ISSN 1818-1015
eISSN 2313-5417

2024 Volume 31 No. 2

MODELING AND ANALYSIS OF INFORMATION SYSTEMS

SCIENTIFIC JOURNAL

Start date of publication — 1999
Published quarterly

FOUNDER
P.G. Demidov Yaroslavl State University

EDITORIAL OFFICE
14 Sovetskaya str., Yaroslavl 150003, Russian Federation

Website: http://mais-journal.ru
E-mail: mais@uniyar.ac.ru
Phone: +7 (4852) 79-77-73

2024 Том 31 № 2

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ

НАУЧНЫЙ ЖУРНАЛ

Издается с 1999 года
Выходит 4 раза в год

УЧРЕДИТЕЛЬ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Ярославский государственный университет им. П. Г. Демидова»

РЕДАКЦИЯ
ул. Советская, 14, Ярославль, 150003, Российская Федерация
Website: http://mais-journal.ru
E-mail: mais@uniyar.ac.ru
Телефон: +7 (4852) 79-77-73

Свидетельство о регистрации СМИ ПИ № ФС77-66186 от 20.06.2016 выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций. Подписной индекс в каталоге «Урал-Пресс» — 31907. Технический редактор, компьютерная вёрстка — К. В. Лагутина. Дата выхода в свет 30.06.2024. Формат 200×265 мм. Объем 106 с. Тираж 28 экз. Свободная цена. Заказ 24066. Издатель и его адрес: Ярославский государственный университет им. П. Г. Демидова; ул. Советская, 14, Ярославль, 150003, Россия. Типография и ее адрес: ООО «Филигрань»; ул. Свободы, 91, Ярославль, 150049, Россия.
Содержание предназначено для детей старше 12 лет.

Editor-in-Chief

Egor V. Kuzmin . . . . . Doctor of Sciences, P.G. Demidov Yaroslavl State University (Russia)

Deputy Editor-in-Chief

Vladimir A. Bashkin . . . . . Doctor of Sciences, P.G. Demidov Yaroslavl State University (Russia)

Editorial Board Secretary

Ilya V. Paramonov . . . . . Ph.D., P.G. Demidov Yaroslavl State University (Russia)

The Editorial Board

Sergei M. Abramov . . . . . Professor, Doctor of Sciences, Corresponding Member of Russian Academy of Sciences, Program Systems Institute of RAS (Pereslavl-Zalesskiy, Russia)
Lilian Aveneau . . . . . Professor, XLIM Laboratory, University of Poitiers (Poitiers, France)
Thomas Baar . . . . . Professor, Doctor, Hochschule für Technik und Wirtschaft Berlin, University of Applied Sciences (Berlin, Germany)
Olga L. Bandman . . . . . Professor, Doctor of Sciences, Supercomputer Software Department, Institute of Computational Mathematics and Mathematical Geophysics SB RAS (Novosibirsk, Russia)
Vladimir N. Belykh . . . . . Professor, Doctor of Sciences, Volga State Academy of Water Transport (Nizhny Novgorod, Russia)
Vladimir A. Bondarenko . . . Professor, Doctor of Sciences, P.G. Demidov Yaroslavl State University (Russia)
Richard R. Brooks . . . . . Professor, Clemson University (South Carolina, USA)
Alex Dekhtyar . . . . . Professor, California Polytechnic State University (Cal Poly, California, USA)
Mikhail Dmitriev . . . . . Professor, Doctor of Sciences, Higher School of Economics (Moscow, Russia)
Vladimir L. Dolnikov . . . . . Doctor of Sciences, Moscow Institute of Physics and Technology (Moscow, Russia)
Valery G. Durnev . . . . . Professor, Doctor of Sciences, P.G. Demidov Yaroslavl State University (Russia)
Sergey D. Glyzin . . . . . Professor, Doctor of Sciences, P.G. Demidov Yaroslavl State University (Russia)
Yuri G. Karpov . . . . . Professor, Doctor of Sciences, St-Petersburg State Polytechnical University (Russia)
Sergey A. Kashchenko . . . . . Professor, Doctor of Sciences, P.G. Demidov Yaroslavl State University (Russia)
Lev S. Kazarin . . . . . Professor, Doctor of Sciences, P.G. Demidov Yaroslavl State University (Russia)
Andrei Yu. Kolesov . . . . . Professor, Doctor of Sciences, P.G. Demidov Yaroslavl State University (Russia)
Olga Kouchnarenko . . . . . Professor at the Burgundy-Franche-Comte University, The FEMTO-ST Institute (CNRS 6174) (Besancon, France)
Nikolai A. Kudryashov . . . . Professor, Doctor of Sciences, MEPhI (Russia)
Irina A. Lomazova . . . . . Professor, Doctor of Sciences, Higher School of Economics (Moscow, Russia)
George G. Malinetskiy . . . . Professor, Doctor of Sciences, M.V. Keldysh Institute of Applied Mathematics RAS (Moscow, Russia)
Victor E. Malyshkin . . . . . Professor, Doctor of Sciences, Institute of Computational Mathematics and Mathematical Geophysics SB RAS (Novosibirsk, Russia)
Alexander V. Mikhailov . . . Professor, Doctor of Sciences, University of Leeds, School of Mathematics (Leeds, Great Britain)
Nikolai Kh. Rozov . . . . . Professor, Doctor of Sciences, Lomonosov Moscow State University (Russia)
Philippe Schnoebelen . . . . . Senior Researcher, LSV, CNRS & ENS de Cachan (CACHAN, France)
Natalia Sidorova . . . . . Dr., Assistant Professor, Architecture of Information Systems group, Technische universiteit Eindhoven (Eindhoven, Netherlands)
Ruslan L. Smeliansky . . . . . Professor, Doctor of Sciences, Corresponding Member of RAS, Lomonosov Moscow State University (Russia)
Valery A. Sokolov . . . . . Professor, Doctor of Sciences, P.G. Demidov Yaroslavl State University (Russia)
Javid Taheri . . . . . Associate Professor, Ph.D., Karlstad University (Sweden)
Eugeniy A. Timofeev . . . . . Professor, Doctor of Sciences, P.G. Demidov Yaroslavl State University (Russia)
Mark Trakhtenbrot . . . . . Dr., Holon Institute of Technology (Holon, Israel)
Dimitry Turaev . . . . . Professor of Applied Mathematics & Mathematical Physics, Imperial College (London, Great Britain)
Vladimir Zakharov . . . . . Doctor of Sciences, Professor, Lomonosov Moscow State University (Russia)

Главный редактор

Е. В. Кузьмин . . . . . д-р физ.-мат. наук, ЯрГУ (Россия)

Заместитель главного редактора

В. А. Башкин . . . . . д-р физ.-мат. наук, ЯрГУ (Россия)

Ответственный секретарь

И. В. Парамонов . . . . . канд. физ.-мат. наук, ЯрГУ (Россия)

Редакционная коллегия

С. М. Абрамов . . . . . д-р физ.-мат. наук, чл.-корр. РАН, Институт программных систем РАН им. А.К. Айламазяна (Россия)
L. Aveneau . . . . . проф., Университет Пуатье (Франция)
T. Baar . . . . . д-р наук, проф., Университет прикладных технических и экономических наук Берлина (Германия)
О. Л. Бандман . . . . . д-р техн. наук, Институт вычислительной математики и математической геофизики СО РАН (Россия)
В. Н. Белых . . . . . д-р физ.-мат. наук, проф., Волжская государственная академия водного транспорта (Россия)
В. А. Бондаренко . . . . . д-р физ.-мат. наук, проф., ЯрГУ (Россия)
R. Brooks . . . . . проф., Университет Клемсона (США)
С. Д. Глызин . . . . . д-р физ.-мат. наук, проф., ЯрГУ (Россия)
A. Dekhtyar . . . . . проф., Калифорнийский политехнический университет, департамент компьютерных наук (США)
М. Г. Дмитриев . . . . . д-р физ.-мат. наук, проф., ВШЭ (Россия)
В. Л. Дольников . . . . . д-р физ.-мат. наук, проф., МФТИ (Россия)
В. Г. Дурнев . . . . . д-р физ.-мат. наук, проф., ЯрГУ (Россия)
В. А. Захаров . . . . . д-р физ.-мат. наук, проф., МГУ (Россия)
Л. С. Казарин . . . . . д-р физ.-мат. наук, проф., ЯрГУ (Россия)
Ю. Г. Карпов . . . . . д-р техн. наук, проф., Санкт-Петербургский государственный технический университет (Россия)
С. А. Кащенко . . . . . д-р физ.-мат. наук, проф., ЯрГУ (Россия)
А. Ю. Колесов . . . . . д-р физ.-мат. наук, проф., ЯрГУ (Россия)
Н. А. Кудряшов . . . . . д-р физ.-мат. наук, проф., Засл. деятель науки РФ, МИФИ (Россия)
O. Kouchnarenko . . . . . проф., Университет Бургундии - Франш-Комтэ (Франция)
И. А. Ломазова . . . . . д-р физ.-мат. наук, проф., ВШЭ (Россия)
Г. Г. Малинецкий . . . . . д-р физ.-мат. наук, проф., Институт прикладной математики им. М.В. Келдыша РАН (Россия)
В. Э. Малышкин . . . . . д-р техн. наук, проф., Институт вычислительной математики и математической геофизики СО РАН (Россия)
A. Mikhailov . . . . . д-р физ.-мат. наук, проф., Университет Лидса (Великобритания)
Н. Х. Розов . . . . . д-р физ.-мат. наук, проф., чл.-корр. РАО, МГУ (Россия)
N. Sidorova . . . . . д-р наук, университет Эйндховена (Нидерланды)
Р. Л. Смелянский . . . . . д-р физ.-мат. наук, проф., член-корр. РАН, академик РАЕН, МГУ (Россия)
В. А. Соколов . . . . . д-р физ.-мат. наук, проф., ЯрГУ (Россия)
J. Taheri . . . . . доцент, Университет Карлстада (Швеция)
Е. А. Тимофеев . . . . . д-р физ.-мат. наук, проф., ЯрГУ (Россия)
M. Trakhtenbrot . . . . . д-р комп. наук, Холонский технологический институт (Израиль)
D. Turaev . . . . . проф., Имперский колледж Лондона (Великобритания)
Ph. Schnoebelen . . . . . проф., Национальный центр научных исследований и Высшая нормальная школа Кашана (Франция)

Contents

Содержание

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 2, 2024
journal homepage: www.mais-journal.ru

Verification of Declarative LTL-specification of Control Programs Behavior

M. V. Neyzov[1], E. V. Kuzmin[2]

DOI: 10.18255/1818-1015-2024-2-120-141

[1]Institute of Automation and Electrometry SB RAS, Novosibirsk, Russia
[2]P.G. Demidov Yaroslavl State University, Yaroslavl, Russia

MSC2020: 68N30
Research article
Full text in Russian

Received May 3, 2024
Revised May 17, 2024
Accepted May 22, 2024

The article continues the series of works on development and verification of control programs based on LTL-specifications of a special type. Previously, it was proposed a declarative LTL-specification, which allows describing the behavior of control programs and building program code based on it in the imperative ST-language for programmable logic controllers. The LTL-specification can be directly verified for compliance with specified temporal properties by the model checking method using the nuXmv symbolic verification tool. In general, it is not required translating LTL-formulas of the specification into another formalism — an SMV-specification (code in the input language of the nuXmv tool).

The purpose of this work is to explore alternative ways of representing a program behavior model corresponding to the declarative LTL-specification during its verification within the nuXmv tool.

In the article, we transform the declarative LTL-specification into various SMV-specifications with accompanying changes of formulation of the verification problem, what leads to a significant reduction in time costs when checking temporal properties by using the nuXmv tool. The acceleration of verification is due to the reduction of the state space of a model being verified. The SMV-specifications obtained as a result of the proposed transformations specify identical or bisimulationally equivalent transition systems. It is ensuring the same verification results when replacing one SMV-specification with another.

Keywords: control software; PLC program; declarative LTL-specification; imperative LTL-specification; complete transition system; pseudo-complete transition system; state space; model checking; nuXmv verifier; SMV-specification; bisimulation equivalence; bisimulation

INFORMATION ABOUT THE AUTHORS

Neyzov, Maxim V. (corresponding author) | ORCID iD: 0009-0000-6893-6137. E-mail: neyzov.max@gmail.com
Researcher

Kuzmin, Egor V. | ORCID iD: 0000-0003-0500-306X. E-mail: kuzmin@uniyar.ac.ru
Head of the Chair of Theoretical Informatics, Dr. Sc.

Funding: State task IAaE SB RAS, project No. 122031600173-8; Yaroslavl State University (project VIP-016).

For citation: M. V. Neyzov and E. V. Kuzmin, "Verification of declarative LTL-specification of control programs behavior", *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 120–141, 2024. DOI: 10.18255/1818-1015-2024-2-120-141.

© Neyzov M. V., Kuzmin E. V., 2024
This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 2, 2024
сайт журнала: www.mais-journal.ru

THEORY OF SOFTWARE

Верификация декларативной LTL-спецификации поведения управляющих программ

М. В. Нейзов[1], Е. В. Кузьмин[2]

DOI: 10.18255/1818-1015-2024-2-120-141

[1]Институт автоматики и электрометрии СО РАН, Новосибирск, Россия
[2]Ярославский государственный университет им. П.Г. Демидова, Ярославль, Россия

УДК 004.424+519.683.8
Научная статья
Полный текст на русском языке

Получена 3 мая 2024 г.
После доработки 17 мая 2024 г.
Принята к публикации 22 мая 2024 г.

Статья продолжает цикл трудов по разработке и верификации управляющих программ на основе LTL-спецификаций специального вида. Ранее была предложена декларативная LTL-спецификация, позволяющая описывать поведение управляющих программ и выполнять построение по ней программного кода на императивном языке ST для программируемых логических контроллеров. Данная LTL-спецификация может быть непосредственно верифицирована на предмет соответствия заданным темпоральным свойствам методом проверки модели (model checking) с помощью инструмента символьной верификации nuXmv. При этом не требуется переводить LTL-формулы спецификации в другой формализм — SMV-спецификацию (код на входном языке инструмента nuXmv).

Цель настоящей работы состоит в исследовании альтернативных способов представления модели поведения программы, соответствующей декларативной LTL-спецификации, при её верификации в рамках инструментального средства nuXmv.

В статье выполняются преобразования декларативной LTL-спецификации в различные SMV-спецификации с сопутствующими изменениями постановки задачи верификации, что приводит к значительному снижению временных затрат при проверке темпоральных свойств с использованием инструмента nuXmv. Ускорение верификации обусловлено сокращением пространства состояний проверяемой модели. Полученные в результате предложенных преобразований SMV-спецификации задают одинаковые или бисимуляционно эквивалентные системы переходов, обеспечивая неизменность результатов верификации при замене одной SMV-спецификации на другую.

Ключевые слова: управляющее программное обеспечение; ПЛК-программа; декларативная LTL-спецификация; императивная LTL-спецификация; полная система переходов; псевдополная система переходов; пространство состояний; проверка модели; верификатор nuXmv; SMV-спецификация; бисимуляционная эквивалентность; бисимуляция

ИНФОРМАЦИЯ ОБ АВТОРАХ

Нейзов, Максим Вячеславович (автор для корреспонденции)	ORCID iD: 0009-0000-6893-6137. E-mail: neyzov.max@gmail.com Исследователь
Кузьмин, Егор Владимирович	ORCID iD: 0000-0003-0500-306X. E-mail: kuzmin@uniyar.ac.ru Заведующий кафедрой теоретической информатики, доктор физ.-мат. наук

Финансирование: Госзадание ИАиЭ СО РАН, проект № 122031600173-8; ЯрГУ (проект VIP-016).

Для цитирования: M. V. Neyzov and E. V. Kuzmin, “Verification of declarative LTL-specification of control programs behavior”, *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 120–141, 2024. DOI: 10.18255/1818-1015-2024-2-120-141.

© Нейзов М. В., Кузьмин Е. В., 2024
Эта статья открытого доступа под лицензией CC BY license (https://creativecommons.org/licenses/by/4.0/).

Введение

В связи с ростом внедрений ответственных программно-управляемых технических систем [1] повышается актуальность задачи разработки и верификации [2] *управляющего программного обеспечения* (УПО). Работа УПО выполняется циклически. *Цикл управления* состоит в получении информации об управляемом объекте (информация фиксируется во входных переменных), непосредственной работе программы (вычислении новых значений внутренних/выходных переменных) и передаче команд на объект управления. Объектом управления может выступать как программный, так и физический объект. УПО при этом является частью программной или *киберфизической системы* [3, 4] соответственно.

В работе [5] была предложена LTL-спецификация специального вида, предназначенная для описания поведения УПО *трансформационных* и *реактивных систем* [6, 7]. Эта LTL-спецификация является конструктивной в том смысле, что по ней может быть построен программный код УПО на некотором стандартном [8] языке программирования. Предварительно LTL-спецификация может быть непосредственно верифицирована на предмет соответствия заданным *темпоральным* свойствам методом *проверки модели* (model checking) [9—11] с помощью инструмента символьной верификации nuXmv [12]. При этом не требуется переводить LTL-формулы спецификации в другой формализм — SMV-спецификацию (код на входном языке инструмента nuXmv).

Цель настоящей работы — исследовать альтернативные способы представления модели поведения программы, соответствующей предложенной LTL-спецификации, для её верификации в рамках инструментального средства nuXmv. Замена декларативной LTL-спецификации на SMV-спецификацию и изменение постановки задачи верификации могут привести к значительному сокращению времени при проверке темпоральных свойств программы.

Содержание работы. Раздел 1 представляет собой краткое содержание работы [5]. В разделе 2 приведены примеры декларативной и императивной LTL-спецификаций поведения программы возведения числа в квадрат для *программируемого логического контроллера* (ПЛК) [13]. В разделе 3 представлены LTL-свойства и проведена верификация данных LTL-спецификаций согласно схеме, предложенной в работе [5]. Раздел 4 содержит процедуры преобразования LTL-спецификаций в соответствующие декларативную и императивную SMV-спецификации. Проводится их верификация. Раздел 5 содержит дополнительные процедуры преобразования SMV-спецификаций. Доказывается, что данные процедуры сохраняют бисимуляционную эквивалентность между системами переходов, которые задаются этими SMV-спецификациями. Проводится верификация новых SMV-спецификаций. Демонстрируется, что представленные в статье преобразования приводят к снижению времени верификации. В заключительном разделе подводятся итоги.

1. LTL-спецификация поведения управляющих программ

LTL-спецификация предназначена для описания и верификации модели поведения УПО. В работе [5] представлены следующие базовые положения. Программа содержит основные $V = \{v_1, \ldots, v_n\}$ и вспомогательные $\_V = \{\_v_1, \ldots, \_v_n\}$ переменные. Основные переменные из V предназначены для хранения всей необходимой информации для построения программы — это могут быть входные, внутренние и выходные переменные. Вспомогательные переменные из $\_V$ всегда хранят значения соответствующих переменных из V, полученные на предыдущем цикле управления. В LTL-спецификации используются все переменные $V \cup \_V$. Вспомогательные переменные из $\_V$ позволяют детектировать изменения значений основных переменных из V: если значение v_i не равно значению $\_v_i$, где $i = 1, \ldots, n$, то значение переменной v_i изменилось. Также вспомогательные переменные множества $\_V$ могут использоваться для определения значений основных переменных из V. Например, выражение $v_1 = \_v_1 + 1$ утверждает, что значение переменной v_1 увеличивается на единицу.

Пусть переменные $v_1, \ldots, v_n$ и $\_v_1, \ldots, \_v_n$ принимают значения из соответствующих бесконечных множеств $D_1, \ldots, D_n$. Тогда множество $S = (D_1 \times \ldots \times D_n)^2$ является пространством возможных состояний программы. Состояние $s = (\boldsymbol{a}, \_\boldsymbol{a}) \in S$, где $\boldsymbol{a}$ и $\_\boldsymbol{a}$ — значения соответствующих векторов $\mathrm{v} = (v_1, \ldots, v_n)$ и $\_\mathrm{v} = (\_v_1, \ldots, \_v_n)$. Каждый цикл управления приводит к *переходу* между состояниями из S. Если состояние в результате цикла управления не изменилось, то происходит переход в это же самое состояние. Совокупность всех возможных переходов образует поведение программы. Моделью поведения в [5] является *размеченная система переходов* (labelled transition system) $LTS = \langle S, S_0, R, P, L \rangle$, где S — множество состояний, $S_0 \subseteq S$ — множество начальных состояний, $R \subseteq S \times S$ — *тотальное* отношение переходов, $P = \{p_i \mid i \in \mathbb{N}\}$ — множество произвольных атомарных утверждений относительно значений переменных из $V \cup \_V$, $L\colon S \to 2^P$ — функция разметки состояний атомарными утверждениями, истинными в этих состояниях. Свойство тотальности отношения переходов R имеет вид: $(\forall s \in S)\ (\exists s' \in S)\ (s, s') \in R$, т. е. из любого состояния $s \in S$ существует переход из R.

Обозначим S^ω множество всех бесконечных слов в алфавите S. *Путь* — бесконечная последовательность $\pi \in S^\omega$. Система переходов LTS задаёт множество всех возможных в ней путей Π_{LTS} — путей, начинающихся в начальных состояниях:

$$\Pi_{LTS} = \{\, \pi \in S^\omega \mid \big(\pi(0) \in S_0\big) \wedge (\forall i \ \in \mathbb{N}_0)\ \big(\pi(i), \pi(i+1)\big) \in R \,\},$$

где $\pi(i)$ — i-ое состояние пути π, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Абсолютно недетерминированной программой назовём программу с недетерминированным поведением всех её переменных из $V \cup \_V$. Моделью поведения такой программы будет *полная* [5] система переходов $cLTS = \langle S, S_0, R_c, P, L \rangle$, где $S_0 = S$, отношение переходов $R_c = S \times S$, т. е. в $cLTS$ возможен переход из любого состояния $s \in S$ в любое состояние $s' \in S$, что соответствует абсолютно недетерминированному поведению.

Наложим на полную систему переходов $cLTS$ ранее оговорённое ограничение на поведение всех её вспомогательных переменных из $\_V$, которое заключается в том, что их значения всегда равны соответствущим значениям переменных из V на предыдущем цикле управления. В итоге получим *псевдополную* [5] систему переходов $pLTS = \langle S, S_0, R'_c, P, L \rangle$, где $S_0 = S$, $R'_c = \{\big((\boldsymbol{a}, \_\boldsymbol{a}), (\boldsymbol{a}', \boldsymbol{a})\big) \in R_c\}$, векторы $(\boldsymbol{a}, \_\boldsymbol{a}) \in S$ и $(\boldsymbol{a}', \boldsymbol{a}) \in S$. Псевдополная система переходов $pLTS$ описывает поведение абсолютно недетерминированной программы, в которой предыдущие значения переменных $v_1, \ldots, v_n$ сохраняются в переменных $\_v_1, \ldots, \_v_n$.

Определим синтаксис и семантику формул линейной темпоральной логики LTL (Linear Temporal Logic). Пусть $p \in P$, тогда с учётом этого формулы LTL имеют следующую грамматику:

$$\varphi, \psi ::= \ \mathit{true} \mid \mathit{false} \mid p \mid \neg\, \varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \Rightarrow \psi \mid \mathbf{X}\varphi \mid \psi \mathbf{U} \varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi.$$

Индуктивно определим отношение выполнимости $\models$ формулы φ логики LTL для произвольного состояния s_i, где $i \in \mathbb{N}_0$, некоторого пути $\pi = s_0\, s_1\, s_2 \ldots$ системы переходов:

$$\begin{array}{ll}
s_i \models \mathit{true}; & s_i \not\models \mathit{false}; \\
s_i \models p & \Longleftrightarrow p \in L(s_i); \\
s_i \models \neg\, \varphi & \Longleftrightarrow s_i \not\models \varphi; \\
s_i \models \varphi \wedge \psi & \Longleftrightarrow s_i \models \varphi \ \text{ и } \ s_i \models \psi; \\
s_i \models \varphi \vee \psi & \Longleftrightarrow s_i \models \varphi \ \text{ или } \ s_i \models \psi; \\
s_i \models \varphi \Rightarrow \psi & \Longleftrightarrow s_i \models \neg\varphi \ \text{ или } \ s_i \models \psi; \\
s_i \models \mathbf{X}\varphi & \Longleftrightarrow s_{i+1} \models \varphi; \\
s_i \models \psi \mathbf{U} \varphi & \Longleftrightarrow (\exists k \geq i)\ s_k \models \varphi \ \text{ и } \ (\forall j,\ i\ \leq j < k)\ s_j \models \psi;
\end{array}$$

$$s_i \models \mathbf{F}\varphi \iff (\exists k \geq i)\ s_k \models \varphi;$$
$$s_i \models \mathbf{G}\varphi \iff (\forall j \geq i)\ s_j \models \varphi.$$

Также определим семантику отношения $\models$ на путях и системах переходов:

$$\pi \models \varphi \iff \pi(0) \models \varphi;$$
$$\Pi \models \varphi \iff (\forall \pi \in \Pi)\ \pi \models \varphi;$$
$$LTS, s \models \varphi \iff (\forall \pi \in \Pi_{LTS})\ [(\pi(0) = s) \Rightarrow (\pi \models \varphi)];$$
$$LTS \models \varphi \iff (\forall s \in S_0)\ LTS, s \models \varphi.$$

В работе [5] LTL-формулы используются для:

1. Задания ограничений для псевдополной системы переходов *pLTS*. Таким образом, на основе *pLTS* формируется модель поведения программы *LTS*.
2. Формализации требуемых свойств модели поведения программы *LTS*.

LTL-спецификация накладывает ограничения на псевдополную систему переходов *pLTS* путём индивидуального ограничения поведения переменных из V. *Декларативная* LTL-спецификация поведения переменной $v \in V$ имеет следующий вид [5]:

$$\mathbf{GX}\big(\neg(v = \_v) \Rightarrow cond_1 \wedge (v = expr_1) \vee \ldots \vee cond_k \wedge (v = expr_k)\big) \wedge$$
$$\mathbf{GX}\big((v = \_v) \Rightarrow \neg\,(cond_1 \vee \ldots \vee cond_k)\big). \tag{1}$$

Данная формула описывает, каким образом происходит изменение значения переменной $v \in V$. Логическое выражение $cond_i$ является необходимым и достаточным условием для изменения значения переменной v согласно выражению $expr_i$, где $i = 1, \ldots, k$. В выражениях $cond_i$ и $expr_i$ могут использоваться любые переменные из $(V \cup \_V) \setminus \{v\}$, константы, логические и арифметические операторы, а также операторы сравнения. Отметим, что в (1) выражения вида $v = \_v$ и $v = expr_i$ являются элементарными высказываниями из множества P, а выражения вида $cond_i$ — пропозициональными формулами (LTL-формулами без темпоральных операторов).

Декларативная LTL-спецификация (1) поведения переменной $v \in V$ должна удовлетворять условию *изменчивости* значения переменной [5]:

$$\mathbf{GX}\big(cond_1 \Rightarrow \neg\,(\_v = expr_1)\big) \wedge \ldots \wedge \mathbf{GX}\big(cond_k \Rightarrow \neg\,(\_v = expr_k)\big), \tag{2}$$

т. е. при истинном условии $cond_i$ выражение $expr_i$ должно возвращать значение, отличное от прошлого значения переменной v. Также для (1) должна выполняться *ортогональность* условий изменения значения переменной для любых $i, j = 1, \ldots, k$ при $i \neq j$ [5]:

$$\mathbf{GX}\big(cond_i \Rightarrow \neg cond_j\big), \tag{3}$$

т. е. одновременно истинным может быть не более одного условия $cond_i$.

Изначально поведение всех переменных из V недетерминировано — моделью поведения такой программы является псевдополная система переходов *pLTS*. Далее на псевдополную систему переходов *pLTS* накладывается ряд ограничений в виде декларативных LTL-спецификаций поведения переменных из V. Задекларированное поведение переменной становится строго детерминированным. В конечном итоге должно быть задекларировано поведение всех переменных из V, кроме входных. Сценарии работы программы зависят от последовательностей значений входных переменных. Поэтому для сохранения всего многообразия сценариев работы поведение всех входных переменных должно оставаться абсолютно недетерминированным. *Декларативная* LTL-спецификация поведения *неинициализированной* программы φ_{var} — конъюнкция формул вида (1) для переменных из V при соблюдении условий (2) и (3).

LTL-формула φ_0 специального вида $I(v_1, \ldots, v_n) \wedge (\_v_1 = v_1) \wedge \ldots \wedge (\_v_n = v_n)$ без темпоральных операторов используется для инициализации модели программы. Здесь $I(v_1, \ldots, v_n)$ — предикат, ограничивающий значения переменных из V. Начальные значения переменных из $\_V$ всегда равны начальным значениям соответствующих переменных из V. Формула $\varphi = \varphi_{var} \wedge \varphi_0$ является *декларативной* LTL-спецификацией (обозначим *dcl_LTL*) поведения программы.

Если в декларативной LTL-спецификации (1) $cond_i$ или $expr_i$, где $i = 1, \ldots, k$, содержит переменную $v' \in V$, то переменная v непосредственно зависит от переменной v'. Обозначим $(v, v') \in Dep$, где $Dep \subseteq V \times V$, бинарное отношение непосредственной зависимости переменных (Dep от англ. *Dependency*). Обозначим Dep^T транзитивное замыкание отношения Dep:

$$Dep^T = \{(v \in V, v' \in V) \mid \exists\, (v_1, \ldots, v_n)\, [(v_1 = v) \wedge (v_n = v') \wedge \forall i \ \in \{1, \ldots, n-1\}\, (v_i, v_{i+1}) \in Dep]\},$$

т. е. $(v, v') \in Dep^T$ тогда и только тогда, когда v непосредственно зависит от v' или существует цепочка зависимостей между v и v'. Таким образом, $Dep^T \subseteq V \times V$ — бинарное отношение непосредственной и опосредованной зависимости переменных.

При построении программ никакие переменные не должны зависеть сами от себя ни непосредственно, ни опосредованно через другие переменные. Поэтому декларативная LTL-спецификация φ поведения программы должна удовлетворять условию *запрета циклических зависимостей*:

$$\forall v \in V\ [(v, v) \notin Dep^T]. \tag{4}$$

С помощью декларативной LTL-спецификации φ на основе псевдополной системы переходов *pLTS* определяется система переходов *LTS*, которая является моделью поведения программы. Пусть $pLTS = \langle S_{pLTS}, S_0, R_{pLTS}, P, L\rangle$, а $LTS = \langle S, S'_0, R_\varphi, P, L'\rangle$, тогда отношение переходов R_φ имеет вид:

$$R_\varphi = \{\, (s_1, s_2) \in R_{pLTS} \mid (\exists \pi \in \Pi_{pLTS})\ \pi = \sigma s_1 s_2 \ldots \models \varphi \,\},$$

где Π_{pLTS} — множество всех путей системы переходов *pLTS*, σ — конечный фрагмент пути, $|\sigma| \in \mathbb{N}_0$. Таким образом, система переходов *LTS* содержит только те пути из *pLTS*, которые удовлетворяют декларативной LTL-спецификации φ. По определению R_φ является тотальным отношением переходов, так как содержит переходы, которые являются частью бесконечной последовательности состояний.

Множество состояний S определяется следующим образом:

$$S = \{\, s \in S_{pLTS} \mid (\exists s' \in S_{pLTS})\ [(s, s') \in R_\varphi \vee (s', s) \in R_\varphi] \,\},$$

т. е. множество S содержит только те состояния, которые участвуют в переходах (присутствуют в R_φ). Множество начальных состояний $S'_0 = \{\, s \in S \mid s \models \varphi_0 \,\}$, где φ_0 — формула инициализации. Функция разметки $L' : S \to 2^P$ совпадает с L на множестве S, т. е. $(\forall s \in S)\ L'(s) = L(s)$.

В итоге полученная система переходов *LTS* задаёт то же самое множество путей, что и декларативная LTL-спецификация φ на псевдополной системе переходов: $\Pi_{LTS} = [\![\varphi]\!]_{pLTS}$. Нотация $[\![\varphi]\!]_{LTS}$ означает множество всех путей в *LTS*, на которых истинна формула φ, т. е. $[\![\varphi]\!]_{LTS} = \{\pi \in \Pi_{LTS} \mid \pi \models \varphi\}$.

Императивная LTL-спецификация поведения переменной $v \in V$ [5]:

$$\begin{aligned}\mathbf{GX}\big(cond_1 \Rightarrow (v = expr_1)\big) \wedge \ldots \wedge \mathbf{GX}\big(cond_k \Rightarrow (v = expr_k)\big) \wedge \\ \mathbf{GX}\big(\neg\, cond_1 \wedge \ldots \wedge \neg\, cond_k \Rightarrow (v = \_v)\big).\end{aligned} \tag{5}$$

Данная спецификация описывает поведение переменной в императивном стиле «если. . . , то. . . ». Императивная LTL-спецификация (обозначим *imp_LTL*) поведения программы является *конструктивной*, т. е. по ней непосредственно может быть построен код программы/модели на императивном языке программирования/моделирования. Декларативная и императивная LTL-спецификации

задают одно и то же поведение — в работе [5] доказана их эквивалентность. Проведена оценка выразительности этих спецификаций — обе спецификации являются Тьюринг-полными [5] при определении логики LTL над бесконечным множеством элементарных высказываний $P = \{p_i \mid i \in \mathbb{N}\}$. Заметим, что Тьюринг-полнота LTL-спецификаций позволяет с их помощью задавать поведение любой машины Тьюринга.

В работе [5] для разрешимости в общем случае задачи проверки модели (model checking) система переходов *cLTS* была ограничена — множества значений переменных $D_1, \ldots, D_n$ имеют конечное число элементов. В этом случае пространство состояний программы S также является конечным. Для соблюдения этого ограничения LTL-спецификация поведения переменной $v_i \in V$ должна удовлетворять условию *ограниченности*:

$$\mathbf{GX}\big(cond_1 \Rightarrow (val(expr_1) \in D_i)\big) \wedge \ldots \wedge \mathbf{GX}\big(cond_k \Rightarrow (val(expr_k) \in D_i)\big), \tag{6}$$

т. е. если условие $cond_j$ истинно ($j = 1, \ldots, k$), то выражение $expr_j$ возвращает значение $val(expr_j)$, которое принадлежит области значений данной переменной. Формула инициализации φ_0 также должна задавать начальные значения переменных, попадающие в допустимую область значений.

В данной статье в качестве области значений некоторой переменной рассматривается либо булево множество $\mathbb{B} = \{0, 1\}$, либо конечное подмножество множества натуральных чисел $D \subset \mathbb{N}_0$.

При задании LTL-спецификации φ поведения конкретной программы и свойств $\Psi = \{\psi_1, \ldots, \psi_h\}$ для её проверки из бесконечного множества атомарных высказываний $\{p_i \mid i \in \mathbb{N}\}$ отбирается конечное множество высказываний $P = \{p_1, \ldots, p_m\}$, необходимых для построения φ и Ψ.

2. LTL-спецификация поведения программы возведения числа в квадрат

В работе [5] была разработана декларативная LTL-спецификация поведения ПЛК-программы возведения числа n в квадрат. Спецификация содержит основные $V = \{n, a, b, c, q, PBStart, PBReset, PBPls, PBMns\}$ и вспомогательные $\_V = \{\_n, \_a, \_b, \_c, \_q, \_PBStart, \_PBReset, \_PBPls, \_PBMns\}$ переменные. Переменная n хранит входное значение, переменные a, b, c — результаты промежуточных вычислений, q — управляющее состояние. При завершении работы алгоритма результат содержится в переменной c. Булевы переменные *PBStart, PBReset, PBPls, PBMns* предназначены соответственно для запуска вычисления, перевода алгоритма в начальное управляющее состояние, увеличения и уменьшения на единицу значения переменной n.

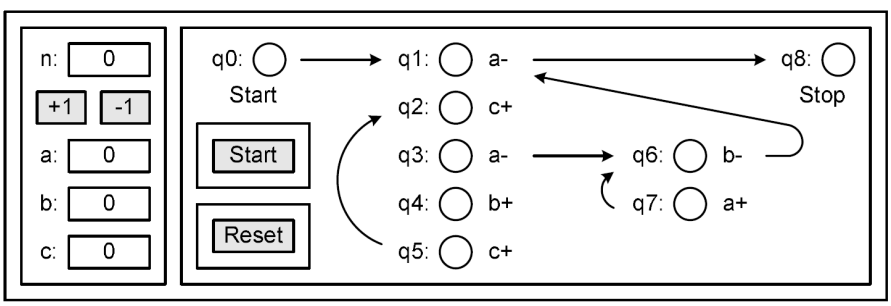


Fig. 1. PLC control panel

Рис. 1. Панель управления ПЛК

К ПЛК подключена панель управления (рис. 1). Интерфейс ПЛК для её подключения представлен на рис. 2. Индикаторы «n», «a», «b», «c» отображают текущие значения одноименных переменных, лампы «q0», ..., «q8» — текущее значение переменной q. Кнопки «Start», «Reset», «+1» и «-1» связаны с булевыми переменными *PBStart, PBReset, PBPls, PBMns* соответственно. Нажатие кнопки

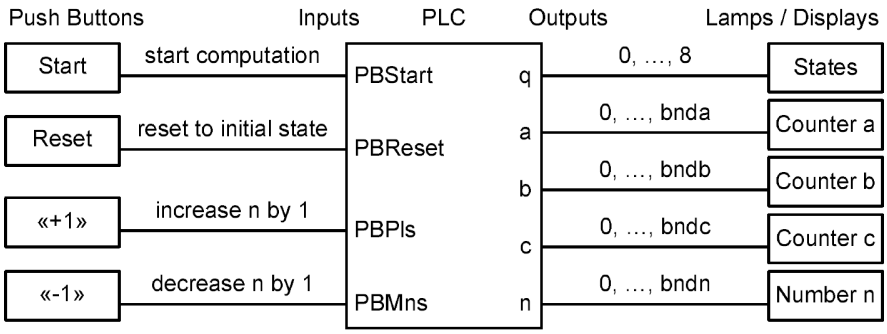


Fig. 2. PLC interface

Рис. 2. Интерфейс ПЛК

выставляет для булевой переменной значение «истина». Спецификация поведения была построена на базе счётчиковой машины, поэтому панель управления также содержит графическое представление её работы. Видно, что $q0$ — начальное управляющее состояние, $q8$ — финальное.

Декларативная LTL-спецификация ПЛК-программы возведения числа в квадрат в синтаксисе инструмента nuXmv представлена в листинге 1, императивная LTL-спецификация — в листинге 2.

Листинг 1 (dcl_LTL-спецификация ПЛК-программы)**:**

```
-- S0:
   q=0 & n=0 & a=0 & b=0 & c=0 & _q=q & _n=n & _a=a & _b=b & _c=c &
   !PBStart & !PBReset & !PBPls & !PBMns & !_PBStart & !_PBReset & !_PBPls & !_PBMns &
-- Sn:
   G X( !(n = _n) ->   _q=0 & _n<bndn  &
                       !_PBPls & PBPls & !PBMns & !PBStart & (n = _n + 1) |
                       _q=0 & _n>0     &
                       !_PBMns & PBMns & !PBPls & !PBStart & (n = _n - 1)  ) &
   G X(  (n = _n) -> !(_q=0 & _n<bndn & !_PBPls & PBPls & !PBMns & !PBStart |
                       _q=0 & _n>0    & !_PBMns & PBMns & !PBPls & !PBStart) ) &
-- Sa:
   G X( !(a = _a) ->   _q=0 & PBStart & !(_a=n) & (a = n)      |
                       _q=7 & _a<bnda           & (a = _a + 1) |
                       _q=1 & _a>0              & (a = _a - 1) |
                       _q=3 & _a>0              & (a = _a - 1) ) &

   G X(  (a = _a) -> !(_q=0 & PBStart & !(_a=n) | _q=7 & _a<bnda |
                       _q=1 & _a>0              | _q=3 & _a>0     ) ) &
-- Sb:
   G X( !(b = _b) ->   _q=4 & _b<bndb & (b = _b + 1) |
                       _q=6 & _b>0    & (b = _b - 1)  ) &
   G X(  (b = _b) -> !(_q=4 & _b<bndb | _q=6 & _b>0)  ) &
-- Sc:
   G X( !(c = _c) ->   _q=2 & _c<bndc        & (c = _c + 1) |
                       _q=5 & _c<bndc        & (c = _c + 1) |
                       _q=8 & _c>0 & PBReset & (c = 0)       ) &
   G X(  (c = _c) -> !(_q=2 & _c<bndc | _q=5 & _c<bndc | _q=8 & _c>0 & PBReset) ) &
-- Sq:
   G X( !(q = _q) ->   _q=1 &  (_a>0) & (q=2) |
                       _q=1 & !(_a>0) & (q=8) |
                       _q=2           & (q=3) |
                       _q=3 &  (_a>0) & (q=4) |
```

```
                      _q=3 & !(_a>0) & (q=6) |
                      _q=4           & (q=5) |
                      _q=5           & (q=2) |
                      _q=6 &  (_b>0) & (q=7) |
                      _q=6 & !(_b>0) & (q=1) |
                      _q=7           & (q=6) |
                      _q=8 & PBReset & (q=0) |
                      _q=0 & PBStart & (q=1)  ) &
   G X(  (q = _q) -> !(_q=1 &  (_a>0) | _q=1 & !(_a>0) | _q=2 | _q=3 & (_a>0) |
                      _q=3 & !(_a>0) | _q=4 | _q=5 | _q=6 & (_b>0) |
                      _q=6 & !(_b>0) | _q=7 | _q=8 & PBReset | _q=0 & PBStart) )
```

Листинг 2 (imp_LTL-спецификация ПЛК-программы)**:**

```
-- S0:
   q=0 & n=0 & a=0 & b=0 & c=0 & _q=q & _n=n & _a=a & _b=b & _c=c &
   !PBStart & !PBReset & !PBPls & !PBMns & !_PBStart & !_PBReset & !_PBPls & !_PBMns &
-- Sn:
   G X( (_q=0 & _n<bndn & !_PBPls & PBPls & !PBMns & !PBStart -> (n = _n + 1)) &
        (_q=0 & _n>0    & !_PBMns & PBMns & !PBPls & !PBStart -> (n = _n - 1)) ) &
   G X(!(_q=0 & _n<bndn & !_PBPls & PBPls & !PBMns & !PBStart) &
       !(_q=0 & _n>0    & !_PBMns & PBMns & !PBPls & !PBStart) -> (n = _n) ) &
-- Sa:
   G X( (_q=0 & PBStart & !(_a=n) -> (a = n)) &
        (_q=7 & _a<bnda           -> (a = _a + 1)) &
        (_q=1 & _a>0              -> (a = _a - 1)) &
        (_q=3 & _a>0              -> (a = _a - 1)) ) &
   G X(!(_q=0 & PBStart & !(_a=n)) & !(_q=7 & _a<bnda) &
       !(_q=1 & _a>0) & !(_q=3 & _a>0) -> (a = _a) ) &
-- Sb:
   G X( (_q=4 & _b<bndb -> (b = _b + 1)) &
        (_q=6 & _b>0    -> (b = _b - 1)) ) &
   G X(!(_q=4 & _b<bndb) & !(_q=6 & _b>0) -> (b = _b) ) &
-- Sc:
   G X( (_q=2 & _c<bndc        -> (c = _c + 1)) &
        (_q=5 & _c<bndc        -> (c = _c + 1)) &
        (_q=8 & PBReset & _c>0 -> (c = 0)) ) &
   G X(!(_q=2 & _c<bndc) & !(_q=5 & _c<bndc) & !(_q=8 & PBReset & _c>0) -> (c = _c) ) &
-- Sq:
   G X( (_q=1 &  (_a>0) -> q=2) & (_q=1 & !(_a>0) -> q=8) &
        (_q=2           -> q=3) &
        (_q=3 &  (_a>0) -> q=4) & (_q=3 & !(_a>0) -> q=6) &
        (_q=4           -> q=5) &
        (_q=5           -> q=2) &
        (_q=6 &  (_b>0) -> q=7) & (_q=6 & !(_b>0) -> q=1) &
        (_q=7           -> q=6) &
        (_q=8 & PBReset -> q=0) &
        (_q=0 & PBStart -> q=1)) ) &
   G X(!(_q=1 &  (_a>0)) & !(_q=1 & !(_a>0)) & !(_q=2) & !(_q=3 &  (_a>0)) &
       !(_q=3 & !(_a>0)) & !(_q=4) & !(_q=5) & !(_q=6 &  (_b>0)) &
       !(_q=6 & !(_b>0)) & !(_q=7) & !(_q=8 & PBReset) & !(_q=0 & PBStart) -> (q = _q) )
```

Символы «&», «|», «!» и «->» означают логические операторы «$\wedge$», «$\vee$», «$\neg$» и «$\Rightarrow$» соответственно. Здесь спецификация поведения программы — формула $\varphi = \mathrm{S0} \wedge \mathrm{S}n \wedge \mathrm{S}a \wedge \mathrm{S}b \wedge \mathrm{S}c \wedge \mathrm{S}q$.

