.ф.-м.н., доцент, директор Высшей школы электроники и компьютерных наук, заведующий кафедрой Электронных вычислительных машин, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Черных Андрей Николаевич, к.т.н., доцент, заведующий лабораторией «НИЛ Проблемно-ориентированных облачных сред», Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация), профессор, научно-исследовательский центр Энсенады (Энсенада, Мексика)

DOI: 10.14529/cmse190407

MICRO-WORKFLOWS: A COMBINATION OF WORKFLOWS AND DATA STREAMING TO SUPPORT DIGITAL TWINS OF PRODUCTION PROCESSES

© 2019 A.B.A. Alaasam[1], G.I. Radchenko[1], A.N. Tchernykh[1,2]

[1]*South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)*,

[2]*Ensenada Research Center*

(Carretera Ensenada - Tijuana No. 3918, Ensenada, 22860 Mexico),

E-mail: *alaasamab@susu.ru, gleb.radchenko@susu.ru, chernykh@cicese.mx*

Received by: 12.11.2019

Recently, there has been an explosive growth in the development of the concept of "Digital Industry". One of the most important elements of this concept is the application of methods of mathematical modeling and intelligent data analysis to create models of production processes and final products based on the processing of signals coming from intelligent sensors. A set of such models, which represent a virtual representation of industrial processes, systems and equipment, is called "Digital twin". Digital twins use data from sensors installed on production lines or on the basis of final products to predict equipment failures, optimize product quality and reduce the environmental impact of production processes. The model sets, underlying digital twins, can be described as workflows consisting of a set of computational services, each of which is a model of a process step. To provide flexible support for cloud computing to perform digital twins, we offer the concept of "Micro-Workflows", which combines the power of the Scientific Workflows concept, the flexibility of container technology and the sustainability of the Stream Processing approach in distributed computing systems.

Keywords: digital twins, microservices, containerization, micro-workflow, Kafka, Kepler.

FOR CITATION

Alaasam A.B.A., Radchenko G.I., Tchernykh A.N. Micro-Workflows: A Combination of Workflows and Data Streaming to Support Digital Twins of Production Processes. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 4. pp. 100–116. (in Russian) DOI: 10.14529/cmse190407.

References

1. Al-Jarrah O.Y. et al. Efficient Machine Learning for Big Data: A Review. Big Data Res. 2015. vol. 2, no. 3. pp. 87–93. DOI: 10.1016/j.bdr.2015.04.001.
2. Alaasam A.B.A. et al. Scientific Micro-Workflows : Where Event-Driven Approach Meets Workflows to Support Digital Twins. Proc. Int. Conf. Russ. Supercomput. Days, RuSCDays'18 (Moscow, Russia, September, 24–25, 2018). MSU. 2018. vol. 1. pp. 489–495.

3. Altintas I. et al. Kepler: an extensible system for design and execution of scientific workflows. Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 2004. vol. I. pp. 423–424. DOI: 10.1109/SSDM.2004.1311241.
4. Boettiger C. An introduction to Docker for reproducible research. ACM SIGOPS Oper. Syst. Rev. 2015. vol. 49, no. 1. pp. 71–79. DOI: 10.1145/2723872.2723882.
5. Borodulin K. et al. Towards Digital Twins Cloud Platform : Microservices and Computational Workflows to Rule a Smart Factory. Proc. The 10th Int. Conf. Util. Cloud Comput., UCC '17 (Austin, Texas, USA, December, 5–8, 2017). pp. 209–210. DOI: 10.1145/3147213.3149234.
6. Bryner M. Smart manufacturing: The next revolution. Chem. Eng. Prog. 2012. vol. 108, no. 10. pp. 4–12.
7. Carvalho O., Roloff E., Navaux P.O.A. A Distributed Stream Processing based Architecture for IoT Smart Grids Monitoring. Companion Proc. 10th Int. Conf. Util. Cloud Comput. (Austin, Texas, USA, December, 5–8, 2017). New York, NY, USA: ACM Press, 2017. pp. 9–14. DOI: 10.1145/3147234.3148105.
8. Davis J. et al. Smart manufacturing, manufacturing intelligence and demand-dynamic performance. Comput. Chem. Eng. 2012. vol. 47. pp. 145–156. DOI: 10.1016/j.compchemeng.2012.06.037.
9. Deelman E. et al. Pegasus, a workflow management system for science automation. Futur. Gener. Comput. Syst. 2015. vol. 46. pp. 17–35. DOI: 10.1016/j.future.2014.10.008.
10. Fahringer T. et al. ASKALON: a Grid application development and computing environment. 6th IEEE/ACM Int. Work. Grid Comput. 2005. pp. 10. DOI: 10.1109/GRID.2005.1542733.
11. Filgueira R. et al. IoT-Hub: New IoT data-platform for Virtual Research Environments. 10th Int. Work. Sci. Gateways, IWSG 2018. 2018. pp. 13–15.
12. Filguiera R. et al. Dispel4py: A Python framework for data-intensive scientific computing. Int. J. High Perform. Comput. Appl. 2017. vol. 31, no. 4. pp. 316–334. DOI: 10.1177/1094342016649766.
13. Foster I.T., Gannon D.B. Cloud Computing for Science and Engineering. MIT Press, 2017. 392 p.
14. Glaessgen E.H., Stargel D.S. The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. 53rd AIAA/ASME/ASCE/AHS/ASC Struct. Struct. Dyn. Mater. Conf. 20th AIAA/ASME/AHS Adapt. Struct. Conf. 14th AIAA. Reston, Virigina: American Institute of Aeronautics and Astronautics, 2012. pp. 1–14. DOI: 10.2514/6.2012-1818.
15. Hirales-Carbajal A. et al. A Grid simulation framework to study advance scheduling strategies for complex workflow applications. 2010 IEEE Int. Symp. Parallel Distrib. Process. Work. Phd Forum. IEEE, 2010. pp. 1–8. DOI: 10.1109/IPDPSW.2010.5470918.
16. Hirales-Carbajal A. et al. Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid. J. Grid Comput. 2012. vol. 10, no. 2. pp. 325–346. DOI: 10.1007/s10723-012-9215-6.
17. Huber N. et al. Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments. Proc. 1st Int. Conf. Cloud Comput. Serv. Sci. Noordwijkerhout, The Netherlands: SciTePress - Science and and Technology Publications, 2011. pp. 563–573.