3. Непосредственная верификация LTL-спецификаций

Декларативная или императивная LTL-спецификация φ может быть непосредственно верифицирована с помощью инструмента символьной проверки модели nuXmv. Для этого в работе [5] постановка задачи верификации имеет следующий вид:

$$pLTS \models (\varphi \Rightarrow \psi). \tag{7}$$

Здесь верификатор проверяет выполнимость импликации $\varphi \Rightarrow \psi$ на псевдополной системе переходов $pLTS$, т. е. проверяется истинность свойства ψ только на тех путях из множества Π_{pLTS}, на которых истинна спецификация поведения φ. Если формула (7) верна, то спецификация поведения φ удовлетворяет свойству ψ, иначе спецификация φ нарушает свойство ψ.

Проведём верификацию декларативной (Листинг 1) и императивной (Листинг 2) LTL-спецификаций ПЛК-программы возведения числа в квадрат согласно (7). Будем проверять набор из девяти LTL-свойств $P1, \ldots, P9$, которые в синтаксисе nuXmv представлены в листинге 3.

Листинг 3 (LTL-свойства ПЛК-программы)**:**

```
G( q=8 -> c=n*n & a=0 & b=0 )                                  -- P1;
G( a+b <= n )                                                  -- P2;
G( c <= n*n )                                                  -- P3;
G( q=8 -> X(q=8 | PBReset & q=0 & a=0 & b=0 & c=0) )           -- P4;
G( ((q=2 | q=5) -> c<bndc) & (q=4 -> b<bndb) & (q=7 -> a<bnda) )  -- P5;
G( !(q=0) -> F(q=8) )                                          -- P6;
G( q=0 & X(PBStart) -> F(q=8) )                                -- P7;
F G(PBReset & PBStart) -> (G F q=0) & (G F q=8)                 -- P8;
(G F PBStart) & (G F PBReset) -> (G F q=8) & (G F q=0)          -- P9.
```

Свойство *P1* утверждает, что всегда в заключительном управляющем состоянии переменная c содержит результат возведения числа n в квадрат, а переменные a и b равны нулю. Свойство *P2* задаёт инвариант — сумма значений переменных a и b никогда не превышает n. Свойство *P3* задаёт другой инвариант — значение переменной c не превышает n^2. Свойство *P4* — после достижения финального состояния его значение больше не изменяется, либо значения всех переменных обнуляются при нажатии кнопки сброса. Свойство *P5* — значения переменных a, b, c ограничены при заданных значениях переменной q. Свойство *P6* утверждает, что всегда если вычисление началось (управляющее состояние q отлично от нуля), то оно гарантированно завершится (управляющее состояние q станет заключительным). Свойство *P7* — если в начальном управляющем состоянии будет нажата кнопка запуска, то вычисление в будущем гарантированно завершится. Свойство *P8* — если когда-то в будущем навсегда зажать кнопки сброса и старта, то бесконечно часто будут происходить вычисления: всегда в будущем будет встречаться начальное ($q = 0$) и финальное ($q = 8$) состояния. Свойство *P9* — если всегда в будущем нажимать на кнопки старта и сброса (по отдельности или вместе), то также бесконечно часто будут происходить вычисления.

Свойства $P1, \ldots, P5$ — свойства *безопасности (Safety)*, $P6, \ldots, P9$ — свойства *живости (Liveness)* [14]. Более точно, *P6*, *P7*, *P8* — свойства *рекуррентности (Recurrence)* [15], *P9* — свойство *реактивности (Reactivity)* [15]. Классификация осуществлялась с помощью программного средства Spot [16].

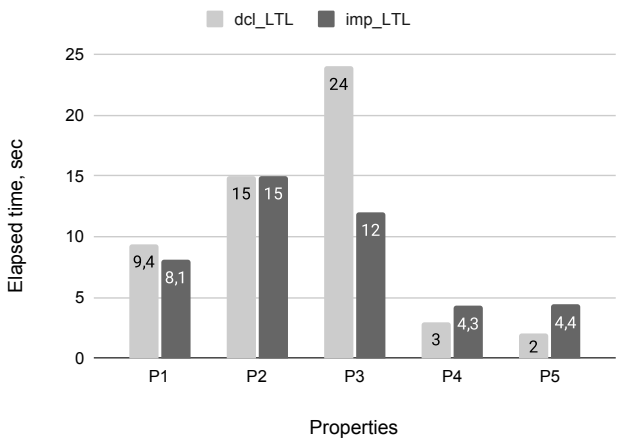
Для непосредственной верификации декларативной/императивной LTL-спецификации φ согласно (7) необходимо задать псевдополную систему переходов $pLTS$ и проверяемое свойство ψ. В синтаксисе nuXmv это было сделано в работе [5]. Результат представлен в листинге 4.

В разделе VAR объявлены основные и вспомогательные переменные. Раздел DEFINE содержит ограничения значений переменных. В разделе ASSIGN задана псевдополная система переходов. Ключевое слово LTLSPEC задаёт проверяемую LTL-формулу. В ней Spec — это dcl_LTL-спецификация φ программы возведения числа в квадрат, а Prop — любое свойство $P1, \ldots, P9$.

Листинг 4 (nuXmv-модель для верификации dcl_LTL-спецификации ПЛК-программы):

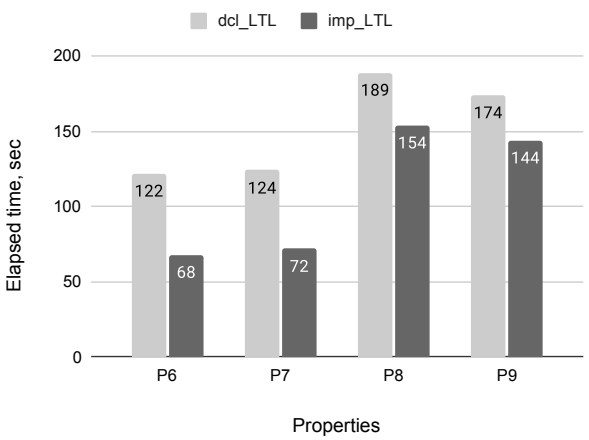
```
MODULE main
VAR
    q : 0..8;    _q : 0..8;                  -- State q
    n : 0..15;   _n : 0..15;                 -- Number n
    a : 0..15;   _a : 0..15;                 -- Counter a
    b : 0..15;   _b : 0..15;                 -- Counter b
    c : 0..255;  _c : 0..255;                -- Counter c
    PBStart : boolean;  _PBStart : boolean;  -- Push Button "Start"
    PBReset : boolean;  _PBReset : boolean;  -- Push Button "Reset"
    PBPls   : boolean;  _PBPls   : boolean;  -- Push Button "+1"
    PBMns   : boolean;  _PBMns   : boolean;  -- Push Button "-1"
DEFINE
    bndn := 15; bnda := 15; bndb := 15; bndc := 255;  -- Bounds
ASSIGN -- pLTS
    next(_q) := q;  next(_n) := n;  next(_a) := a;  next(_b) := b;  next(_c) := c;
    next(_PBStart) := PBStart;  next(_PBPls) := PBPls;
    next(_PBReset) := PBReset;  next(_PBMns) := PBMns;
LTLSPEC (Spec -> Prop)
```

Свойства $P1, P2, \ldots, P9$ выполняются для обеих LTL-спецификаций. Верификация проводилась на персональном компьютере с процессором Intel Core i5-3570 3.4 ГГц и 8 ГБ оперативной памяти. Время, затраченное на проверку свойств, представлено на рис. 3. Рис. 3,a содержит свойства безопасности (Safety), а рис. 3,b — свойства живости (Liveness). По горизонтальной оси расположены свойства (Properties), по вертикальной оси откладывается время в секундах, затраченное на верификацию (Elapsed time). Первый столбец отражает длительность верификации dcl_LTL-спецификации, второй — imp_LTL-спецификации.



(a)

Fig. 3. Experiment 1. Elapsed time for properties verification: safety (a), liveness (b)



(b)

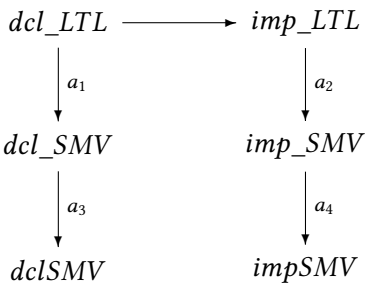
Рис. 3. Эксперимент 1. Затраченное время на верификацию свойств: безопасности (a), живости (b)

Из рис. 3 видно, что время, затраченное на проверку свойства $P2$, совпадает для обеих спецификаций, часть свойств ($P1, P3, P6, \ldots, P9$) проверяются немного быстрее на imp_LTL-спецификации, другая часть ($P4, P5$) наоборот, немного медленнее. В итоге можно считать, что время, затраченное на верификацию декларативной LTL-спецификации, сопоставимо со временем верификации

императивной LTL-спецификации, т. е. ни одна из этих спецификаций не обладает существенным преимуществом в скорости проверки свойств.

4. Преобразование LTL-спецификаций в SMV-код

В разделе 2 по декларативной LTL-спецификации (dcl_LTL) была построена эквивалентная ей императивная LTL-спецификация (imp_LTL) [5]. Выполним перевод обеих LTL-спецификаций в SMV-код. Под SMV-кодом будем понимать код на входном языке инструмента nuXmv. В результате преобразования a_1 получим декларативную SMV-спецификацию (обозначим dcl_SMV), в результате преобразования a_2 — императивную SMV-спецификацию (обозначим imp_SMV). Вышеуказанные преобразования схематично изображены на диаграмме (8).



(8)

Преобразование a_1. В декларативной LTL-спецификации (1) поведения переменной $v \in V$ заменим темпоральный оператор «**G**» на ключевое слово «**TRANS**», которое описывает отношение переходов модели [17]. Полученное отношение переходов содержит только те переходы, которые удовлетворяют формуле, находящейся под действием ключевого слова TRANS. Далее заменим темпоральный LTL-оператор «**X**» на SMV-оператор «**next**», который определяет значение выражения в следующий момент времени. Удалим знак конъюнкции между формулами. В итоге получим следующий (эквивалентный исходной LTL-спецификации) шаблон для перевода:

$$\begin{aligned}&\mathbf{TRANS}\big(\mathbf{next}\big(\neg(v = \_v) \Rightarrow cond_1 \wedge (v = expr_1) \vee \ldots \vee cond_k \wedge (v = expr_k)\big)\big)\\ &\mathbf{TRANS}\big(\mathbf{next}\big(\ \ (v = \_v) \Rightarrow \neg\,(cond_1 \vee \ldots \vee cond_k)\big)\big)\end{aligned} \tag{9}$$

Формулу инициализации поместим под действие ключевого слова INIT. Преобразуем согласно шаблону (9) декларативную LTL-спецификацию поведения программы возведения числа в квадрат (листинг 1). Результирующий SMV-код представлен в листинге 5. Для работы верификатора данный код необходимо перенести из раздела LTLSPEC в раздел ASSIGN.

Листинг 5 (dcl_SMV-спецификация ПЛК-программы)**:**

```
-- S0:
INIT(q=0 & n=0 & a=0 & b=0 & c=0 & _q=q & _n=n & _a=a & _b=b & _c=c &
    !PBStart & !PBReset & !PBPls & !PBMns & !_PBStart & !_PBReset & !_PBPls & !_PBMns)
-- Sn:
TRANS(next(!(n = _n) -> _q=0 & _n<bndn & !_PBPls & PBPls & !PBMns & !PBStart & (n = _n + 1) |
                        _q=0 & _n>0 & !_PBMns & PBMns & !PBPls & !PBStart & (n = _n - 1) ))
TRANS(next( (n = _n) -> !(_q=0 & _n<bndn & !_PBPls & PBPls & !PBMns & !PBStart |
                          _q=0 & _n>0 & !_PBMns & PBMns & !PBPls & !PBStart) ))
-- Sa:
TRANS(next( !(a = _a) -> _q=0 & PBStart & !(_a=n) & (a = n) |
                         _q=7 & _a<bnda           & (a = _a + 1) |
                         _q=1 & _a>0              & (a = _a - 1) |
                         _q=3 & _a>0              & (a = _a - 1) ))
```

```
TRANS(next(  (a = _a) -> !(_q=0 & PBStart & !(_a=n) | _q=7 & _a<bnda |
                           _q=1 & _a>0 | _q=3 & _a>0) ))
-- Sb:
TRANS(next( !(b = _b) -> _q=4 & _b<bndb & (b = _b + 1) |
                         _q=6 & _b>0    & (b = _b - 1) ))
TRANS(next(  (b = _b) -> !(_q=4 & _b<bndb | _q=6 & _b>0) ))
-- Sc:
TRANS(next( !(c = _c) -> _q=2 & _c<bndc        & (c = _c + 1) |
                         _q=5 & _c<bndc        & (c = _c + 1) |
                         _q=8 & PBReset & _c>0 & (c = 0) ))
TRANS(next(  (c = _c) -> !(_q=2 & _c<bndc | _q=5 & _c<bndc | _q=8 & PBReset & _c>0) ))
-- Sq:
TRANS(next( !(q = _q) -> _q=1 &  (_a>0) & (q=2) | _q=1 & !(_a>0) & (q=8) |
                         _q=2           & (q=3) |
                         _q=3 &  (_a>0) & (q=4) | _q=3 & !(_a>0) & (q=6) |
                         _q=4           & (q=5) |
                         _q=5           & (q=2) |
                         _q=6 &  (_b>0) & (q=7) | _q=6 & !(_b>0) & (q=1) |
                         _q=7           & (q=6) |
                         _q=8 & PBReset & (q=0) |
                         _q=0 & PBStart & (q=1) ))
TRANS(next(  (q = _q) -> !(_q=1 &  (_a>0) | _q=1 & !(_a>0) | _q=2 | _q=3 & (_a>0) |
                           _q=3 & !(_a>0) | _q=4 | _q=5 | _q=6 & (_b>0) |
                           _q=6 & !(_b>0) | _q=7 | _q=8 & PBReset | _q=0 & PBStart ) ))
```

Преобразование a_2**.** Декларативной LTL-спецификации (1) поведения переменной $v \in V$ соответствует эквивалентная ей императивная LTL-спецификация (5). Преобразуем формулу (5) — внесём оператор «**X**» внутрь скобки, получим:

$$\mathbf{G}\big(\mathbf{X}(cond_1) \Rightarrow \mathbf{X}(v = expr_1)\big) \wedge \ldots \wedge \mathbf{G}\big(\mathbf{X}(cond_k) \Rightarrow \mathbf{X}(v = expr_k)\big) \wedge \\ \mathbf{G}\big(\neg\,\mathbf{X}(cond_1) \wedge \ldots \wedge \neg\,\mathbf{X}(cond_k) \Rightarrow \mathbf{X}(v = \_v)\big). \tag{10}$$

В формуле (10) заменим темпоральный LTL-оператор «**X**» на SMV-оператор «**next**», раскроем скобки, где присутствует переменная v, и получим следующий императивный шаблон SMV-кода:

$$\begin{aligned} &\mathbf{next}\,(v) := \mathbf{case} \\ &\quad \mathbf{next}(cond_1) : \mathbf{next}(expr_1); \\ &\quad \ldots \\ &\quad \mathbf{next}(cond_k) : \mathbf{next}(expr_k); \\ &\quad \text{TRUE} \qquad\quad : \mathbf{next}(\_v); \\ &\mathbf{esac;} \end{aligned} \tag{11}$$

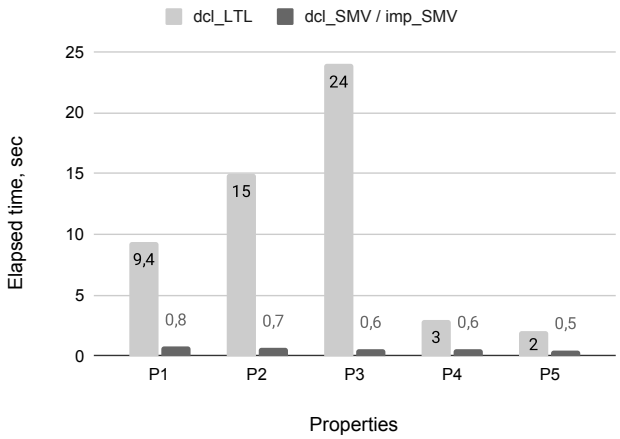
Из декларативной LTL-спецификации поведения программы возведения числа в квадрат (листинг 1) получим императивную LTL-спецификацию и выполним преобразование согласно шаблону (11). Инициализацию выполним для каждой переменной с помощью ключевого слова **init**. Результирующий SMV-код представлен в листинге 6. Данный код необходимо перенести из раздела LTLSPEC в раздел ASSIGN.

Листинг 6 (imp_SMV-спецификация ПЛК-программы)**:**

```
-- S0:
   init(q) := 0; init(_q) := q; init(n) := 0; init(_n) := n;
   init(a) := 0; init(_a) := a; init(b) := 0; init(_b) := b; init(c) := 0; init(_c) := c;
```

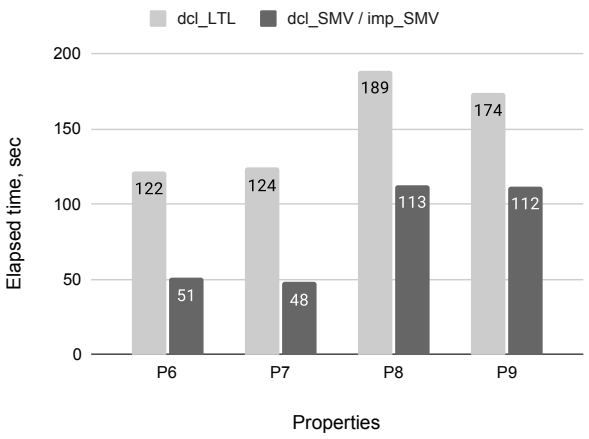
```
    init(PBStart) := FALSE;  init(_PBStart) := PBStart;
    init(PBReset) := FALSE;  init(_PBReset) := PBReset;
    init(PBPls)   := FALSE;  init(_PBPls)   := PBPls;
    init(PBMns)   := FALSE;  init(_PBMns)   := PBMns;
 -- Sn:
    next(n) := case
        next(_q=0 & _n<bndn & !_PBPls & PBPls & !PBMns & !PBStart) : next(_n + 1);
        next(_q=0 & _n>0 & !_PBMns & PBMns & !PBPls & !PBStart)    : next(_n - 1);
        TRUE                                                       : next(_n);
    esac;
 -- Sa:
    next(a) := case
        next(_q=0 & PBStart & !(_a=n)) : next(n);
        next(_q=7 & _a<bnda)           : next(_a + 1);
        next(_q=1 & _a>0)              : next(_a - 1);
        next(_q=3 & _a>0)              : next(_a - 1);
        TRUE                           : next(_a);
    esac;
 -- Sb:
    next(b) := case
        next(_q=4 & _b<bndb) : next(_b + 1);
        next(_q=6 & _b>0)    : next(_b - 1);
        TRUE                 : next(_b);
    esac;
 -- Sc:
    next(c) := case
        next(_q=2 & _c<bndc)        : next(_c + 1);
        next(_q=5 & _c<bndc)        : next(_c + 1);
        next(_q=8 & PBReset & _c>0) : 0;
        TRUE                        : next(_c);
    esac;
 -- Sq:
    next(q) := case
        next(_q=1 &  (_a>0)) : 2;
        next(_q=1 & !(_a>0)) : 8;
        next(_q=2          ) : 3;
        next(_q=3 &  (_a>0)) : 4;
        next(_q=3 & !(_a>0)) : 6;
        next(_q=4          ) : 5;
        next(_q=5          ) : 2;
        next(_q=6 &  (_b>0)) : 7;
        next(_q=6 & !(_b>0)) : 1;
        next(_q=7          ) : 6;
        next(_q=8 & PBReset) : 0;
        next(_q=0 & PBStart) : 1;
        TRUE                 : next(_q);
    esac;
```

С помощью инструмента nuXmv выполним верификацию dcl_SMV-спецификации и imp_SMV-спецификации. Сравним время их верификации со временем верификации dcl_LTL-спецификации. Результаты проверки свойств *P1*, . . . , *P9* представлены на рис. 4. Первый столбец показывает время верификации dcl_LTL-спецификации. Верификация спецификаций dcl_SMV и imp_SMV занимает одинаковое время, которое представлено во втором столбце.



(a)

Fig. 4. Experiment 2. Elapsed time for properties verification: safety (a), liveness (b)



(b)

Рис. 4. Эксперимент 2. Затраченное время на верификацию свойств: безопасности (a), живости (b)

Из рис. 4 видно, что время проверки SMV-спецификаций меньше времени проверки LTL-спецификации. Имеем следующий вектор коэффициентов сокращения времени верификации:

$$(xP_1 = 12;\ xP_2 = 21;\ xP_3 = 40;\ xP_4 = 5;\ xP_5 = 4;\ xP_6 = 2.4;\ xP_7 = 2.6;\ xP_8 = 1.6;\ xP_9 = 1.5),$$

где xP_i — коэффициент сокращения времени верификации свойства Pi, $i = 1, \ldots, 9$. Время проверки свойства $P1$ сократилось в двенадцать раз, свойства $P2$ — в двадцать один раз, $P3$ — в сорок раз, $P4$ — в пять раз, $P5$ — в четыре раза, т. е. время верификации свойств безопасности сократилось до нескольких десятков раз, что является достаточно хорошим показателем. Время проверки свойств $P6$ и $P7$ сократилось примерно в два с половиной раза, $P8$ и $P9$ — примерно в полтора раза. Свойства живости $P6, \ldots, P9$ имеют относительно небольшой коэффициент сокращения времени.

Также заметим, что свойства безопасности $P1, \ldots, P5$ проверяются значительно быстрее свойств живости $P6, \ldots, P9$. У свойств безопасности наблюдается следующая тенденция — при увеличении времени верификации dcl_LTL-спецификации также увеличивается и коэффициент сокращения времени для SMV-спецификаций, что положительно сказывается при проверке свойств безопасности, занимающих длительное время.

Причиной ускорения верификации является сокращение пространства достижимых состояний проверяемой модели. При верификации декларативной LTL-спецификации (dcl_LTL — листинг 1) согласно (7) выполняется анализ псевдополной системы переходов $pLTS$, которая имеет $2.27995 \cdot 10^{16}$ $(2^{54.3399})$ достижимых состояний. Оценку количества достижимых состояний выполняет инструментальное средство nuXmv при верификации.

Задача верификации SMV-спецификаций выглядит следующим образом:

$$LTS \models \psi, \tag{12}$$

где LTS — система переходов, описывающая то же самое поведение, что и декларативная LTL-спецификация φ, т. е. $\Pi_{LTS} = [\![\varphi]\!]_{pLTS}$, ψ — проверяемое свойство. SMV-спецификация непосредственно задаёт систему переходов LTS. При верификации SMV-спецификаций (dcl_SMV — листинг 5, imp_SMV — листинг 6) согласно (12) выполняется анализ системы переходов LTS, которая имеет $9.92256 \cdot 10^5$ $(2^{19.9204})$ достижимых состояний. Таким образом, пространство состояний сократилось примерно в $2.4 \cdot 10^{10}$ (2^{35}) раз.

5. Бисимуляционное преобразование SMV-спецификаций

Из листинга 3 видно, что свойства $P1, \ldots, P9$ формулируются только относительно переменных из $V = \{v_1, \ldots, v_n\}$, переменные из $\_V = \{\_v_1, \ldots, \_v_n\}$ отсутствуют, — данный набор свойств сосредоточен на поведении только основных переменных V. Далее будет показано, что в этом случае переменные множества $\_V$ могут быть удалены из SMV-спецификации.

Выполним преобразование полученных в разделе 4 SMV-спецификаций (dcl_SMV, imp_SMV) согласно диаграмме (8). В результате преобразования a_3 получим новую декларативную SMV-спецификацию (обозначим dclSMV), в результате преобразования a_4 — новую императивную SMV-спецификацию (обозначим impSMV). В обозначениях новых спецификаций отсутствует знак лидирующего подчёркивания «_», что указывает на отсутствие в спецификациях переменных из множества $\_V = \{\_v_1, \ldots, \_v_n\}$.

Преобразование a_3. Выполним эквивалентное преобразование — в шаблоне (9) перенесём оператор **next** внутрь всех скобок и выражений $cond_i$, $expr_i$, где $i = 1, \ldots, k$. В итоге оператор **next** будет применён к каждой переменной. Согласно псевдополной системе переходов $\mathbf{next}(\_v) = v$. Выполним данную подстановку — таким образом, удалим все переменные множества $\_V = \{\_v_1, \ldots, \_v_n\}$ из спецификации. Получим следующий шаблон:

$$\begin{aligned}
&\mathbf{TRANS}\big(\neg(\mathbf{next}(v) = v) \Rightarrow Xcond_1 \wedge (\mathbf{next}(v) = Xexpr_1) \vee \ldots \vee Xcond_k \wedge (\mathbf{next}(v) = Xexpr_k)\big)\\
&\mathbf{TRANS}\big(\ \ (\mathbf{next}(v) = v) \Rightarrow \neg\,(Xcond_1 \vee \ldots \vee Xcond_k)\big)
\end{aligned} \tag{13}$$

В этом шаблоне выражения $Xcond_i$ и $Xexpr_i$ представляют собой исходные выражения $cond_i$ и $expr_i$ соответственно ($i = 1, \ldots, k$), в которых каждая переменная v без лидирующего подчёркивания заменяется на $\mathbf{next}(v)$, а вместо переменных вида $\_v$ подставляются соответствующие переменные вида v. Преобразуем согласно шаблону (13) декларативную SMV-спецификацию поведения программы возведения числа в квадрат (dcl_SMV — листинг 5). В конструкции INIT удалим инициализацию переменных из $\_V = \{\_v_1, \ldots, \_v_n\}$. Результирующий SMV-код представлен в листинге 7. Из раздела ASSIGN исключим выражения вида $\mathbf{next}(\_v) = v$. Теперь переменные из $\_V = \{\_v_1, \ldots, \_v_n\}$ больше нигде не используются, поэтому удалим их из раздела объявления переменных VAR. После чего можно проводить верификацию.

Листинг 7 (dclSMV-спецификация ПЛК-программы)**:**

```
-- S0:
INIT(q=0 & n=0 & a=0 & b=0 & c=0 & !PBStart & !PBReset & !PBPls & !PBMns)
-- Sn:
TRANS(!(next(n) = n) ->
  q=0 & n<bndn & !PBPls & next(PBPls) & !next(PBMns) & !next(PBStart) & (next(n) = n + 1) |
  q=0 & n>0    & !PBMns & next(PBMns) & !next(PBPls) & !next(PBStart) & (next(n) = n - 1) )
TRANS( (next(n) = n) ->
       !(q=0 & n<bndn & !PBPls & next(PBPls) & !next(PBMns) & !next(PBStart) |
         q=0 & n>0    & !PBMns & next(PBMns) & !next(PBPls) & !next(PBStart)) )
-- Sa:
TRANS(!(next(a) = a) -> q=0 & next(PBStart) & !(a=next(n)) & (next(a) = next(n)) |
                        q=7 & a<bnda                       & (next(a) = a + 1) |
                        q=1 & a>0                          & (next(a) = a - 1) |
                        q=3 & a>0                          & (next(a) = a - 1) )
TRANS( (next(a) = a) -> !(q=0 & next(PBStart) & !(a=next(n)) |
                          q=7 & a<bnda | q=1 & a>0 | q=3 & a>0) )
-- Sb:
TRANS(!(next(b) = b) -> q=4 & b<bndb & (next(b) = b + 1) | q=6 & b>0 & (next(b) = b - 1) )
TRANS( (next(b) = b) -> !(q=4 & b<bndb | q=6 & b>0) )
```

```
-- Sc:
TRANS(!(next(c) = c) -> q=2 & c<bndc              & (next(c) = c + 1) |
                        q=5 & c<bndc              & (next(c) = c + 1) |
                        q=8 & next(PBReset) & c>0 & (next(c) = 0) )
TRANS( (next(c) = c) -> !(q=2 & c<bndc | q=5 & c<bndc | q=8 & next(PBReset) & c>0) )
-- Sq:
TRANS(!(next(q) = q) -> q=1 &  (a>0)        & next(q)=2 |
                        q=1 & !(a>0)        & next(q)=8 |
                        q=2                 & next(q)=3 |
                        q=3 &  (a>0)        & next(q)=4 |
                        q=3 & !(a>0)        & next(q)=6 |
                        q=4                 & next(q)=5 |
                        q=5                 & next(q)=2 |
                        q=6 &  (b>0)        & next(q)=7 |
                        q=6 & !(b>0)        & next(q)=1 |
                        q=7                 & next(q)=6 |
                        q=8 & next(PBReset) & next(q)=0 |
                        q=0 & next(PBStart) & next(q)=1 )
TRANS( (next(q) = q) -> !(q=1 &  (a>0) | q=1 & !(a>0) | q=2 | q=3 & (a>0) |
                          q=3 & !(a>0) | q=4 | q=5 | q=6 & (b>0) | q=6 & !(b>0) |
                          q=7 | q=8 & next(PBReset) | q=0 & next(PBStart) ) )
```

Преобразование a$_4$. Аналогично преобразованию a_3 в шаблоне (11) перенесём оператор **next** внутрь всех выражений $cond_i$, $expr_i$, получим выражения $Xcond_i$ и $Xexpr_i$ соответственно, где $i = 1, \ldots, k$. С учётом **next**$(\_v) = v$ получим шаблон:

$$
\begin{aligned}
&\textbf{next}\,(v) := \textbf{case}\\
&\quad Xcond_1 : Xexpr_1;\\
&\quad \ldots\\
&\quad Xcond_k : Xexpr_k;\\
&\quad \text{TRUE} \quad : v;\\
&\textbf{esac;}
\end{aligned}
\tag{14}
$$

Преобразуем по шаблону (14) императивную SMV-спецификацию поведения программы возведения числа в квадрат (imp_SMV — листинг 6). Из раздела ASSIGN исключим выражения вида **init**$(\_v) = v$ и **next**$(\_v) = v$. И удалим все переменные множества $\_V = \{\_v_1, \ldots, \_v_n\}$ из раздела объявления переменных VAR. Результирующий SMV-код представлен в листинге 8.

Листинг 8 (impSMV-спецификация ПЛК-программы)**:**

```
-- S0:
   init(q) := 0;   init(n) := 0;   init(a) := 0;   init(b) := 0;   init(c) := 0;
   init(PBStart) := FALSE;     init(PBPls) := FALSE;
   init(PBReset) := FALSE;     init(PBMns) := FALSE;
-- Sn:
   next(n) := case
       q=0 & n<bndn & !PBPls & next(PBPls) & !next(PBMns) & !next(PBStart) : n + 1;
       q=0 & n>0    & !PBMns & next(PBMns) & !next(PBPls) & !next(PBStart) : n - 1;
       TRUE                                                                : n;
   esac;
-- Sa:
   next(a) := case
       q=0 & next(PBStart) & !(a=next(n)) : next(n);
       q=7 & a<bnda                       : a + 1;
```

```
        q=1 & a>0                          : a - 1;
        q=3 & a>0                          : a - 1;
        TRUE                               : a;
    esac;
-- Sb:
    next(b) := case
        q=4 & b<bndb : b + 1;
        q=6 & b>0    : b - 1;
        TRUE         : b;
    esac;
-- Sc:
    next(c) := case
        q=2 & c<bndc               : c + 1;
        q=5 & c<bndc               : c + 1;
        q=8 & next(PBReset) & c>0 : 0;
        TRUE                       : c;
    esac;
-- Sq:
    next(q) := case
        q=1 &  (a>0)        : 2;
        q=1 & !(a>0)        : 8;
        q=2                 : 3;
        q=3 &  (a>0)        : 4;
        q=3 & !(a>0)        : 6;
        q=4                 : 5;
        q=5                 : 2;
        q=6 &  (b>0)        : 7;
        q=6 & !(b>0)        : 1;
        q=7                 : 6;
        q=8 & next(PBReset) : 0;
        q=0 & next(PBStart) : 1;
        TRUE                : q;
    esac;
```

С помощью инструмента nuXmv выполним верификацию полученных спецификаций: dclSMV и impSMV. На рис. 5 представлены результаты верификации свойств *P1*, . . . , *P9*. Первый столбец показывает время верификации спецификаций dcl_SMV и imp_SMV. Верификация спецификаций dclSMV и impSMV занимает одинаковое время, которое представлено во втором столбце.

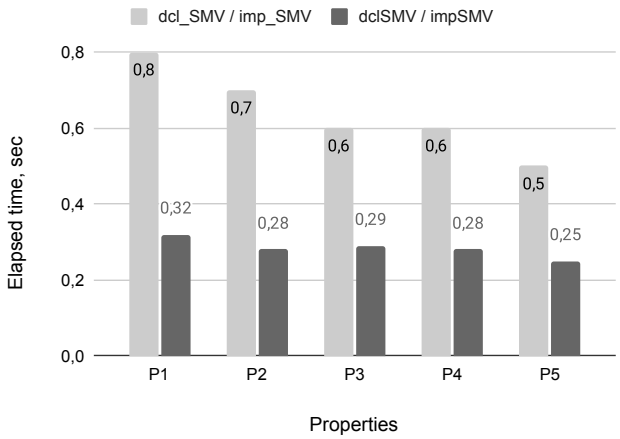
Из рис. 5 видно, что время проверки новых SMV-спецификаций меньше времени проверки предыдущих SMV-спецификаций. Имеем следующий вектор коэффициентов сокращения времени верификации:

$$(xP_1 = 2.5;\ xP_2 = 2.5;\ xP_3 = 2;\ xP_4 = 2.1;\ xP_5 = 2;\ xP_6 = 4.6;\ xP_7 = 4.8;\ xP_8 = 5.4;\ xP_9 = 8.6),$$

где xP_i — коэффициент сокращения времени верификации свойства *Pi*, $i = 1, \ldots, 9$. Время верификации свойств безопасности *P1*, . . . , *P5* сократилось примерно в 2–2.5 раза, время верификации свойств живости *P6*, . . . , *P9* — примерно в 4.5–8.5 раз. На этот раз свойства живости имеют более высокий коэффициент сокращения времени верификации.

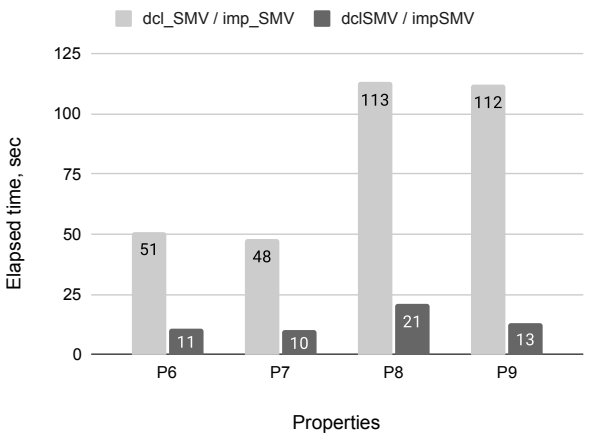
При сравнении времени верификации исходной dcl_LTL-спецификации со временем верификации спецификаций dclSMV и impSMV получим следующий вектор коэффициентов:

$$(xP_1 = 29;\ xP_2 = 53;\ xP_3 = 82;\ xP_4 = 10;\ xP_5 = 8;\ xP_6 = 11;\ xP_7 = 12;\ xP_8 = 9;\ xP_9 = 13),$$



(a)

Fig. 5. Experiment 3. Elapsed time for properties verification: safety (a), liveness (b)



(b)

Рис. 5. Эксперимент 3. Затраченное время на верификацию свойств: безопасности (a), живости (b)

где xP_i — коэффициент сокращения времени верификации свойства Pi, $i = 1, \ldots, 9$. Итоговое время верификации свойств безопасности $P1, \ldots, P5$ сократилось от 8 до 82 раз, время верификации свойств живости $P6, \ldots, P9$ — от 9 до 13 раз.

Причиной ускорения верификации также является сокращение пространства достижимых состояний проверяемой модели. Полученные спецификации (dclSMV — листинг 7, impSMV — листинг 8) задают систему переходов LTS', которая описывает поведение только основных переменных $v_1, \ldots, v_n$. Система переходов LTS имеет $9.92256 \cdot 10^5$ $(2^{19.9204})$ достижимых состояний, а система переходов LTS' $6.2016 \cdot 10^4$ $(2^{15.9204})$ достижимых состояний. Таким образом, пространство состояний сократилось в 16 (2^4) раз. Если сравнивать с псевдополной системой переходов $pLTS$, которая имеет $2^{54.3399}$ достижимых состояний, то пространство состояний сократилось в $2^{38.4195}$ раз.

Удаление вспомогательных переменных $\_v_1, \ldots, \_v_n$ из спецификации привело к изменению системы переходов — как минимум уменьшилось число достижимых состояний, что сократило отношение переходов. Убедимся, что такое изменение модели поведения программы не повлияет на результаты верификации. Только в этом случае можно заменить LTS на LTS' при проверке свойств $P1, \ldots, P9$. Рассмотрим преобразования a_3, a_4 на уровне систем переходов.

Процедура преобразования систем переходов. Пусть дана исходная система переходов $LTS = \langle S, S_0, R, P, L \rangle$, где $S = (D_1 \times \ldots \times D_n)^2$. Вектор состояния $s = (\boldsymbol{a}, \boldsymbol{\_a})$ принимает значения из S, где $\boldsymbol{a}$ и $\boldsymbol{\_a}$ — значения соответствующих векторов $\mathrm{v} = (v_1, \ldots, v_n)$ и $\mathrm{\_v} = (\_v_1, \ldots, \_v_n)$. Преобразуем исходную систему переходов LTS в систему переходов $LTS' = \langle S', S'_0, R', P', L' \rangle$, где $S' = D_1 \times \ldots \times D_n$, с помощью отображения $f : S \to S'$ следующего вида:

$$f(\mathrm{v}, \mathrm{\_v}) = \mathrm{v}, \tag{15}$$

т. е. берётся проекция вектора $(\mathrm{v}, \mathrm{\_v})$, в которой присутствуют только компоненты $v_1, \ldots, v_n$ и отсутствуют компоненты $\_v_1, \ldots, \_v_n$. Функция (15) формализует удаление вспомогательных переменных, что было произведено на уровне SMV-спецификации.

Множество начальных состояний S'_0 также определяется с помощью f на основании S_0, а именно:

$$S'_0 = \{f(s) \in S' \mid s \in S_0\}, \tag{16}$$

т. е. все начальные состояния из LTS и только они преобразуются в начальные состояния в LTS'.

Все переходы в R удовлетворяют спецификациям dcl_SMV и imp_SMV. Так как преобразования a_3 и a_4 сохраняют эквивалентность между исходным и результирующим TRANS-выражениями, то полученные спецификации dclSMV и impSMV задают то же поведение основных переменных $v_1, \ldots, v_n$, несмотря на то, что они уже не зависят от вспомогательных переменных $\_v_1, \ldots, \_v_n$. Таким образом, отношение переходов R' сохраняет переходы из R — для каждого перехода из R с помощью f строится соответствующий ему переход в R' согласно диаграмме:

$$\begin{array}{ccc} s = (\boldsymbol{a}, \_\boldsymbol{a}) & \xrightarrow{\quad f \quad} & s' = \boldsymbol{a} \\ {\scriptstyle \in R}\big\downarrow & & \big\downarrow{\scriptstyle \in R'} \\ s_1 = (\boldsymbol{a}_1, \boldsymbol{a}) & \xrightarrow{\quad f \quad} & s'_1 = \boldsymbol{a}_1 \end{array}$$

Здесь $\boldsymbol{a}, \boldsymbol{a}_1$ — значения вектора $(v_1, \ldots, v_n)$, $\_\boldsymbol{a}$ — вектора $(\_v_1, \ldots, \_v_n)$. В итоге отношение переходов R' определяется следующим образом:

$$R' = \{\big(f(s), f(s_1)\big) \in (S')^2 \mid (s, s_1) \in R\}. \tag{17}$$

Компонент P — множество атомарных утверждений над переменными множества $V \cup \_V$. Здесь $P = P_\varphi \cup P_\psi$, где P_φ — утверждения, используемые при задании LTL-спецификации φ поведения программы, а P_ψ — при задании LTL-спецификации проверяемых свойств $\psi_1, \ldots, \psi_h$. Множество $P_\psi \subset P$ содержит утверждения только относительно переменных из V. В LTS' компонент $P' = P_\psi$.

Разметка состояний из LTS сохраняется и в LTS' для множества P' — если утверждение $p' \in P'$ истинно в $s \in S$, то оно истинно и в $f(s)$:

$$(\forall p' \in P')\ (\forall s \in S)\ [p' \in L(s) \Rightarrow p' \in L'\big(f(s)\big)]. \tag{18}$$

Определение бисимуляции. Системы переходов $LTS = \langle S, S_0, R, P', L\rangle$ и $LTS' = \langle S', S'_0, R', P', L'\rangle$ находятся в отношении *бисимуляции* $B \subseteq S \times S'$, если выполняются следующие три условия [10]:

$$(\forall s \in S)\ (\forall s' \in S')\ [B(s, s') \Rightarrow L(s) = L'(s')], \tag{19}$$

$$(\forall s, s_1 \in S)\ (\forall s' \in S')\ [B(s, s') \wedge R(s, s_1) \Rightarrow (\exists s'_1 \in S')\ R'(s', s'_1) \wedge B(s_1, s'_1)], \tag{20}$$

$$(\forall s \in S)\ (\forall s', s'_1 \in S')\ [B(s, s') \wedge R'(s', s'_1) \Rightarrow (\exists s_1 \in S)\ R(s, s_1) \wedge B(s_1, s'_1)]. \tag{21}$$

Условие (19) является условием *совпадения разметки* соответствующих состояний. Условие (20) утверждает, что если существует переход в LTS, то существует и соответствующий ему переход в LTS'. То же верно и в обратном направлении — условие (21).

Системы переходов LTS и LTS' *бисимуляционно эквивалентны* [10, 18] (обозначим $LTS \equiv LTS'$), если существует такое отношение бисимуляции B, что:

$$(\forall s_0 \in S_0)\ (\exists s'_0 \in S'_0)\ B(s_0, s'_0), \tag{22}$$

$$(\forall s'_0 \in S'_0)\ (\exists s_0 \in S_0)\ B(s_0, s'_0), \tag{23}$$

т. е. для любого начального состояния в LTS найдётся (в соответствии с отношением бисимуляции B) начальное состояние в LTS', и наоборот.

Известна теорема [10], которая утверждает, что если системы переходов бисимуляционно эквивалентны $LTS \equiv LTS'$, то для любой формулы ψ логики CTL* верно:

$$(LTS \models \psi) \Leftrightarrow (LTS' \models \psi), \tag{24}$$

т. е. если свойство ψ выполняется на LTS, то оно выполняется и на LTS', и если свойство ψ нарушается на LTS, то оно нарушается и на LTS'. Так как логика LTL является подмножеством логики CTL*, то данная теорема верна и для любой LTL-формулы.

Утверждение 1 (О сохранении бисимуляционной эквивалентности)**.** *Преобразование систем переходов с помощью отображения* $f : S \to S'$ *вида* $f(\mathrm{v}, \_\mathrm{v}) = \mathrm{v}$, *удовлетворяющее условиям* (16), (17), (18), *сохраняет бисимуляционную эквивалентность, т. е. имеем* $LTS = \langle S, S_0, R, P', L\rangle \equiv LTS' = \langle S', S'_0, R', P', L'\rangle$.

Доказательство. Формально сохранение бисимуляционной эквивалентности — это утверждение (15), (16), (17), (18) $\vdash$ (19), (20), (21), (22), (23).

Согласно (18) для P' при преобразовании разметка состояний сохраняется. Так как истинность/ложность любого атомарного утверждения $p' \in P'$ зависит только от v и не зависит от _v, а состояния $s \in S$ и $f(s) \in S'$ всегда имеют одно и то же значение вектора v согласно (15), то все прообразы для $f(s) \in S'$ всегда имеют одинаковую разметку, т. е. верно утверждение:

$$(\forall p' \in P')\,(\forall s \in S)\,[p' \in L'\big(f(s)\big) \Rightarrow p' \in L(s)]. \tag{25}$$

Для множества P' оба утверждения (18) и (25) образуют условие *равенства разметки* состояний:

$$(\forall p' \in P')\,(\forall s \in S)\,[L(s) = L'\big(f(s)\big)]. \tag{26}$$

Из (26) следует (19) при $s' = f(s)$.

Согласно (17) переходы в R' образуются только из переходов из R, поэтому по построению любому переходу из R соответствует ровно один переход в R', и любому переходу из R' соответствует хотя бы один переход в R, т. е. верны утверждения (20) и (21).

В соответствии с (16) все начальные состояния из LTS и только они преобразуются в начальные состояния в LTS', поэтому по построению любому начальному состоянию из S_0 соответствует ровно одно состояние в S'_0, и любому начальному состоянию из S'_0 соответствует хотя бы одно начальное состояние в S_0, т. е. верны утверждения (22) и (23). Таким образом, предложенное преобразование систем переходов сохраняет бисимуляционную эквивалентность относительно множества P'. □

Бисимуляционная эквивалентность $LTS \equiv LTS'$ согласно теореме (24) позволяет заменить систему переходов LTS на систему переходов LTS' без изменения результатов проверки LTL-свойств.

Заключение

В настоящей работе предложены преобразования декларативной (dcl_LTL) и императивной (imp_LTL) LTL-спецификаций [5] в декларативную (dcl_SMV) и императивную (imp_SMV) SMV-спецификации соответственно. Данные SMV-спецификации задают систему переходов LTS, которая по сравнению с псевдополной системой переходов $pLTS$ имеет меньший размер пространства состояний. Система переходов $pLTS$ по определению содержит состояния и переходы, которые не удовлетворяют LTL-спецификации поведения программы. В системе переходов LTS исключены такие состояния и переходы. Уменьшение размера пространства состояний приводит к сокращению времени верификации модели программы.

Если проверяемые LTL-свойства не содержат вспомогательных переменных, то могут быть выполнены дальнейшие преобразования SMV-спецификаций: перевод dcl_SMV и imp_SMV в спецификации dclSMV и impSMV соответственно. Вновь полученные SMV-спецификации задают систему переходов LTS', которая по сравнению с LTS имеет меньший размер пространства состояний. Это также приводит к сокращению времени верификации. Система переходов LTS описывает поведение основных и вспомогательных переменных. Система переходов LTS' описывает поведение только основных переменных. Предложенное преобразование SMV-спецификаций сохраняет бисимуляционную эквивалентность систем переходов: $LTS \equiv LTS'$. Это позволяет заменить одну SMV-спецификацию на другую без каких-либо изменений результатов верификации.

Из dcl_LTL-спецификации проще получить dclSMV-спецификацию, чем impSMV-спецификацию. Преимуществом impSMV-спецификации является её компактность по сравнению с dclSMV-спецификацией. Выбор декларативности или императивности спецификации зависит от целей и удобства

её использования. Для описания требуемого поведения переменных УПО больше подходит декларативный стиль, для построения кода программы — императивный.

Таким образом, в настоящей работе были предложены четыре новые SMV-спецификации, которые позволяют значительно сократить время верификации декларативной LTL-спецификации.

References

[1] D. J. Smith and K. G. Simpson, *The Safety Critical Systems Handbook*, 5th. Butterworth-Heinemann, 2020, ISBN: 978-0-12-820700-0.

[2] V. D'Silva, D. Kroening, and G. Weissenbacher, "A Survey of Automated Techniques for Formal Software Verification", in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, 2008, pp. 1165–1178. DOI: 10.1109/TCAD.2008.923410.

[3] S. Oks *et al.*, "Cyber-Physical Systems in the Context of Industry 4.0: A Review, Categorization and Outlook", *Information Systems Frontiers*, 2022. DOI: 10.1007/s10796-022-10252-x.

[4] E. Lee and S. Seshia, *Introduction to Embedded Systems – A Cyber-Physical Systems Approach*, 2nd. MIT Press, 2017, ISBN: 978-0-262-53381-2.

[5] M. Neyzov and E. Kuzmin, "LTL-specification for Development and Verification of Control Programs", *Modeling and Analysis of Information Systems*, vol. 30, no. 4, pp. 308–339, 2023, in Russian. DOI: 10.18255/1818-1015-2023-4-308-339.

[6] D. Harel and A. Pnueli, "On the Development of Reactive Systems", in *Logics and Models of Concurrent Systems*, vol. 13, 1985, pp. 477–498. DOI: 10.1007/978-3-642-82453-1_17.

[7] A. Pnueli, "Applications of Temporal Logic to the Specification and Verification of Reactive Systems: A Survey of Current Trends", in *Current Trends in Concurrency*, vol. 224, Springer, 1986, pp. 510–584. DOI: 10.1007/BFb0027047.

[8] *IEC 61131-3:2013 Programmable controllers – Part 3: Programming languages.* [Online]. Available: https://webstore.iec.ch/publication/4552.

[9] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, *Handbook of Model Checking*, 1st. Springer Publishing Company, Incorporated, 2018, ISBN: 3319105744.

[10] E. M. Clarke, O. Grumberg, and D. Peled, *Verification of Program Models: Model Checking*. MCNMO, 2002, p. 416, translated from English to Russian, ISBN: 5-94057-054-2.

[11] D. Beyer and A. Podelski, "Software model checking: 20 years and beyond", in *Principles of Systems Design: Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, Springer, 2022, pp. 554–582, ISBN: 978-3-031-22337-2. DOI: 10.1007/978-3-031-22337-2_27.

[12] *nuXmv home.* [Online]. Available: https://nuxmv.fbk.eu/.

[13] *IEC 61131-1:2003 Programmable controllers – Part 1: General information.* [Online]. Available: https://webstore.iec.ch/publication/4550.

[14] B. Alpern and F. B. Schneider, "Defining Liveness", *Information Processing Letters*, vol. 21, no. 4, pp. 181–185, 1985. DOI: 10.1016/0020-0190(85)90056-0.

[15] Z. Manna and A. Pnueli, "A Hierarchy of Temporal Properties", in *Proceedings of the ninth annual ACM symposium on Principles of distributed computing*, 1990, pp. 377–410. DOI: 10.1145/93385.93442.

[16] *Spot home.* [Online]. Available: https://spot.lre.epita.fr/.

[17] *nuXmv User Manual.* [Online]. Available: https://nuxmv.fbk.eu/downloads/nuxmv-user-manual.pdf.

[18] D. Park, "Concurrency and Automata on Infinite Sequences", in *Theoretical Computer Science: 5th GI-Conference Karlsruhe*, vol. 104, 1981, pp. 167–183. DOI: 10.1007/BFb0017309.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 2, 2024
journal homepage: www.mais-journal.ru

ALGORITHMS IN COMPUTER SCIENCE

Mathematical Properties of the Agent-based Model of Extinction — Recolonization for Population Genetics

N. V. Gaianov[1]

DOI: 10.18255/1818-1015-2024-2-142-151

[1]National Research University Higher School of Economics, Moscow, Russia

MSC2020: 60J70
Research article
Full text in Russian

Received April 26, 2024
Revised May 7, 2024
Accepted May 15, 2024

The individual-based model describes the dynamics of genetic diversity of a population scattered on a spatial continuum in the case of a finite number of individuals. During extinction events in a certain area, a portion of the population dies, after which new individuals with the genotype of the parent are born during recolonization event. In this paper we examine the model, as well as its modification, and derive properties related to population parameters. The study demonstrates that the lifespan of individuals follows an exponential distribution, allele probabilities remain constant over time, and the average heterozygosity, constrained by the number of individuals during extinction and recolonization, equals a similar quantity in the Moran model. The joint distribution of alleles is generalized for populations continuously scattered in space. Joint allele distribution and heterozygosity are computed through simulations.

Keywords: population model; recolonisation; spatial continuum; Moran model

INFORMATION ABOUT THE AUTHORS

Gaianov, Nikita V. (corresponding author) | ORCID iD: 0009-0001-0522-6856. E-mail: nvgayanov@edu.hse.ru
Research Assistant

Funding: Russian Science Foundation (Project no. 22-71-10056).

For citation: N. V. Gaianov, "Mathematical properties of the agent-based model of extinction — recolonization for population genetics", *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 142–151, 2024. DOI: 10.18255/1818-1015-2024-2-142-151.

© Gaianov N. V., 2024
This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 2, 2024
сайт журнала: www.mais-journal.ru

ALGORITHMS IN COMPUTER SCIENCE

Математические свойства агентной модели вымирания — реколонизации для популяционной генетики

Н. В. Гаянов[1]

DOI: 10.18255/1818-1015-2024-2-142-151

[1]Национальный исследовательский университет «Высшая школа экономики», Москва, Россия

УДК 51-76
Научная статья
Полный текст на русском языке

Получена 26 апреля 2024 г.
После доработки 7 мая 2024 г.
Принята к публикации 15 мая 2024 г.

Агентная модель описывает динамику генетического разнообразия непрерывно распределенной популяции в случае конечного числа особей. В событии вымирания в некоторой области умирает часть популяции, после чего в ходе реколонизации рождаются новые особи с генотипом родителя. Мы рассматриваем модель, а также её модификацию, и получаем свойства, связанные с популяционными параметрами. В работе показано, что время жизни особей имеет экспоненциальное распределение, вероятности аллелей сохраняются во времени, средняя гетерозиготность при ограничении, связанном с числом особей при вымирании и реколонизации, равна аналогичной величине в модели Морана. Совместное распределение аллелей обобщено на случай популяций, непрерывно расположенных в пространстве. Совместное распределение аллелей и гетерозиготность посчитаны на симуляциях.

Ключевые слова: популяционная модель; реколонизация; непрерывное пространство; модель Морана

ИНФОРМАЦИЯ ОБ АВТОРАХ

Гаянов, Никита Владимирович (автор для корреспонденции) | ORCID iD: 0009-0001-0522-6856. E-mail: nvgayanov@edu.hse.ru
Стажер-исследователь

Финансирование: Российский научный фонд (проект № 22-71-10056).

Для цитирования: N. V. Gaianov, “Mathematical properties of the agent-based model of extinction — recolonization for population genetics”, *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 142–151, 2024. DOI: 10.18255/1818-1015-2024-2-142-151.

© Гаянов Н. В., 2024
Эта статья открытого доступа под лицензией CC BY license (https://creativecommons.org/licenses/by/4.0/).

Введение

Популяционные модели описывают изменения генетического разнообразия популяции во времени. Такие модели используются во многих задачах популяционной генетики. Наиболее известные модели: модель Райта — Фишера, Морана, ступенька Кимуры (stepping stone) и др. [1] — либо не учитывают пространственную структуру популяции, либо разделяют её на подпопуляции, которые развиваются независимо друг от друга, но при этом между ними существует миграция. Такие модели не учитывают сценарии, в которых подпопуляции распределены непрерывно и граничат друг с другом.

Проблемы построения моделей с непрерывным расположением особей связаны с утверждением, доказанным в [2] о том, что не существует модели, в которой выполняются 3 свойства:

1) особи распределены с одинаковой плотностью;
2) особи размножаются независимо, со средним числом потомков равным единице;
3) новорождённые особи мигрируют согласно нормальному распределению с матожиданием, равным положению родителя.

Существуют модели, в которых существует локальная регуляризация плотности, например [3], однако для таких моделей сложно строить генеалогические деревья. В работе [4] был предложен новый класс моделей (пространственная Λ-Флеминг — Вайот модель), в которой изменение генетического разнообразия происходит благодаря событиям вымирания — реколонизации. Такой подход обеспечивает выполнение свойств (1) и (3), а события размножения становятся зависимыми. Для данной модели, а также для двойственной ей Λ-коалесценции было получено множество результатов [5—7], существуют работы связанные с изучением применимости модели [8, 9]

Пространственный Λ-Флеминг — Вайот процесс описывает случай бесконечной плотности и получается как предельный переход процесса с конечным числом особей, который будем называть агентным (individual-based) процессом вымирания — реколонизации, в дальнейшем просто процесс вымирания — реколонизации.

Наша цель заключается в изучении свойств процесса вымирания — реколонизации, обобщении популяционных параметров на случай непрерывно распределённой популяции и написании симулятора для данного процесса.

1. Описание процесса

Рассмотрим двумерный тор $\mathbb{T}$, реализованный как квадрат с ребром длины L, склеенный по противоположным сторонам. В момент времени $t = 0$ особи распределены согласно однородному пуассоновскому точечному процессу [10] на $\mathbb{T}$ с плотностью ρ. Особь описывается координатой на торе (x^1, x^2) и генотипом, представляющим собой вектор из нулей и единиц длины N_ℓ. В начальный момент времени все координаты векторов генотипа для всех особей независимо и одинаково распределены согласно распределению Бернулли с параметром p. Изменение популяции происходит благодаря событиям вымирания — реколонизации. Назовем временами событий вымирания — реколонизации последовательность $\{t_n, n \in \mathbb{N}\} \subset \mathbb{R}_+$ случайных величин с независимыми приращениями, такую что

$$t_1 \sim Exp(\lambda), \tag{1}$$

$$t_{n+1} - t_n \sim Exp(\lambda), n = 1, 2, \ldots \tag{2}$$

Пусть $x = (x^1, x^2), z = (z^1, z^2) \in \mathbb{T}$. Расстояние на торе между x и z будем формально обозначать как $||x - z||$, положим его равным

$$||x - z|| = \sqrt{\min(|x^1 - z^1|, L - |x^1 - z^1|)^2 + \min(|x^2 - z^2|, L - |x^2 - z^2|)^2}.$$

В момент времени t_n происходит событие вымирания — реколонизации.

1. На торе из равномерного распределения выбирается точка z — центр события. Каждой особи, живущей в момент времени до события назначается величина

$$v(x,z) = \exp\left(-\frac{||x-z||^2}{2\alpha^2\theta^2}\right),$$

где x — координаты особи. Согласно взвешенной вероятности, полученной по значениям $v(x,z)$, выбирается родитель.
2. Каждая особь умирает с вероятностью $u(x,z)$, где

$$u(x,z) = u_0 \exp\left(-\frac{||x-z||^2}{2\theta^2}\right).$$

3. Из обобщенного пуассоновского распределения на торе с плотностью $u(x,z)$ (z фиксировано) генерируются новые особи. Генотип новых особей равен генотипу родителя.

Параметр u_0 описывает интенсивность, с которой умирают особи, а θ и α описывают ширину области, в которой умирают особи и выбирается родитель, соответственно.

2. Теоретические свойства модели

2.1. Время жизни

Время жизни особи считается как разность момента времени события, в котором она умерла и момента времени события, в котором она родилась.

Теорема 1. *Время жизни особи распределено экспоненциально с параметром* $\frac{\lambda}{L^2}\iint_{\mathbb{T}} u_0 \exp -\frac{||s||^2}{2\theta^2} ds^1 ds^2$.

Доказательство. Пусть $P(D|x,z)$ — вероятность, что в ходе события вымирания — реколонизации с центром в z особь, расположенная в x умрет. Эта вероятность равна $u(x,z)$. Так как события вымирания — реколонизации расположены равномерно на $\mathbb{T}$, то

$$P(D|x) = \frac{1}{L^2}\iint_{\mathbb{T}} u(x,z)dz = \frac{1}{L^2}\iint_{\mathbb{T}} u_0 \exp -\frac{||x-z||^2}{2\theta^2} dz^1 dz^2.$$

Интеграл берется по тору, так что, сделав замену $s = x - z$, мы получим, что

$$P(D|x) = \frac{1}{L^2}\iint_{\mathbb{T}} u_0 \exp -\frac{||s||^2}{2\theta^2} ds^1 ds^2.$$

Вероятность не зависит от x, и, так как особи распределены согласно однородному пуассоновскому процессу на торе, то положим

$$p_d = P(D) = \frac{1}{L^2}\iint_{\mathbb{T}} u_0 \exp -\frac{||s||^2}{2\theta^2} ds^1 ds^2.$$

Вероятности смерти особи в каждом событии вымирания — реколонизации равны, а сами события взаимоисключающие, значит, вероятность умереть через k событий вымирания — реколонизации равна

$$p_k = (1-p_d)^{k-1} p_d, k = 1, 2, \ldots$$

Пусть $f_k(t)$ — плотность вероятности суммы k независимых экспоненциально распределенных случайных величин с параметом λ. Такая сумма имеет гамма распределение с параметрами $(k, 1/\lambda)$.

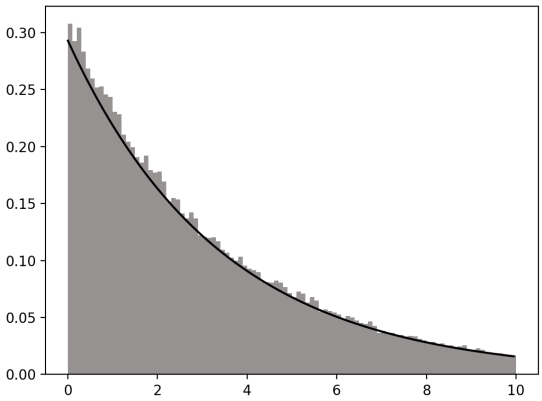


Fig. 1. Lifetime distribution based on simulations with parameters $u_0 = 0.4, \rho = 10000, \theta = 0.4$, and 10000 extinction — recolonization events. Bold line is a theoretical value.

Рис. 1. Распределение времени жизни в симуляции при параметрах $u_0 = 0.4, \rho = 10000, \theta = 0.4$, и 10000 событий вымирания — реколонизации. Выделенная линия — теоретическое значение.

Пусть $P(D_k)$ — вероятность умереть в k-ое событие вымирания — реколонизации, тогда плотность вероятности времени жизни $g(t)$ вычисляется по формуле полной вероятности:

$$g(t) = \sum_{k=1}^{\infty} P(D_k) f_k(t),$$

т. е.

$$g(t) = \sum_{k=1}^{\infty} (1 - p_d)^{k-1} p_d \frac{t^{k-1} \lambda^k e^{-\lambda t}}{\Gamma(k)} = \lambda p_d e^{-\lambda p_d t}.$$

Получается, что время жизни особи имеет экспоненциальное распределение с параметром λp_d. Среднее время жизни равно $\frac{1}{\lambda p_d}$. □

Замечание. Частота смертей равна произведению частоты событий вымирания — реколонизации и вероятности умереть во время этого события, то есть λp_d, значит среднее время жизни равно $1/\lambda p_d$.

2.2. Сохранение вероятностей аллелей

Будем называть координаты генотипа аллелями. Говорим что аллель производная, если соответствующая координата равна единице.

Лемма 1. *Вероятность производного аллеля у особей не зависит от времени.*

Доказательство. При $0 \le t < t_1$ вероятность что аллель производная равна p по начальному условию. При $t_1 \le t < t_2$ для особей, переживших событие вымирания — реколонизации, вероятность производной аллели равна p по начальному условию, а для особей, родившихся в момент вымирания — реколонизации вероятность равна вероятности производного аллеля у родителя, то есть тоже равна p. Аналогично рассматривая остальные события вымирания — реколонизации, получаем, что вероятности не зависят от времени для любого t. □

Лемма 2. *Пусть в момент времени t в популяции существует ровно N индивидуумов, обозначим за G_l^i значение l-ой координаты генотипа i особи тогда* $P(G_l^{i_0} = 1 | \sum_{i=1}^{N} G_l^i = k) = \frac{k}{N}$.

Доказательство.

$$P\left(G_l^{i_0}=1|\sum_{i=1}^{N}G_l^i=k\right)=\frac{P(G_l^{i_0}=1)P(\sum_{i=1,i\neq i_0}^{N}G_l^i=k-1)}{P(\sum_{i=1}^{N}G_l^i=k)}=$$

$$=\frac{p\binom{N-1}{k-1}p^{k-1}(1-p)^{N-k}}{\binom{N}{k}p^k(1-p)^{N-k}}=\frac{\binom{N-1}{k-1}}{\binom{N}{k}}=\frac{k}{N}. \quad (3)$$

□

2.3. Связь с моделью Морана

Зафиксируем один аллель (координату в генотипе). Пусть X_t — число особей с производным аллелем в момент времени t, тогда назовем величины $p_{ml}(n)=P(X_{t_{n+1}}=m|X_{t_n}=l)$ переходными вероятностями числа особей с производным аллелем. Рассмотрим следующую (неэквивалентную) модификацию. Пусть $t_n=n$, а в событиях вымирания — реколонизации умирает и рождается ровно одна особь.

Теорема 2. *Переходные вероятности для модификации процесса равны переходным вероятностям для модели Морана.*

Доказательство. Зафиксируем число особей N. Согласно лемме 2, вероятность выбрать особь с производным аллелем при условии того, что в популяции их k равна k/N. Пусть X_t — число особей с производным аллелем в момент времени t, тогда

$$P(X_{t+1}=k+1|X_t=k)=\frac{k}{N}\left(1-\frac{k}{N}\right), \quad (4)$$

$$P(X_{t+1}=k-1|X_t=k)=\frac{k}{N}\left(1-\frac{k}{N}\right), \quad (5)$$

$$P(X_{t+1}=k|X_t=k)=1-2\frac{k}{N}\left(1-\frac{k}{N}\right). \quad (6)$$

В остальных случаях вероятность равна нулю. □

Так как вероятности равны вероятностям в модели Морана, то все свойства, такие как время фиксации, гетерозиготность и т. д. переносятся на модель вымирания — реколонизации с этими ограничениями.

3. Популяционные характеристики для анализа непрерывно расположенных популяций

Существует множество характеристик для анализа популяций, таких как распределение аллелей, совместное распределение аллелей, гетерозиготность и т. д. Эти характеристики не учитывают пространственное расположение: в случае нескольких популяций они жестко разделяют их. Для модели вымирания — реколонизации и других пространственных моделей необходимо разработать обобщения характеристик, учитывающих пространственную структуру популяций.

3.1. Распределение аллелей

Предположим что особи в популяции расположены согласно нормальному распределению с центром в точке z. Рассмотрим функцию

$$h(z,x;\beta)=\exp\left(-\frac{||x-z||^2}{2\beta^2}\right).$$

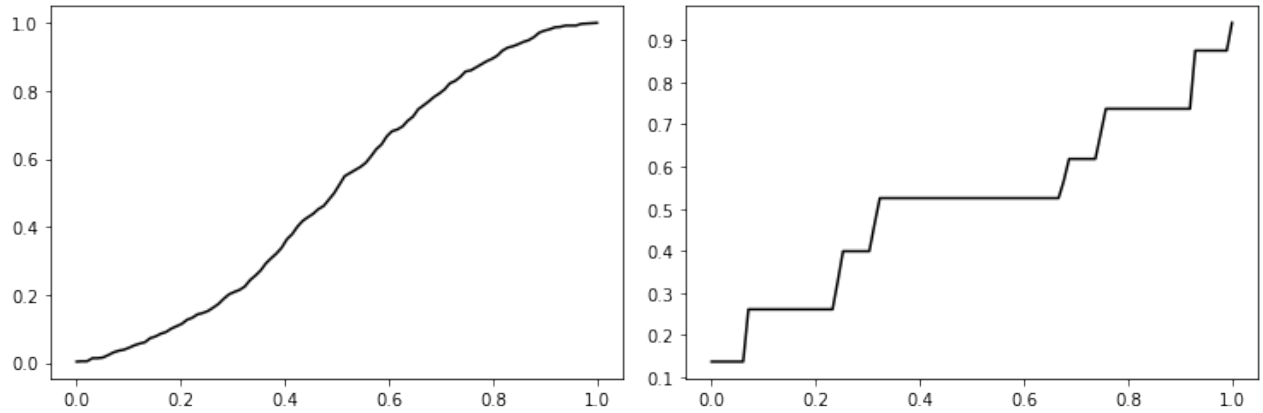


Fig. 2. Allele distribution with $u_0 = 0.1, \rho = 10000, \theta = 0.15, \alpha = 1, N_\ell = 1000$. Left is distribution after 1000 events of extinction — recolonization, right is distribution after 5000 events of extinction — recolonization.

Рис. 2. Распределение аллелей при $u_0 = 0.1, \rho = 10000, \theta = 0.15, \alpha = 1, N_\ell = 1000$. Слева — после 1000 событий вымирания реколонизации, справа — после 5000 событий вымирания — реколонизации.

Такая функция будет придавать больший вес особям, находящимся ближе к центру, и чем дальше особь от центра популяции, тем меньше её вклад.

Мы определим частоту аллеля l в точке x как

$$\theta^l = \frac{\sum_{x_k^{(t)} \sim l1} h(x, x_k^{(t)})}{\sum_{x_k^{(t)}} h(x, x_k^{(t)})},$$

где сумма в числителе берется по всем живым особям с производным аллелем в позиции l в момент времени t, а знаменатель берется по всем живым особям в момент времени t.

Затем распределение аллелей определяется как

$$D^t(x, \theta) = \#\{l : \theta_l \le \theta\}/N_\ell,$$

где # обозначает количество элементов в множестве.

Замечание. Классическое распределение аллелей ξ_i, как правило, вводится как число сайтов (координат), которые являются производными ровно в i особях. Т. е, стандартное определение будет представлять гистограмму, а нововведённое — функцию распределения. Ясно, что

$$\frac{\xi_i}{N_\ell} = \lim_{\beta \to \infty, \varepsilon \to 0+} D(x, i/N) - D(x, i/N - \varepsilon).$$

Мы также вводим совместное распределение аллелей для двух популяций, расположенных в точках x и y:

$$\theta_x^l = \frac{\sum_{x_k^{(t)} \sim l1} h(x, x_k^{(t)})}{\sum_{x_k^{(t)}} h(x, x_k^{(t)})},$$

$$\theta_y^l = \frac{\sum_{x_k^{(t)} \sim l1} h(y, x_k^{(t)})}{\sum_{x_k^{(t)}} h(y, x_k^{(t)})},$$

$$D^t(x, y, \theta_1, \theta_2) = \#\{l : \theta_x^l \le \theta_1, \theta_y^l \le \theta_2\}/N_{al}.$$

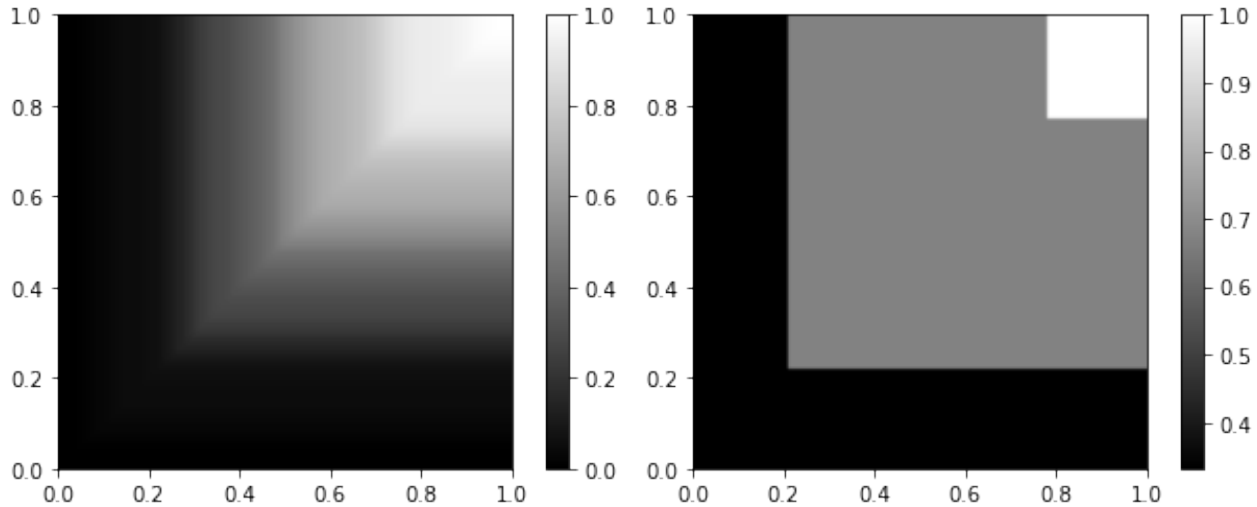


Fig. 3. Joint allele distribution for $\rho = 10000, \theta = 0.15, u_0 = 0.1$. On the left — after 1000 extinction — recolonization events, on the right — after 5000 extinction — recolonization events.

Рис. 3. Совместное распределение аллелей для $\rho = 10000, \theta = 0.15, u_0 = 0.1$. Слева — после 1000 событий вымирания — реколонизация, справа — после 5000 событий вымирания — реколонизации.

3.2. Гетерозиготность

Пусть X_t — количество особей с производным аллелем, а N_t — количество живых особей в момент времени t. Тогда величина $H_t = \frac{X_t(N_t - X_t)}{N^2}$ называется гетерозиготностью. Нулевая гетерозиготность указывает на фиксацию варианта аллеля. Вычисление средней гетерозиготности для популяции осложняется случайным размером популяции и случайным числом новых индивидуумов на каждом шаге. Однако, при наложенных ограничениях в 2.3, средняя гетерозиготность вычисляется аналогично модели Морана (см. например [11]):

$$\mathbb{E}(H_t) = H_0(1 - 2/N^2)^{\lfloor t \rfloor},$$

где $\lfloor \cdot \rfloor$ — функция округления в меньшую сторону. В общем случае были проведены симуляции с различными параметрами u_0 и θ, показывающие уменьшение гетерозиготности.

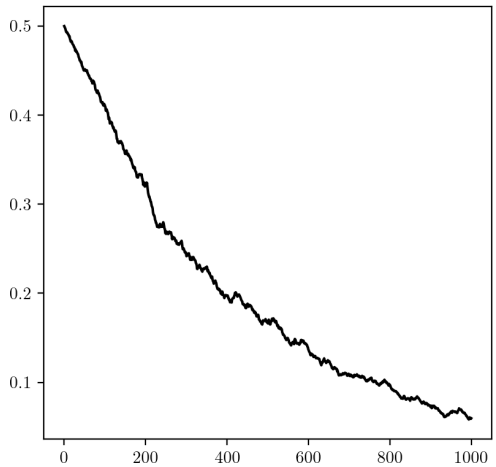
4. Программный симулятор

Был разработан симулятор для модели вымирания — реколонизации. Код написан на языке Python с использованием Cython для оптимизации. Для генерации индивидуумов с использованием неоднородного обобщенного пуассоновского процесса используется метод выборки с отклонением (acceptance-rejection) [10]: количество новых индивидуумов получается из распределения Пуассона с параметром $\Lambda = \iint_{\mathbb{T}} u(x,z)dx^1dx^2 = L^2 p_d$. Новые индивидуумы генерируются равномерно на торе, а затем либо принимаются с вероятностью $u(z,x)/\Lambda$, либо отклоняются и генерируются заново.

Код симулятора доступен на GitHub по ссылке: https://github.com/Genomics-HSE/slfvp.

5. Численные результаты

Модель вымирания — реколонизации представляет новый способ исследования генетического разнообразия популяции, учитывая пространственное расположение её особей. Для модели вымирания-реколонизации были получены некоторые свойства. Было обнаружено, что продолжительность жизни индивидуумов имеет экспоненциальное распределение с средним, равным $1/\lambda p_d$. Вероятности производных аллелей у индивидуумов сохраняются со временем. Было доказа-



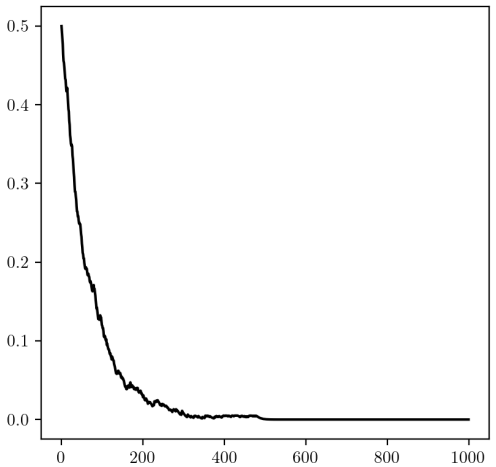
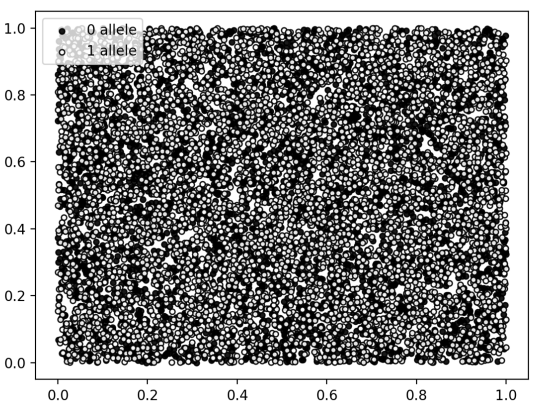


Fig. 4. Average heterozygosity at parameters $\theta = 0.6$, $\rho = 2000$, and 1000 extinction — recolonization events. Averaging was performed over 100 runs. On the left, $u_0 = 0.2$, on the right, $u_0 = 0.4$.

Рис. 4. Усредненная гетерозиготность при параметрах $\theta = 0.6$, $\rho = 2000$ и 1000 событиям вымирания — реколонизации. Усреднение проводилось по 100 запускам. Слева $u_0 = 0.2$, справа $u_0 = 0.4$



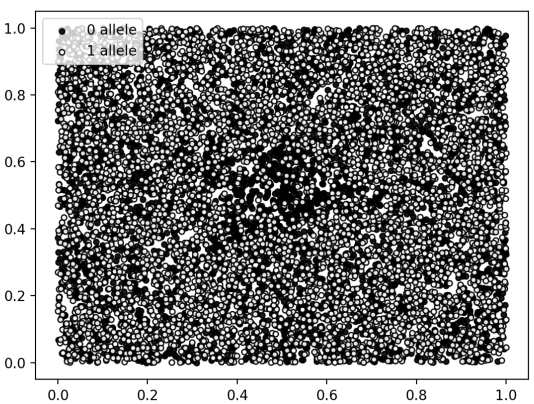


Fig. 5. Individuals with their alleles after one event of extinction — recolonization with parameters $\theta = 0.1, u_0 = 0.9, \rho = 10000$.

Рис. 5. Особи и их аллели до и после одного события вымирания — реколонизации с различными $\theta = 0.1, u_0 = 0.9, \rho = 10000$.

но, что процесс вымирания — реколонизации, при дополнительных ограничениях, сводится к модели Морана. Были определены пространственные аналоги распределения аллелей и совместного распределения аллелей.

Для анализа модели вымирания — реколонизации был разработан симулятор. Теоретические результаты были сопоставлены с результатами симуляции. Время симуляции (в среднем) масштабируется линейно с числом событий, плотностями и квадратично с длиной тора. Значительный вклад в оптимизацию времени симуляции внес выбор тора в качестве пространства, на котором живут особи, что делает интеграл $\iint_{\mathbb{T}} v(z,x)dz$ независимым от x. Для различных параметров модели были получены графики гетерозиготности.

Выводы

Популяционные модели, учитывающие пространственное расположение, представляют собой новое направление в популяционной генетике. Несмотря на ограничения, выявленные в [2], существуют процессы, успешно моделирующие генетическое разнообразие в пространстве. Рассмотренная модель вымирания — реколонизации относится к этому классу. Она, некоторой модификацией, рассмотренной в 2.3, сводится к модели Морана, и может рассматриваться как её обобщение. Всё ещё остаются вопросы по оценке средней гетерозиготности в общем случае. Неизвестно, насколько структура тора применима к реальным сценариям. В качестве дальнейшего изучения пространственных популяционных моделей интересно привлечение мутаций, отбора, рекомбинации для моделирования популяции. Интересно, как наши новые пространственные распределения частот аллелей совпадают с исходными.

References

[1] R. Durrett and R. Durrett, *Probability models for DNA sequence evolution*. Springer, 2008. DOI: 10.1007/978-0-387-78168-6.

[2] J. Felsenstein, "A pain in the torus: Some difficulties with models of isolation by distance", *The American Naturalist*, vol. 109, no. 967, pp. 359–368, 1975.

[3] A. M. Etheridge, "Survival and extinction in a locally regulated population", *The Annals of Applied Probability*, vol. 14, no. 1, pp. 188–214, 2004.

[4] A. Etheridge, *Some Mathematical Models from Population Genetics: École D'Été de Probabilités de Saint-Flour XXXIX-2009*. Springer Science & Business Media, 2011. DOI: 10.1007/978-3-642-16632-7.

[5] N. Barton, A. Etheridge, and A. Véber, "A new model for evolution in a spatial continuum", *Electronic Journal of Probability*, vol. 15, pp. 162–216, 2010. DOI: 10.1214/EJP.v15-741.

[6] N. Biswas, A. Etheridge, and A. Klimek, "The spatial Lambda-Fleming-Viot process with fluctuating selection", *Electronic Journal of Probability*, vol. 26, pp. 1–51, 2021. DOI: 10.1214/21-EJP593.

[7] A. M. Etheridge, A. Véber, and F. Yu, "Rescaling limits of the spatial Lambda-Fleming-Viot process with selection", *Electronic Journal of Probability*, vol. 25, pp. 1–89, 2020. DOI: 10.1214/20-EJP523.

[8] S. Guindon, H. Guo, and D. Welch, "Demographic inference under the coalescent in a spatial continuum", *Theoretical population biology*, vol. 111, pp. 43–50, 2016.

[9] T. Joseph, M. Hickerson, and D. Alvarado-Serrano, "Demographic inference under a spatially continuous coalescent model", *Heredity*, vol. 117, no. 2, pp. 94–99, 2016.

[10] R. L. Streit and R. L. Streit, *The Poisson point process*. Springer, 2010. DOI: 10.1007/978-1-4419-6923-1_2.

[11] J. Wakely, *Coalescent Theory: An Introduction*. Macmillan Learning, 2016, ISBN: 9780974707754.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 2, 2024
journal homepage: www.mais-journal.ru

COMPUTER SYSTEM ORGANIZATION

On the Study of One Way to Detect Anomalous Program Execution

Y. V. Kosolapov[1], T. A. Pavlova[1]

DOI: 10.18255/1818-1015-2024-2-152-163

[1]Southern Federal University, Rostov-on-Don, Russia

MSC2020: 93B11
Research article
Full text in Russian

Received May 18, 2024
Revised May 27, 2024
Accepted May 29, 2024

Developing more accurate and adaptive methods for detecting malicious code is a critical challenge in the context of constantly evolving cybersecurity threats. This requires constant attention to new vulnerabilities and attack methods, as well as the search for innovative approaches to detecting and preventing cyber threats. The paper examines an algorithm for detecting the execution of malicious code in the process of a protected program. This algorithm is based on a previously proposed approach, when the legitimate execution of a protected program is described by a profile of differences in the return addresses of called functions, also called a distance profile. A concept has been introduced called positional distance, which is determined by the difference between the call numbers in the program trace. The main change was the ability to add to the profile the distances between the return addresses of not only neighboring functions, but also several previous ones with a given positional distance. In addition to modifying the detection algorithm, the work developed a tool for automating the construction of a distance profile and experimentally studied the dependence of the probability of false detection of an atypical distance on the training duration for four well-known browsers. Experiments confirm that with a slight increase in verification time, the number of atypical distances detected by the proposed algorithm can be significantly less than the number of atypical distances detected by the basic algorithm. However, it should be noted that the effect of the transition from the basic algorithm to the proposed one, as the results showed, depends on the characteristics of the specific program being protected. The study highlights the importance of continually improving malware detection techniques to adapt them to changing threats and software operating conditions. As a result, this will ensure more reliable protection of information and systems from cyber attacks and other cyber threats.

Keywords: exploits; program protection; abnormal program execution

INFORMATION ABOUT THE AUTHORS

Kosolapov, Yury V. (corresponding author)	ORCID iD: 0000-0002-1491-524X. E-mail: itaim@mail.ru Associated professor, PhD
Pavlova, Tatjyana A.	ORCID iD: 0009-0007-4565-6950. E-mail: tapavlova@sfedu.ru Student

For citation: Y. V. Kosolapov and T. A. Pavlova, "On the study of one way to detect anomalous program execution", *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 152–163, 2024. DOI: 10.18255/1818-1015-2024-2-152-163.

© Kosolapov Y. V., Pavlova T. A., 2024
This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 2, 2024
сайт журнала: www.mais-journal.ru

COMPUTER SYSTEM ORGANIZATION

Об исследовании одного способа выявления аномального выполнения программы

Ю. В. Косолапов [1], Т. А. Павлова[1]

DOI: 10.18255/1818-1015-2024-2-152-163

[1]Южный федеральный университет, Ростов-на-Дону, Россия

УДК 004.056.5
Научная статья
Полный текст на русском языке

Получена 18 мая 2024 г.
После доработки 27 мая 2024 г.
Принята к публикации 29 мая 2024 г.

Разработка более точных и адаптивных методов обнаружения вредоносного кода является критической задачей в контексте постоянно эволюционирующих угроз кибербезопасности. Это требует постоянного внимания к новым уязвимостям и методам атак, а также поиска инновационных подходов к обнаружению и предотвращению киберугроз. В работе исследуется алгоритм обнаружения исполнения вредоносного кода в процессе защищаемой программы. Этот алгоритм основан на ранее предложенном подходе, когда легитимное исполнение защищаемой программы описывается профилем разностей адресов возврата вызываемых функций, называемым также профилем расстояний. Введено такое понятие, как позиционное расстояние, которое определяется разницей между номерами вызовов в трассе программы. Основным изменением стала возможность добавления в профиль расстояний между адресами возврата не только соседних функций, а также нескольких предыдущих с заданным позиционным расстоянием. Кроме модификации алгоритма обнаружения, в работе разработано средство автоматизации построения профиля расстояний и экспериментально исследуется зависимость вероятности ложного обнаружения нетипичного расстояния от длительности обучения для четырех известных браузеров. Эксперименты подтверждают, что при незначительном увеличении времени проверки число нетипичных расстояний, обнаруживаемых предложенным алгоритмом, может быть существенно меньше числа нетипичных расстояний, выявляемых базовым алгоритмом. Однако следует отметить, что при этом эффект перехода от базового алгоритма к предложенному, как показали результаты, зависит от характеристик конкретной защищаемой программы. Исследование подчеркивает важность постоянного совершенствования методов обнаружения вредоносного кода, чтобы адаптировать их к изменяющимся угрозам и условиям эксплуатации программного обеспечения. В итоге это позволит обеспечить более надежную защиту информации и систем от кибератак и других киберугроз.

Ключевые слова: эксплойты; защита программ; аномальное выполнение программы

ИНФОРМАЦИЯ ОБ АВТОРАХ

Косолапов, Юрий Владимирович (автор для корреспонденции) | ORCID iD: 0000-0002-1491-524X. E-mail: itaim@mail.ru
Доцент, канд. техн. наук

Павлова, Татьяна Александровна | ORCID iD: 0009-0007-4565-6950. E-mail: tapavlova@sfedu.ru
Студент

Для цитирования: Y. V. Kosolapov and T. A. Pavlova, “On the study of one way to detect anomalous program execution”, *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 152–163, 2024. DOI: 10.18255/1818-1015-2024-2-152-163.