18. Korambath P. et al. A smart manufacturing use case: Furnace temperature balancing in steam methane reforming process via kepler workflows. Procedia Comput. Sci. Elsevier Masson SAS, 2016. vol. 80. pp. 680–689. DOI: 10.1016/j.procs.2016.05.357.
19. Kostenetskiy P.S., Safonov A.Y. SUSU Supercomputer Resources. Proc. 10th Annu. Int. Sci. Conf. Parallel Comput. Technol., PCT 2016 (Arkhangelsk, Russia, March, 29–31, 2016). CEUR Workshop Proceedings, 2016. vol. 1576. pp. 561–573.
20. Lee J., Bagheri B., Kao H.A. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. Manuf. Lett. 2015. vol. 3. pp. 18–23. DOI: 10.1016/j.mfglet.2014.12.001.
21. Lewis J., Fowler M. Microservices. Available at: http://martinfowler.com/articles/microservices.html (accessed: 08.11.2019).
22. Li S., Xu L. Da, Zhao S. The internet of things: a survey . Inf. Syst. Front. 2015. vol. 17, no. 2. pp. 243–259. DOI: 10.1007/s10796-014-9492-7.
23. Perera C. et al. Sensing as a service model for smart cities supported by Internet of Things. Trans. Emerg. Telecommun. Technol. 2014. vol. 25, no. 1. pp. 81–93. DOI: 10.1002/ett.2704.
24. Radchenko G., Alaasam A., Tchernykh A. Micro-Workflows: Kafka and Kepler Fusion to Support Digital Twins of Industrial Processes. 2018 IEEE/ACM Int. Conf. Util. Cloud Comput. Companion, UCC Companion. 2018. pp. 83–88.
25. Radchenko G., Hudyakova E. A service-oriented approach of integration of computer-aided engineering systems in distributed computing environments. UNICORE Summit 2012, Proc. 2012. vol. 15. pp. 57–66. DOI: 10.1109/UCC-Companion.2018.00039.
26. Rahman M. et al. Adaptive workflow scheduling for dynamic grid and cloud computing environment. Concurr. Comput. Pract. Exp. 2013. vol. 25, no. 13. pp. 1816–1842. DOI: 10.1002/cpe.3003.
27. Sinha N., Pujitha K.E., Alex J.S.R. Xively based sensing and monitoring system for IoT. 2015 Int. Conf. Comput. Commun. Informatics, ICCCI 2015. 2015. pp. 1–6. DOI: 10.1109/ICCCI.2015.7218144.
28. Soldatos J. et al. OpenIoT: Open source internet-of-things in the cloud. Lect. Notes Comput. Sci. 2015. vol. 9001. pp. 13–25. DOI: 10.1007/978-3-319-16546-2_3.
29. Tao F. et al. Digital twin-driven product design, manufacturing and service with big data. Int. J. Adv. Manuf. Technol. 2017. vol. 94, no. 9–12. pp. 3563–3576. DOI: 10.1007/s00170-017-0233-1.
30. Taylor I. et al. Workflows for e-Science. Springer London, 2007. 523 p.
31. Tuegel E.J. et al. Reengineering aircraft structural life prediction using a digital twin. Int. J. Aerosp. Eng. 2011. pp. 1–14. DOI: 10.1155/2011/154798.
32. Xavier M.G. et al. Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments. 2013 21st Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. Belfast, UK: IEEE, 2013. pp. 233–240. DOI: 10.1109/PDP.2013.41.
33. Savchenko D.I., Radchenko G. Testing methodology for microservice cloud applications. Supercomputer Days in Russia Proceedings of the International Conference. Moscow: Moscow State University, 2015. pp. 245–256. (in Russian)

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет» (национальный исследовательский университет).

Главный редактор — Л.Б. Соколинский.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по научным специальностям и соответствующим им отраслям науки: 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки), 05.13.17 – Теоретические основы информатики (физико-математические науки).

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год.

Адрес редакции, издателя: 454080, г. Челябинск, проспект Ленина, 76, Издательский центр ЮУрГУ, каб. 32.

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `http://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.**
2. Адрес редакционной коллегии научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, кафедра СП,

   ответственному секретарю Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**