© Косолапов Ю. В., Павлова Т. А., 2024
Эта статья открытого доступа под лицензией CC BY license (https://creativecommons.org/licenses/by/4.0/).

Введение

Актуальным направлением защиты программного обеспечения (ПО) является защита от эксплойтов — исполнимого кода, использующего уязвимости в ПО с целью нарушения его штатной работы. Помимо установки обновлений, устраняющих уязвимости в ПО, и поиска различными техниками возможных уязвимостей, можно выделить два основных способа борьбы с эксплойтами [1]: сигнатурный способ и способ на основе выявления аномалий. Первый предполагает, что разработчикам средств защиты уже известны особенности эксплойта (сигнатура), и поэтому защита программ выполняется путем сравнения входных данных с сигнатурой. Недостатком такого подхода является невозможность обнаружения новых эксплойтов («эксплойтов нулевого дня»). Во втором способе для защищаемой программы в период обучения строится профиль «нормального» состояния, а защита программы заключается в выявлении отклонений от этого состояния (аномалий).

Одним из известных подходов построения профиля является подход на основе цепочек системных вызовов [2]. При таком подходе возможно обнаружение новых эксплойтов, но также имеется ненулевая вероятность принять легитимное исполнение программы за аномалию. Однако более существенной проблемой этого подхода является невозможность обнаружения мимикрирующих эксплойтов [3].

В работах [4, 5] предложен способ обнаружения эксплойтов, в котором на этапе обучения строятся профили расстояний между системными вызовами, а на этапе тестирования (эксплуатации) выполняется проверка вызовов на типичность — принадлежность профилям. В [4] предлагается использовать взвешенный показатель нетипичности вызова, значение которого зависит от числа профилей, по которым выполняется проверка, и количества учитываемых предыдущих вызовов. В этом подходе расстояния до всех учитываемых предыдущих вызовов сравниваются со всеми профилями, построенными при обучении. В [5] используется пороговое значение на количество предыдущих типичных вызовов, при превышении которого вызов считается типичным. При этом расстояния до предыдущих вызовов проверяются не по всем профилям: если разность номеров позиций вызовов в тестируемой трассе вызовов равна i, то и проверка выполняется по профилю, в котором расстояния вычисляются между вызовами в обучающей трассе, разность номеров позиций которых также равна i.

В [4, 5] с помощью средства перехвата вызовов API Monitor [6] получены предварительные экспериментальные результаты для защищаемой программы Firefox. При этом в [4] целью экспериментов было исследование зависимости вероятности ложного обнаружения от числа предыдущих вызовов и числа профилей, а в [5] — зависимость этой вероятности от длительности обучения. Формат хранения данных API Monitor не позволяет автоматизировать накопление данных для построения профилей. Поэтому в [7] подход, лежащий в основе алгоритмов [4, 5], реализован для 32-битных операционных систем Windows: реализован механизм отслеживания вызовов в защищаемой программе и разработаны утилиты для построения и обновления профиля по обучающим трассам, а также утилита для проверки типичности вызовов в тестируемой трассе. Отметим, что в [7] строится только один профиль расстояний — для соседних вызовов; проверка типичности вызовов выполняется также только по соседним вызовам. Результаты экспериментального исследования, проведенного в [7] для защищаемой программы Firefox, показали нулевую частоту ложного пропуска смоделированного поведения эксплойта, при этом частота ложного обнаружения эксплойта имеет тенденцию к снижению в зависимости от длительности обучения, что подтвердило предварительные результаты работы [5] для программы Firefox. Одним из препятствий более полного исследования зависимости вероятности ложного обнаружения от длительности обучения является ручной ввод входных данных для защищаемой программы: в экспериментах [4, 5, 7] все действия на сайтах, загружаемых Firefox, выполнялись вручную.

В настоящей работе ставятся следующие задачи: 1) дальнейшее исследование характеристик способа, лежащего в основе алгоритмов [4, 5], с целью изучения возможностей уменьшения вероятности ложного обнаружения эксплойтов; 2) автоматизация построения профиля расстояний для защищаемых программ с графическим пользовательским интерфейсом; 3) экспериментальное исследование зависимости вероятности ложного обнаружения от длительности обучения на большем количестве входных данных.

1. Базовый алгоритм из [4, 5]

Сначала в удобном виде опишем базовый способ обнаружения, лежащий в основе алгоритмов из [4, 5]. Пусть P — защищаемая программа, L — множество имен отслеживаемых функций, используемых программой. Во множество отслеживаемых функций могут входить, например функции, которые обычно используются эксплойтами [8]. Вызов функции определим в виде пары $c = (n, r)$, где $n \in L$ — имя вызываемой функции, r — адрес возврата из этой функции. Отметим, что модули программы P в силу технологии рандомизации размещения адресного пространства могут не иметь постоянного адреса загрузки. Однако, так как информация о размещении модулей известна после запуска программы, то без потери общности можно считать, что каждый модуль всегда имеет фиксированный адрес загрузки [7]. Тогда программа P может быть представлена в виде ориентированного графа $G(P) = (V, E \subseteq V \times V)$, где множество узлов V — это множество разных вызовов функций с именами функций из множества L, а пара вызовов $(c = (n, r), \tilde{c} = (\tilde{n}, \tilde{r}))$ принадлежит множеству ребер E, если существуют такие входные данные, при которых после вызова $c = (n, r)$ следует вызов $\tilde{c} = (\tilde{n}, \tilde{r})$. Все вызовы из V, а также все возможные пути в графе $G(P)$ будем называть легитимными. Ребру $e = (c, \tilde{c}) \in E$, где $c = (n, r)$, $\tilde{c} = (\tilde{n}, \tilde{r})$, поставим в соответствие число $d(c, \tilde{c}) = d(e) = \tilde{r} - r$, которое в [4, 5] для удобства названо расстоянием между вызовами $\tilde{c}$ и c. (Отметим, что в математическом смысле, $d(c, \tilde{c})$ не является расстоянием, так как $d(c, \tilde{c}) \neq d(\tilde{c}, c)$.)

Граф $G(P)$, для каждого ребра v которого известно число $d(e)$, может следующим образом использоваться для выявления исполнения эксплойта: если при выполнении программы найден вызов $\tilde{c} = (\tilde{n}, \tilde{r})$, расстояние от которого до предыдущего вызова $c = (n, r)$ не равно $d(e)$ для всех $e = ((n, r'), (\tilde{n}, r'')) \in E$, то вызов $\tilde{c}$ считается нелегитимным или нетипичным. Однако построение графа $G(P)$ для больших программ представляется сложной задачей. Информация о расстояниях между вызовами (не вся) может быть получена из трасс исполнения программы P путем многократного запуска этой программы на разных входных данных. Трассой вызовов T назовем путь в графе $G(P)$ при заданных входных данных. Трасса T может быть представлена в виде последовательности вызовов $(c_i)_{i=1}^{N}$. Для двух последовательных вызовов $c_i = (n_i, r_i)$ и $c_{i+1} = (n_{i+1}, r_{i+1})$ расстоянием между ними в [4, 5] названо число $d(c_i, c_{i+1}) = r_{i+1} - r_i$. В общем случае расстояние между вызовами c_i и c_j, $i \leqslant j$, естественно определить так: $d(c_i, c_j) = r_j - r_i$. Здесь предполагается, что расстояния могут вычисляться только для вызовов, совершенных из одного потока [4, 5]. Без потери общности считается, что трасса T состоит из вызовов одного потока. Трассу вызовов, используемую для построения профиля на этапе обучения или дообучения, обозначим T_{learn}, а трассу, вызовы которой проверяются на типичность, обозначим T_{test}. На основе обучающей трассы вызовов $T_{learn} = (c_i)_{i=1}^{N}$, полученной путем запуска программы P с разными входными данными в период обучения, может быть построен *профиль расстояний* $D(P)$, представляющий собой набор множеств вида

$$D_{(n,m)} = \{d(c_i, c_{i+1}) : c_i, c_{i+1} \in T_{learn}, i \in [1, N-1], \text{name}(c_i) = n, \text{name}(c_{i+1}) = m\}, \tag{1}$$

где $\text{name}(c) = n$ для $c = (n, r)$. Таким образом, профиль $D(P)$ состоит из $2 \cdot \binom{|L|}{2}$ множеств вида (1): $D(P) = \left(D_{(n,m)} : n, m \in L\right)$. Некоторые из этих множеств могут быть пустыми, что означает, что в трассе T_{learn} программы P не встречалось пары соседних вызовов с соответствующими именами. Профиль $D(P)$ может строиться по разным легитимным запускам программы P и представляет

собой описание её нормального поведения. Способ выявления эксплойтов через аномалии в поведении сводится к обнаружению нетипичных вызовов в трассе T_{test}, то есть таких вызовов, расстояния от которых до предыдущих вызовов не входят в профиль $D(P)$.

Отсутствие расстояния в профиле $D(P)$ может свидетельствовать о том, что был произведен запуск эксплойта или о том, что при легитимном исполнении программы была задействована ветвь исполнения кода, не встречавшаяся при обучении (ложное обнаружение). В работах [5, 7] на примере программы Firefox экспериментально продемонстрировано, что, с одной стороны, выполнение эксплойта (на примере одного реального эксплойта и одной модели эксплойта) всегда обнаруживается, а с другой стороны, при увеличении длительности обучения вероятность ложного обнаружения снижается. Для практического использования вероятность ложного обнаружения должна быть как можно меньше, в идеале она должна быть нулевой. В следующих двух разделах строится и экспериментально исследуется одна модификация подхода из работы [4].

2. Модификация алгоритма из [4]

Пусть $T_{learn} = (c_i)_{i=1}^{N}$ — обучающая трасса защищаемой программы P. Для вызовов c_i и c_j, $j > i$, из трассы вызовов T *позиционным расстоянием* между этими вызовами назовем число $pd(c_i, c_j) = j - i$. Введем три целочисленных параметра: глубину построения профиля при обучении (LMD — learning model depth), глубину проверки трассы (TTD — test trace depth) и глубину проверки профилей (TMD — test model depth), при этом $TTD \leqslant LMD$ и $TMD \leqslant LMD$. Профилем расстояний $D_{learn}(P)$ программы P назовем семейство множеств

$$D_{learn}(P) = \left(D^{j}_{(n,m)} | n, m \in L, j \in [1, LMD]\right), \tag{2}$$

где множество $D^{j}_{(n,m)}$ имеет вид

$$D^{j}_{(n,m)} = \{d(c, \tilde{c}) | c, \tilde{c} \in T_{learn}, \text{name}(c) = n, \text{name}(\tilde{c}) = m, pd(c, \tilde{c}) = j\}.$$

Таким образом, $D^{j}_{(n,m)}$ состоит из возможных расстояний между вызовами c и $\tilde{c}$, находящимися на позиционном расстоянии j, причем в вызове c имя функции равно n, а в вызове $\tilde{c}$ — m. Построение профиля описано в алгоритме 1. Алгоритм Train принимает на вход начальное значение профиля, обучающую трассу вызовов и параметр LMD. Начальное значение профиля может быть пустым, когда профиль создается впервые, или непустым, когда требуется пополнить (дообучить) профиль на основе новых трасс. Отметим, что такой подход построения профиля используется в [4, 5].

Способ выявления эксплойтов через аномалии в поведении можно свести к построению профиля расстояний тестового запуска и обнаружению расстояний из этого профиля, не входящих в профиль $D_{learn}(P)$, а значит ни в одно множество $D^{j}_{(n,m)}$, где параметр j меняется от 1 до $TMD(\leqslant LMD)$. Профиль расстояний тестового запуска имеет вид

$$D_{test}(P) = \{\tilde{D}^{j}_{(n,m)} | n, m \in L, j \in [1, TTD]\}, \tag{3}$$

где переменная TTD отвечает за глубину профиля (т. е. какое количество предыдущих вызовов для текущего участвуют в профиле), аналогично переменной LMD при обучении трассы, а

$$\tilde{D}^{j}_{(n,m)} = \{d(c, \tilde{c}) | c, \tilde{c} \in T_{test}, \text{name}(c) = n, \text{name}(\tilde{c}) = m, pd(c, \tilde{c}) = j\}.$$

Отметим, что при таком подходе выявления аномалий фиксируется число нетипичных расстояний, а не число нетипичных вызовов, однако очевидно, что этот подход может быть модифицирован для подсчета числа нетипичных вызовов.

Algorithm 1. Train — the algorithm for constructing (updating) a distance profile

Алгоритм 1. Train — алгоритм построения (обновления) профиля расстояний

Data: $D_{learn}(P)$, $T_{train} = (c_i)_{i=1}^{N}$, $LMD \in \mathbb{N}$
Result: $D_{learn}(P)$ — обновленный профиль расстояний
/* Цикл по всем вызовам трассы T_{train}, начиная со второго */
for $i \in [2, N]$ **do**
  /* Получить имя функции в c_i */
  $m = \text{name}(c_i)$
  /* Цикл по вызовам, находящимся на позиционном расстоянии не более LMD */
  for $k \in [1, \min\{LMD, i-1\}]$ **do**
    /* Получить имя функции в c_{i-k}, находящемся на позиционном расстоянии k от c_i */
    $n = \text{name}(c_{i-k})$
    /* Если в $D_{learn}(P)$ еще нет элемента $D^k_{(n,m)}$, то добавить в список $D_{learn}(P)$ элемент с номером $D^k_{(n,m)}$, и инициализировать этот элемент пустым значением */
    if $D^k_{(n,m)} \notin D_{learn}(P)$ **then**
      $D^k_{(n,m)} = \varnothing$
      $D_{learn}(P) = D_{learn}(P) \parallel D^k_{(n,m)}$
    /* Во множество $D^k_{(n,m)}$ добавить расстояние $d(c_{i-k}, c_i)$ */
    $D^k_{(n,m)} = D^j_{(n,m)} \cup \{d(c_{i-k}, c_i)\}$
return $D_{learn}(P)$

Алгоритм CheckTraceBack проверки трассы на наличие нетипичных расстояний представлен в виде псевдокода 2. Кроме профиля расстояний, тестовой трассы, параметров *TMD* и *TTD*, этот алгоритм принимает на вход булев аргумент *learn*, при истинности которого выполняется дообучение профилей в момент проверки трассы. Под дообучением здесь понимается добавление во множество $D^i_{(n,m)}$ таких расстояний из $\tilde{D}^i_{(n,m)}$, которые были определены как типичные, то есть были найдены в $D^j_{(n,m)}$, где $j \in [1, TMD]$. Отличие предлагаемого способа от способа из [4] заключается в том, что 1) введен дополнительный параметр *TMD*, который регулирует глубину проверки расстояний, 2) при обучении обновляется не профиль $D^1_{(n,m)}$, а соответствующий $D^i_{(n,m)}$, что, как представляется, в дальнейшем позволит более тонко проводить анализ типичности; 3) нетипичность определяется не на основе взвешенного значения типичности, а на основе простого правила: вызов считается нетипичным, если ни одно из *TTD* расстояний до этого вызова не принадлежит ни одному из *TMD* профилей.

Для каждого вызова c_i, где $i \geqslant 1$ — номер вызова в трассе T программы, определим список из r предыдущих вызовов $\text{prev}_r(c_i, T) = (pc^i_1, pc^i_2, \ldots, pc^i_k)$, где $k = r$, если $r < i$, в противном случае $k = i$. Таким образом, $pc^i_j = c_{i-j}$, то есть вызов c_i находится на позиционном расстоянии j от вызова pc^i_j. Из построения алгоритма CheckTraceBack вытекают следующие утверждения.

Утверждение 1. *Пусть* $LMD, TMD, TTD \in \mathbb{N}$ *— такие, что* $TMD, TTD \leqslant LMD$. *Легитимный вызов* $c \in T_{test}$ *считается типичным в* CheckTraceBack, *если найдется* $p \in \text{prev}_{TTD}(c, T_{test}) \cap \text{prev}_{TMD}(c, T_{learn})$.

Утверждение 2. *Пусть* $LMD, TMD, TTD \in \mathbb{N}$ *— такие, что* $TMD \leqslant LMD$, $TTD \leqslant LMD$. *Вызов* $c \in T_{test}$ *в* CheckTraceBack *считается типичным, если найдется* $p \in \text{prev}_{TTD}(c, T_{test})$, *для которого в* T_{learn} *найдется* $\tilde{c}$ *и такой* $\tilde{p} \in \text{prev}_{TMD}(\tilde{c}, T_{learn})$, *что* $\text{name}(\tilde{p}) = \text{name}(p)$, $\text{name}(\tilde{c}) = \text{name}(c)$ *и* $d(\tilde{p}, \tilde{c}) = d(p, c)$.

Algorithm 2. CheckTraceBack

Data: $D_{learn}(P)$, $T_{test} = (c_i)_{i=1}^{M}$, $TMD \in \mathbb{N}$, $TTD \in \mathbb{N}$, $learn \in \{\text{true}, \text{false}\}$
Result: *atypical* — число нетипичных расстояний
/* Инициализировать число нетипичных расстояний */
1 $atypical = 0$
/* Построить профиль расстояний для тестового запуска */
2 $D_{test}(P) = \text{Train}(\varnothing, T_{test}, TTD)$
/* Цикл по всем упорядоченным парам возможных имен функций */
3 **for** $(n, m) \in L \times L$ **do**
  /* Построить общее множество D возможных расстояний для вызовов, находящихся в обучающей трассе на позиционном расстоянии не более TMD */
4 $D = \varnothing$
  /* Расстояния из $\tilde{D}^i_{(n,m)}$, полученного на тестовой трассе, не входящие в D, считаются нетипичными */
5 **for** $i \in [1, TTD]$ **do**
    /* Получить множество $\tilde{D}^i_{(n,m)}$ из $D_{test}(P)$ */
6 $\tilde{D}^i_{(n,m)} \leftarrow D_{test}(P)$
7 $D = \tilde{D}^i_{(n,m)}$
8 **for** $j \in [1, TMD]$ **do**
      /* Получить множество $D^j_{(n,m)}$ из $D_{learn}(P)$ */
9 $D^j_{(n,m)} \leftarrow D_{learn}(P)$
10 **if** *learn* **then**
        /* Произвести дообучение */
11 $D^i_{(n,m)} = D^i_{(n,m)} \cup \left(\tilde{D}^i_{(n,m)} \cap D^j_{(n,m)}\right)$
12 $D = D \setminus D^j_{(n,m)}$
    /* Обновить число нетипичных расстояний */
13 $atypical = atypical + |D|$
14 **return** *atypical*

Из утверждения 1 вытекает, что часть легитимных путей из $G(P)$, которые не встретились в трассе T_{learn}, будут признаны легитимными в алгоритме CheckTraceBack. На рис. 1 изображена часть графа $G(P)$ с тремя легитимными путями исполнения защищаемой программы P: сплошной линией показаны пути $a-b-c-d$ и $a-e-f-g$, которые выполнялись при обучении, а пунктирной линией показан путь $a-b-c-g$, который при обучении не встретился. Расстояние между соседними вызовами c и g, скорее всего, будет нетипичным, однако при $LMD \geqslant 3$ расстояние между вызовами a и g будет во множестве $D^3_{(n,m)}$, где $n = \text{name}(a)$ и $m = \text{name}(g)$. Поэтому исполнение пути $a-b-c-g$ при $TTD, TMD \geqslant 3$ будет признано легитимным.

Из утверждения 2 вытекает, что возможны случаи, когда $c \neq \tilde{c}$, но при этом c считается типичным. К таким случаям относятся: 1) $c \notin T_{learn}$; 2) $c \in T_{learn}$, причем $\text{prev}_{TMD}(c, T_{learn}) \cap \text{prev}_{TTD}(c, T_{test}) = \varnothing$. Типичным вызов $c \in T_{test}$ в таких ситуация может быть признан только в случае, когда найдется $\tilde{c} \in T_{learn}$, что $\text{name}(\tilde{c}) = \text{name}(c)$, причем в $\text{prev}_{TMD}(\tilde{c}, T_{learn})$ имеется хотя бы один такой вызов $\tilde{p}$, что $\text{name}(\tilde{p}) = \text{name}(p)$, $p \in \text{prev}_{TTD}(c, T_{test})$, и $d(p, c) = d(\tilde{p}, \tilde{c})$. Отметим, что случаи 1) и 2) создают потенциальную лазейку для обхода защиты на основе профиля расстояний.

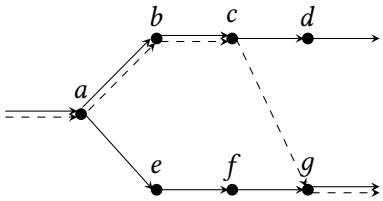


Fig. 1. An example of a legitimate path (shown as a dotted line) in $G(P)$ that would be recognized as legitimate by the CheckTraceBack algorithm for $LMD \geqslant TTD \geqslant 3$

Рис. 1. Пример легитимного пути (показан пунктирной линией) в $G(P)$, который будет распознан как легитимный в алгоритме CheckTraceBack при $LMD \geqslant TTD \geqslant 3$

Таким образом, алгоритм CheckTraceBack может как обнаруживать неизвестные на этапе обучения легитимные пути графа $G(P)$, так и принимать нелегитимные вызовы за легитимные. Похожая особенность отмечена и в [4] для соответствующего алгоритма проверки. Следующий раздел посвящен исследованию зависимости вероятности ложного обнаружения алгоритмом CheckTraceBack от длительности обучения.

3. Экспериментальное исследование

В работах [4, 5] предварительные эксперименты для защищаемой программы Firefox проводились на небольшом числе сайтов. Разработанное в [7] средство позволило автоматизировать процесс накопления профиля для Windows-программ платформы x86 и, в частности, увеличить число сайтов. Однако в [7] также, как и в [4, 5], при экспериментальном исследовании действия на сайтах выполнялись вручную. В настоящей работе доработано средство из [7] с целью автоматизации построения профилей в соответствии с алгоритмом 1 и автоматизации проверки трассы в соответствии с алгоритмом 2. Также разработано средство эмуляции действий пользователя при работе с программами, имеющими графический интерфейс.

В работе в качестве защищаемых программ выбраны четыре Интернет-браузера для операционной системы Windows на платформе x86: Mozilla Firefox 52.9 ESR, MiniBrowser 1.0.0.127, Maxton 5.3.8.2000, Vivaldi 1.0.435.46. В качестве входных данных выбран набор из 2000 сайтов наиболее популярных сайтов по данным www.similarweb.com. Выбранные защищаемые программы обладают графическим пользовательским интерфейсом, поэтому для автоматизации построения профиля разработано средство эмуляции действий пользователя на сайте. В основе разработанного средства лежит библиотека PyAutoGUI [9] для Python, которая предоставляет возможности эмуляции действий пользователя-человека. Она позволяет программно управлять мышью и клавиатурой, выполнять другие действия в графическом пользовательском интерфейсе и получать информацию об экране. При помощи библиотеки PyAutoGUI были разработаны функции, позволяющие перемещать указатель мыши на случайно сгенерированные координаты на экране, кликать мышью, прокручивать колесико мыши, менять масштаб открытой вкладки, а также выделять часть объектов на экране и копировать их в буфер обмена. Для каждого из выбранных браузеров с помощью разработанного средства автоматизации получены трассы вызовов отслеживаемых функций, где в качестве набора отслеживаемых функций L выбраны функции, которые обычно вызываются из эксплойтов [8]. Для каждого браузера было выполнено не менее 100 запусков, где в каждом запуске открывалось 20 сайтов. На каждом сайте выполнялась эмуляция действий пользователя: нажатия мыши в случайных координатах, прокрутка колеса мыши на случайную величину, выделение и копирование случайной области страницы, увеличение и уменьшение масштаба. В результате этих запусков были получены файлы с трассами вызовов. При экспериментальном исследовании трассы вызовов размера меньше 10 МБ не учитывались, а трассы более 400МБ разбивались на части

Table 1. Call trace statistics **Таблица 1.** Статистика трасс вызовов

P	K	S
Mozilla Firefox 52.9 ESR	100	28551930
Maxton 5.3.8.2000	95	57725599
MiniBrowser 1.0.0.127	109	123871995
Vivaldi 1.0.435.46	117	56680562

Table 2. Statistics of called functions **Таблица 2.** Статистика вызываемых функций

name(c) $\in L$	**Firefox**	**MiniBrowser**	**Maxthon**	**Vivaldi**
CreateFileMappingA	0,000742332	3,50239E-08	0	0
CreateFileMappingW	0,002046657	0,004444233	0,013014573	0,006753462
CreateFileMappingNumaW	0	0	0	0
DecodePointer	0,000240964	0,004816537	0,000683787	0,000362417
GetDC	0,003635026	0,001073011	7,81802E-05	9,91522E-06
GetModuleHandleA	6,48292E-05	0,001963777	9,97824E-06	3,55148E-05
GetModuleHandleW	0,002033978	0,684743868	0,000834482	0,000681945
HeapCreate	0	2,29269E-06	1,47248E-06	1,23499E-07
IsDebuggerPresent	0,003687351	6,64476E-05	7,26541E-05	4,56947E-06
LoadResource	0	0	6,65216E-06	8,82137E-08
MapViewOfFile	0,000466729	0,002017736	0,009120979	0,004527637
MapViewOfFileEx	0,001987606	0	0	0
MapViewOfFileFromApp	0	0	0	0
OpenFile	0	0	0	0
ReadFile	0,235920444	0,125970006	0,449466986	0,438129724
SizeofResource	0	0	6,65216E-06	8,82137E-08
VirtualAlloc	0,156430721	2,36615E-05	0,000145343	3,52855E-07
VirtualAllocEx	0,000354932	0,00010016	0,000165438	2,8987E-05
VirtualProtect	0,06968289	2,77948E-05	0,000121956	4,80941E-05
VirtualProtectEx	0,003072472	0	0	0,000163213
WinExec	0	0	0	0
WriteFileEx	0	0	0	0
WriteFile	0,519633069	0,174750467	0,526270866	0,549253869

по 200 МБ. В итоге для каждого из браузеров было построено K файлов с трассами вызовов, где значение K указано в таблице 1. Также в этой таблице приводится общее число вызовов функций S.

Статистика вызываемых функций, приведенная в таблице 2, показывает, что для некоторых защищаемых программ появление функции в трассе может уже свидетельствовать о нетипичном выполнении кода. Для рассмотренных программ таким функциями являются CreateFileMappingNumaW, MapViewOfFileFromApp, OpenFile, WinExec и WriteFileEx.

В [5, 7] зависимость вероятности ложного обнаружения от длительности обучения проводилась путем построения профиля на K трассах и тестирования на X новых трассах, где K растет, а X фиксировано. В настоящей работе исследование зависимости вероятности ложного обнаружения от длительности обучения выполнено по следующей схеме: 1) набор из K файлов трасс случайным образом перемешивается с помощью перестановки π; 2) для каждого $F \in \{1, \ldots, K-1\}$ строится профиль расстояний $D_{learn}(P)$ вида (2) по первым F файлам трасс при $LMD = 20$; 3) на основе файла с номером $F+1$ строится профиль расстояний $D_{test}(P)$ вида (3) для заданного параметра $TTD \in \{1, 2, 5, 10, 20\}$;

Table 3. Average time to check a trace by profile (in milliseconds)

Таблица 3. Среднее время проверки трассы по профилю (в миллисекундах)

	(TTD, TMD)						
	$(1, 1)$	$(2, 5)$	$(5, 2)$	$(5, 10)$	$(10, 5)$	$(10, 20)$	$(20, 10)$
Firefox	8919,39	8998,10	8957,56	9236,05	9237,89	9183,709	9766,80
MiniBrowser	36117,21	35659,88	35856,69	37209,40	37179,05	36238,13	40028,02
Maxthon	15172,39	15642,39	15636,33	16140,23	16165,13	15944,82	17170,20
Vivaldi	17768,15	17899,98	17925,43	18433,87	18450,35	18135,84	17758,96

4) для заданного параметра $TMD \in \{1, 2, 5, 10, 20\}$ с помощью алгоритма CheckTraceBack находится количество нетипичных E_π расстояний, то есть расстояний, не содержащихся в профиле $D_{learn}(P)$; 5) выполняется дообучение профиля $D_{learn}(P)$ путем объединения с профилем $D_{test}(P)$ (шаг 11 алгоритма CheckTraceBack). Шаги 1)-5) выполняются 10 раз для 10 разных случайных перестановок π. Таким образом было получено 10 наборов по $K - 1$ измерению числа нетипичных расстояний. По этим наборам строится набор усредненных значений E числа нетипичных расстояний из $K - 1$ измерения. Использование случайных перестановок позволяет частично нивелировать резкие всплески числа нетипичных расстояний. Такие всплески соответствуют ситуациям, когда трассы для тестируемого профиля $D_{test}(P)$ получены при открытии сайтов, задействующих функционал браузера, не используемый на этапе предыдущего обучения. Отметим, что проверка принадлежности расстояний к профилю может выполняться как в ходе исполнения программы, так и по сгенерированной программой трассе. В первом случае имеется возможность предотвратить выполнение эксплойта, но, как представляется, может замедлиться производительность самой программы. Во втором обнаружение эксплойта выполняется постфактум. В настоящей работе нетипичность расстояний оценивается по сгенерированной трассе. В ходе эксперимента замеряется время проверки наличия нетипичных расстояний в профиле (3) и усредняется по 10 перестановкам. Результаты экспериментов показали, что среднее время проверки сгенерированной трассы имеет тенденцию к росту с увеличением параметров TTD и TMD (см. таблицу 3).

Результаты оценки зависимости вероятности ложного обнаружения от длительности обучения путем нахождения среднего числа нетипичных расстояний в зависимости от длительности обучения показаны на рис. 2. На первых этапах обучения для всех браузеров фиксировалось большое число нетипичных расстояний, поэтому на графиках показаны этапы при $F \geqslant 10$. На рисунке для наглядности изображены только графики при ($TTD = 1$, $TMD = 1$) и ($TTD = 20$, $TMD = 10$). Графики для других сочетаний параметров (TTD, TMD) не приводятся, так как, учитывая незначительность в приросте времени проверки, для оценки эффекта предлагаемого алгоритма целесообразно сравнивать базовый алгоритм ($TTD = 1$, $TMD = 1$) с алгоритмом при больших значениях этих параметров.

Анализ графиков показывает, что 1) большие значения параметров TTD и TMD снижают среднее число нетипичных расстояний, по сравнению с базовым алгоритмом из [7]; 2) эффект от использования больших значений TTD и TMD зависит от защищаемой программы: например, для Firefox, MiniBrowser и Maxthon число нетипичных расстояний, особенно на ранних этапах обучения, заметно ниже для больших значений TTD и TMD, в то время как для Vivaldi этот эффект проявляется в меньшей степени; 3) число нетипичных расстояний имеет тенденцию к снижению в зависимости от длительности обучения, особенно эта тенденция заметна на ранних этапах обучения; 4) обучение на трассах со всех запусков не исключает появления нетипичных расстояний. Последнее наблюдение говорит о том, что без дополнительной информации о вызываемых функциях выявление нетипичного выполнения на основе расстояний между функциями может приводить к нежелательным ложным обнаружениям. В то же время как источник дополнительной информации о выполнении защищаемой программы в системах типа SeismoMeter[10] или в гибридных системах,

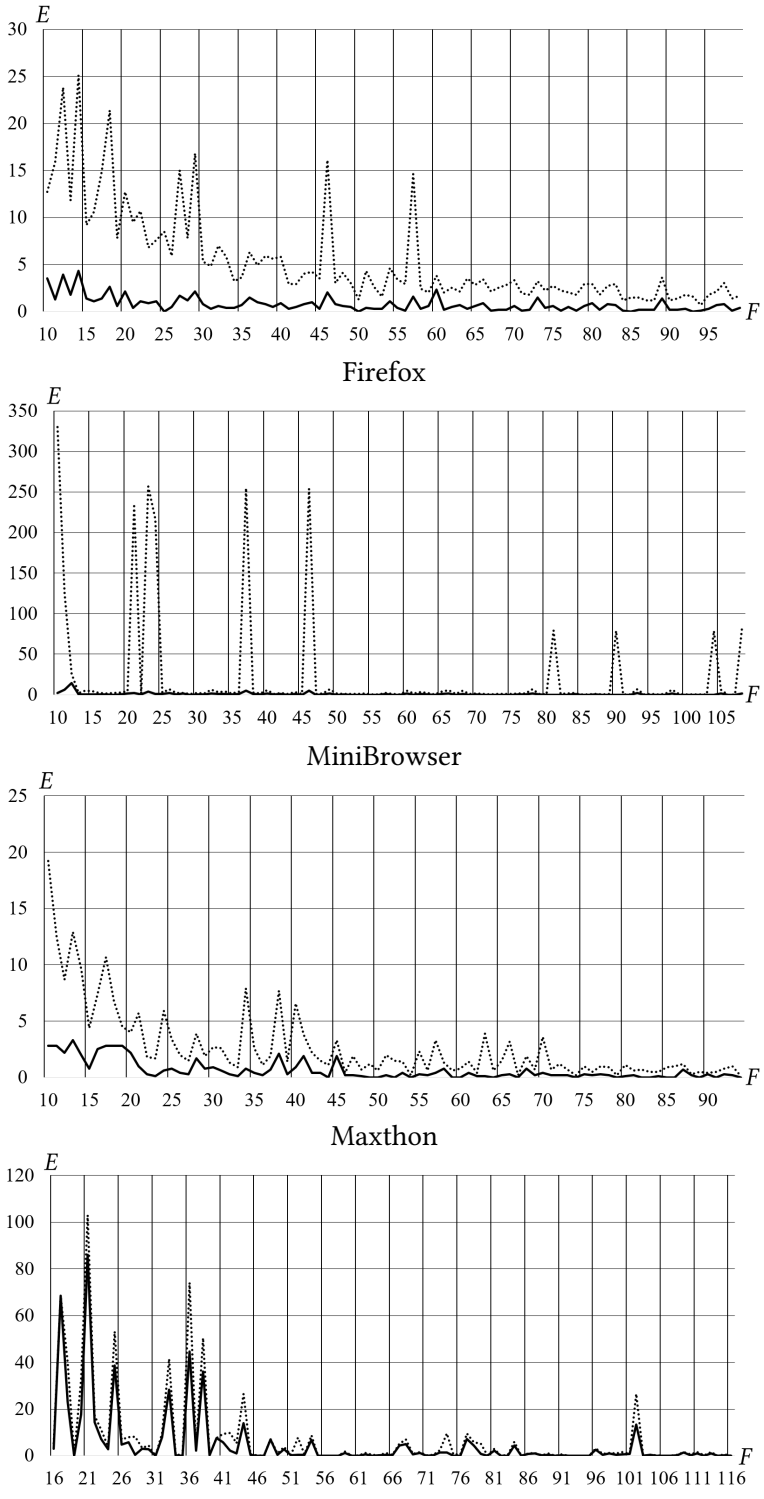


Fig. 2. Average number of atypical distances, where the dotted line shows the graph for the parameters $TTD = 1$ and $TMD = 1$, which corresponds to the basic algorithm implemented in [7], and the solid line shows the graph at $TTD = 20$ and $TMD = 10$

Рис. 2. Среднее число нетипичных расстояний, где пунктирной линией изображен график для $TTD = 1$ и $TMD = 1$, что соответствует алгоритму, реализованному в [7], а сплошной линией показан график при $TTD = 20$ и $TMD = 10$

использующих сигнатурный подход и подход на основе аномалий, такой способ использоваться может.

Заключение

Результаты исследования алгоритмов обнаружения нетипичного выполнения кода на основе построения профиля расстояний между системными функциями, полученные в [4, 5, 7], и результаты настоящей работе, свидетельствуют о возможном применении этого способа как дополнительном источнике информации о легитимном исполнении кода. Особенностью этого подхода является отсутствие необходимости доступа к исходному коду программы, так как широко известны разные способы получения информации о вызываемых в программе функциях. Длительность обучения естественно влияет на уменьшение вероятности ложного обнаружения нетипичного исполнения. Как продемонстрировано в настоящей работе, средства эмуляции действий пользователя позволяют автоматизировать построение такого профиля и сократить время его построения. Отметим, что в [5] приводятся положительные результаты выявления запуска реального эксплойта, а в [7] положительные результаты получены для смоделированного поведения эксплойта, когда вызов из «эксплойта» выполнялся по случайному адресу. Однако наибольший интерес представляет вопрос: может ли атакующий, зная систему защиты, адаптировать эксплойт так, чтобы его запуск остался незамеченным. Таким образом, дальнейшим направлением исследования является анализ возможных путей обхода предлагаемой защиты программы и пути совершенствования защиты.

References

[1] K. Lee, J. Lee, and K. Yim, “Classification and analysis of malicious code detection techniques based on the APT attack”, *Applied Sciences*, vol. 13, no. 5, p. 2894, 2023.

[2] A. Hofmeyr, S. Forrest, and A. Somayaji, “Intrusion detection using sequences of system calls”, *Journal of computer security*, vol. 6, no. 3, pp. 151–180, 1998.

[3] D. Wagner and P. Soto, “Mimicry attacks on host-based intrusion detection systems”, in *Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 255–264.

[4] Y. Kosolapov, “On one method for detecting exploitation of vulnerabilities and its parameters”, *Sistemy i Sredstva Informatiki [Systems and Means of Informatics]*, vol. 31, no. 4, pp. 48–60, 2021, in Russian.

[5] Y. Kosolapov, “On the detection of exploitation of vulnerabilities that leads to the execution of a malicious code”, *Automatic Control and Computer Sciences*, vol. 55, pp. 827–837, 2021.

[6] R. Batra. “Api monitor”. (2013), [Online]. Available: http://www.rohitab.com/apimonitor (visited on 04/21/2024).

[7] A. Kechahmadze and Y. Kosolapov, “Method for detecting exploits based on the profile of differences between function call addresses”, *Informatika i sistemy upravleniya*, vol. 73, no. 3, pp. 106–116, 2022, in Russian.

[8] “Exploit protection reference”. (2023), [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/exploit-protection-reference?view=o365-worldwide (visited on 04/21/2024).

[9] A. Sweigart. “Pyautogui documentation”. (2021), [Online]. Available: https://readthedocs.org/projects/pyautogui/downloads/pdf/latest/ (visited on 04/21/2024).

[10] Y. Ding, T. Wei, H. Xue, Y. Zhang, C. Zhang, and X. Han, “Accurate and efficient exploit capture and classification”, *Science China. Information Sciences*, vol. 60, 052110:1–052110:17, 2017.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 2, 2024
journal homepage: www.mais-journal.ru

COMPUTING METHODOLOGIES AND APPLICATIONS

Suppression of Additive Periodic Low-frequency Interference on Eddy Current Defectograms

L. Y. Bystrov[1], A. N. Gladkov[1], E. V. Kuzmin[1]

DOI: 10.18255/1818-1015-2024-2-164-181

[1]P.G. Demidov Yaroslavl State University, Yaroslavl, Russia

MSC2020: 62-07
Research article
Full text in Russian

Received April 29, 2024
Revised May 14, 2024
Accepted May 22, 2024

Traffic safety in railway transport requires regular inspection of the rail condition to identify and timely eliminate defects. Eddy current flaw detection is one of the popular methods of non-destructive rail testing. The data (defectograms) obtained from eddy current flaw detectors during rail testing are produced in large volume and therefore need in efficient automatic analysis. The analysis means the process of detecting on defectograms the presence of defective areas and identification of structural elements of the rail track, taking into account noise and possible interferences of various natures. The threshold noise level is found to isolate signals from defects and structural elements. Its value can be distorted by electromagnetic influences superimposed on the signals, which have pronounced low frequency and periodicity. This interferences raises the threshold noise level, complicating to detect useful signals. In this regard, there is a need to suppress the effects of the described type. Therefore, there is a need for described interferences reduction. In this paper spectral subtraction was used as a method for interference reduction on eddy current defectograms. The interference function was defined as the sum of the low-frequency harmonics of the Discrete Fourier Transform of the original signal. Then the interference-free signal can be found by subtracting of low-frequency range harmonics. The right boundary of this range was called the threshold harmonic frequency. It was found minimizing the distance of the signal's autocorrelation function and expected autocorrelation. For it two kinds were proposed: the noise autocorrelation and the reference autocorrelation. Both approaches allow to determine the threshold harmonic frequency so that periodic interferences will be best reduced. The based on the Gaussian noise autocorrelation method is somewhat universal for eddy current defectograms. Whereas the based on reference autocorrelation method depends on specific data and recording equipment. For the eddy current defectogram data under consideration, the most suitable threshold harmonic frequency was found. The described approaches to periodic low-frequency interference reduction, in addition to eddy current testing, can be successfully used for other areas.

Keywords: eddy current testing; rail surface defects; periodic interference; spectral subtraction; authocorrelation function

INFORMATION ABOUT THE AUTHORS

Bystrov, Leonid Y. (corresponding author)	ORCID iD: 0000-0002-0610-5466. E-mail: bystrovl0306@mail.ru Student
Gladkov, Artemy N.	ORCID iD: 0009-0007-0211-5660. E-mail: gladkov.art76@gmail.com Student
Kuzmin, Egor V.	ORCID iD: 0000-0003-0500-306X. E-mail: kuzmin@uniyar.ac.ru Head of the Chair of Theoretical Informatics, Dr. Sc.

Funding: Yaroslavl State University (project VIP-016).

For citation: L. Y. Bystrov, A. N. Gladkov, and E. V. Kuzmin, "Suppression of additive periodic low-frequency interference on eddy current defectograms", *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 164–181, 2024. DOI: 10.18255/1818-1015-2024-2-164-181.

© Bystrov L. Y., Gladkov A. N., Kuzmin E. V., 2024
This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 2, 2024
сайт журнала: www.mais-journal.ru

Подавление аддитивных периодических низкочастотных помех на вихретоковых дефектограммах

Л. Ю. Быстров[1], А. Н. Гладков[1], Е. В. Кузьмин[1]

DOI: 10.18255/1818-1015-2024-2-164-181

[1]Ярославский государственный университет им. П.Г. Демидова, Ярославль, Россия

УДК 519.254
Научная статья
Полный текст на русском языке

Получена 29 апреля 2024 г.
После доработки 14 мая 2024 г.
Принята к публикации 22 мая 2024 г.

Безопасность движения на железнодорожном транспорте требует регулярной проверки состояния рельсов для отслеживания и своевременного устранения возникающих на них дефектов. Вихретоковая дефектоскопия — один из популярных методов проведения неразрушающего контроля рельсов. Данные (дефектограммы), поступающие от вихретоковых дефектоскопов при тестировании рельсов, характеризуются большим объёмом и нуждаются в эффективном автоматическом анализе. Под анализом понимается процесс определения по дефектограммам наличия дефектных участков наряду с выявлением конструктивных элементов рельсового пути с учётом шума и возможных помех разной природы. Для выделения полезных сигналов (от дефектов и конструктивных элементов) находится пороговый уровень шума, значение которого может быть искажено накладывающимися на сигналы электромагнитными помехами, обладающими выраженной низкочастотностью и периодичностью. Указанные помехи завышают пороговый уровень шума, осложняя выявление полезных сигналов. В связи с этим возникает необходимость в подавлении помех описанного типа. В данной работе в качестве метода устранения помех на вихретоковых дефектограммах используется спектральное вычитание. Функция помех определяется как сумма низкочастотных гармоник дискретного преобразования Фурье исходных сигналов. Очищенные от помех сигналы получаются вычитанием гармоник низкочастотного диапазона. Правая граница этого диапазона названа частотой пороговой гармоники. Она находится с помощью минимизации расстояния между автокорреляционной функцией сигналов и ожидаемой автокорреляцией. Предложены два вида ожидаемой автокорреляции: автокорреляция гауссовского шума и эталонная автокорреляция. Оба подхода позволяют определить частоту пороговой гармоники, при которой периодические помехи будут подавляться наилучшим образом. Метод, основанный на автокорреляции гауссовского шума, является в некотором роде универсальным для вихретоковых дефектограмм. Эталонная автокорреляция привязана к конкретным данным и пишущему оборудованию. Для рассматриваемых данных вихретоковых дефектограмм найдена наиболее подходящая частота пороговой гармоники. Описанные подходы к подавлению периодических низкочастотных помех помимо вихретоковой дефектоскопии могут успешно применяться и в других областях.

Ключевые слова: вихретоковая дефектоскопия; поверхностные дефекты рельсов; периодические помехи; спектральное вычитание; автокорреляционная функция

ИНФОРМАЦИЯ ОБ АВТОРАХ

Быстров, Леонид Юрьевич (автор для корреспонденции)	ORCID iD: 0000-0002-0610-5466. E-mail: bystrovl0306@mail.ru Студент
Гладков, Артемий Николаевич	ORCID iD: 0009-0007-0211-5660. E-mail: gladkov.art76@gmail.com Студент
Кузьмин, Егор Владимирович	ORCID iD: 0000-0003-0500-306X. E-mail: kuzmin@uniyar.ac.ru Заведующий кафедрой теоретической информатики, доктор физ.-мат. наук

Финансирование: ЯрГУ (проект VIP-016).

Для цитирования: L. Y. Bystrov, A. N. Gladkov, and E. V. Kuzmin, “Suppression of additive periodic low-frequency interference on eddy current defectograms”, *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 164–181, 2024. DOI: 10.18255/1818-1015-2024-2-164-181.

© Быстров Л. Ю., Гладков А. Н., Кузьмин Е. В., 2024
Эта статья открытого доступа под лицензией CC BY license (https://creativecommons.org/licenses/by/4.0/).

Введение

Безопасность движения на железнодорожном транспорте требует своевременного обнаружения рельсовых дефектов. Для этого регулярно проводится неразрушающий контроль рельсов. Одним из широко применяемых методов для обнаружения поверхностных и подповерхностных дефектов в объектах железнодорожного транспорта является вихретоковая дефектоскопия [1].

Данные (дефектограммы), собранные дефектоскопом, нуждаются в анализе, то есть в выявлении среди них участков, указывающих на дефекты рельсов, на фоне всевозможных помех и сигналов от конструктивных элементов [2].

В этой работе рассматривается 12-разрядный вихретоковый дефектоскоп с 15 каналами данных для каждого рельса. Каналы данных соответствуют физическим датчикам, которые последовательно располагаются на поверхности рельса перпендикулярно направлению движения дефектоскопа. Дефектоскоп формирует сигналы (поканально) для каждого миллиметра железнодорожного пути. Значения сигналов в каждом канале являются целыми числами от -2048 до 2047 (рис. 1). Ввиду большого объёма данных дефектограммы разбиваются на фрагменты по 50 метров, каждому каналу одного такого фрагмента соответствует массив из 50000 элементов $x(t), t \in \overline{0, 49999}$.

Полезными являются сигналы от дефектов и конструктивных элементов (различные сварки и стыки рельсов). Обычно они составляют менее 10 % данных дефектограммы. Большинство же сигналов представлено рельсовым шумом. На практике могут встречаться участки рельсов с протяжёнными поверхностными дефектами, но в данной работе они не рассматриваются, так как представляют отдельный объект исследования.

Сигнал считается полезным, если отклонение его значения (амплитуда) от среднего значения всех сигналов (в рамках одного канала) как минимум в два раза превосходит среднее отклонение значений (амплитуду) шума на данном участке. Значение амплитуды, разграничивающее шум и полезные сигналы на данном участке, называется *пороговым уровнем шума* (рис. 2). Он может быть найден с помощью статистических методов [3]. Пороговый уровень шума не одинаков на разных участках дефектограммы, поэтому его необходимо вычислять отдельно для каждого участка.

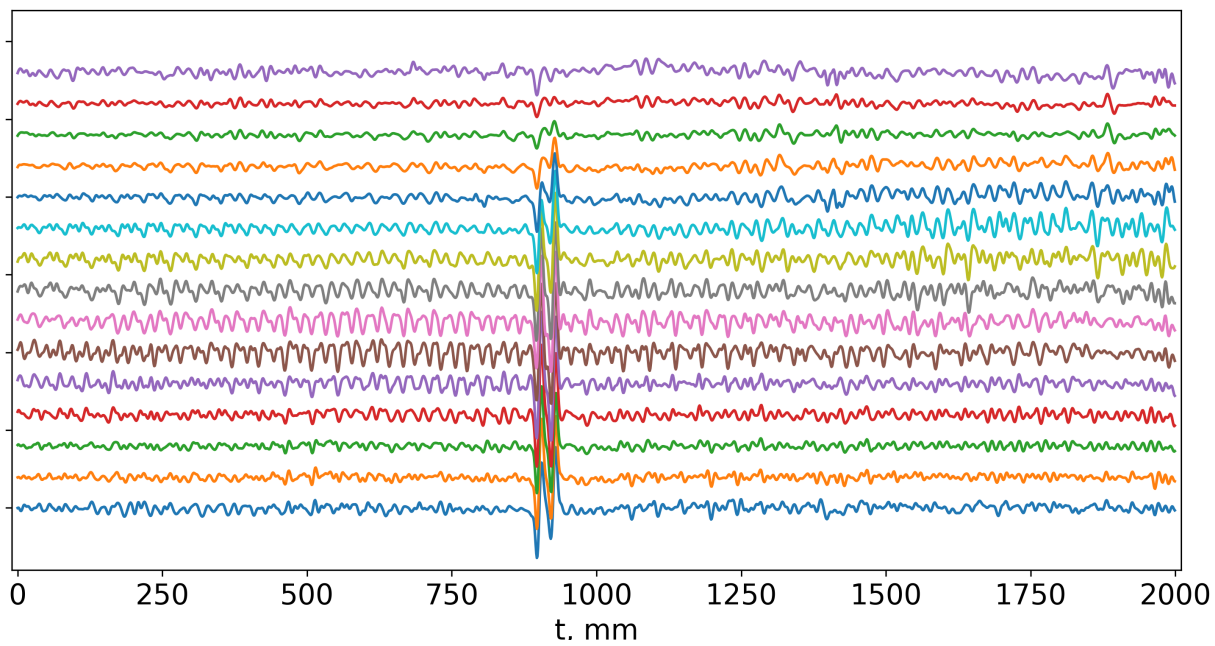


Fig. 1. The example of a flaw detector record of a welded rail joint

Рис. 1. Пример записи дефектоскопом сварного стыка рельсов

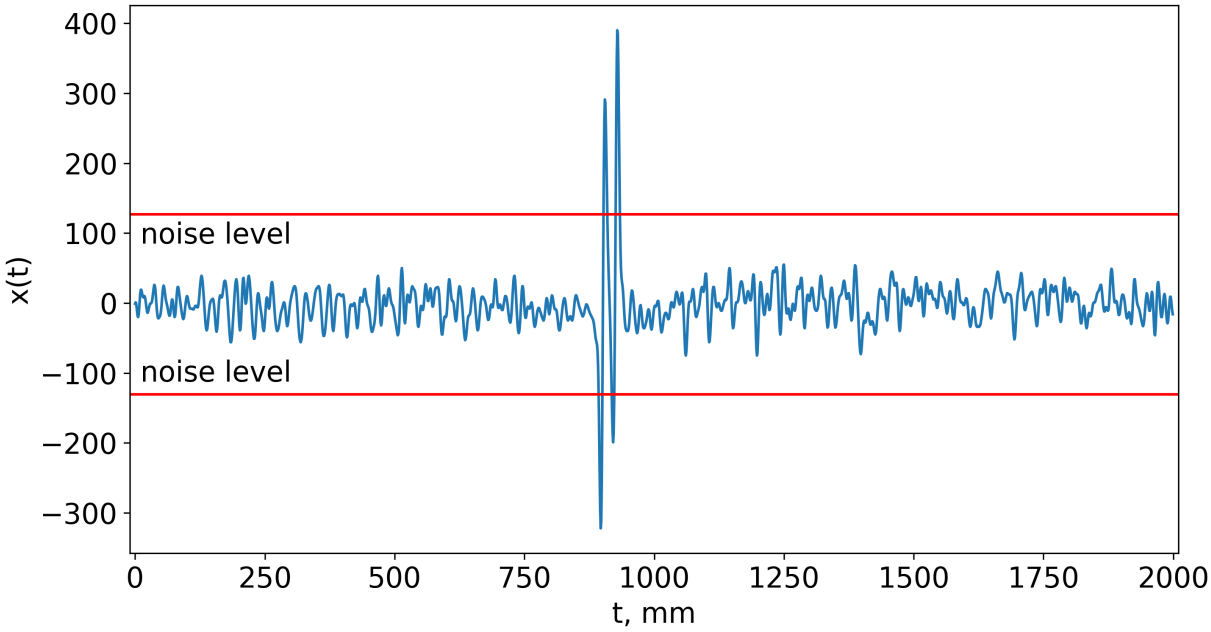


Fig. 2. The threshold noise level for one defectogram channel

Рис. 2. Пороговый уровень шума для одного канала дефектограммы

Определение порогового уровня шума рельсов может быть осложнено присутствием периодических помех, искажающих реальные значения амплитуд.

В данной статье рассматриваются помехи, вызванные электромагнитными наводками от оборудования, сопровождающего дефектоскоп, и внешних источников электромагнитного излучения. Всё это внешнее по отношению к рельсу и к системе дефектоскопа воздействие, поэтому можно говорить об аддитивной природе рассматриваемых помех.

Указанные помехи на дефектограммах характеризуются периодичностью и низкочастотностью. Тогда как не затронутый помехами сигнал дефектоскопа представляет собой непериодическую функцию, заданную преимущественно в высокочастотном диапазоне. Помехи искажают значения исходного сигнала, внося в него вредную чужеродную периодическую и низкочастотную составляющую (рис. 3).

Наличие этих помех не позволяет достоверно установить пороговый уровень шума (например, с помощью алгоритма из статьи [3]) и, соответственно, обнаружить полезные сигналы (рис. 4). Поэтому возникает задача, состоящая в предварительном «выпрямлении» данных — подавлении возникающих в сигнале помех с сохранением формы конструктивных элементов и восстановлением реального порогового уровня шума. Настоящая статья посвящена решению этой задачи.

Для подавления помех предлагается использовать метод спектрального вычитания, широко применяющийся в цифровой обработке сигналов и многократно доказавший свою эффективность [4, 5]. Данный подход является универсальным и эффективно использует вычислительные ресурсы [6]. Спектральное вычитание способно исключить из сигнала низкочастотную составляющую, порождённую помехами. Границы низкочастотного диапазона определяются с помощью функции автокорреляции, которая позволяет оценить периодичность сигнала. Таким образом, использование метода спектрального вычитания даёт возможность отслеживать и контролировать характерные черты помех — низкочастотность и периодичность — и через подавление этих характеристик исключать из сигнала сами помехи.

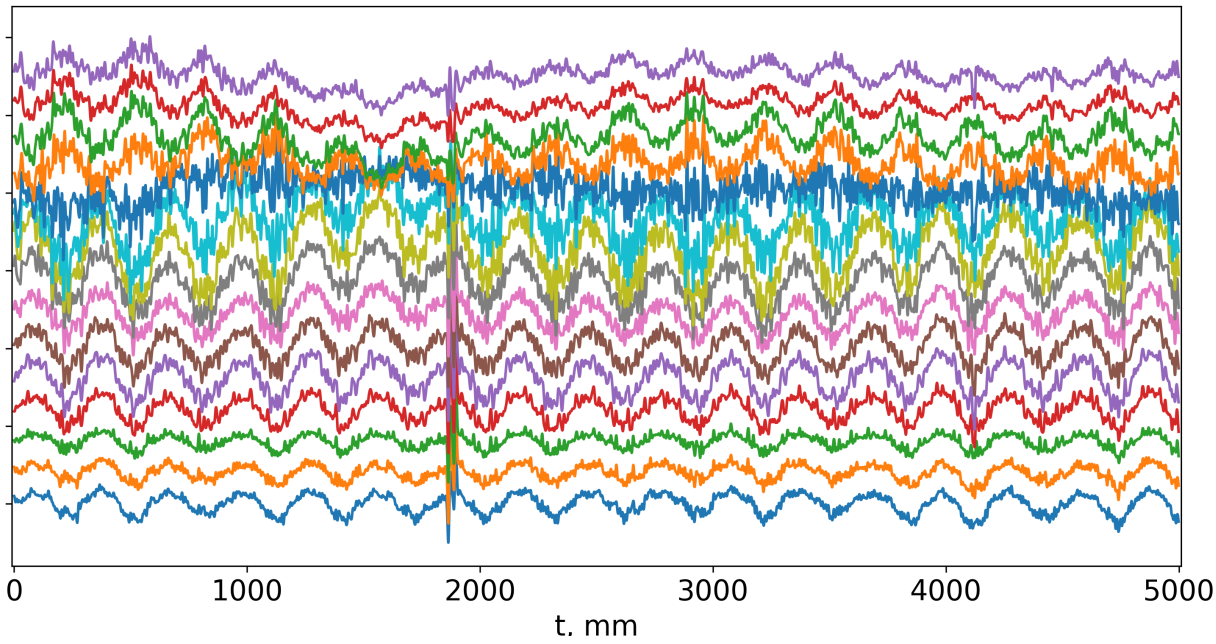


Fig. 3. The example of periodic interferences on the defectogram

Рис. 3. Пример периодических помех на дефектограмме

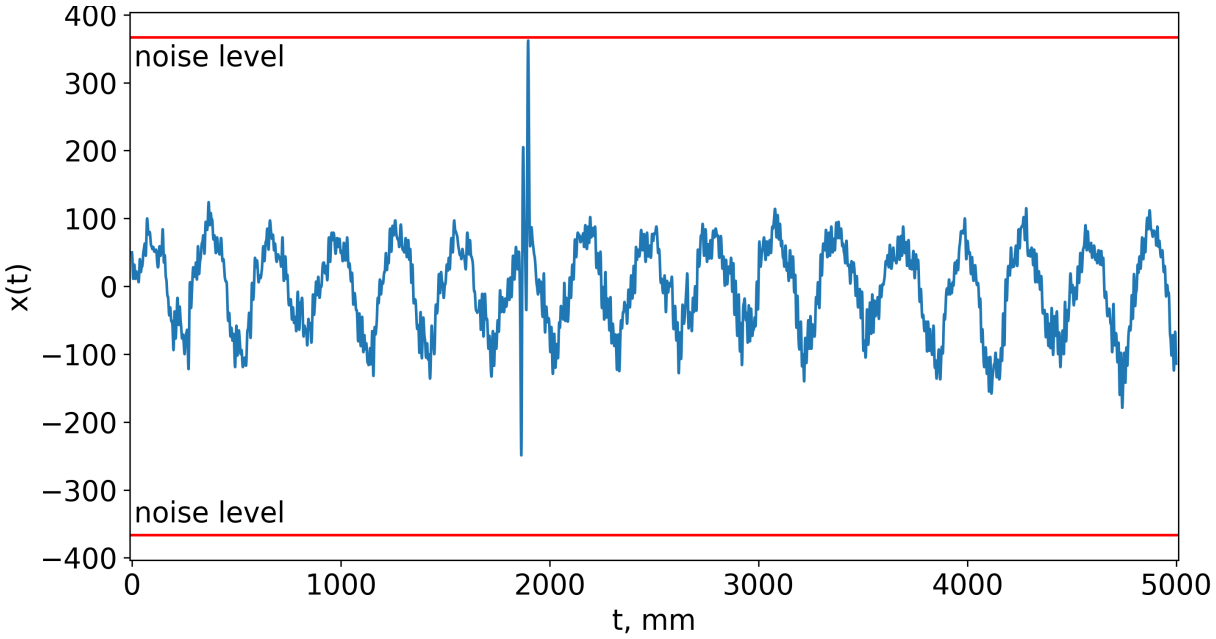


Fig. 4. Interference Overstatement of the Threshold Noise Level.

Рис. 4. Завышение помехами порогового уровня шума.

1. Основные понятия

Пусть $x(t)$, $t = 0, 1, \ldots, T-1$, — последовательность конечных действительных или комплексных чисел, тогда *дискретное преобразование Фурье (ДПФ)* этой последовательности определяется как [7]

$$X(\omega) = \frac{1}{T}\sum_{j=0}^{T-1} x(t)W^{\omega j}, \text{ где } W = e^{-i\frac{2\pi}{T}}, \ \omega \in \overline{0, T-1}, \ i^2 = -1. \tag{1}$$

ДПФ позволяет перевести сигнал $x(t)$ из временной области в частотную. Значение ω обозначает ω-ую гармонику ряда Фурье, а $X(\omega)$ — комплексный вес ω-ой гармоники в разложении сигнала $x(t)$.

Функция $|X(\omega)|$ является комплексным модулем числа $X(\omega)$ и называется *амплитудным спектром* сигнала $x(t)$. Величина $|X(\omega)|^2$ — *спектр мощности* сигнала $x(t)$. *Фазовым же спектром* сигнала называется функция $\Theta(\omega)$, которая определяется как комплексный аргумент $X(\omega)$

$$\Theta(\omega) = \arctan\frac{Im(X(\omega))}{Re(X(\omega))},$$

где $Im(k)$ и $Re(k)$ — мнимая и действительная части числа k соответственно.

Обратное дискретное преобразование Фурье (ОДПФ) определяется следующим образом:

$$x(t) = \sum_{\omega=0}^{T-1} X(\omega)W^{-\omega t}, t \in \overline{0, T-1}.$$

Автокорреляционной функцией $r_x(\tau)$ дискретного сигнала $x(t)$ называется последовательность

$$r_x(\tau) = \frac{1}{T-1}\sum_{t=0}^{T-1} x(t)x(t+\tau),$$

где $\tau \in \overline{0, T-1}$. Переменная τ называется *смещением отсчётов*.

Функция $r_x(\tau)$ позволяет делать выводы о периодичности сигнала. Известно, что автокорреляционная функция периодического сигнала сама является периодической, причём с тем же самым периодом [8].

2. Спектральное вычитание помех

Обозначим через $x(t)$ сигнал, сформированный дефектоскопом на 50-метровом участке рельса для одного канала ($t = 0, 1, \ldots, 49999$). Очищенный от помех сигнал $x(t)$ обозначим через $s(t)$, функцию помех — через $p(t)$.

В виду того, что помехи «накладываются» на сигнал, то есть обладают аддитивной природой, можно считать, что сигнал $x(t)$ раскладывается в сумму чистого сигнала $s(t)$ и функции помех $p(t)$

$$x(t) = s(t) + p(t). \tag{2}$$

Метод спектрального вычитания [9] основан на предположении о том, что спектр мощности $|X(\omega)|^2$ сигнала $x(t)$ связан со спектрами мощности $|S(\omega)|^2$ и $|P(\omega)|^2$ сигналов $s(t)$ и $p(t)$ следующим соотношением:

$$|X(\omega)|^2 = |S(\omega)|^2 + |P(\omega)|^2. \tag{3}$$

Отсюда можно выразить спектр $|S(\omega)|$ и выполнить для него обратное преобразование:

$$\hat{s}(t) = \text{ОДПФ}\left(|S(\omega)| \cdot e^{i\Theta(\omega)}\right) = \text{ОДПФ}\left(\sqrt{|X(\omega)|^2 - |P(\omega)|^2} \cdot e^{i\Theta(\omega)}\right), \ \Theta(\omega) - \text{фазовый спектр } x(t). \tag{4}$$

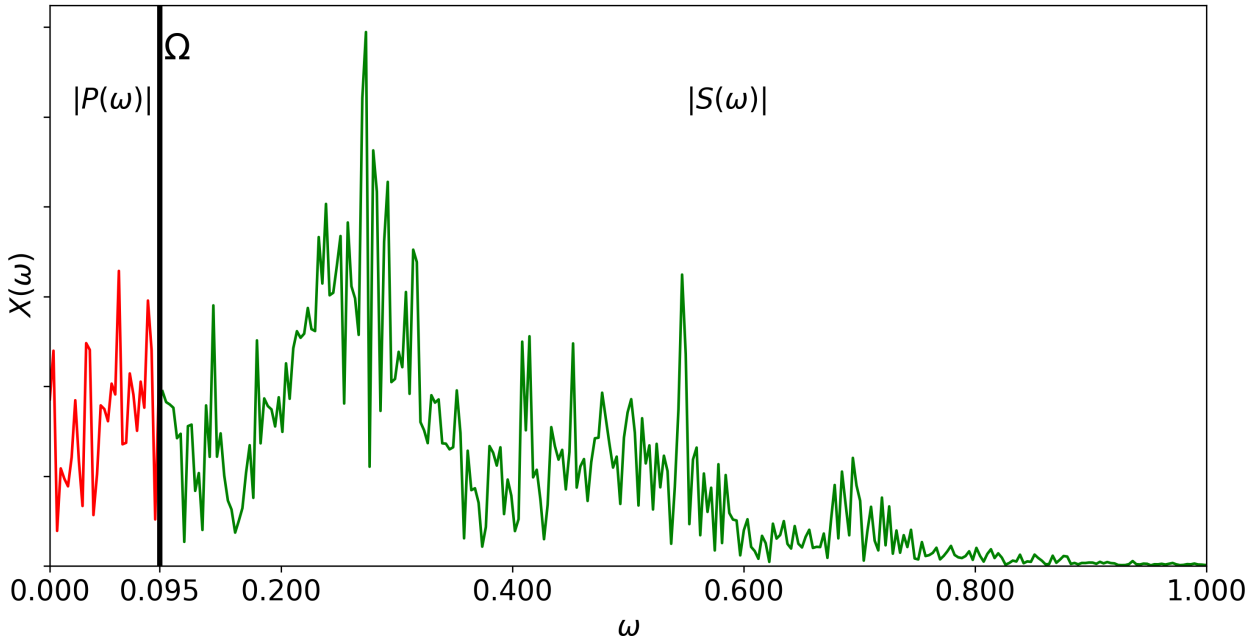


Fig. 5. The separation of the spectrum $|X(\omega)|$ into the interfence spectrum $|P(\omega)|$ and the pure signal spectrum $|S(\omega)|$. Ω — the harmonic separating $|P(\omega)|$ from $|S(\omega)|$.

Рис. 5. Разделение спектра $|X(\omega)|$ на спектр помех $|P(\omega)|$ и спектр чистого сигнала $|S(\omega)|$. Ω — гармоника, отделяющая $|P(\omega)|$ от $|S(\omega)|$.

Таким образом, если удастся аппроксимировать амплитудный спектр $P(\omega)$ функции помех $p(t)$, то с помощью формулы (4) можно получить очищенный от помех сигнал $s(t)$.

Помехи на вихретоковых дефектограммах являются низкочастотными, что обуславливает выраженный характер их влияния на форму сигнала. Тогда как шум и полезные сигналы представлены преимущественно в высокочастотном диапазоне. Поэтому можно считать, что в спектре $|X(\omega)|$ гармоники из низкочастотного диапазона определяют помехи, а все прочие гармоники определяют чистый сигнал (рис. 5).

В таком случае спектр $X(\omega)$ разделяется на две непересекающиеся части

$$|X(\omega)| = \begin{cases} |P(\omega)|, \omega \leqslant \Omega, \\ |S(\omega)|, \omega > \Omega. \end{cases} \tag{5}$$

Число Ω разграничивает спектр помех $|P(\omega)|$ и спектр чистого сигнала $|S(\omega)|$. Будем называть Ω *пороговой гармоникой.*

Формула (5) не нарушает равенства (3):

$$|X(\omega)|^2 = \begin{cases} |P(\omega)|^2 + 0^2, \omega \leqslant \Omega, \\ |S(\omega)|^2 + 0^2, \omega > \Omega. \end{cases}$$

Таким образом, разделение спектра $X(\omega)$ на две непересекающиеся части позволяет упростить формулу спектрального вычитания

$$\hat{s}(t) = \text{ОДПФ}\left(|S(\omega)| \cdot e^{i\Theta(\omega)}\right) = \text{ОДПФ}\left(|X(\omega)| \cdot e^{i\Theta(\omega)}\right), \omega > \Omega = \text{ОДПФ}\left(X(\omega)\right), \omega > \Omega. \tag{6}$$

Теперь возникает задача нахождения пороговой гармоники Ω. Найдём частоту $\hat{\nu}$ данной гармоники, используя формулу (1):

$$\hat{\nu} = \frac{2\pi}{T} \cdot \Omega. \tag{7}$$

Отсюда выразим Ω и подставим её в формулу (6). Получим

$$\hat{s}(t) = \text{ОДПФ } (X(\omega)), \omega > \frac{\hat{\nu}T}{2\pi}. \tag{8}$$

Практический опыт работы с данными показал, что гармоники, определяющие форму полезных сигналов, имеют частоту $\nu \geqslant 0,3$. Частоты гармоник полезных сигналов лежат в спектре $|S(\omega)|$, который не пересекается со спектром помех $|P(\omega)|$. Следовательно, частота $\hat{\nu}$ пороговой гармоники Ω должна быть меньше числа $0,3$: $\hat{\nu} \in (0;0,3)$. Для аппроксимации числа $\hat{\nu}$ предлагаются два автокорреляционных метода. Один из них основан на автокорреляции идеального гауссовского шума, другой — на эталонной автокорреляции.

3. Нахождение частоты пороговой гармоники

Главными характеристиками помех на вихретоковых дефектограммах являются низкочастотность и периодичность. Первое свойство было использовано для разделения спектра $|X(\omega)|$ на две части, но граница этого разделения найдена не была. Для её нахождения воспользуемся вторым свойством помех — периодичностью.

Сведения о периодичности сигнала $x(t)$ даёт функция автокорреляции $r_x(\tau)$. Для каждого смещения отсчётов τ она показывает, насколько исходный сигнал $x(t)$ коррелируем со своим смещением $x(t+\tau)$. Для смещения τ, совпадающего с периодом функции, автокорреляция периодических функций близка к 1. Кроме того, функция автокорреляции периодических функций сама периодична. Что же касается непериодических функций, то их автокорреляционная функция не периодична. И она уже на малых смещениях τ становится близка к 0, с увеличением смещения не покидая окрестности 0.

В связи с тем, что полезные сигналы на вихретоковых дефектограммах не периодичны, можно считать, что их значения для разных отсчётов слабо коррелируемы и автокорреляция полезных сигналов практически не отличима от автокорреляции чистого шума. Известно, что на бесконечном объёме данных рельсовый шум распределяется по нормальному закону [3], то есть в идеальном случае представляет собой гауссовский шум. Таким образом, периодичность сигнала $x(t)$ можно оценивать через «близость» расстояние между его функцией автокорреляции $r_x(\tau)$ и функцией автокорреляции гауссовского шума $r_{noise}(\tau)$.

Известно, что для каждого смещения τ значение автокорреляции гауссовского шума $r_{noise}(\tau)$ распределено по нормальному закону

$$r_{noise}(\tau) \sim N\left(-\frac{T-\tau}{T(T-1)}, \frac{(T-\tau)(T-2)(T^2+T-2\tau) - 2T(T-1)(T-2\tau)^+}{T^2(T-1)^2(T+1)}\right), \tag{9}$$

где $(a)^+ = \max\{a, 0\}$, $N(m, \sigma^2)$ — нормальный закон распределения с математическим ожиданием m и дисперсией σ^2, T — количество отсчётов сигнала [10].

Случайное распределение значений автокорреляции гауссовского шума не позволяет получить аналитической детерминированной записи функции $r_{noise}(\tau)$. Будем считать, что величина $r_{noise}(\tau)$ приблизительно описывается функцией

$$\hat{r}_{noise}(\tau) = M(r_{noise}(\tau)) \pm \sigma(r_{noise}(\tau)), \tag{10}$$

где $M(X)$ — математическое ожидание случайной величины X, $\sigma(X)$ — её среднеквадратическое отклонение.

Функция $\hat{r}_{noise}(\tau)$ моделирует кривую среднего отклонения случайной величины $r_{noise}(\tau)$ от её математического ожидания. Она описывает «среднюю» форму автокорреляции гауссовского шума.

Подставим значения математического ожидания и среднеквадратического отклонения из формулы (9) в выражение (10):

$$\hat{r}_{noise}(\tau) = -\frac{T-\tau}{T(T-1)} \pm \sqrt{\frac{(T-\tau)(T-2)(T^2+T-2\tau) - 2T(T-1)(T-2\tau)^+}{T^2(T-1)^2(T+1)}}. \tag{11}$$

Расстояние между функциями будем измерять с помощью среднеквадратического расстояния d:

$$d(f(t), g(t))) = \frac{1}{T}\sum_{t=0}^{T-1}(f(t)-g(t))^2, \text{где } T\text{— количество отсчётов.} \tag{12}$$

Ввиду того, что функция $\hat{r}_{noise}(\tau)$ разделяется на две практически симметричные дуги (при больших T: $M(r_{noise}(\tau)) = 0$), формулы (12) для оценки близости $r_x(\tau)$ к $\hat{r}_{noise}(\tau)$ недостаточно, требуются дополнительные соображения. Например, можно использовать абсолютные величины. Будем рассматривать только положительные значения $\hat{r}_{noise}(\tau)$, то есть $|\hat{r}_{noise}(\tau)| = M(r_{noise}(\tau)) + \sigma(r_{noise}(\tau))$. Расстояние между $r_x(\tau)$ и $\hat{r}_{noise}(\tau)$ можно искать по формуле

$$d(r_x(\tau), \hat{r}_{noise}(\tau))) = \frac{1}{T}\sum_{t=0}^{T-1}(|r_x(\tau)| - |\hat{r}_{noise}(\tau)|)^2. \tag{13}$$

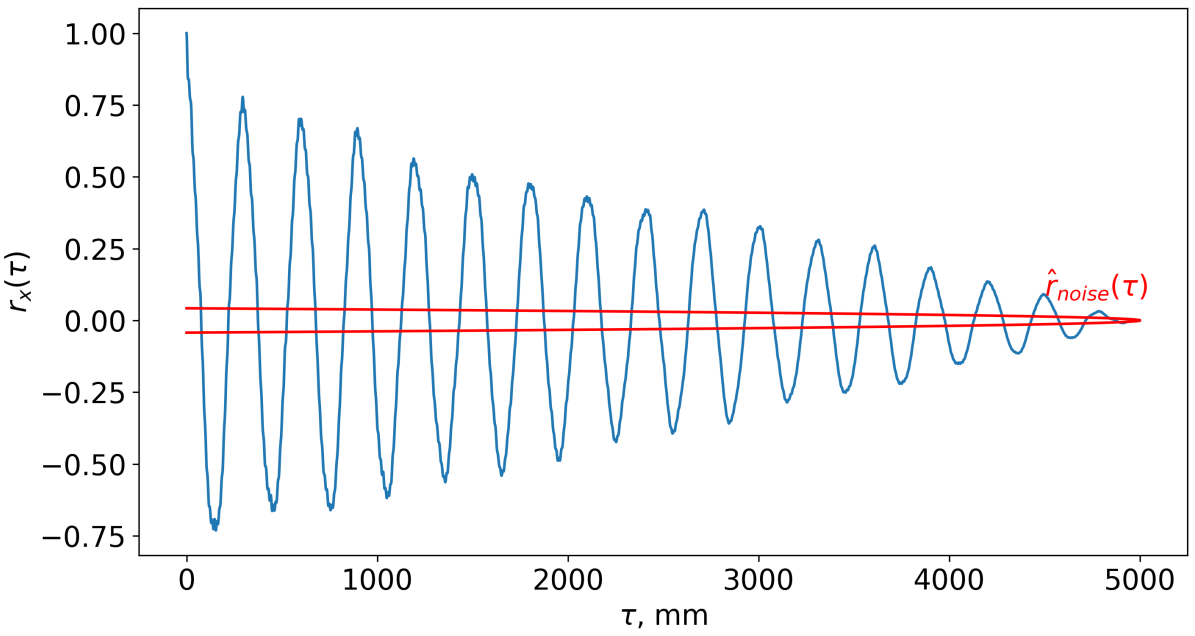
Периодичность сигнала $x(t)$ можно оценивать через расстояние между функциями $r_x(\tau)$ и $\hat{r}_{noise}(\tau)$ по формуле (13). Чем меньше значение $d(r_x(\tau), \hat{r}_{noise}(\tau))$, тем менее периодичен сигнал $x(t)$ и, соответственно, меньше искажён помехами (рис. 6).

Применим формулу (8), подставляя в неё разные значения частоты $\hat{\nu}$ пороговой гармоники Ω из диапазона $\nu \in (0; 0,3)$. В результате сигнал $x(t)$ будет преобразован в $\hat{s}(t)$. Ту частоту $\hat{\nu}$, для которой расстояние $d(r_{\hat{s}}(\tau), \hat{r}_{noise}(\tau))$ будет минимально, примем за искомую частоту пороговой гармоники Ω. Ниже приведена реализация описанного алгоритма на языке Python.

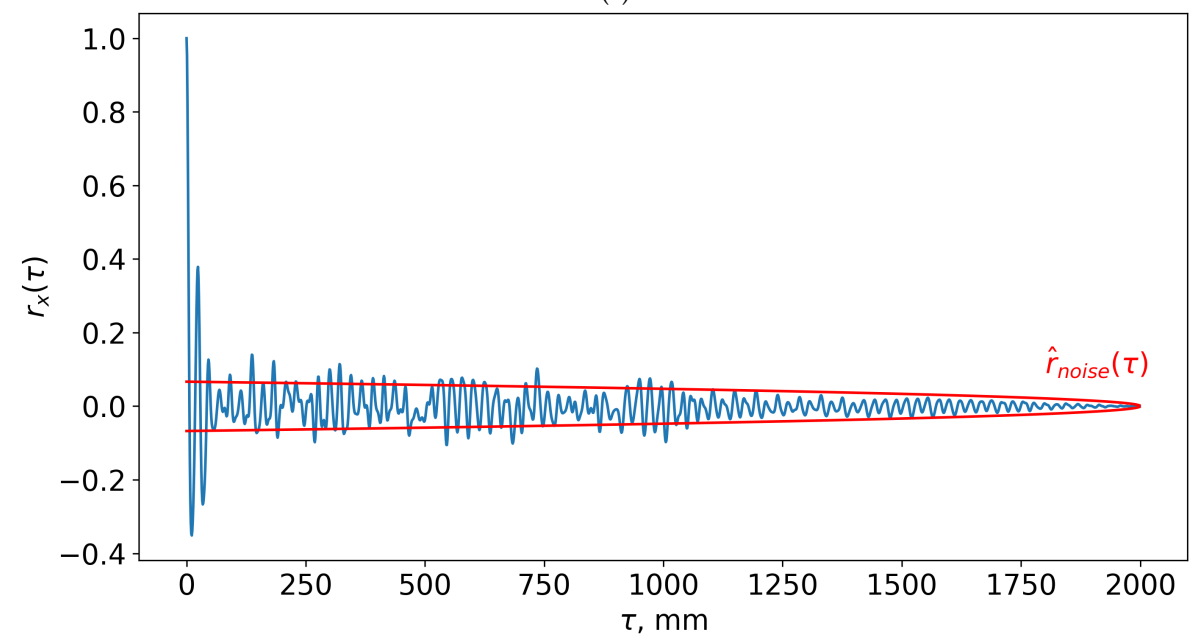
```
import sys
import math
#  Библиотека, предоставляющая удобные инструменты для работы с массивами данных
import numpy as np
#  Библиотека, предоставляющая инструменты для обработки сигналов
from scipy.fft import irfft, rfft
#  Библиотека, предоставляющая инструменты для анализа временных рядов
from statsmodels.tsa.stattools import acf

def get_harmonic(frequency, T):
   #  frequency - частота, T - число гармоник в преобразовании Фурье.
   #  Нахождение гармоники, соответствующей данной частоте frequency
   return math.ceil(frequency / 2 / math.pi * T)

def metric_noise(signal_acf, r_noise):
   #  Вычисление среднеквадратического расстояния
   #  между абсолютным значением функциии signal_acf и функцией r_noise
   T = len(signal_acf)
   sum = 0
   for i in range(0, T):
      sum += (abs(signal_acf[i]) - r_noise[i]) ** 2
   return (sum / T)
```



(a)



(b)

Fig. 6. The signal autocorrelation function $r_x(\tau)$ and the estimating Gaussian noise autocorrelation function $\hat{r}_{noise}(\tau)$. **(a)** The defectogram section with interferences. **(b)** The defectogram section without interferences.

Рис. 6. Функция автокорреляции сигнала $r_x(\tau)$ и функция, оценивающая автокорреляцию гауссовского шума $\hat{r}_{noise}(\tau)$. **(a)** Участок дефектограммы с помехами. **(b)** Участок дефектограммы без помех.

```
def get_r_noise(T):
   def average(t):
      #  Матожидание автокорреляции гауссовского шума
      return - (T - t) / (T * (T - 1))
   def sigma(t):
      #  Среднеквадратичное отклонение автокорреляции гауссовского шума
      return math.sqrt(
            (
               ((T - t) * (T - 2) * ((T ** 2) + T - 2 * t))
               - 2 * T * (T - 1) * max(T - 2 * t, 0)
            )
               / ((T ** 2) * ((T - 1) ** 2) * (T + 1))
         )
   at = np.vectorize(average)
   st = np.vectorize(sigma)
   t = np.arange(T, dtype = np.double)
   mean = at(t)
   sigma = st(t)
   r_noise = mean + sigma  # Автокорреляция шума
   return r_noise

def optmize_noise(x, frm = 0.0, to = 0.35, step = 0.005):
   #  Поиск threshold частоты пороговой гармоники
   #  frm, to - задают границы отрезка перебора
   #  step - задаёт шаг перебора
   T = len(x)
   r_noise = get_r_noise(T)
   frequency, threshold, min_metric = frm, T, sys.float_info.max
   X = rfft(x)  #  Прямое преобразование Фурье
   while frequency <= to:
      X[0: get_harmonic(frequency, T)] = 0 #  Спектральное вычитание
      x1 = irfft(X)                        #  Обратное преобразование Фурье
      ac = acf(x1, T - 1)                  #  Построение автокорреляции нового сигнала
      metric = metric_noise(ac, r_noise)   #  Вычисление расстояния между функциями
      if metric < min_metric:              #  Минимизация расстояния
         threshold = frequency
         min_metric = metric
         frequency += step
   return threshold
```

Таким образом, спектр помех $|P(\omega)|$ будет определён в таком низкочастотном диапазоне, который соответствует как можно большему количеству выраженных периодических составляющих сигнала $x(t)$. Это позволяет связать низкочастотность помех с их периодичностью.

В рамках данной работы был проведён эксперимент, в ходе которого перебирались значения частот ν в отрезке $[0; 0,35]$ с шагом $0,005$ для одного канала на 8040 пятидесятиметровых участках дефектограммы. Для каждого участка было выбрано наиболее подходящее значение частоты $\hat{\nu}$ пороговой гармоники Ω, соответствующее минимальному расстоянию $d(r_{\hat{s}}(\tau), \hat{r}_{noise}(\tau))$. Полученные результаты отображены на графике 7.

Как можно видеть, для разных участков дефектограммы получаются разные значения $\hat{\nu}$. Но тем не менее, отсюда ещё не следует, что наше предположение о разграничении спектров $|P(\omega)|$ и $|S(\omega)|$ не верно.

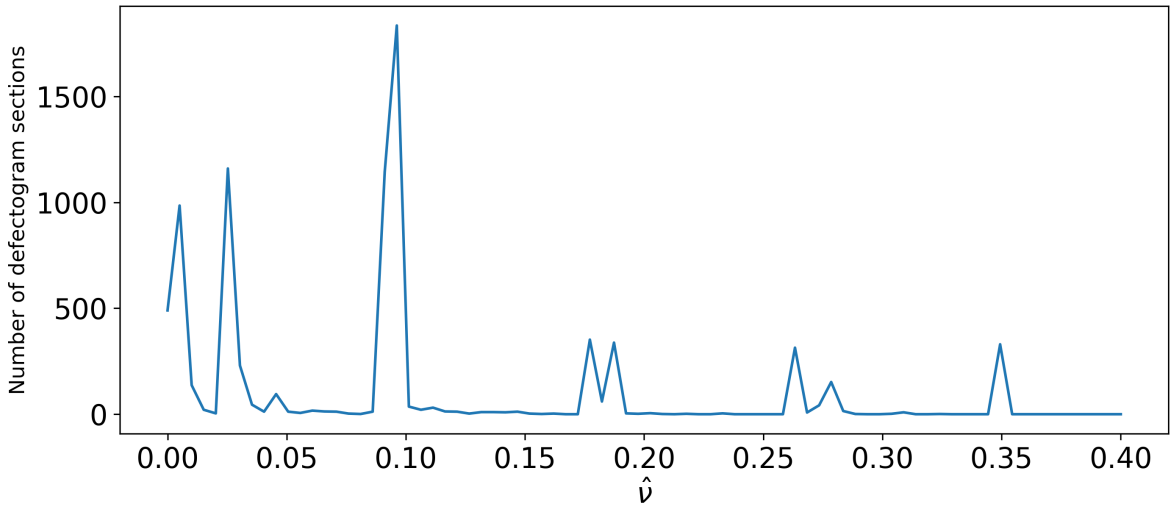


Fig. 7. The values of the threshold harmonic frequency for the method based on Gaussian noise autocorrelation

Рис. 7. Значения частоты пороговой гармоники для метода, основанного на автокорреляции гауссовского шума

Частоты, сосредоточенные на графике 7 в окрестности нуля, соответствуют участкам дефектограммы, не затронутым или мало затронутым помехами. Можно считать, что в таком случае спектр $|S(\omega)|$ чистого от помех сигнала $s(t)$ полностью представлен спектром $|X(\omega)|$ исходного сигнала $x(t)$, и, действительно, никакой пороговой гармоники Ω искать не требуется.

Для тех же участков, где помехи заметны и оказывают существенное влияние на сигнал, значение $\hat{\nu}$ оказывается равным $0,095$ и $0,09$ (первое значение встречается чаще). На графике 7 эти значения представлены наибольшим «пиком». Полученные числа лучше всего подходят на роль искомого значения $\hat{\nu}$. Следует отметить, что на других данных, при работе с другим оборудованием, может получиться иное значение $\hat{\nu}$. Подходящее значение будет представлено самым высоким «пиком» на графике частот $\hat{\nu}$.

Частоты $0,175$, $0,185$, $0,26$ и $0,345$ соответствуют около $10\,\%$ рассмотренных участков дефектограммы. Они определяют высокочастотные помехи, которые практически не влияют на форму сигнала и не изменяют пороговый уровень шума (рис. 8).

Таким образом, эксперимент показал, что, строго говоря, пороговой гармоники, отделяющей спектр помех $|P(\omega)|$ от чистого спектра $|S(\omega)|$, не существует. Но для помех, обладающих выраженной низкочастотностью и периодичностью, которые оказывают на сигнал негативное воздействие, искажая форму полезных сигналов и завышая пороговый уровень шума, можно указать такое значение. Для рассмотренных вихретоковых дефектограмм оно приблизительно равно $0,095$.

4. Эталонная автокорреляция

В классическом методе спектрального вычитания в качестве вычитаемого амплитудного спектра берётся спектр некоторого «эталонного» участка сигнала, где искажения представлены наиболее отчётливо. Можно использовать данную идею применительно не к спектру, а к автокорреляции.

Возьмём чистый участок дефектограммы, не содержащий помех (рис. 10). Примем его функцию автокорреляции $r_{ref}(\tau)$ за «эталон» того, как должна выглядеть функция автокорреляции сигнала $s(t)$, очищенного от помех. Сигнал $s(t)$ будем оценивать с помощью $\hat{s}(t)$ по формуле (8). Расстояние $d(r_{\hat{s}}(\tau), r_{ref}(\tau)$ будем считать по формуле (12). Минимизируем расстояние и примем за искомую частоту $\hat{\nu}$ значение, соответствующее минимальному $d(r_{\hat{s}}(\tau), r_{ref}(\tau)$. Ниже приведена реализация описанного алгоритма на языке Python.

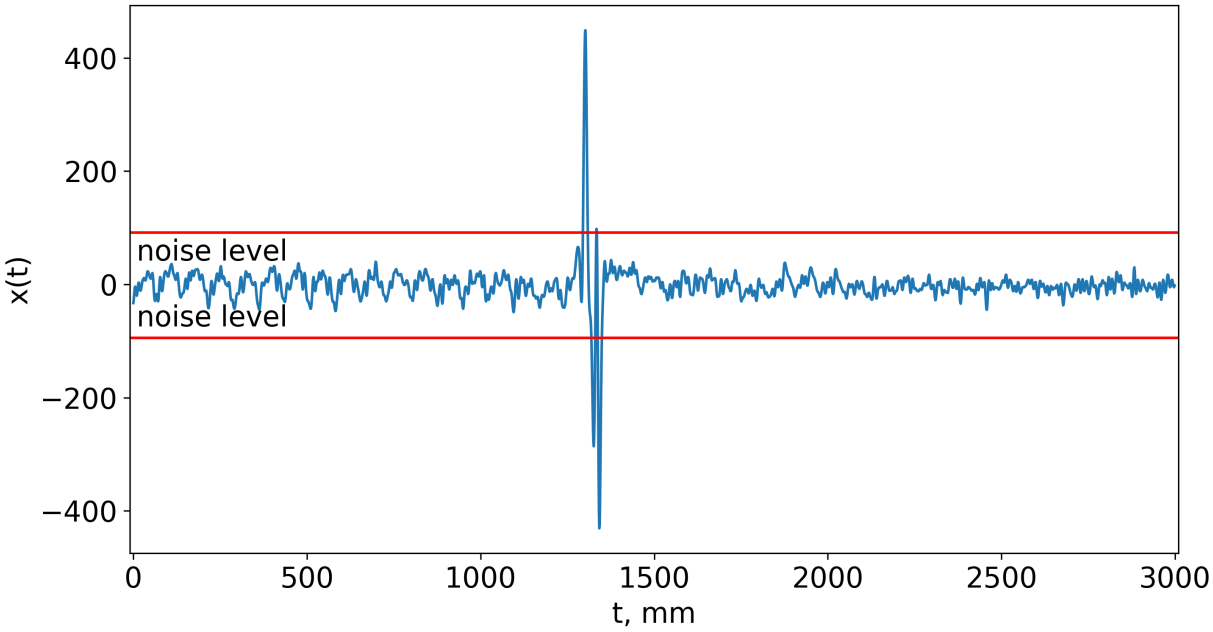


Fig. 8. The section of the defectogram with high-frequency interferences

Рис. 8. Участок дефектограммы с высокочастотными помехами

```
#  Библиотека, предоставляющая инструменты для обработки сигналов
from scipy.fft import irfft, rfft
#  Библиотека, предоставляющая инструменты для анализа временных рядов
from statsmodels.tsa.stattools import acf

def metric_standard(signal_acf, standard):
   #  Вычисление среднеквадратического расстояния между функциями signal_acf и standard
   T = len(signal_acf)
   sum = 0
   for i in range(0, T):
      sum += (signal_acf[i] - standard[i]) ** 2
   return (sum / T)

def optmize_standard(x, standard, frm = 0.0, to = 0.35, step = 0.005):
   #  Поиск threshold частоты пороговой гармоники
   #  frm, to - задают границы отрезка перебора
   #  step - задаёт шаг перебора
   if(len(x) != len(standard)):
      raise ValueError("Длины эталонного участка и участка данных должны совпадать")
   T = len(x)
   standard_acf = acf(standard, T-1)  #  Формирование эталонной автокорреляции
   frequency, threshold, min_metric = frm, T, sys.float_info.max
   X = rfft(x)  #  Прямое преобразование Фурье
   while frequency <= to:
      X[0: get_harmonic(frequency, T)] = 0 #  Спектральное вычитание
      x1 = irfft(X)                        #  Обратное преобразование Фурье
      ac = acf(x1, T - 1)                  #  Построение автокорреляции нового сигнала
      metric = metric_standard(ac, standard_acf)  #  Вычисление расстояния между функциями
      if metric < min_metric:              #  Минимизация расстояния
         threshold = frequency
         min_metric = metric
         frequency += step
   return threshold
```

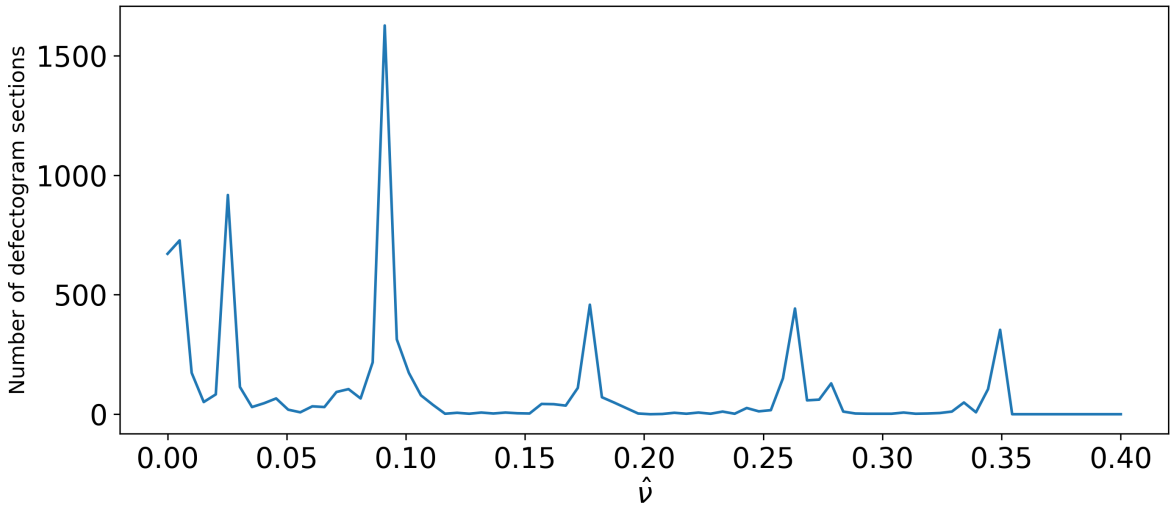


Fig. 9. The values of the threshold harmonic frequency for the method based on reference autocorrelation

Рис. 9. Значения частоты пороговой гармоники для метода, основанного на эталонной автокорреляции

Повторим описанный в главе 3 эксперимент на тех же участках дефектограммы, но вместо величины $d(r_{\hat{s}}(\tau), \hat{r}_{noise}(\tau))$ будем минимизировать $d(r_{\hat{s}}(\tau), r_{ref}(\tau)$. Полученные результаты эксперимента представлены на графике 9.

В отличие от метода, основанного на автокорреляции гауссовского шума, метод эталонной автокорреляции не раздваивает наиболее часто встречающееся значение частоты $\hat{\nu}$ пороговой гармоники Ω на $0,09$ и $0,095$, а указывает одно значение $0,09$. «Сдвоенные пики» также пропадают на участках с высокочастотными помехами. Это может говорить как о том, что метод эталонной автокореляции обладает большей точностью по сравнению с методом, основанном на автокорреляции гауссовского шума, так и наоборот о том, что данный метод искажает реальные значения, сливая пики, расположенные близко друг к другу, в один пик. В остальном же график 9 практически ничем не отличается от графика 7.

Использование автокорреляции гауссовского шума является в некотором смысле универсальным подходом для вихретоковых дефектограмм, тогда как эталонная автокорреляция будет уникальной в зависимости от применяемого оборудования и качества анализируемых данных.

Итак, оба метода как шумовой, так и эталонной автокорреляции наиболее подходящим значением на роль частоты $\hat{\nu}$ пороговой гармоники Ω указывают число в окрестности $0,09$ (первый метод — $0,095$, второй — $0,09$). При использовании любого из этих чисел ($0,09$ или $0,095$) в спектральном вычитании (8) удаётся устранить искажение, вызванное периодическими помехами. Сигнал «выравнивается». При этом форма полезных сигналов остаётся неизменной (рис. 11). Также восстанавливается пороговый уровень шума, что позволяет правильно обнаруживать полезные сигналы на фоне рельсового шума (рис. 12).

Заключение

Главными свойствами рассмотренных на вихретоковых дефектограммах помех являются низкочастотность и периодичность. Предложенный в работе метод подавления помех основан на этих двух характеристиках. Спектральное вычитание позволяет исключить из сигнала низкочастотный диапазон спектра, которым представлены помехи, а функция автокорреляции, оценивающая периодичность сигнала, — определить границы этого диапазона.

Правая граница низкочастотного диапазона, в котором представлены помехи, названа пороговой гармоникой. Она находится с помощью автокорреляционных методов, один из которых основан

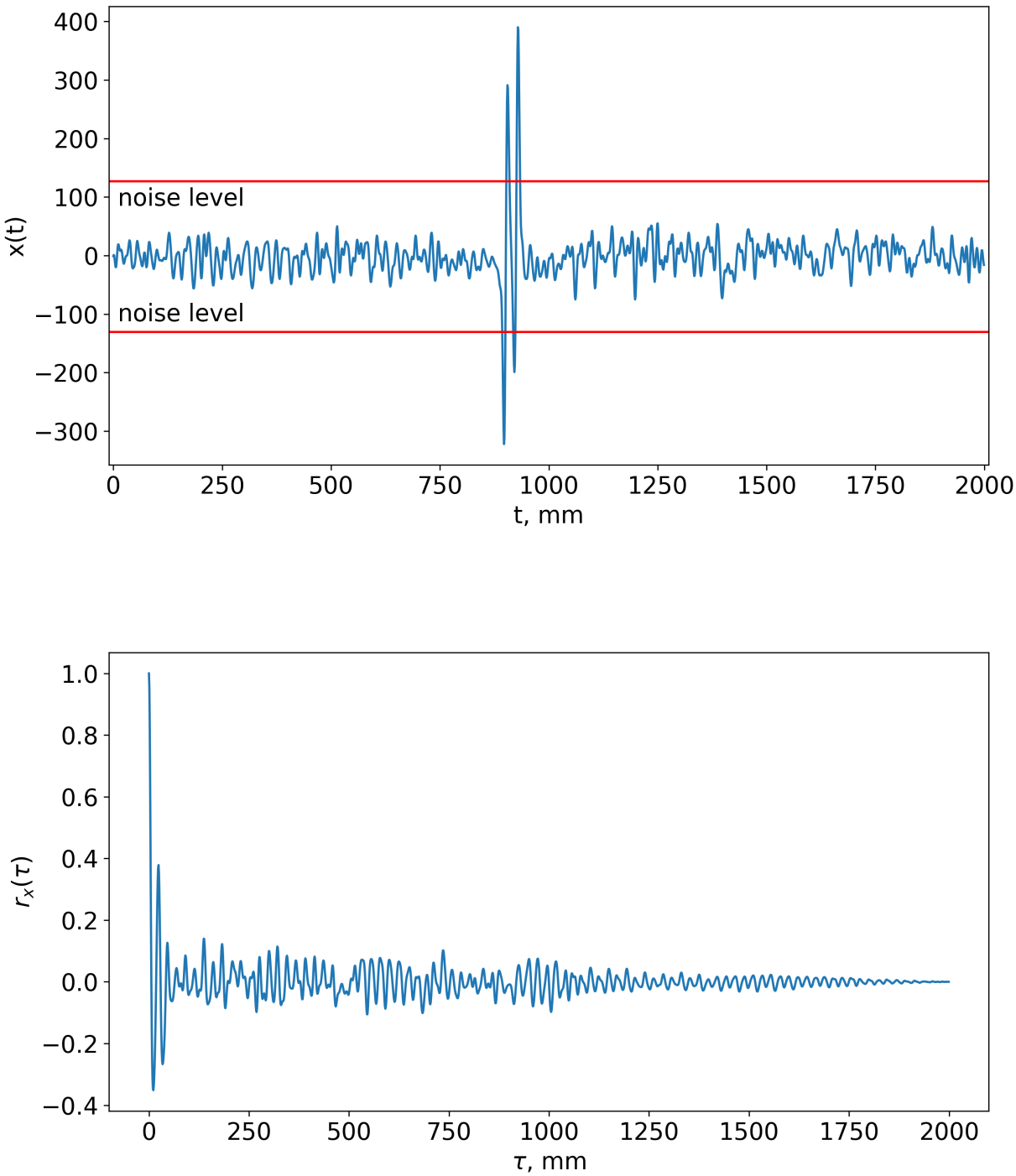


Fig. 10. The reference defectogram section without interferences and its autocorrelation function

Рис. 10. Эталонный участок дефектограммы без помех и его функция автокорреляции

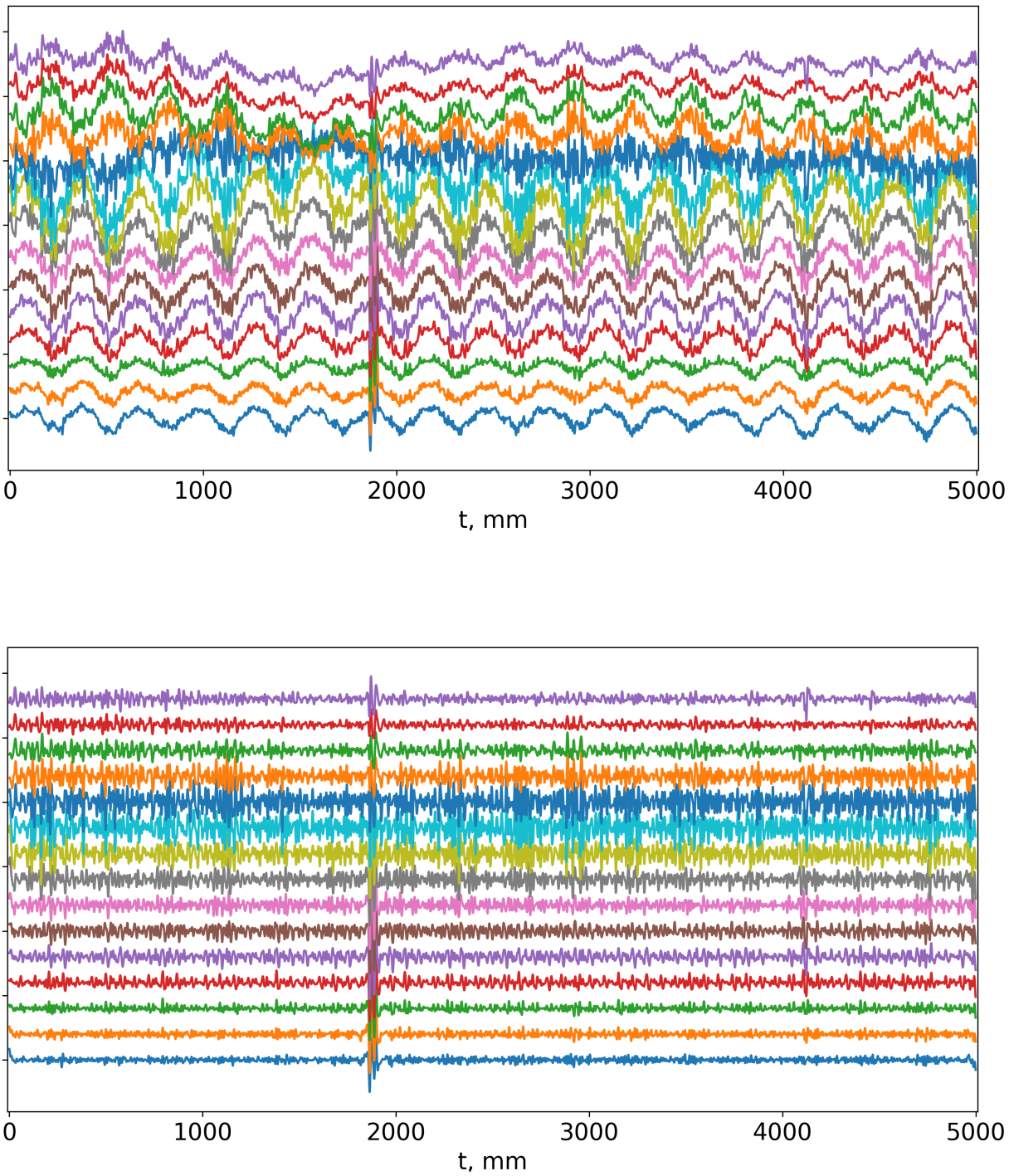


Fig. 11. The example of periodic interference reduction on the defectogram

Рис. 11. Пример подавления периодических помех на дефектограмме

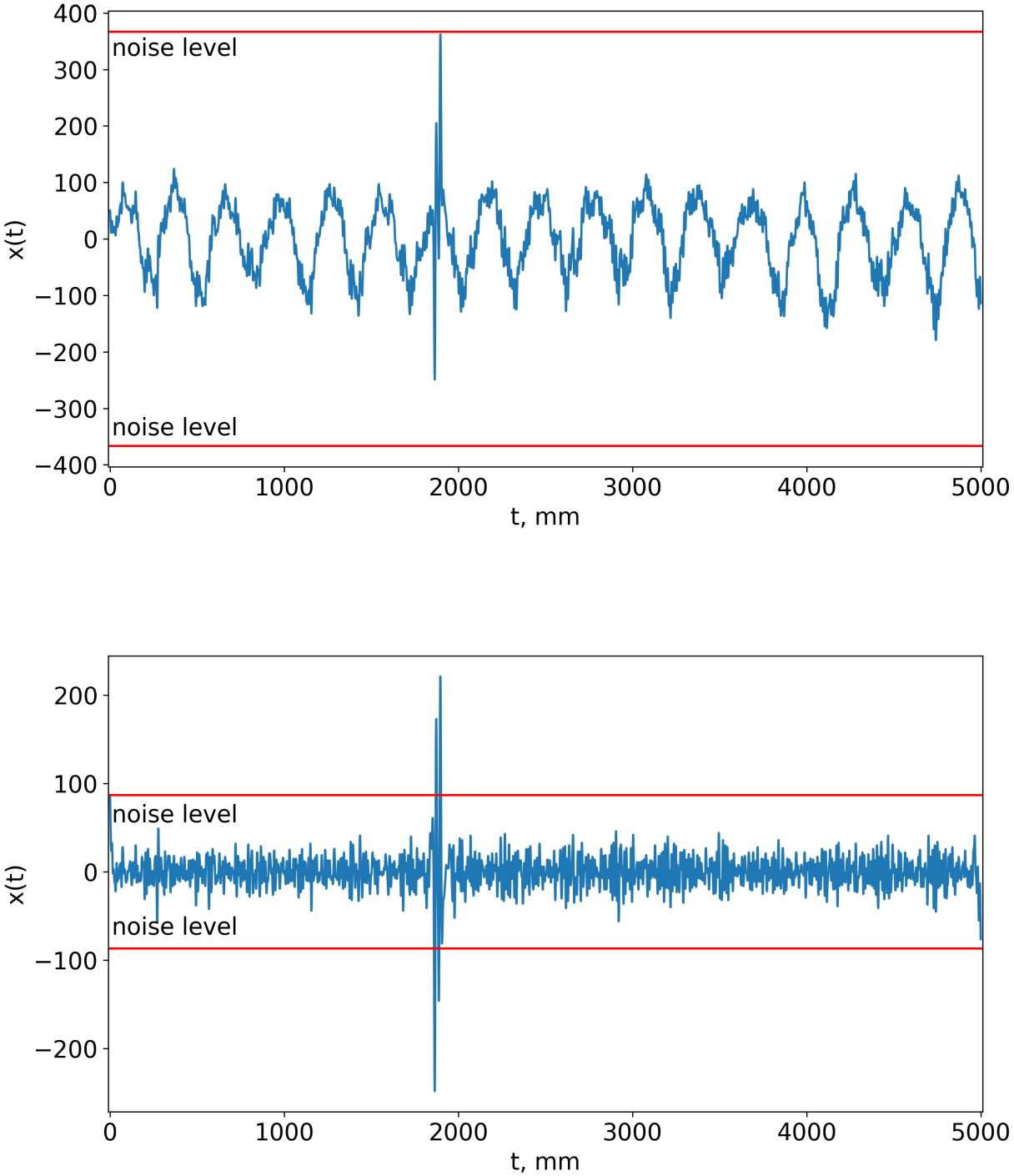


Fig. 12. The example of threshold noise level restoration for one defectogram channel

Рис. 12. Пример восстановления порогового уровня шума для одного канала дефектограммы

на автокорреляции гауссовского шума, а другой — на эталонной автокорреляции. Данные подходы в качестве частоты пороговой гармоники на рассмотренных участках вихретоковых дефектограмм указывают схожие значения: один метод — 0, 095, второй — 0, 09. Метод, основанный на автокорреляции гауссовского шума, в отличие от метода, основанного на эталонной автокорреляции, не зависит от применяемого оборудования и качества собранных данных.

Спектральное вычитание решает задачу подавления низкочастотных и периодических помех. В сочетании с автокорреляционными подходами оно позволяет грамотно отлавливать и исключать помехи на вихретоковых дефектограммах. При этом описанные методы борьбы с периодическими помехами нельзя назвать универсальными. Они основаны на нескольких предположениях, среди которых гипотеза спектрального вычитания (4), разделение спектра сигнала на две непересекающиеся части (от этого прямым образом зависит существование пороговой гармоники), нормальное распределение рельсового шума, непериодичность полезных сигналов и наличие чистого участка данных. Всё это создаёт серьёзные ограничения для применимости описанных подходов в других видах сигнала. Тем не менее, при соблюдении всех необходимых условий спектральное вычитание в сочетании с автокорреляционными методами может использоваться и в других областях, не ограничиваясь вихретоковой дефектоскопией.

References

[1] K. V. Vlasov and A. L. Bobrov, “Influence of object physical properties instability on edge current method sensitivity”, *Vestnik IzhGTU imeni M. T. Kalashnikova*, vol. 27, no. 1, pp. 55–62, 2024, in Russian. DOI: 10.22213/2413-1172-2024-1-55-62.

[2] A. A. Markov and E. A. Kuznecova, *Defektoskopiya rel’sov. Formirovanie i analiz signalov. Kniga 2. Rasshifrovka defektogramm*. Ul’tra Print, 2014, 332 pp., in Russian.

[3] E. V. Kuzmin, O. E. Gorbunov, P. O. Plotnikov, and V. A. Tyukin, “On finding a threshold of useful signals in the analysis of magnetic and eddy current defectograms”, *Modeling and Analysis of Information Systems*, vol. 24, no. 6, pp. 760–771, 2017, in Russian. DOI: 10.18255/1818-1015-2017-6-760-771.

[4] Y. Xu, Y. Jing, Y. Wand, R. He, J. Wang, and Y. Geng, “Novel denoizing method for partial discharge signals using singular value decomposition and spectral subtraction”, *IET Science, Measurement and Technology*, vol. 17, no. 3, pp. 105–114, 2022. DOI: 10.1049/smt2.12134.

[5] Q. Liu, Y. Yu, B. S. Han, and W. Zhou, “An improved spectral subtraction method for eliminating additive noise in condition monitoring system using fiber bragg grating sensors”, *Sensors*, vol. 24, no. 2, p. 443, 2024. DOI: 10.3390/s24020443.

[6] D. Shah and B. Shah, “Comparison of spectral subtraction noise reduction algorithms”, *Journal of Emerging Investigators*, vol. 6, pp. 22–262, 2023. DOI: 10.59720/22-262.

[7] N. Ahmed and K. R. Rao, *Ortogonal’nye preobrazovaniya pri obrabotke cifrovyh signalov*. Svyaz’, 1980, 248 pp., in Russian.

[8] G. M. Jenkins, *Spectral Analysis and Its Applications*. Holden-Day, 1968, 541 pp.

[9] S. F. Boll, “Suppression of acoustic noise in speech using spectral subtraction”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 2, pp. 113–120, 1979. DOI: 10.1109/TASSP.1979.1163209.

[10] O. D. Anderson, “Moments of the sampled autocovariances and autocorrelations for a Gaussian white noise process”, *The Canadian Journal of Statistics*, vol. 18, no. 3, pp. 271–284, 1990. DOI: 10.2307/3315458.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 2, 2024
journal homepage: www.mais-journal.ru

ARTIFICIAL INTELLIGENCE

UAV Detection Using Neural Networks

M. D. Averina[1], O. A. Levanova [1], D. V. Grushevskaya [1], K. A. Kukharev[1], D. M. Murin[1], M. A. Kalinin[2]

DOI: 10.18255/1818-1015-2024-2-182-193

[1]P.G. Demidov Yaroslavl State University, Yaroslavl, Russia
[2]National Research University Higher School of Economics, Moscow, Russia

MSC2020: 68T07
Research article
Full text in Russian

Received May 6, 2024
Revised May 24, 2024
Accepted May 29, 2024

The availability of unmanned aerial vehicles (UAVs) has led to a significant increase in the number of offenses involving their use. This makes the development of UAV detection systems relevant. Solutions based on deep neural networks show the best results in detecting UAVs on video. This article presents a study of various neural network detectors and focuses on identifying objects as small as possible, up to the size of 4×4 and even 3×3 pixels. The work investigates architectures SSD (VGG16) and YOLOv3 and it's modifications. Precision and recall metrics are calculated separately for different intervals of the object areas. The best result have been shown by YOLOv3 model with bbox parameters chosen as the result of object sizes clustering. Small (3×3 px) drones have been successfully identified with 76 % precision and a very small recall of 26 %. For objects between 10 and 20 pixels in area, the recall is 64 % with an accuracy of 75 %. For objects with an area more than 20px the recall is about 90 %, the precision is 89 %, and the F1 score is 90 %. These results show that it is possible to recognize even 4×4 pixel drones, which can be used in video surveillance systems.

Keywords: UAV detection

INFORMATION ABOUT THE AUTHORS

Averina, Maria D. (corresponding author)	ORCID iD: 0009-0005-3111-1488. E-mail: maverina518@gmail.com Postgraduate student
Levanova, Olga A.	ORCID iD: 0000-0001-8078-4447. E-mail: olaydy@gmail.com PhD, associate professor
Grushevskaya, Darya V.	ORCID iD: 0009-0006-5696-5452. E-mail: grushevskaya.d.v@mail.ru M. Sc.
Kukharev, Kirill A.	ORCID iD: 0009-0006-8764-7860. E-mail: kirillkukharev76@gmail.com M. Sc.
Murin, Dmitriy M.	ORCID iD: 0000-0002-8068-0784. E-mail: nirum87@mail.ru PhD, associate professor
Kalinin, Maksim A.	ORCID iD: 0009-0007-7890-6418. E-mail: maksim.kalinin.2110@gmail.com M. Sc.

Funding: Yaroslavl State University (project VIP-016).

For citation: M. D. Averina, O. A. Levanova , D. V. Grushevskaya , K. A. Kukharev, D. M. Murin, and M. A. Kalinin, "UAV detection using neural networks", *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 182–193, 2024. DOI: 10.18255/1818-1015-2024-2-182-193.

© Averina M. D., Levanova O. A., Grushevskaya D. V., Kukharev K. A., Murin D. M., Kalinin M. A., 2024
This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 2, 2024
сайт журнала: www.mais-journal.ru

ARTIFICIAL INTELLIGENCE

Детекция БПЛА при помощи нейронных сетей

М. Д. Аверина[1], О. А. Леванова[1], Д. В. Грушевская [1], К. А. Кухарев[1], Д. М. Мурин[1], М. А. Калинин[2]

DOI: 10.18255/1818-1015-2024-2-182-193

[1]Ярославский государственный университет им. П.Г. Демидова, Ярославль, Россия
[2]Национальный исследовательский университет «Высшая школа экономики», Москва, Россия

УДК 004.93'12
Научная статья
Полный текст на русском языке

Получена 6 мая 2024 г.
После доработки 24 мая 2024 г.
Принята к публикации 29 мая 2024 г.

Доступность беспилотных летательных аппаратов (БПЛА) привела к тому, что их стали часто использовать при совершении правонарушений. Такая ситуация делает актуальной разработку систем обнаружения БПЛА. При обнаружении БПЛА применяются различные подходы на основе анализа радиочастотных и акустических сигналов, а также обработки видео данных. Лучшие результаты в обнаружении БПЛА на видео показывают решения, основанные на глубоких нейронных сетях. В этой статье мы представляем исследование различных нейросетевых детекторов и оценку возможности их практического использования в системах видеонаблюдения. Основной акцент работы направлен на выявление как можно более маленьких объектов, вплоть до размеров 4×4 пикселя. В работе представлены результаты анализа архитектур SSD(VGG16), YOLOv3 и их модификаций. В качестве метрик качества использовались полнота и точность, которые вычислялись отдельно для разных размеров объекта. Лучший результат был получен для модели YOLOv3 со значениями параметров bbox, подобранных в результате кластеризации размеров объектов. При распознавании дронов размера 3×3 удалось достичь точности 76 % при очень маленьком значении полноты 26 %. Для объектов, площадь которых составляет от 10 до 20 пикселей полнота составила 64 % при точности 75 %. Для объектов большего размера в среднем получилась полнота 90 %, точность 89 % и F1-мера 90 %. Данные результаты показывают, что распознавание дронов возможно даже при размере 4×4, что может быть успешно использовано в системах видеонаблюдения.

Ключевые слова: детекция БПЛА

ИНФОРМАЦИЯ ОБ АВТОРАХ

Аверина, Мария Дмитриевна (автор для корреспонденции)	ORCID iD: 0009-0005-3111-1488. E-mail: maverina518@gmail.com Аспирант
Леванова, Ольга Александровна	ORCID iD: 0000-0001-8078-4447. E-mail: olaydy@gmail.com Кандидат физ.-мат. наук, доцент
Грушевская, Дарья Владимировна	ORCID iD: 0009-0006-5696-5452. E-mail: grushevskaya.d.v@mail.ru Магистр
Кухарев, Кирилл Александрович	ORCID iD: 0009-0006-8764-7860. E-mail: kirillkukharev76@gmail.com Магистр
Мурин, Дмитрий Михайлович	ORCID iD: 0000-0002-8068-0784. E-mail: nirum87@mail.ru Кандидат физ.-мат. наук, доцент
Калинин, Максим Александрович	ORCID iD: 0009-0007-7890-6418. E-mail: maksim.kalinin.2110@gmail.com Магистр

Финансирование: ЯрГУ (проект VIP-016).

Для цитирования: M. D. Averina, O. A. Levanova , D. V. Grushevskaya , K. A. Kukharev, D. M. Murin, and M. A. Kalinin, "UAV detection using neural networks", *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 182–193, 2024. DOI: 10.18255/1818-1015-2024-2-182-193.

© Аверина М. Д., Леванова О. А., Грушевская Д. В., Кухарев К. А., Мурин Д. М., Калинин М. А., 2024
Эта статья открытого доступа под лицензией CC BY license (https://creativecommons.org/licenses/by/4.0/).

Введение

В последние годы неуклонно растет число частных лиц, имеющих беспилотные летательные аппараты (БПЛА, дроны, квадрокоптеры). По данным Федерального управления гражданской авиации США (Federal Aviation Administration, FAA) на 29 февраля 2024 года в США был зарегистрирован 781781 дрон (375226 Commercial Drones Registered, 400858 Recreational Drones Registered, 5697 Paper Registrations)[1]. По данным «European ATM Master Plan» SESAR Joint Undertaking, опубликованным в декабре 2019 года, количество коммерческих дронов только в Европе вырастет с 10000 (2015 год) до почти 400000 к 2050 году [1]. В декабре 2023 года был утвержден национальный проект «Беспилотные авиационные системы»[2], согласно которому в России планируется к 2030 году увеличить производство беспилотников до 32000 в год.

В то же время широкое распространение квадрокоптеров привело к значительному увеличению числа нарушений безопасности в общественных местах и на объектах критически важной инфраструктуры. БПЛА могут переносить различную «полезную» нагрузку, а также могут быть оснащены различными датчиками и средствами негласного получения информации. С применением данных устройств нарушаются права на конфиденциальность персональных данных и преодолеваются установленные режимы охраны и защиты зданий и сооружений, а также охраняемых территорий и технологических объектов. В России чаще всего совершаются преступления, связанные с доставкой осужденным в учреждения уголовно-исполнительной системы запрещенных предметов с помощью БПЛА.

Обнаружение и контроль действий БПЛА становятся существенными задачами для обеспечения безопасности, тем самым стимулируя рост рынка средств противодействия им. По данным опроса компании Ernst & Young[3] рост спроса на борьбу с дронами обгоняет рост спроса на них самих. Противодействие им состоит из обнаружения, идентификации и нейтрализации. На текущий момент интенсивно развиваются области обнаружения и нейтрализации. При этом системы обнаружения представляют, как самостоятельный интерес, так и включаются в системы противодействия.

Наиболее известными методами обнаружения БПЛА являются: радиолокационное, радиочастотное обнаружение, обнаружение инфракрасного излучения, электрооптическое, акустическое, и оптическое обнаружение. Оптические методы обнаружения способны эффективно справляться с их обнаружением в условиях дневного света, шума, наличия птиц. К недостаткам оптического метода можно отнести относительно небольшую дальность обнаружения.

Для раннего обнаружения данных необходима возможность распознавания объекта, имеющего малый размер. Поэтому акцент нашей работы сделан на исследовании подходов обнаружения на изображениях дронов маленького размера.

1. Системы видеонаблюдения

Для обеспечения возможности раннего обнаружения особенный интерес представляет распознавание квадрокоптеров, занимающих малую площадь кадра. Пусть разрешение камеры составляет 640 пикселей, мы хотим обеспечить распознавание дрона на расстоянии 200 метров. Предположим, что для обнаружения квадрокоптера необходимо, чтобы он занимал площадь кадра 7×7 пикселей. Такая плотность пикселей может быть достигнута при угле обзора 7 градусов, тогда для кругового обзора объекта видеонаблюдения необходимо 52 камеры. В этих условиях по горизонтали и вертикали одна камера обозревает расстояние приблизительно 23 метра. Таким образом, система видеонаблюдения не сможет «увидеть» квадрокоптер, поднятый на высоту 25 метров.

[1] https://www.faa.gov/uas
[2] http://government.ru/rugovclassifier/906/events/
[3] https://assets.ey.com/content/dam/ey-sites/ey-com/en_ru/topics/advisory/ey-uav-survey-eng.pdf

Теперь предположим, что для обнаружения квадрокоптера занимаемая им площадь должна составлять только 3×3 пикселя (предельный случай). Тогда угол обзора будет равен 15 градусам (для кругового обзора потребуются 24 камеры), а обозреваемое расстояние составит 53 метра. Чтобы «обойти» такую систему нужно поднять дрон уже на 55 метров. Заметим, что при размещении камер видеонаблюдения вдоль периметра объекта через 70 м высота границы обнаружения будет колебаться как раз в пределах от 35 до 53 метров. Таким образом, возможно построение системы видеонаблюдения, которая обнаруживает БПЛА на расстоянии 200 метров, при этом очень важно распознавать в видео объекты очень маленьких размеров.

Также стоит отметить, что для практического применения в системах видеонаблюдения критически важна точность распознавания алгоритма (precision). Если величина $1 - \mathit{precision} > 0.2$, то каждый пятый сигнал о наличии дрона в видео будет ошибочным, однако потребует некоторой реакции (оператора, включения сирены или средств РЭБ). Таким образом, точность ниже 80 % неприемлема для интеграции алгоритмов распознавания в системы видеонаблюдения. Если мы говорим про объекты критически важной инфраструктуры, то более важным становится показатель полноты (recall). Показатели точности и полноты сильно связаны, и при увеличении одного показателя обычно другой уменьшается. В данной работе не отдавалось преимущество ни одному из этих показателей, они анализировались в совокупности.

1.1. Существующие решения

Задача создания системы видеонаблюдения с обнаружением дронов на видео не является новой. Имеющиеся подобные системы, как правило, принадлежат коммерческим организациям и являются платными, что является серьезным препятствием при оценке существующих решений.

Существующие готовые системы в открытом доступе найти достаточно трудно, отметим лишь дрон-детектор от компании «Спецлаб»[4], однако качество обнаружения оставляет желать лучшего. Существенно проще в научных статьях найти информацию о различных подходах к задаче распознавания дронов. Благодаря соревнованию Drone vs Bird[5] и IEEE International Conference on Advanced Video and Signal-based Surveillance (AVSS) тематика обнаружения дронов активно развивается начиная с 2017 года.

В 2024 году вышла большая обзорная статья по их обнаружению [2], как по данным с радара, так и по звуку и видео (обнаружению в видео в статье уделено мало внимания). Для распознавания БПЛА на видео было предпринято множество различных решений: использовались классические алгоритмы, например, SVM [3], нейронные сети, и комбинированный подход [4]. Есть работы, в которых используется только визуальная информация [5], в других предлагается совместное использование видео и звука [6]. Авторы еще одной статьи [4] разделили задачу на обнаружение движущихся объектов (решение основано на вычитании фона) и классификацию обнаруженного объекта на дрон, птицу и фон (за счет нейронной сети). Однако построенное решение работает только для статичной камеры. В некоторых исследованиях была важнее скорость распознавания, а в других — точность и качество обнаружения. В целом, наилучшие показатели достигнуты с помощью сверточных нейронных сетей. Также есть исследования по разработке системы обнаружения БПЛА на основе информации от двух камер (широкоугольной статической и узкоугольной поворотной с увеличением). Использование только оптической информации на основе архитектуры YOLOv3 позволило достичь хороших показателей [7].

Еще стоит отметить статью, посвященную усовершенствованию модели YOLOv5 [8] с целью уменьшения числа параметров и ускорения. В работе используется набор данных, содержащий порядка 8000 дронов среднего размера (до 32×32) и порядка 1000 штук размера 8×8 и меньше.

[4] https://www.goal.ru/neuronet/drone-detector
[5] https://wosdetc2023.wordpress.com/

Улучшенный вариант YOLOv5 показал качество mAP 90.6%, Recall 89.3% (FPS 12.6), после ускорения — mAP 87.2 % (FPS 27.6).

В статье [9] модифицировали сеть YOLOv8s и сравнили качество работы моделей с другими известными архитектурами (TIB-Net, YOLOv5-s, YOLOX-s, YOLOv7, YOLOv7-tiny, YOLOv8s). Авторам удалось достичь точности (P) и полноты (R) по 93.3 %, у оригинальной сети YOLOv8s результаты ощутимо хуже (P = 81.4 %, R = 78.1 %). Среди других моделей лучший же результат показала YOLOv5 P = 88.1 % и R = 90.9 % при тестировании уже на их наборе данных.

2. Постановка задачи

Перед авторами была поставлена задача разработки системы по распознаванию беспилотных летательных аппаратов на видео, полученному в результате статичной съемки без записи звука. Целью данной работы является исследование применимости нейросетевого детектора для систем видеонаблюдения. Поэтому для данной задачи очень важно обеспечить обнаружение дронов как можно меньшего размера, для более раннего реагирования. Также существенным является требование малого числа ложных срабатываний ($presision > 0.8$). Прежде всего необходимо решить, как оценивать качество работы построенного решения. В задачах обнаружения объектов чаще всего для определения качества модели используются классические метрики AP и mAP, которые при решении данной задачи менее информативны. Для системы видеонаблюдения привычнее оперировать такими широко известными метриками как Precision и Recall. Они дают разностороннюю и довольно полную оценку качества. Также было интересно исследовать качество обнаружения для разных размеров дронов. Поэтому отдельно для каждого интервала площади, занимаемой дроном, рассчитывалось общее число размеченных квадрокоптеров (total), число правильно распознанных дронов (TP), число ложно распознанных дронов (FP), число нераспознанных дронов (FN). И по полученным значениям уже рассчитывались основные характеристики Precision и Recall также для каждого размера. Отметим, что в различных статьях по обнаружению дронов используются разные метрики.

2.1. Набор данных

Для создания системы по распознаванию дронов на видеоизображениях необходим большой объем размеченных данных. Для качественного обучения необходим был набор, состоящий из нескольких сотен тысяч размеченных изображений / видеокадров. В ходе работы был самостоятельно создан набор размеченных изображений, а также предприняты попытки поиска готовых наборов данных.

2.2. Готовые наборы данных

В свободном доступе было найдено несколько подходящих наборов изображений с размеченными БПЛА.

- Набор изображений дронов «Drone Dataset (UAV)» с сайта Kaggle[6], созданный в 2019 году. Набор состоит из 1359 фотографий дронов и такого же количества файлов txt и xml с разметкой в формате, необходимом для обучения нейронных сетей. Размер набора: 725.28 Мб.
- Набор данных с беспилотными летательными аппаратами «Drone Dataset: Amateur Unmanned Air Vehicle Detection» с сайта Mendeley Data[7]. Данный набор был создан в 2019 году группой авторов для проекта «Amateur Drone Detection and Tracking». В наборе данных более 4000 снимков любительских дронов (например, квадрокоптер DJI Phantom). Кроме того, набор данных содержит «негативные» объекты, не относящиеся к дронам. Размер набора: 157 Мб.

Перечисленные наборы данных в основном содержат объекты с площадью больше 1000 пикселей и не такие многочисленные, как необходимо.

[6] https://www.kaggle.com/datasets/dasmehdixtr/drone-dataset-uav
[7] https://data.mendeley.com/datasets/zcsj2g2m4c/4

Также был проанализирован набор 2020 года с размеченными видеороликами, предоставляемый организаторами соревнования Drone vs Bird. Во втором столбце таблицы 1 представлена статистика по размерам дронов в данном наборе (после приведения картинок к размеру 608 × 608). Из таблицы видно, что дроны чаще всего имеют крупный размер, что не совсем подходит для решения поставленной задачи.

Поиск остальных открытых наборов данных был затруднен тем, что информация о них содержится только в научных статьях по обнаружению дронов. Так в открытом доступе был найден набор данных FL-dataset[8], который содержит 14 черно-белых видео с дронами (38948 кадров) разрешения 752×480, съемка с дрона, в кадре одновременно до 2-х дронов. Минимальный, средний и максимальный размеры дронов: 9×9, 25.5×16.4 и 259×197 соответственно. Данный набор исключили, поскольку видео черно-белые и съемка с нестационарной камеры. Набор данных[9] содержит 50 видео (70250 кадров) с несколькими дронами в кадре, типичный размер дронов от 10×8 до 65×21. Однако съемка производилась с дрона, т. е. камера нестационарная, поэтому этот набор данных не был взят, хотя для построения детектора с покадровой обработкой он вполне подходит. В статье [10] был использован данный набор, рассмотрена модель построения как детектора, так и трекера для слежения за БПЛА. Детектор содержит много шагов, в частности используются: нахождение движения с помощью оптического потока, нахождение ключевых точек, небольшие полносвязные сети, которые обучают. Трекер строится на основе фильтра Калмана. Авторам удалось достичь следующих показателей качества: точность — 88.8 %, полнота — 89.0 %, F-мера — 88.9 %. Отметим, что в работе идет сравнение только с одной моделью из статьи [11], которую они обучили на предложенном наборе данных.

Набор данных TIB-Net[10] содержит 2850 картинок, съемки на стационарную камеру с расстояния около 500 метров. Набор данных в основном содержит изображения площадью более 100 пикселей при разрешении 1920 × 1080. В статье [12] предложена архитектура нейронной сети, которая показала качество $mAP = 89.2$. Набор данных состоит из отдельных изображений, но визуально можно однозначно определить, что кадры взяты из нескольких видео. Набор изначально не разделен на обучающую и тестовую выборки. Вероятно, авторы статьи владели информацией из какого видео взято изображение, и в обучающем наборе не было кадров похожих на изображения из тестовой выборки. Однако пользоваться этим набором данных можно только целиком в качестве либо тестовой выборки, либо обучающего набора. Иначе, если разделить кадры случайным образом, то очень вероятна ситуация, при которой в тестовой выборке окажутся кадры очень похожие на те, которые были в обучающей выборке. Данный набор не использовался в нашей работе, поскольку был найден уже после проведения исследований.

2.3. Авторский набор данных

Было принято решение о создании собственного набора данных. Членами команды было отснято 13 видео с полетом дрона в различных локациях суммарной длительностью 3.5 часа. Съемки производились с помощью двух уличных видеокамер HIKVISION HiWatch (модель DS-I200(C) 2.8 мм; модель DS I200(B) 4 мм) с максимальным разрешением 1920 × 1080 пикселей и частотой смены кадров 25 к/с. В съемках участвовало два дрона: DJI Phantom 4 Pro (размер около 247.5 × 247.5 мм) и DJI Mini 2 Fly More Combo (размер: 203 × 159 × 56 мм). В кадре всегда присутствовал один квадрокоптер.

Полученные видеофайлы были сначала предварительно размечены специально написанным для этой задачи инструментом трекинга за одним объектом. После разметка уточнялась с помощью готового инструмента Computer Vision Annotation Tool (CVAT)[11].

[8] https://drive.switch.ch/index.php/s/3b3bdbd6f8fb61e05d8b0560667ea992
[9] https://engineering.purdue.edu/~bouman/UAV_Dataset/
[10] https://drive.google.com/drive/folders/1ro-S2lwBmn83HLSppr5i-hBHLlYLAobg
[11] https://github.com/openvinotoolkit/cvat

Table 1. The distribution of the drones number by area occupied

Таблица 1. Распределение числа дронов по занимаемой площади

Площадь дрона	**Количество дронов Drone-vs-bird**	**Количество дронов авторский набор**	**Размер дрона, пиксели**
от 0 до 10	3776	31008	$\sim 3 \times 3$
от 11 до 50	19205	67370	$\sim 5 \times 5$
от 51 до 100	18656	23790	$\sim 10 \times 10$
от 101 до 500	54414	52487	$\sim 22 \times 22$
от 501 до 1000	10775	2755	$\sim 100 \times 100$
более 1000	10994	3221	$> 100 \times 100$
Всего	117820	180619	

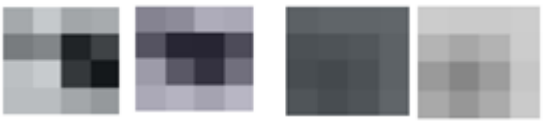


Fig. 1. Drone (left) and bird (right) are the size of 4×4 pixels

Рис. 1. Дрон (слева) и птица (справа) размером 4×4 пикселя

Всего было размечено порядка 180 тысяч дронов. В таблице 1 в третьей колонке приведена статистика количества дронов созданного набора данных для различных размеров по площади. Как можно заметить из таблицы при создании базы упор был сделан на то, чтобы квадрокоптер в кадре был маленького размера.

Самый минимальный размер объекта в созданной базе данных 3×3 пикселя, человеку визуально невозможно определить тип такого маленького объекта. При разметке была использована дополнительная информация о траектории объекта, например, когда дрон удаляется. Начиная с размера 4×4 пикселя можно попробовать отличать объект, опираясь только на изображение. На рисунке 1 представлено несколько примеров объектов такого размера. Дроны (слева) чаще всего имеют силуэт равнобедренного треугольника с вершиной, направленной вниз, а птицы (справа) более овальные. С совсем маленькими изображениями очень легко ошибиться, наиболее уверенно можно различать изображения размером 10×10 пикселей. При таком размере уже зачастую легко распознать объект, примеры изображений можно видеть на рисунке 2. Полученная база видеоданных с разметкой беспилотных летательных аппаратов (квадрокоптеров) была зарегистрирована в 2021 году в Федеральной службе по интеллектуальной собственности (Роспатент) [13].

Финально в качестве набора данных были взяты как авторский набор, так и Drone-vs-bird, которые в совокупности содержат около 300 тысяч дронов (встречаются изображения с несколькими объектами). В тренировочную выборку вошло 80 % изображений (225277), а остальные использовались для валидации (17026) и тестирования (38201). Подчеркнем, что тестовый и валидационный наборы данных состояли из видео, которых не было в обучении.

Отметим, что при использовании авторского набора данных сравнительный анализ с другими результатами становится затруднительным. Если обучать на авторском наборе данных, а тестировать на открытом, то логично будет получить качество лучше, поскольку сеть обучалась на большем наборе данных. При обучении на общедоступных наборах результаты можно будет сравнить с другими моделями, но тогда авторский набор данных можно использовать только для тестирования. И максимум, что можно показать в этом случае, — это то, что предложенная модель работает лучше для наших данных. Авторы ставили задачу построения коммерческого решения, которое будет работать в условиях, приближенных к съемке камерой видеонаблюдения. Поэтому цель работы — это добиться наилучшего качества для нашего набора данных, в котором много небольших и да-

Fig. 2. Drone (left) and bird (right) are the size of 10×10 pixels

Рис. 2. Дрон (слева) и птица (справа) размером 10×10 пикселей

же крошечных объектов. В виду этих причин сравнение качества производилось только с работой классической базовой архитектуры, которую в дальнейшем улучшали.

3. Обнаружение дронов

Современные модели нейронных сетей, используемые для решения задачи обнаружения и классификации объектов на изображении, подразделяют на 3 основные группы.

- Двухэтапные детекторы (Two-Stage object detectors) используют алгоритм генерации регионов интересов (region-proposal). Основным представителем этого семейства является модель R-CNN (region-based convolutional networks) и ее модификации — Fast R-CNN, Faster R-CNN и Mask R-CNN.
- Одноэтапные детекторы (One-stage object detectors) — это полностью сверточные нейронные сети, что позволяет обрабатывать изображения в реальном времени. Выделяют такие модели, как SSD, YOLO, RFBNet.
- Детекторы объектов на основе точек (Points-based object detectors) основаны на обнаружении ключевых точек, по которым определяют остальные свойства объекта: центр и размер ограничивающего прямоугольника, трехмерное местоположение, ориентацию и даже позу.

Для систем видеонаблюдения немаловажным фактором является скорость работы, поскольку предполагается возможность функционирования алгоритмов в режиме реального времени. Поэтому были выбраны наиболее быстрые одноэтапные детекторы, которые основаны на концепции заранее заготовленных anchor bboxes и предсказаниях по сетке (predictions on a grid). В рамках данной работы был проведен ряд экспериментов при обучении моделей SSD и YOLOv3.

3.1. SSD

В качестве базовой модели для дообучения была взята модель SSD (backbone VGG16) с размером входного изображения 512×512. Поэтому набор данных был преобразован к этим размерам. В таблице 2 представлена статистика, полученная для модели SSD при пороге доверия классификатору confidence = 0.3. Обучение проходило 20 эпох (далее наблюдалось переобучение), batch_size = 12, lr = 0.0025. Для объектов размера порядка 7×7 точность и полнота уже близки к 80 %. Для объектов меньшего размера начинает существенно падать полнота. Для размеров порядка 4×4 и 5×5 около половины дронов сеть находит, а для размера 3×3 обнаружение маловероятно.

Также были проанализированы результаты работы при сниженных порогах confidence = 0.2 и 0.1 (см. таблица 3). Снижением порога можно добиться увеличения полноты, но при этом ощутимо ухудшается точность.

3.2. YOLO v3

Также была протестирована модель YOLO v3, размер входного изображения 608×608. Были рассмотрены несколько подходов. Сначала была обучена оригинальная модель, в таблице 4 представлены результаты на разных эпохах. При обучении использовались следующие значения параметров: lr = 0.01, batch_size = 12, confidence = 0.3. При сравнении 1-й и 5-й эпохи можно видеть, что средние показатели почти не изменились, однако произошло существенное улучшение точности для объектов 3×3 при падении точности для объектов большего размера. На 25-й эпохе видно продолжение этой тенденции. Как обсуждалось выше, дроны размера 3×3 сложно отличить

Table 2. SSD model results

Таблица 2. Результаты модели SSD

Площадь	Всего	TP	FP	FN	Точность	Полнота	Размер
от 0 до 10	6490	959	1359	5531	0.41	0.15	$\sim 3 \times 3$
от 11 до 20	10005	5056	2585	4949	0.66	0.51	$\sim 4 \times 4$
от 21 до 30	6050	3724	888	2326	**0.81**	0.62	$\sim 5 \times 5$
от 31 до 50	4227	3360	927	867	**0.87**	**0.80**	$\sim 7 \times 7$
от 51 до 100	3253	2965	941	288	**0.76**	**0.91**	$\sim 10 \times 10$
от 101 до 150	3011	2678	326	333	**0.89**	**0.89**	$\sim 12 \times 12$
от 151 до 200	1092	1039	97	53	**0.92**	**0.95**	$\sim 14 \times 14$
от 201 до 500	3351	3247	101	104	**0.97**	**0.97**	$\sim 22 \times 22$
более 500	722	678	28	44	**0.96**	**0.94**	
Всего	38201	23706	7252	14495	0.77	0.62	

Table 3. SSD model test results with less confidence threshold

Таблица 3. Результаты модели SSD при уменьшенном confidence

Confidence	0.2		0.1	
Площадь, пиксель	Полнота	Точность	Полнота	Точность
от 0 до 10	0.20	0.35	0.27	0.19
от 11 до 20	0.60	0.58	0.70	0.38
от 21 до 30	0.77	0.77	0.88	0.63
от 31 до 50	0.90	0.75	0.93	0.54
от 51 до 100	0.95	0.70	0.96	0.49
от 101 до 150	0.91	0.80	0.92	0.62
от 151 до 200	0.96	0.83	0.97	0.56
от 201 до 500	0.98	0.94	0.98	0.71
более 500	0.94	0.93	0.96	0.85
Всего	0.69	0.69	0.76	0.47

от шума или птицы. А модель при обучении стала смещать внимание именно в сторону маленьких объектов (их больше в выборке). Чтобы убрать перекос, был проведен эксперимент, при котором из обучающей выборки были удалены объекты размера 3×3 пикселя. Однако в результате было получено еще большее ухудшение при обучении для больших объектов.

Именно в этот момент было принято решение о сборе подробной статистики по размерам объектов в выборке. В результате модель YOLOv3 обучили с уменьшенными размерами якорных боксов (anchor boxes): $[(4,6), (10,10), (16,16), (45,20), (20,51), (50,79), (80,90), (156,198), (200,200)]$. В этот раз перекоса при обучении не стало, результаты после 7-й эпохи показаны в таблице 5, левый столбец. Для дронов с площадью больше 20 удалось достичь показателей точности и полноты больше 80 %. Отметим также, что уменьшение порога IoU при обучении до 0.3 дало еще небольшое улучшение в качестве, правый столбец таблицы 5.

Поскольку уменьшение размеров базовых bbox улучшило результаты, то мы провели кластеризацию bbox на 9 классов. В качестве якорных боксов были взяты центры кластеров: $[(4,6), (11,13), (21,23), (39,42), (86,71), (225,194), (321,369), (558,290), (551,505)]$. В таблице 6 указан лучший результат после 9-й эпохи обучения. Отметим, что при обучении был выставлен порог IoU = 0.5, таким образом, также удалось повысить качество работы модели благодаря подбору размеров bbox. Для объектов с размером площади более 20 пикселей точность и полнота уже превышают 87 %.

Таким образом, более точная настройка внутренних параметров модели YOLOv3 позволила улучшить качество работы для дронов маленьких размеров. Так для размеров порядка 4×4 сеть распознает 64 % дронов (улучшили на 8 % по сравнению с базовой моделью), при точности в 75 %

Table 4. Results of the YOLOv3 model for different epoch number

Таблица 4. Результаты модели YOLOv3 при разном числе эпох

Число эпох	**1**		**5**		**25**	
Площадь, пиксель	**Полнота**	**Точность**	**Полнота**	**Точность**	**Полнота**	**Точность**
от 0 до 10	0.29	0.12	0.27	0.56	0.18	0.57
от 11 до 20	0.66	0.66	0.65	0.65	0.56	0.71
от 21 до 30	0.84	0.87	0.86	0.74	0.82	0.65
от 31 до 50	0.86	0.82	0.89	0.68	0.91	0.71
от 51 до 100	0.90	0.76	0.94	0.24	0.95	0.51
от 101 до 150	0.88	0.89	0.90	0.74	0.91	0.82
от 151 до 200	0.90	0.87	0.94	0.74	0.94	0.80
от 201 до 500	0.85	0.98	0.90	0.95	0.94	0.88
более 500	0.77	0.82	0.85	0.81	0.90	0.94
Всего	0.71	0.58	0.72	0.58	0.69	0.70

Table 5. Model YOLOv3 results with reduced bbox

Таблица 5. Результаты YOLOv3 с уменьшенными bbox

IoU	**0.5**		**0.3**	
Площадь, пиксель	**Полнота**	**Точность**	**Полнота**	**Точность**
от 0 до 10	0.24	0.79	0.23	0.77
от 11 до 20	0.58	0.72	0.57	0.70
от 21 до 30	0.82	0.80	0.88	0.81
от 31 до 50	0.82	0.76	0.91	0.83
от 51 до 100	0.94	0.80	0.95	0.85
от 101 до 150	0.89	0.96	0.91	0.96
от 151 до 200	0.94	0.97	0.97	0.92
от 201 до 500	0.93	0.88	0.96	0.88
более 500	0.85	0.96	0.91	0.92
Всего	0.68	0.81	0.71	0.81

(улучшили на 4 %). Для объектов 3×3 удалось существенно поднять точность (76 % против 57 %), однако полнота остается крайне низкой (26 %, улучшили на 8 %). В целом можно отметить, что вне зависимости от используемой модели размер 3×3 распознается плохо, поэтому его возможно использовать лишь как предварительный сигнал, а принятие решения о нахождении дрона уже делать на основе трекинга такого объекта.

После обнаружения БПЛА нейронной сетью можно наложить ограничение на размер объекта для получения подходящих характеристик системы. Зададим максимальный размер БПЛА, обнаружение которых будет игнорироваться, и система не будет подавать сигнал о распознавании. В таблице 7 можно видеть сравнение качества обученных моделей в зависимости от порогового значения размера дрона. Так, например, во второй строке показаны метрики моделей, в случае когда дроны размеров 4×4 и меньше будут игнорироваться (не учитываться при оценке качества). Видно, что для таких размеров БПЛА более точная настройка внутренних параметров модели YOLOv3 позволила улучшить точность обнаружения с 70 % у базовой модели до 89 %. Для дронов размера больше 3×3 полнота была улучшена на 3 %, точность — на 15 %. А для БПЛА размера больше 5×5 лучшая модель показала качество выше 90 % (и полнота и точность). Еще раз отметим, что для всех моделей есть возможность менять соотношение метрик полноты и точности за счет изменения параметра confidence у уже обученной сети, (здесь у всех моделей задано значение 0.3).

Table 6. Model YOLOv3 results with bbox from clustering

Таблица 6. Результаты YOLOv3 с bbox по кластеризации

Площадь, пиксель	Полнота	Точность	F1
от 0 до 10	0.26	0.76	0.39
от 11 до 20	0.64	0.75	0.69
от 21 до 30	0.88	0.88	0.86
от 31 до 50	0.90	0.84	0.87
от 51 до 100	0.95	0.87	0.91
от 101 до 150	0.90	0.93	0.92
от 151 до 200	0.94	0.94	0.94
от 201 до 500	0.93	0.95	0.94
более 500	0.88	0.96	0.92
Всего	0.72	0.80	0.78

Table 7. Comparative analysis of models

Таблица 7. Сравнительный анализ моделей

Модель	SSD 0.3		YOLOv3 base		YOLOv3 less bbox		YOLOv3 clust bbox	
Размер	П	Т	П	Т	П	Т	П	Т
$\sim 3 \times 3$	0.72	0.80	0.79	0.70	0.78	0.81	0.82	0.85
$\sim 4 \times 4$	0.82	0.84	0.90	0.70	0.87	0.84	0.90	0.89
$\sim 5 \times 5$	0.89	0.85	0.92	0.72	0.89	0.85	0.92	0.90
$\sim 7 \times 7$	0.93	0.88	0.93	0.72	0.92	0.89	0.92	0.92
$\sim 10 \times 10$	0.93	0.93	0.93	0.85	0.91	0.93	0.92	0.94

Заключение

В статье приведены результаты исследования задачи обнаружения дронов, занимающих малую площадь кадра, в видеопотоке. Были рассмотрены нейросетевые архитектуры SSD (VGG16) и YOLOv3, а также их модификации с целью улучшения качества обнаружения малых объектов.

Имеющиеся в открытом доступе наборы данных содержали недостаточное для решения поставленной задачи число размеченных дронов малых размеров, поэтому авторами был создан новый датасет, содержащий порядка 180 тысяч размеченных дронов малых размеров (до 100×100). Для обучения моделей он был дополнен материалами из открытых источников.

Акцент при составлении набора данных был сделан на изображениях с дронами и квадрокоптерами. При этом в наборе данных отсутствовали изображения беспилотников самолетного и вертолетного типов. Поскольку различные виды беспилотников имеют свои отличительные признаки, ожидается снижение точности распознавания для дронов самолетного и вертолетного типа. Авторами исследования не проводились тесты, связанные с другими видами беспилотных летательных аппаратов.

В ходе экспериментов было установлено, что улучшение точности обнаружения для объектов 3×3 может существенно влиять на точность распознавания объектов большего размера. Значения метрик качества вычислялись для разных размеров дронов в отдельности. При распознавании дронов размера 3×3 удалось достичь точности 76 % при очень маленьком значении полноты 26 %. Для объектов площадью из интервала (10, 20] полнота составила 64 % при точности 75 %. Для объектов большего размера в среднем получилась полнота 90%, точность 89 % и F1-мера 90 %. Таким образом, эффективное распознавание дронов возможно даже при размере 4×4. Однако вне зависимости от используемой модели результаты обнаружения дрона размера 3×3 на текущий момент оказываются неудовлетворительными и могут использоваться лишь как предварительный сигнал.

В рамках данного проекта продолжается работа по улучшению качества обнаружения. Рассматривается возможность добавить в набор данных изображения с другими видами беспилотников. С помощью 3D моделирования в Unity уже создана модель квадрокоптера, однако реалистичность фоновой сцены требует существенного улучшения. Также планируется создать набор различных моделей квадрокоптеров, беспилотников вертолетного и самолетного типа.

Оригинальная модель YOLOv5 (без всяких настроек) была обучена, но показала результаты ощутимо хуже, поэтому результаты в статье не приведены. Сейчас производится настройка параметров модели YOLOv5, которая, по заявлению авторов, должна лучше работать с маленькими объектами, что подтверждается публикациями по распознаванию дронов (без модификаций лучше чем YOLOv8 [9]). Также планируется добавить использование трекеров с целью улучшения показателей качества распознавания.

References

[1] *European ATM master plan: Digitalising Europe's aviation infrastructure. Executive view*, SESAR Joint Undertaking, ISBN: 978-92-9216-134-7, 2020. DOI: 10.2829/695700.

[2] N. Al-lQubaydhi *et al.*, "Deep learning for unmanned aerial vehicles detection: A review", *Computer Science Review*, vol. 51, p. 100 614, 2024. DOI: https://doi.org/10.1016/j.cosrev.2023.100614.

[3] F. Mahdavi and R. Rajabi, "Drone detection using convolutional neural networks", in *2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, 2020, pp. 1–5. DOI: 10.1109/ICSPIS51611.2020.9349620.

[4] U. Seidaliyeva, D. Akhmetov, L. Ilipbayeva, and E. Matson, "Real-time and accurate drone detection in a video with a static background", *Sensors*, vol. 20, no. 14, p. 3856, 2020. DOI: 10.3390/s20143856.

[5] A. Coluccia *et al.*, "Drone vs. bird detection: Deep learning algorithms and results from a grand challenge", *Sensors*, vol. 21, no. 8, p. 2824, 2021. DOI: 10.3390/s21082824.

[6] S. Jamil *et al.*, "Malicious UAV detection using integrated audio and visual features for public safety applications", *Sensors*, vol. 20, no. 14, p. 3923, 2020.

[7] E. Unlu, E. Zenou, N. Riviere, and P.-E. Dupouy, "Deep learning-based strategies for the detection and tracking of drones using several cameras", *IPSJ T Comput Vis Appl*, vol. 11, no. 7, Unlu2019, 2019. DOI: 10.1186/s41074-019-0059-x.

[8] Y. Shang, C. Liu, D. Qiu, Z. Zhao, R. Wu, and S. Tang, "AD-YOLOv5s based UAV detection for low altitude security", *International Journal of Micro Air Vehicles*, vol. 15, p. 17 568 293 231 190 017, 2023. DOI: 10.1177/17568293231190017.

[9] X. Zhai, Z. Huang, T. Li, H. Liu, and S. Wang, "YOLO-drone: An optimized YOLOv8 network for tiny UAV object detection", *Electronics*, vol. 12, p. 3664, 2023. DOI: 10.3390/electronics12173664.

[10] J. Li, D. H. Ye, M. Kolsch, J. P. Wachs, and C. A. Bouman, "Fast and robust UAV to UAV detection and tracking from video", *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 3, pp. 1519–1531, 2022. DOI: 10.1109/TETC.2021.3104555.

[11] A. Rozantsev, V. Lepetit, and P. Fua, "Detecting flying objects using a single moving camera", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 5, pp. 879–892, 2017. DOI: 10.1109/TPAMI.2016.2564408.

[12] H. Sun, J. Yang, J. Shen, D. Liang, L. Ning-Zhong, and H. Zhou, "Tib-net: Drone detection network with tiny iterative backbone", *IEEE Access*, vol. 8, pp. 130 697–130 707, 2020.

[13] O. A. Levanova, D. V. Grushevskaya, A. A. Britnev, M. D. Averina, and D. M. Murin, *Baza videodannyh s razmetkoj bespilotnyh letatel'nyh apparatov (kvadrokopterov). Svidetelstvo o gosudarstvennoy registratsii bazy dannyh no. 2021620329, date 2021-02-25.*

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 2, 2024
journal homepage: www.mais-journal.ru

ARTIFICIAL INTELLIGENCE

Automatic Determination of Semantic Similarity of Student Answers With the Standard One Using Modern Models

N. S. Lagutina[1], K. V. Lagutina[1], V. N. Kopnin[1]

DOI: 10.18255/1818-1015-2024-2-194-205

[1]P.G. Demidov Yaroslavl State University, Yaroslavl, Russia

MSC2020: 68T50
Research article
Full text in Russian

Received March 20, 2024
Revised April 11, 2024
Accepted April 17, 2024

The paper presents the results of a study of modern text models in order to identify, on their basis, the semantic similarity of English-language texts. The task of determining semantic similarity of texts is an important component of many areas of natural language processing: machine translation, information retrieval, question and answer systems, artificial intelligence in education. The authors solved the problem of classifying the proximity of student answers to the teacher's standard answer. The neural network language models BERT and GPT, previously used to determine the semantic similarity of texts, the new neural network model Mamba, as well as stylometric features of the text were chosen for the study. Experiments were carried out with two text corpora: the Text Similarity corpus from open sources and the custom corpus, collected with the help of philologists. The quality of the problem solution was assessed by precision, recall, and F-measure. All neural network language models showed a similar F-measure quality of about 86 % for the larger Text Similarity corpus and 50–56 % for the custom corpus. A completely new result was the successful application of the Mamba model. However, the most interesting achievement was the use of vectors of stylometric features of the text, which showed 80 % F-measure for the custom corpus and the same quality of problem solving as neural network models for another corpus.

Keywords: natural language processing; text similarity; text classification; neural network language models; assessing students' open responses; artificial intelligence in education

INFORMATION ABOUT THE AUTHORS

Lagutina, Nadezhda S.	ORCID iD: 0000-0002-6137-8643. E-mail: lagutinans@gmail.com PhD, associate professor
Lagutina, Ksenia V. (corresponding author)	ORCID iD: 0000-0002-1742-3240. E-mail: lagutinakv@mail.ru PhD, associate professor
Kopnin, Vladislav N.	ORCID iD: 0009-0007-5451-775X. E-mail: vlad.kopnen@mail.ru Student

Funding: Yaroslavl State University (project VIP-016).

For citation: N. S. Lagutina, K. V. Lagutina, and V. N. Kopnin, "Automatic determination of semantic similarity of student answers with the standard one using modern models", *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 194–205, 2024. DOI: 10.18255/1818-1015-2024-2-194-205.

© Lagutina N. S., Lagutina K. V., Kopnin V. N., 2024
This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 2, 2024
сайт журнала: www.mais-journal.ru

Автоматическое определение семантического сходства ответов учащихся с эталонным с помощью современных моделей

Н. С. Лагутина[1], К. В. Лагутина[1], В. Н. Копнин[1]

DOI: 10.18255/1818-1015-2024-2-194-205

[1]Ярославский государственный университет им. П.Г. Демидова, Ярославль, Россия

УДК 004.912
Научная статья
Полный текст на русском языке

Получена 20 марта 2024 г.
После доработки 11 апреля 2024 г.
Принята к публикации 17 апреля 2024 г.

В работе представлены результаты исследования современных моделей текста с целью выявления на их основе семантической близости текстов на английском языке. Задача определения семантического сходства текстов является важной составляющей многих областей обработки естественного языка: машинного перевода, поиска информации, систем вопросов и ответов, искусственного интеллекта в образовании. Авторы решали задачу классификации близости ответов учащихся к эталонному ответу учителя. Для исследования были выбраны нейросетевые языковые модели BERT и GPT, ранее применявшиеся к определению семантического сходства текстов, новая нейросетевая модель Mamba, а так же стилометрические характеристики текста. Эксперименты проводились с двумя корпусами текстов: корпус Text Similarity из открытых источников и собственный корпус, собранный с помощью филологов. Качество решения задачи оценивалось точностью, полнотой и F-мерой. Все нейросетевые языковые модели показали близкое качество F-меры около 86 % для большего по размеру корпуса Text Similarity и 50–56 % для собственного корпуса авторов. Совсем новым результатом оказалось успешное применение модели Mamba. Однако, самым интересным достижением стало применение векторов стилометрических характеристик текста, показавшее 80 % F-меры для авторского корпуса и одинаковое с нейросетевыми моделями качество решения задачи для другого корпуса.

Ключевые слова: обработка естественного языка; сходство текстов; классификация текстов; нейросетевые языковые модели; оценка открытых ответов учащихся; искусственный интеллект в образовании

ИНФОРМАЦИЯ ОБ АВТОРАХ

Лагутина, Надежда Станиславовна	ORCID iD: 0000-0002-6137-8643. E-mail: lagutinans@gmail.com Канд. физ.-мат. наук, доцент
Лагутина, Ксения Владимировна (автор для корреспонденции)	ORCID iD: 0000-0002-1742-3240. E-mail: lagutinakv@mail.ru Канд. тех. наук, доцент
Копнин, Владислав Николаевич	ORCID iD: 0009-0007-5451-775X. E-mail: vlad.kopnen@mail.ru Студент

Финансирование: ЯрГУ (проект VIP-016).

Для цитирования: N. S. Lagutina, K. V. Lagutina, and V. N. Kopnin, “Automatic determination of semantic similarity of student answers with the standard one using modern models”, *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 194–205, 2024. DOI: 10.18255/1818-1015-2024-2-194-205.

© Лагутина Н. С., Лагутина К. В., Копнин В. Н., 2024
Эта статья открытого доступа под лицензией CC BY license (https://creativecommons.org/licenses/by/4.0/).

Введение

Эффективный контроль знаний обучающихся в тестовых и письменных заданиях обязательно включает ответы в виде связного текста. Классический подход к проверке выполнения таких заданий преподавателем очень трудоёмкий, утомительный и субъективный. По мере развития методов обработки естественного языка и искусственного интеллекта осуществляются исследования в области автоматизации этого процесса [1].

Основной задачей проверки ответов учащихся является определение их сходства с эталонным ответом учителя. Автоматическое определение сходства текста — это, как правило, вычисление расстояния между двумя фрагментами текста, представляющего степень их близости в естественном языке [2]. Решение этой задачи можно рассматривать в двух аспектах: лексическое сходство и семантическое сходство [3].

Тексты схожи лексически, если они имеют одинаковые или однокоренные слова или последовательности слов. Автоматическое определение лексического сходства базируется на статистических методах анализа текста на символьном уровне и уровне слов. Однако в области ответов на вопросы этот аспект сильно сужает набор правильных вариантов и оставляет за его пределами фразы, отличающиеся от эталона структурой, синонимами, особенностями словоупотреблений.

Тексты обладают семантическим сходством, если они используются в одном и том же контексте, содержат одинаковую информацию и значение. Определение семантического сходства текстов представляет собой сложную задачу, так как требует учёта значений слов, контекстной информации, синтаксической структуры фраз. Огромный потенциал для её решения появляется с развитием больших языковых моделей (LLM): BERT, GPT, Mamba.

Исследователи предлагают большое количество вариантов определения сходства текстов, но часто ограничиваются решением в конкретной предметной области, строя узкоспециализированную модель текста [4]. Такой подход сильно ограничивает рамки применения разработанного решения и осложняет выбор модели в каждом конкретном случае. Поэтому, когда в ходе разработки программной системы построения языкового профиля обучающегося [5] возникла необходимость проверки открытых ответов, авторы работы поставили перед собой задачу исследовать, насколько качественно современные модели текста могут определять близость текстов на английском языке. В данной работе под сходством текстов понимается смысловая (семантическая) близость свободно сконструированного ответа учащегося с заведомо правильным, эталонным ответом преподавателя [6].

1. Обзор научных исследований

Первые методы определения сходства текста измеряли близость слов и предложений на основе оцифровки последовательности символов или строк, из которых они состоят. Полученные числовые векторы сравнивались математическими метриками расстояний: евклидово, косинусное, манхэттенское, расстояние Хэмминга [7]. Часто сходство вычислялось путем выделения самой длинной общей подстроки [8] или с использованием n-грамм [9].

Авторы работы [10] заметили, что данный подход мало учитывает семантику слов и добавили информацию о близости слов на основе WordNet. Эту же идею использовали исследователи сходства арабских текстов для оценки ответов учащихся [8]. Однако, выявление семантики текста с помощью тезаурусов и онтологий больше характерно для задач по поиску сходных данных в предметных областях [11].

Классическими методами выявления семантической информации считаются вычисление характеристик TF-IDF и латентный семантический анализ (latent semantic analysis, LSA) [11]. Молер и Михалча [12] исследовали LSA, но также предложили методы классификации путем извлечения лексических и синтаксических характеристик. Исследователи подготовили корпус Mohler's Dataset

эталонных ответов и ответов учащихся в области компьютерных наук на английском языке, размеченных по пятибальной системе.

Большая часть исследований в области оценки открытых ответов учащихся классифицирует тексты по оценкам или баллам [13]. В последнее десятилетие наиболее популярными моделями для решения этой задачи стали эмбеддинги, сначала, такие как Word2Vec и GloVe [14], потом построенные с помощью методов глубокого обучения [15]. Авторы работы [16] показали, что эмбеддинги BERT превосходят по качеству GloVe при определении сходства ответов с эталонными для корпуса Mohler's Dataset. Цамус и Филигхера [17] классифицировали набор данных SemEval-2013 из ответов учащихся, размеченных на три класса: правильный, неправильный и противоречивый. Они использовали различные модели BERT и получили F-меру от 67 % до 79 %.

Классификация ответов с точки зрения оценок важна для автоматизации контроля знаний, но непосредственное определение сходства текста с эталонным позволяет решать и другие задачи в области искусственного интеллекта в образовании (Artificial Intelligence in Education), например, организацию обратной связи или оценку профессиональных компетенций.

Современные решения по сравнению текстов используют идею векторного представления текстов и дальнейшего сравнения или классификации векторов. В работе [18] реализован гибридный подход с использованием эмбеддингов BERT в комбинации с моделью на основе нейронной сети Bi-LSTM. Авторы определяли сходство пар вопросов из набора данных Quora. Значение F-меры оказалось достаточно высоким: 91 %. Сравнение аннотаций статей оказалось намного сложнее. Витчард и др. [19] комбинировали эмбеддинги BERT и USE (Universal sentence encoder for English) и получили F-меру всего лишь 46 %. Исследователи отмечают высокую трудоёмкость применения больших языковых моделей и сильную зависимость от объёма и качества используемых корпусов текстов [20].

Сильное различие в качестве определения сходства текста побуждает исследователей строить комплексные и ансамблевые модели. Например, в работе [21] объединяются эмбеддинги FastText и n-граммы. Сравнение слов через ансамбль четырёх мер сходства осуществляют Хассан и др. [22]. Комбинацию эмбеддингов для инструмента визуализации поиска похожих текстов используется в работе [19]. Авторы исследования [23] сопоставляют ключевые слова и фразы с использованием ансамблей эмбеддингов BERT, GPT, параметров связей на основе WordNet, алгоритма сравнения строк Джаро-Винклера, и достигают максимального значения F-меры 74 %. Коллектив учёных [11] обращает внимание на важную роль синтаксической информации при сравнении текстов.

Следует заметить, что в области определения сходства текстов остаются малоисследованными классические лексические характеристики. Поэтому для своей работы авторы выбрали как языковые модели BERT, GPT, совсем новую Mamba, так и стилометрические параметры уровня символов, слов и структуры фраз и предложений.

2. Метод определения близости текстов

2.1. Основные этапы метода

Для определения близости текстов авторы применили одну из методологий, общепринятых в современных исследованиях. Это методология для работы с алгоритмами без обучения. Метод определения близости текстов включает в себя следующие этапы:

1. Векторное моделирование текста. Текст на естественном языке преобразуется в вектор чисел на основе стилометрической или предобученной языковой модели.
2. Вычисление близости векторов текстов. Для пар векторов текстов считается значение метрики близости.

Table 1. Parameters of text corpora **Таблица 1.** Параметры корпусов текстов

Корпус	Кол-во пар текстов	Близкие	Неблизкие	Уникальные	Ср. длина
Text Similarity	2142	1613	529	2597	5.18
Собственный корпус	1813	618	1195	683	6.09

3. Определение близости текстов. Для пар векторов на основе значения метрики близости дается ответ, являются ли тексты близкими (да/нет). Это осуществляется с помощью порогового значения: если метрика близости выше заданного порога, то тексты близкие, иначе нет.
4. Измерение качества метода. Ответы метода сравниваются с правильными с помощью стандартных метрик качества. Правильные ответы имеются в исходных корпусах текстов, где определены пары не только близких текстов, но и текстов, близкими не считающихся.

Близость текстов определяется для двух корпусов, состоящих из англоязычных текстов различных размеров и тематик. В следующих разделах подробно описаны корпуса, модели текстов, метрики близости и метрики качества, применяемые в методе.

2.2. Корпуса текстов

Для экспериментальной апробации векторных моделей были выбраны два корпуса англоязычных текстов: один из открытых источников и один собственный корпус.

Корпус Text Similarity[1] содержит 2142 пары текстов. Тексты представляют собой короткие фрагменты по тематике акций, взятые из источника Quovo. В корпусе содержится 1613 пар близких текстов и 529 пар неблизких.

Собственный корпус был собран авторами и их коллегами-филологами. Он содержит 1813 пар коротких текстов, которые являются ответами студентов на вопросы или эталонными ответами с точки зрения преподавателя. Вопросы задавались преподавателями-филологами с целью узнать уровень английского языка студентов по шкале CEFR. В корпусе содержится 618 пар близких текстов и 1195 пар неблизких.

В таблице 1 приведены сравнительные параметры корпусов: количество пар текстов, количество пар близких и неблизких текстов, число уникальных текстов и средняя длина текстов в словах.

2.3. Векторные модели

Каждый текст из корпуса моделируется как вектор чисел. Используемые модели делятся на две категории: стилометрические и предобученные языковые.

Стилометрические модели подразумевают подсчёт стандартных статистических характеристик стиля текста трёх уровней.

1. Уровень символов. Он включает в себя следующие характеристики:
   - количество букв;
   - количество символов;
   - количество предложений;
   - средняя длина предложения в символах;
   - частоты встречаемости букв;
   - частоты встречаемости знаков препинания.
2. Уровень слов. Он включает в себя следующие характеристики:
   - n-граммы слов, $n = 1, 2, 3$. Среди всех n-грамм в корпусе выбирается 40 самых частых для каждого значения n, а для отдельных текстов считается частота встречаемости отдельных n-грамм относительно отобранных;
   - количество слов;
   - средняя длина предложения в словах;

[1] https://www.kaggle.com/datasets/rishisankineni/text-similarity

- средняя длина предложения в символах.

3. Уровень структуры. Текст разделяется на слова, для слов вычисляются части речи, т. е. текст представляется как последовательность частей речи. Далее считаются n-граммы частей речи, $n = 1, 2, 3, 4$. Среди всех n-грамм в корпусе выбирается 40 самых частых для каждого знначения n, а для отдельных текстов считается частота встречаемости отдельных n-грамм относительно отобранных.

Описанные характеристики уже успешно показали себя в обработке текстов в предыдущих работах авторов [24].

Вычисление характеристик всех уровней происходит с использованием Python-библиотеки Stanza [25]: она разделяет текст на слова и определяет части речи. Каждый из уровней считается как одна модель текста, также уровни комбинируются попарно или в тройку конкатенацией векторов характеристик, результат считается новой моделью. Элементы в векторах характеристик размещены по заданным позициям, каждая позиция соответствует отдельной характеристике, чтобы при сопоставлении векторов числа на одних и тех же позициях соответствовали друг другу по смыслу.

Вторая категория моделей — это предобученные языковые модели, созданные на основе современных нейросетевых архитектур.

- BERT для английского языка и его вариации:
  - bert-base-cased — базовая версия BERT для английского языка, учитывающая регистр;
  - bert-base-uncased — базовая версия BERT для английского языка, не учитывающая регистр;
  - bert-large-cased — увеличенная версия BERT для английского языка, учитывающая регистр;
  - bert-large-uncased — увеличенная версия BERT для английского языка, не учитывающая регистр;
  - bert-large-cased-whole-word-masking — увеличенная версия BERT для английского языка, учитывающая регистр и маскирующая сразу все токены, соответствующие слову;
  - bert-large-uncased-whole-word-masking — увеличенная версия BERT для английского языка, не учитывающая регистр и маскирующая сразу все токены, соответствующие слову.
- GPT для английского языка и его вариации:
  - openai-community/gpt2 — базовая версия GPT-2 для английского языка с 124 миллионами параметров;
  - openai-community/gpt2-medium — средняя версия GPT-2 для английского языка с 355 миллионами параметров;
  - openai-community/gpt2-large — большая версия GPT-2 для английского языка с 774 миллионами параметров;
  - sembeddings/model_gpt_trained — версия GPT-2 для английского языка, предназначенная для задач по анализу значения текстов, в том числе близости текстов;
  - sembeddings/gptops_finetuned_mpnet_gpu_v1 — версия GPT-2 для английского языка, предназначенная для задач кластеризации и семантического поиска.
- Mamba для английского языка и её вариации:
  - Q-bert/Mamba-130M — базовая версия Mamba для английского языка с 130 миллионами параметров;
  - Q-bert/Mamba-1B — базовая версия Mamba для английского языка с миллиардом параметров.

Все языковые модели являются предобученными. Они взяты из открытого ресурса Hugging Face[2]. Модели сопровождаются собственными токенизаторами, которые преобразуют исходный

[2] https://huggingface.co/models

текст на естественном языке в последовательность токенов, подающихся на вход модели. На выходе для каждого текста формируется эмбеддинг — вектор действительных чисел фиксированного размера.

2.4. Метрики близости

Пары текстов, представленные как векторы чисел одинаковой длины, сравниваются с помощью метрик близости:

- косинусное сходство — косинус угла между векторами, принимает значения от −1 до 1;
- коэффициент корреляции Пирсона — метрика линейной зависимости между двумя величинами, принимает значения от −1 до 1;
- метрика Чебышева — максимум модуля разности компонент векторов;
- евклидово расстояние — квадратный корень из суммы квадратов разностей соответствующих компонент векторов;
- метрика Минковского — метрика-обобщение евклидова и манхэттенского расстояний.

Таким образом для корпуса текстов считается набор чисел по одной из указанных метрик, в итоге числа оказываются в одном диапазоне. Каждой паре векторов текстов сопоставляется число. Чтобы вынести вердикт, близки тексты или нет, выбирается порог значения метрики. Если значение метрики близости для пары текстов оказывается выше порога, то тексты считаются близкими, иначе — нет. Порог значения метрики является гиперпараметром описываемого метода и подбирается путём перебора.

2.5. Метрики качества

Метод в результате подсчёта метрики близости и применения порога находит для корпуса из пар текстов ответы да/нет о близости пар. Полученные ответы сравниваются с правильными с помощью следующих метрик качества:

- точность — доля пар текстов, действительно близких, относительно всех пар текстов которые метод отметил как близкие;
- полнота — доля найденных методом пар близких текстов, принадлежащих классу относительно всех пар близких текстов;
- F-мера — среднее гармоническое точности и полноты.

Метрики являются стандартными для классификационных задач.

3. Эксперименты по определению близости текстов

Метод измерения близости текстов применялся для обоих корпусов. С каждым корпусом проводились отдельные эксперименты: строились вектора текстов с помощью всех моделей, затем для пары модель+метрика близости методом перебора значений выбирался порог таким образом, чтобы результат работы метода показывал лучшую F-меру.

Оба корпуса содержат в себе как пары близких, так и пары неблизких текстов. Поэтому результаты классификации для всех моделей получились достаточно показательными. Лучшие характеристики качества приведены в таблицах 2 и 3, где для каждой модели выбирался результат с наибольшей F-мерой.

В первом столбце указывается модель по техническому имени с Hugging Face или по названию уровня стилометрических характеристик/комбинаций. Во втором столбце указываются метрики близости, для которых была достигнута наибольшая F-мера. В большинстве случаев все метрики близости позволили достичь одного и того же максимума по F-мере, для некоторых моделей такими метриками оказались косинусное сходство (cos), коэффициент корреляции Пирсона (Pearson) или метрика Чебышева (Chebyshev). В остальных столбцах приведены значения метрик качества в процентах.

Table 2. Classification results for the Text Similarity corpus

Таблица 2. Результаты классификации для корпуса Text Similarity

Модель	Метрика близости	F1-мера	Точность	Полнота
bert-base-cased	cos, Pearson	**86.01**	**76.21**	**98.70**
bert-base-uncased	все	85.91	75.30	100.00
bert-large-cased	все	85.91	75.30	100.00
bert-large-cased-whole-word-masking	все	85.91	75.30	100.00
bert-large-uncased	cos, Pearson	86.07	76.01	99.19
bert-large-uncased-whole-word-masking	все	85.91	75.30	100.00
openai-community/gpt2	все	85.91	75.30	100.00
openai-community/gpt2-medium	все	85.91	75.30	100.00
openai-community/gpt2-large	cos, Pearson	86.52	77.02	98.70
sembeddings/model_gpt_trained	cos, Pearson	87.65	80.87	95.66
sembeddings/gptops_finetuned_mpnet_gpu_v1	cos, Pearson	**87.89**	**79.91**	**97.64**
Q-bert/Mamba-130M	Pearson	**86.37**	**76.71**	**98.82**
Q-bert/Mamba-1B	Pearson	86.42	76.90	98.64
Символы	все	85.91	75.30	100.00
Структура	Pearson	86.24	76.96	98.07
Слова	Pearson	**86.33**	**76.52**	**99.01**
Символы+Структура	все	85.91	75.30	100.00
Символы+Слова	Chebyshev	85.93	75.36	99.94
Структура+Слова	все	85.91	75.30	100.00
Символы+Структура+Слова	все	85.91	75.30	100.00

На корпусе Text Similarity (2) все модели добились примерно одинакового качества результата: 85–87 % F-меры. Среди BERT-моделей можно признать лучшими bert-base-cased и bert-base-uncased, так как они при меньшем размере показали те же результаты, что и вариации bert-large. Эмбеддинги Mamba сработали аналогично. Для GPT-моделей можно отметить, что эмбеддинги от sembeddings, ориентированные именно на задачи анализа семантики и поиск близких текстов, не дали прироста качества. Стилометрические модели показали то же качество результата, что и эмбеддинги. Комбинации пар и тройки уровней стилометрических характеристик не увеличили точность, полноту и F-меру.

Для собственного корпуса (3) приведены модели, показавшие лучшее качество. Эмбеддинги показали низкое качество, среди них можно отметить только sembeddings/gptops_finetuned_mpnet_gpu_v1, превзошедшую остальные. Зато стилометрические характеристики позволили выбрать близкие тексты с качеством около 80 %. Это ниже на 5 п. п., чем на корпусе Text Similarity, зато существенно выше, чем результаты моделей на основе эмбеддингов. При этом все стилометрические характеристики показали примерно один и тот же результат, комбинация качество определения близости текстов не повысила.

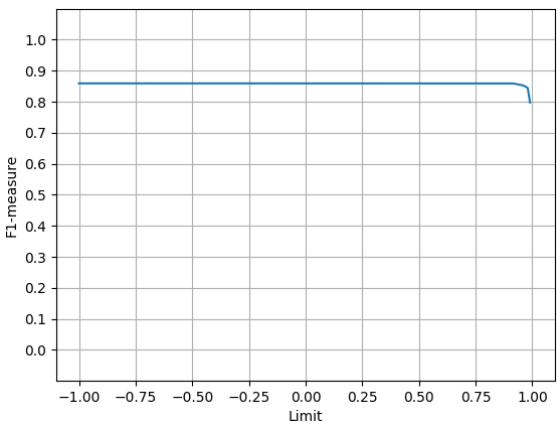
Достигнутые значения F-меры для стилометрических моделей не сильно зависят от порога, по которому происходит определение близости текстов. На рис. 1 для корпуса Text Similarity взяты стилометрические характеристики и порог от −1 до 1. Косинусная метрика на всём диапазоне позволяет достичь 85.91 % F-меры, и такая стабильная ситуация случилась для большинства пар стилометрическая модель-метрика близости. В качестве исключения можно привести косинусную метрику в комбинации с уровнем структуры (1b).

Для моделей на основе эмбеддингов значение порога имеет важную роль для обоих корпусов. На рис. 2a до порога в 0.5 показываются хорошие результаты, и при значении около 0.5 находится

Table 3. Classification results for the custom corpus

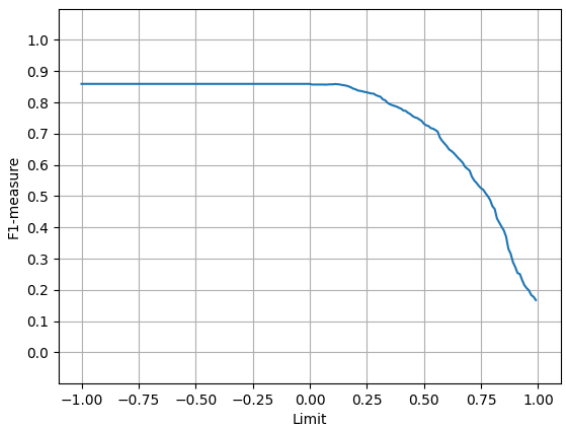
Таблица 3. Результаты классификации для собственного корпуса

Модель	Метрика близости	F1-мера	Точность	Полнота
bert-base-uncased	cos, Pearson	54.33	40.69	81.72
bert-large-uncased	cos, Pearson	51.80	36.64	88.35
bert-large-uncased-whole-word-masking	Chebyshev	51.53	36.60	87.06
openai-community/gpt2	cos, Pearson	50.86	34.11	100.00
openai-community/gpt2-large	cos	51.61	37.96	80.58
sembeddings/gptops_finetuned_mpnet_gpu_v1	cos, Pearson	**56.37**	**46.05**	**72.65**
Q-bert/Mamba-130M	Pearson	50.86	34.11	100.00
Q-bert/Mamba-1B	cos	51.75	35.45	95.79
Символы	все	**80.30**	**67.09**	**100.00**
Структура	все	80.25	67.02	100.00
Слова	все	80.25	67.02	100.00
Символы+Структура+Слова	все	80.28	67.05	100.00



a)

Fig. 1. Dependence of F=measure on the threshold for the Text Similarity corpus and stylometric features a) of all levels b) of the structure level with the cosine metric



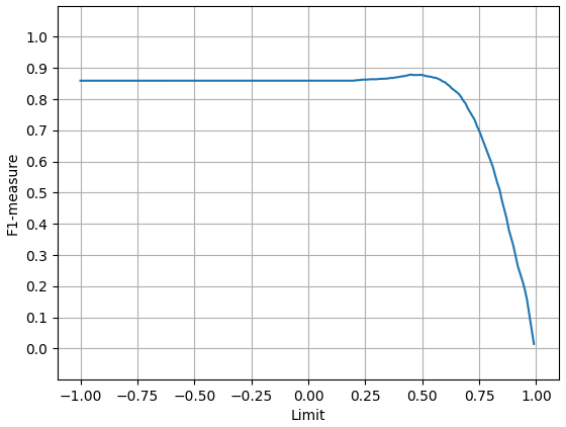
b)

Рис. 1. Зависимость F-меры от порога для корпуса Text Similarity и стилометрических характеристик a) всех уровней b) уровня структуры с косинусной метрикой

наибольшая F-мера. То же происходит и для второго корпуса, как показано на рис. 2b: лучшая F-мера достигается только при значении порога около 0.5. Стоит отметить, что порог оказывается одинаковым для различных метрик близости.

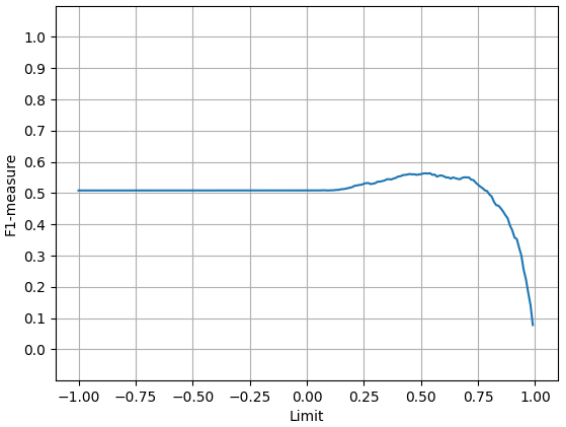
Таким образом, для собственного корпуса с короткими текстами на специфическую тематику лучше оказалось использовать стилометрические модели для определения близости. Нейросетевые модели показали для него слишком низкий результат: 51–56 % F-меры. Вероятно, такое качество работы моделей на основе эмбеддингов связано с размерами корпусов. Обычно успешные эксперименты с эмбеддингами получаются на корпусах из нескольких тысяч текстов, в то время как у собственного корпуса всего несколько сотен близких текстов.

Корпус Text Similarity также является тематическим, но в нём существенно больше близких текстов. Их поиск оказался успешнее для нейростетевых моделей: 87.89 % F-меры при максимуме в 56.37 % для этих же моделей при работе с собственным корпусом. Результат достаточно высокий и стабильный, так как все модели показали близкие результаты качества. Однако порог в 88 % F-меры не преодолела ни одна модель.



a)

Fig. 2. Dependence of F-measure on the threshold for the sembeddings/gptops_finetuned_mpnet_gpu_v1 model a) on the Text Similarity corpus with the cosine metric b) on the custom corpus with the Pearson correlation coefficient



b)

Рис. 2. Зависимость F-меры от порога для модели sembeddings/gptops_finetuned_mpnet_gpu_v1 a) на корпусе Text Similarity с косинусной метрикой b) на собственном корпусе с коэффициентом корреляции Пирсона

Следует отметить, что все модели показали высокое значение полноты определения близости текстов: 80–100 % для собственного корпуса и 98–100 % для корпуса Text Similarity. Это означает, что модели позволяют найти все близкие тексты, но при этом не справляются с некоторыми случаями отсечения неблизких текстов. Анализ ошибок и поиск усовершенствованных лингвистических моделей может повысить точность решения задачи.

Заключение

Результаты работы показывают, что задача сравнения ответа учащегося с эталонным достаточно хорошо решается методами определения сходства числовых характеристических векторов. Языковые нейросетевые модели BERT и GPT показывают лучшие результаты на корпусе большего объёма. Новизной исследования является использование модели Mamba. Качество решения задачи с её помощью практически совпадает с другими нейросетевыми моделями для корпуса Text Similarity и немного превосходит для собственного корпуса авторов. Кроме того, результаты экспериментов показывает успешность этой новой языковой модели.

Интересным результатом является высокая F-мера классификации близких текстов с помощью стилометрических характеристик. Особенно высокий результат получился для авторского корпуса. Скорее всего это обусловлено лексическими особенностями этого корпуса, так как классические нейросетевые модели показали существенно меньший результат, в то время как на другом корпусе качество практически совпадает. Этот факт открывает большие перспективы для более широкого исследования возможности применения стилометрии в области определения сходства текстов.

References

[1] R. Gao, H. E. Merzdorf, S. Anwar, M. C. Hipwell, and A. Srinivasa, “Automatic assessment of text-based responses in post-secondary education: A systematic review”, *Computers and Education: Artificial Intelligence*, vol. 6, p. 100 206, 2024. DOI: 10.1016/j.caeai.2024.100206.

[2] J. Wang and Y. Dong, “Measurement of text similarity: A survey”, *Information*, vol. 11, no. 9, p. 421, 2020. DOI: 10.3390/info11090421.

[3] A. Rozeva and S. Zerkova, “Assessing semantic similarity of texts–methods and algorithms”, *AIP Conference Proceedings*, vol. 1910, no. 1, p. 060 012, 2017. DOI: 10.1063/1.5014006.

[4] P. D. Wibisono, A. Asad, and A. Chintan, "Short text similarity measurement methods: A review", *Soft Computing*, vol. 25, pp. 4699–4723, 2021. DOI: 10.1007/s00500-020-05479-2.

[5] N. S. Lagutina, M. V. Tihomirov, and N. K. Mastakova, "Algoritm avtomaticheskogo postroeniya yazykovogo profilya uchashchegosya", *Zametki po informatike i matematike*, no. 15, pp. 58–65, 2023, in Russian.

[6] O. B. Mishunin, A. P. Savinov, and D. I. Firstov, "Sostoyanie i uroven' razrabotok sistem avtomaticheskoj ocenki svobodnyh otvetov na estestvennom yazyke", *Modern high technologies*, no. 1, pp. 38–44, 2016, in Russian.

[7] L. Zahrotun, "Comparison Jaccard similarity, cosine similarity and combined both of the data clustering with Shared Nearest Neighbor method", *Computer Engineering and Applications Journal*, vol. 5, no. 1, pp. 11–18, 2016. DOI: 10.18495/comengapp.v5i1.160.

[8] H. A. Abdeljaber, "Automatic Arabic short answers scoring using longest common subsequence and Arabic WordNet", *IEEE Access*, vol. 9, pp. 76 433–76 445, 2021. DOI: 10.1109/ACCESS.2021.3082408.

[9] S. Sultana and I. Biskri, "Identifying similar sentences by using n-grams of characters", in *Recent Trends and Future Technology in Applied Intelligence: Proceedings of 31st International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*, Springer, 2018, pp. 833–843. DOI: 10.1007/978-3-319-92058-0_80.

[10] S. Vij, D. Tayal, and A. Jain, "A machine learning approach for automated evaluation of short answers using text similarity based on WordNet graphs", *Wireless Personal Communications*, vol. 111, pp. 1271–1282, 2020. DOI: 10.1007/s11277-019-06913-x.

[11] Y. Zhou, C. Li, G. Huang, Q. Guo, H. Li, and X. Wei, "A short-text similarity model combining semantic and syntactic information", *Electronics*, vol. 12, no. 14, p. 3126, 2023. DOI: 10.3390/electronics12143126.

[12] M. Mohler and R. Mihalcea, "Text-to-text semantic similarity for automatic short answer grading", in *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, 2009, pp. 567–575.

[13] M. Han, X. Zhang, X. Yuan, J. Jiang, W. Yun, and C. Gao, "A survey on the techniques, applications, and performance of short text semantic similarity", *Concurrency and Computation: Practice and Experience*, vol. 33, no. 5, e5971, 2021. DOI: 10.1002/cpe.5971.

[14] S. Roy, S. Dandapat, A. Nagesh, and Y. Narahari, "Wisdom of students: A consistent automatic short answer grading technique", in *Proceedings of the 13th International Conference on Natural Language Processing*, 2016, pp. 178–187.

[15] A. Ahmed, A. Joorabchi, and M. J. Hayes, "On deep learning approaches to automated assessment: Strategies for short answer grading", in *Proceedings of the 14th International Conference on Computer Supported Education*, vol. 2, 2022, pp. 85–94. DOI: 10.5220/0011082100003182.

[16] A. Ahmed, A. Joorabchi, and M. J. Hayes, "On the application of sentence transformers to automatic short answer grading in blended assessment", in *Proceedings of the 33rd Irish Signals and Systems Conference (ISSC)*, IEEE, 2022, pp. 1–6. DOI: 10.1109/ISSC55427.2022.9826194.

[17] L. Camus and A. Filighera, "Investigating transformers for automatic short answer grading", in *Proceedings of the 21st International Conference Artificial Intelligence in Education, Part II 21*, Springer, 2020, pp. 43–48. DOI: 10.1007/978-3-030-52240-7_8.

[18] D. Viji and S. Revathy, "A hybrid approach of weighted fine-tuned BERT extraction with deep Siamese Bi-LSTM model for semantic text similarity identification", *Multimedia Tools and Applications*, vol. 81, no. 5, pp. 6131–6157, 2022. DOI: 10.1007/s11042-021-11771-6.

[19] D. Witschard, I. Jusufi, R. M. Martins, K. Kucher, and A. Kerren, "Interactive optimization of embedding-based text similarity calculations", *Information Visualization*, vol. 21, no. 4, pp. 335–353, 2022. DOI: 10.1177/14738716221114372.

[20] T. Brown *et al.*, "Language models are few-shot learners", *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[21] D. Shashavali *et al.*, "Sentence similarity techniques for short vs variable length text using word embeddings", *Computación y Sistemas*, vol. 23, no. 3, pp. 999–1004, 2019. DOI: 10.13053/cys-23-3-3273.

[22] B. Hassan, S. E. Abdelrahman, R. Bahgat, and I. Farag, "UESTS: An unsupervised ensemble semantic textual similarity method", *IEEE Access*, vol. 7, pp. 85 462–85 482, 2019. DOI: 10.1109/ACCESS.2019.2925006.

[23] I. Gagliardi and M. T. Artese, "Ensemble-based short text similarity: An easy approach for multilingual datasets using transformers and wordnet in real-world scenarios", *Big Data and Cognitive Computing*, vol. 7, no. 4, p. 158, 2023. DOI: 10.3390/bdcc7040158.

[24] N. Lagutina, K. Lagutina, A. Brederman, and N. Kasatkina, "Text classification by CEFR levels using machine learning methods and BERT language model", *Modeling and Analysis of Information Systems*, vol. 30, no. 3, pp. 202–213, 2023. DOI: 10.18255/1818-1015-2023-3-202-213.

[25] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020, pp. 101–108. DOI: 10.18653/v1/2020.acl-demos.14.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 2, 2024
journal homepage: www.mais-journal.ru

ARTIFICIAL INTELLIGENCE

Keywords, Morpheme Parsing and Syntactic Trees: Features for Text Complexity Assessment

D. A. Morozov[1], I. A. Smal[1], T. A. Garipov[1], A. V. Glazkova[2]

DOI: 10.18255/1818-1015-2024-2-206-220

[1]Novosibirsk National Research State University, Novosibirsk, Russia
[2]University of Tyumen, Tyumen, Russia

MSC2020: 68T50
Research article
Full text in Russian

Received February 27, 2024
Revised March 29, 2024
Accepted May 8, 2024

The text complexity assessment is an applied problem of current interest with potential application in the drafting of legal documents, editing textbooks, and selecting books for extracurricular reading. The methods for generating a feature vector when automatically assessing the text complexity are quite diverse. Early approaches relied on easily calculable quantities, such as the average length of a sentence or the average number of syllables per word. With the development of natural language processing algorithms, the space of used features is expanding. In this work, we examined three groups of features: 1) automatically generated keywords, 2) information about the features of morphemic word parsing, and 3) information about the diversity, branching, and depth of syntactic trees. The RuTermExtract algorithm was utilized to generate keywords, a convolutional neural network model was used to generate morphemic parses, and the Stanza model, trained on the SynTagRus corpus, was used to generate syntax trees. We conducted a comparison using four different machine learning algorithms and four annotated Russian-language text corpora. The corpora used differ both in the domain and markup paradigm, due to which the results obtained more objectively reflect the real relationship between the characteristics and the text complexity. The use of keywords performed worse on average than the use of topic markers obtained using latent Dirichlet allocation. In most situations, morphemic characteristics turned out to be more effective than previously described methods for assessing the lexical complexity of a text: the frequency of words and the occurrence of word-formation patterns. The use of an extensive set of syntactic features allowed, in most cases, to improve the quality of work of neural network models in comparison with the previously described set.

Keywords: text complexity; keyword generation; morpheme parsing generation; syntax trees

INFORMATION ABOUT THE AUTHORS

Morozov, Dmitry A. (corresponding author)	ORCID iD: 0000-0003-4464-1355. E-mail: morozowdm@gmail.com Junior Researcher
Smal, Ivan A.	ORCID iD: 0009-0005-1082-0584. E-mail: vanasmal@mail.ru Graduate Student
Garipov, Timur A.	ORCID iD: 0009-0008-4527-2268. E-mail: garipov154@yandex.ru Master's Student
Glazkova, Anna V.	ORCID iD: 0000-0001-8409-6457. E-mail: a.v.glazkova@utmn.ru Associate Professor, PhD

For citation: D. A. Morozov, I. A. Smal, T. A. Garipov, and A. V. Glazkova, "Keywords, morpheme parsing and syntactic trees: features for text complexity assessment", *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 206–220, 2024. DOI: 10.18255/1818-1015-2024-2-206-220.

© Morozov D. A., Smal I. A., Garipov T. A., Glazkova A. V., 2024
This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 2, 2024
сайт журнала: www.mais-journal.ru

Ключевые слова, морфемные разборы и синтаксические деревья в задаче оценки сложности текста

Д. А. Морозов[1], И. А. Смаль[1], Т. А. Гарипов[1], А. В. Глазкова[2] DOI: 10.18255/1818-1015-2024-2-206-220

[1]Новосибирский национальный исследовательский государственный университет, Новосибирск, Россия
[2]Тюменский государственный университет, Тюмень, Россия

УДК 004.912
Научная статья
Полный текст на русском языке

Получена 27 февраля 2024 г.
После доработки 29 марта 2024 г.
Принята к публикации 8 мая 2024 г.

Задача оценки сложности текста является актуальной прикладной задачей с потенциальным применением при составлении юридических документов, редактуре учебников и подборе книг для внеклассного чтения. Способы формирования признакового описания при автоматической оценке сложности текста достаточно разнообразны. Ранние подходы опирались на легко вычислимые величины, такие как средняя длина предложения или среднее число слогов в слове. С развитием алгоритмов обработки естественного языка расширяется и пространство используемых признаков. В рамках настоящей работы мы исследовали три группы признаков: 1) автоматически генерируемые ключевые слова, 2) сведения об особенностях морфемного разбора слов и 3) информацию о разнообразии, разветвлённости и глубине синтаксических деревьев. Для генерации ключевых слов использован алгоритм RuTermExtract, для генерации морфемных разборов — свёрточная нейросетевая модель, для генерации синтаксических деревьев — модель Stanza, обученная на корпусе SynTagRus. Мы провели сравнение на материале четырёх различных моделей машинного обучения и четырёх аннотированных русскоязычных корпусов текстов. Использованные корпусы различаются как по домену, так и по парадигме разметки, благодаря чему полученные результаты объективнее отражают реальную связь характеристик и сложности текста. Использование ключевые слова показало в среднем результат хуже, чем использование тематических маркеров, получаемых при помощи латентного размещения Дирихле. Морфемные характеристики оказались в большинстве ситуаций эффективнее ранее описанных способов оценки лексической сложности текста: учёта частотности слов и встречаемости словообразовательных паттернов. Использование обширного набора синтаксических признаков позволило в большинстве случаев улучшить качество работы нейросетевых моделей в сравнении с ранее описанным набором.

Ключевые слова: сложность текста; генерация ключевых слов; генерация морфемных разборов; синтаксические деревья

ИНФОРМАЦИЯ ОБ АВТОРАХ

Морозов, Дмитрий Алексеевич (автор для корреспонденции)	ORCID iD: 0000-0003-4464-1355. E-mail: morozowdm@gmail.com Младший научный сотрудник
Смаль, Иван Андреевич	ORCID iD: 0009-0005-1082-0584. E-mail: vanasmal@mail.ru Аспирант
Гарипов, Тимур Александрович	ORCID iD: 0009-0008-4527-2268. E-mail: garipov154@yandex.ru Магистрант
Глазкова, Анна Валерьевна	ORCID iD: 0000-0001-8409-6457. E-mail: a.v.glazkova@utmn.ru Доцент, канд. тех. наук

Для цитирования: D. A. Morozov, I. A. Smal, T. A. Garipov, and A. V. Glazkova, “Keywords, morpheme parsing and syntactic trees: features for text complexity assessment”, *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 206–220, 2024. DOI: 10.18255/1818-1015-2024-2-206-220.

© Морозов Д. А., Смаль И. А., Гарипов Т. А., Глазкова А. В., 2024
Эта статья открытого доступа под лицензией CC BY license (https://creativecommons.org/licenses/by/4.0/).

Введение

Сложность текста — величина, оценивать которую требуется при подготовке различных учебных пособий, инструкций, типовых договоров. При этом как экспертная, так и эмпирическая оценка этой величины требует значительных затрат времени и, зачастую, финансов. В связи с этим востребованными являются алгоритмы, позволяющие оценить сложность текста, исходя из вычисляемых (желательно автоматически) характеристик. Первые подобные алгоритмы были предложены в середине двадцатого века. Обычно они являлись линейными регрессиями, опиравшимися на простые статистические признаки, такие как средняя длина слов и предложений. К таким алгоритмам можно отнести индексы удобочитаемости, в частности, индекс Флеша [1], индекс Дейла — Чалла [2], Automated Readability Index [3]. При разработке и использовании алгоритмов оценки сложность текста обычно ассоциируют с возрастом предполагаемого читателя, а оценка проводится в терминах классов школы. При таком подходе читательский опыт носителей языка одного возраста считается слабо различающимся, а шкала ограничивается сверху старшими классами школы или младшими курсами университетов. Это позволяет использовать в качестве источника текстов и разметки различные учебные материалы и списки литературы.

Благодаря появлению всё более производительных компьютеров и развитию методов обработки естественного языка стало возможным улучшение качества оценки с использованием более продвинутых моделей и более сложных, высокоабстрактных и лингвистически мотивированных характеристик текста. Упоминаемые в различных работах признаки чрезвычайно разнообразны. В исследованиях, посвящённых оценке сложности русскоязычных текстов, помимо традиционно используемых признаков, таких как средние длины слов и предложений, встречается использование оценок лексического разнообразия [4—6], морфологических [5—9], синтаксических [6, 8—10], лексических [6—9, 11, 12], семантических [9], тематических [6] признаков.

Для оценки лексической сложности текста обычно используют характеристики, связанные с частотностью слов [6, 9, 12, 13]. При этом ряд исследований показал, что частотность, вычисленная на основании какого-либо корпуса, достаточно часто отличается от реальной распространённости и знакомости слов [11, 13]. В то же время, алгоритмы оценки именно сложности слов отсутствуют, а само понятие семантической сложности слова недостаточно формализовано. В этой ситуации способом оценить лексическую сложность текста целиком может стать вычисление различных особенностей морфемного разбора слов[1]. Так, в работе [9] используется информация о встречаемости различных ассоциированных со сложностью морфемных конструкций, а именно доли лемм вида **ция*, **ние*, **вие*, **тие*, **ист*, **изм*, **ура*, **ище*, **ство*, **ость*, **овка*, **атор*, **итор*, **тель*, **льный*, **овать*. Такой подход вычислительно проще полноценного анализа морфемных разборов, однако остаётся непонятным, насколько вычисленные таким образом признаки значимы.

В исследованиях, затрагивающих использование синтаксических признаков в рамках оценки сложности текста на русском языке [6, 8—10], набор задействованных признаков достаточно узок. Чаще всего авторы используют глубину синтаксического дерева, долю рёбер того или иного типа и расстояние между вершиной и её непосредственным предком, выбор используемых признаков при этом как правило интуитивен. При этом в работе [10] синтаксические признаки оказались среди наиболее значимых как в отдельных корпусах, так и в целом. В работе [6] использование синтаксических признаков позволило улучшить качество модели для большинства корпусов и алгоритмов машинного обучения. Таким образом, представляется интересным изучение более обширных наборов синтаксических признаков, а также поиск наиболее значимых из них.

[1]Морфемным разбором слова называют разбиение слова на несколько непересекающихся подстрок-морфем (возможно, с добавлением дополнительных символов), которые делятся на различные классы согласно своей природе: корни и разнообразные аффиксы (приставки, суффиксы, окончания и т. д.).

Информация о тематике текста относительно редко применяется при оценке сложности текста. В то же время, в работе [6] использование латентного размещения Дирихле (LDA) [14] для формирования тематических кластеров позволило значительно (на 3–10 %) повысить качество работы модели. Это позволяет предположить, что другие механизмы извлечения терминов, описывающих тематику текста, могли бы быть использованы в качестве признаков. Одним из способов кратко описать тематику текста является использование ключевых слов. Ключевые слова представляют собой набор слов и словосочетаний, в совокупности соответствующих высокоуровневому описанию содержания текста. Они широко применяются в научной среде и СМИ, но в общем случае, особенно для длинных художественных текстов, этот инструмент, по-видимому, либо мало применим, либо неприменим вовсе из-за разнообразия обсуждаемых в тексте тем. Однако в случае с оценкой сложности текста многие из исследований проводятся на корпусах, состоящих из небольших отрывков текста. Так, в работе [10] в том числе используются фрагменты по 200 предложений, в работе [6] — по 70 предложений, в работе [9] — 11 ± 7 предложений. При такой длине контекста, сравнимой с длиной аннотаций к научным статьям или газетных текстов[2], генерируемые ключевые слова, вероятно, могут быть использованы в качестве признаков тематического моделирования при оценке сложности текста и сравнены с ранее исследованным LDA.

Итак, целью настоящей работы является анализ возможных расширений признакового описания в задаче автоматической оценки сложности текста в терминах возраста его предполагаемого читателя. Мы рассмотрели три направления таких расширений: добавление признаков, связанных с лексической сложностью, с синтаксической сложностью и с тематикой текста. Для каждого из направлений мы сравнили влияние на качество оценки ранее упоминавшихся признаков с вновь предлагаемыми.

Статья организована следующим образом. В разделе 1 описаны использованные при сравнении значимости корпусы текстов и модели машинного обучения. Раздел 2 состоит из трёх подразделов, в каждом из которых описывается методика извлечения отдельного типа признаков. Подраздел 2.1 содержит описание вычисления ключевых слов и построения на их базе признаков. В подразделе 2.2 описано построение модели морфемных разборов и её применение для извлечения характеристик текста. Наконец, подраздел 2.3 содержит описание экспериментов, направленных на поиск наиболее значимых особенностей синтаксических деревьев. В разделе 3 приводятся и обсуждаются полученные результаты. Выводы к работе представлены в заключении.

1. Модели и данные

Для того, чтобы иметь возможность более прозрачно сравнить вновь полученные результаты с имевшимися ранее, для проверки признаков мы выбрали наборы данных из числа ранее упоминавшихся в контексте оценки сложности текста. Мы использовали четыре корпуса русскоязычных текстов: корпус ознакомительных отрывков (Fic), корпус учебных текстов (TB), корпус рекомендованной литературы (RL) и корпус читаемых книг (BR). Во всех этих корпусах в качестве разметки сложности выступает разметка возраста читателя. Корпус Fic был представлен в работе [15] и состоит из начальных фрагментов различных произведений, открыто доступных для ознакомления в онлайн-библиотеках. Для этого корпуса использовалось два вида имеющейся в нём разметки — а) двухклассовая разметка на тексты для детей и для взрослых (FicChAd) и б) четырёхклассовая разметка, проведённая в соответствии с «Возрастной классификацией информационной продукции в России»[3] (FicRARS). Корпус TB, описанный в том числе в работе [8], состоит из текстов школьных учебников по обществознанию, разбитых на отдельные предложения. Так как корпус TB достаточно мал (менее 50000 предложений), мы объединили тексты из нескольких классов в категории:

[2]Так, средняя длина текстов в Газетном корпусе НКРЯ составляет 20 предложений.

[3]Введена согласно Федеральному закону РФ №436-ФЗ от 2010-12-23 «О защите детей от информации, причиняющей вред их здоровью и развитию».

Table 1. The characteristics of the datasets

Таблица 1. Характеристика использованных корпусов текстов

Характеристика	Fic		TB	RL	BR
Всего фрагментов	58184		449	9230	5795
Всего словоупотреблений	26252666		335436	4888290	2897003
Всего различных слов	304731		18661	103875	55577
Средняя длина текста (в словах)	918.64		747.07	1053.28	984.75
Средняя длина предложения (в словах)	13.12		10.67	14.95	13.92
Количество классов	2	4	3	3	5
Минимальный размер класса (в фрагментах)	28765	7014	98	1499	805
Максимальный размер класса (в фрагментах)	29419	21124	200	5390	1428

5–7, 8–9 и 10–11 классы. Корпусы RL и BR были представлены в работе [6]. Первый из них состоит из текстов произведений, рекомендованных Министерством просвещения РФ для внеклассного чтения, второй — из произведений, которые респонденты школьного возраста чаще всего называли последними из прочитанных. Таким образом, для рассмотрения были выбраны корпусы с различной парадигмой разметки предполагаемого возраста читателя с текстами из различных доменов. В ходе предобработки тексты каждого из корпусов Fic, RL и BR были случайным образом разделены на обучающую и тестовую выборки в соотношении 4:1, а затем разделены на фрагменты длиной 70 предложений аналогично [6]. Корпус TB был разделён на обучающую и тестовую выборки по сериям учебников: учебники за авторством А. Ф. Никитина составили обучающую выборку, за авторством Л. Н. Боголюбова — тестовую. Помимо этого, предложения были объединены в группы по 70 предложений, так как сложность отдельных предложений может значительно отличаться от сложности текста в целом. Краткая характеристика корпусов приведена в таблице 1.

Для сравнения влияния различных лингвистических характеристик мы использовали четыре алгоритма машинного обучения, упомянутые в работе [6]: метод случайного леса, метод опорных векторов, свёрточную нейронную сеть и многослойный персептрон. Для каждой из групп характеристик мы оценили качество а) алгоритмов, обученных только на вычисленных характеристиках (только для методов случайного леса и опорных векторов), б) алгоритмов, обученных только на векторном представлении текстов, и в) алгоритмов, обученных на совокупности характеристик и векторного представления. Как и в работе [6], мы рассматривали задачу классификации. Модели были спроектированы и реализованы следующим образом:

1. **Метод случайного леса (RF).** Мы использовали реализацию алгоритма из библиотеки scikit-learn [16]. Ансамбль состоял из 100 решающих деревьев, для оценки качества разбиения использовался критерий Джини. Для формирования векторного представления текста использовался алгоритма мешка слов с максимальным размером словаря 10000. В постановке с обучением на совокупности векторное представление и значения характеристик конкатенировались перед подачей в модель.
2. **Метод опорных векторов (LSVC).** Аналогично методу случайного леса мы использовали реализацию алгоритма из библиотеки scikit-learn. Для обучения использовался штраф l2 и функция потерь squared hinge. Способ формирования векторного представления текста и способ агрегации данных совпадают с таковыми для метода случайного леса.
3. **Свёрточная нейронная сеть (CNN).** Мы использовали два свёрточных слоя (256 фильтров, размер ядра 2), после каждого из которых расположен слой пулинга. Для формирования векторного представления использовалась модель geowac_lemmas_none_fasttextskipgram_300_5_2020[4] [17]. Для реализации использовался фреймворк Keras[5]. Обучение проводилось с использованием метода ранней остановки со следующими параметрами: максимальное чис-

[4] https://rusvectores.org/ru/models/
[5] https://github.com/fchollet/keras

ло эпох обучения — 100, максимальное количество эпох без улучшения результата (patience) — 20. Мы использовали оптимизатор Adam [18].

4. **Многослойный перцептрон (MLP).** Использованная архитектура состоит из трёх последовательных полносвязных слоёв из 1024 нейронов с функцией активации tanh, слоя прореживания с коэффициентом 0.5 и полносвязного слоя классификации с функцией активации softmax. Для формирования векторного представления использовалась модель distiluse-base-multilingual-cased [19] из семейства Sentence Transformers [20]. Как и в случае CNN, мы использовали фреймворк Keras, оптимизитор Adam и те же параметры обучения. При обучении с добавлением лингвистических характеристик векторное представление и значения характеристик конкатенировались перед подачей в модель.

Для предобработки текстов использовалась модель токенизации Stanza [21], обученная на данных Национального корпуса русского языка[6], для лемматизации текста использовалась библиотека pymorphy2 [22], кроме того, перед векторизацией тексты очищались от стоп-слов, входящих в набор таковых для русского языка в библиотеке NLTK [23].

Для оценки качества работы алгоритмов мы использовали метрику F1, для экспериментов с тремя и более классами использовалось взвешенное среднее, так как размеры классов значительно различаются. Запуск обучения с различными случайными зёрнами показал значительную дисперсию при зафиксированном разбиении на обучающую и тестовую выборки, в связи с чем было принято решение провести для каждой постановки пять запусков с различными случайными зёрнами и усреднить полученный результат.

2. Расширение признакового описания

2.1. Тематическое моделирование и ключевые слова

В рамках исследования признаков, связанных с тематикой текста, мы сравнили влияние на качество модели информации о тематике, полученной из латентного размещения Дирихле (LDA), и информации о ключевых словах. Этот эксперимент проводился только с текстами из корпусов Fic, RL и BR. Мы воспользовались реализацией алгоритма LDA из библиотеки scikit-learn, для каждого корпуса количество возможных тематик было равно 100.

Недавнее сравнение различных алгоритмов генерации [24] показало, что лучшего качества сразу по нескольким метрикам удаётся добиться при помощи дообучения многоязычной модели mT5 [25] на базе архитектуры T5 [26], в свою очередь представляющей собой развитие архитектуры Transformer [27]. Однако использование моделей, обучаемых с учителем, в рамках нашей задачи невозможно, так как нам не удалось найти подходящих по домену корпусов с разметкой ключевых слов. В то же время, результаты работы [24] показывают, что алгоритм RuTermExtract[7] позволяет добиться результатов, лишь немногим уступающих mT5, а по двум метрикам — и превосходящих. Поэтому мы решили использовать его в качестве алгоритма генерации ключевых. Для каждого из текстов, использованных в экспериментах, за исключением корпуса TB, при помощи RuTermExtract были предрасчитаны ключевые слова (не более 15 для каждого текста). Таким образом, для корпуса Fic было сгенерировано в совокупности 145744 ключевых слова/словосочетания, для корпуса RL — 27511, для корпуса BR — 17314. Для векторизации в случае метода случайного леса и метода опорных векторов использовался тот же алгоритм, что и для векторизации всего текста, в случае свёрточной нейронной сети и многослойного перцептрона — модель distiluse-base-multilingual-cased.

[6] https://ruscorpora.ru/

[7] https://github.com/igor-shevchenko/rutermextract

Table 2. Comparison of the performance of morpheme segmentation algorithms on the Morphodict dataset. CNN — ensemble of convolutional neural networks, CatBoost — gradient boosting algorithm over decision trees, LSTM — recurrent neural network with long short-term memory.

Таблица 2. Сравнение качества работы алгоритмов морфемной сегментации на словаре Morphodict. CNN — ансамбль свёрточных нейронных сетей, CatBoost — алгоритм градиентного бустинга над решающими деревьями, LSTM — рекуррентная нейронная сеть с долгой краткосрочной памятью.

Модель	CNN	CatBoost	LSTM
F1	0.9866	0.9326	0.9815
Precision	0.9858	0.9188	0.9800
Recall	0.9874	0.9469	0.9830
Accuracy	0.9740	0.8884	0.9661
WordAccuracy	0.9082	0.6443	0.8802

2.2. Лексические признаки

В рамках эксперимента с характеристиками текста, ассоциированными со сложностью лексики, мы сравнили две группы признаков, упоминаемых в литературе (признаки, связанные с частотностью и со словообразовательными паттернами), и признаки, основанные на анализе морфемных разборов слов. Для вычисления признаков, связанных с частотностью, использовался Частотный словарь современного русского языка: на материалах Национального корпуса русского языка [28], список словообразовательных паттернов совпадает с таковым из работы [9].

Трудность использования результатов морфемного анализа заключается в необходимости вычисления морфемных разборов слов при помощи отдельного алгоритма порождения разборов или же морфемного словаря, претендующего на полноту. При этом важно отметить, что задача построения морфемных разборов с лингвистической точки зрения трудна и, вероятно, не имеет решения [29]. Это связано с недостаточной формализацией задачи и существованием нескольких противоречащих друг другу парадигм выделения морфем.

В то же время, для решения прикладных задач, подобных исследуемой в настоящей работе, могут быть применены приближённые решения. Среди работ, посвящённых порождению морфемных разборов с разметкой типов морфем, лучшего качества удалось добиться алгоритмам машинного обучения. Так, в работе [30] представлен алгоритм генерации разборов на основе ансамбля свёрточных нейронных сетей, в работах [31, 32] проведено сравнение этого алгоритма с алгоритмом градиентного бустинга над решающими деревьями CatBoost и рекуррентной нейронной сетью с долгой краткосрочной памятью соответственно. Проанализировав использовавшийся в этих работах для обучения и тестирования словарь морфемных разборов на базе Словообразовательного словаря русского языка [33], мы обнаружили большую долю некорректно размеченных разборов [34]. В связи с этим, мы провели повторное сравнение моделей на материале словаря морфемных разборов Morphodict, использующегося в Национальном корпусе русского языка[8] (разработан на базе Словаря морфем русского языка [35]). Этот словарь содержит разборы для 75649 лемм. Всего в словаре встречается 8079 различных морфем, из которых 7148 имеют тип «корень». В среднем на слово приходится 4.12 морфем, а каждая морфема встречается в словаре 38.56 раз.

Для сравнения моделей мы провели кросс-валидацию на пяти непересекающихся выборках. Качество работы алгоритма оценивалось по метрикам, предложенным в работе [30]: F1, Precision, Recall — F-мера, точность и полнота для границ морфем без учёта их типа, Accuracy — доля верно выделенных (с учётом типа) морфем, WordAccuracy — доля полностью верных разборов. Результаты сравнения приведены в таблице 2.

[8] https://ruscorpora.ru/

Лучший результат согласно всем пяти метрикам показал алгоритм на основе ансамбля свёрточных нейронных сетей. По результатам эксперимента было решено использовать гибридное решение: для лемм, присутствующих в словаре, разбор берётся из словаря, в противном случае — порождается ансамблем свёрточных нейронных сетей, обученным на полном словаре. Итоговыми вычисляемыми характеристиками, сформировавшими набор морфемных признаков, стали (среднее, максимальное и медиана вычислялись относительно текста целиком):

1. доля уникальных морфем;
2. доля уникальных корней;
3. среднее/максимальное/медианное число морфем в слове;
4. среднее/максимальное/медианное число корней в слове;
5. доля слов, содержащих соединительные гласные;
6. средняя/максимальная/медианная суммарная длина суффиксов в слове;
7. доля слов, содержащих хотя бы один из суффиксов *-ни-*, *-ени-*, *-ц-*, *-ость-*, *-ств-*, *-ящ-*, *-ющ-*, *-еск-*, *-ист-*, *-изм-*, *-лищ-*, *-атор-*, *-тор-*, *-тель-*, *-ирова-*, *-иров-*[9].

2.3. Синтаксические признаки

Для описания синтаксических деревьев мы воспользовались форматом, описанным в рамках проекта UniversalDependencies[10]. Для порождения деревьев использовалась модель Stanza [21], обученная на материале корпуса SynTagRus[11]. В качестве базового набора синтаксических признаков Syn_{old} мы использовали набор признаков, описанный в работе [6]. Его мы сравнили с созданными наборами признаков Syn_{rich} и Syn_{best}.

Для создания набора Syn_{rich} мы выделили 46 синтаксических признаков, 36 из которых соответствуют количеству рёбер соответствующего типа в синтаксическом дереве предложения. Помимо этого для каждого предложения в качестве признаков вычислялись:

- глубина дерева;
- количество вершин;
- количество внутренних вершин;
- количество листьев;
- среднее линейное расстояние между вершиной и её непосредственным предком;
- максимальное линейное расстояние между вершиной и её непосредственным предком;
- количество вершин ровно с одним потомком;
- количество вершин ровно с двумя потомками;
- количество вершин ровно с тремя потомками;
- количество вершин ровно с четырьмя и более потомками.

На основании полученных для каждого предложения значений для текста в целом вычислялись четыре характеристики: среднее, медианное и максимальное значение признака в тексте, а также стандартное отклонение. Полученные 184 признака составили набор Syn_{rich}.

Далее на основании полученного набора Syn_{rich} мы построили набор Syn_{best}. Для этого мы оценили значимость каждого из признаков, входящих в Syn_{rich} при помощи нескольких подходов. В качестве основного метода оценки мы использовали взаимную информацию [36] характеристики и метки текста. Эксперименты проводились на материале корпусов TB и BR. Для каждого из корпусов были найдены пять характеристик с наибольшим значением этой величины. Для корпуса TB такими характеристиками оказались (в скобках указано значение взаимной информации):

- среднее количество рёбер с типом nmod (1.280);

[9] Список суффиксов был определён на основании работы [9] и консультаций с экспертами в области преподавания русского языка.

[10] https://universaldependencies.org/

[11] https://huggingface.co/stanfordnlp/stanza-ru/blob/main/models/depparse/syntagrus_charlm.pt

- средняя глубина дерева (1.068);
- среднее число внутренних вершин (0.956);
- среднее количество рёбер с типом amod (0.951);
- среднее число вершин, имеющих ровно двух потомков (0.825);

для корпуса BR:

- средняя глубина дерева (0.727);
- среднее количество рёбер с типом nmod (0.727);
- среднее количество рёбер с типом det (0.713);
- среднее число внутренних вершин (0.707);
- среднее число вершин, имеющих ровно двух потомков (0.825).

В совокупности таким образом в набор Syn_{best} было добавлено шесть признаков. Далее этот список был расширен при помощи серии экспериментов. В ходе каждого из них мы обучали модель регрессии, оценивали значимость признаков при помощи трёх метрик: Mean Decrease in Impurity [37], Permutation importance [38] и Drop-column importance (эта метрика вычисляется схоже с Permutation importance, однако вместо перемешивания значений в столбце он просто удаляется, а затем модель переучивается без него). Затем для каждой из метрик определяли пять наиболее значимых признаков. Всего было использовано три алгоритма обучения: случайный лес, алгоритм градиентного бустинга над решающими деревьями и метод опорных векторов (для этого алгоритма замерялись только метрики Permutation importance и Drop-column importance). Таким образом, мы собрали 16 списков и дополнили набор Syn_{best} теми признаками, которые оказались хотя бы в двух из них. Тем самым мы расширили набор до 21 признака:

- среднее число вершин, имеющих ровно двух потомков;
- среднее число внутренних вершин;
- среднее число вершин в дереве;
- средняя глубина дерева;
- среднее количество рёбер с типом amod;
- среднее количество рёбер с типом compound;
- среднее количество рёбер с типом conj;
- среднее количество рёбер с типом det;
- среднее количество рёбер с типом nmod;
- среднее количество рёбер с типом nsubj;
- максимальное число листьев;
- максимальное число внутренних вершин;
- максимальное число вершин в дереве;
- медианное количество рёбер с типом nmod;
- медианное число внутренних вершин;
- медианное число вершин в дереве;
- стандартное отклонение числа листьев;
- стандартное отклонение числа вершин в дереве;
- стандартное отклонение числа рёбер с типом conj;
- стандартное отклонение числа рёбер с типом det;
- стандартное отклонение среднего линейного расстояния между вершиной и её непосредственным предком.

3. Результаты

Результаты проведённых экспериментов перечислены в таблицах 3, 4 и 5. В совокупности можно говорить о значительных различиях в результатах в зависимости от конкретной модели и корпуса.

Table 3. The quality of the models trained using text representation, LDA topics (*Topic*), and keywords (*Keywords*). *BoW*, *FT*, *BERT* – text representation using a bag-of-words, FastText and BERT, respectively. In this and the following tables, for each dataset-model pair, the best achieved result and the results that differ from it by no more than 1 percentage point are highlighted in gray.

Таблица 3. Сравнение качества моделей, обученных с использованием векторного представления текста, тематик LDA (*Topic*) и ключевых слов (*Keywords*). *BoW*, *FT*, *BERT* — векторное представление текста при помощи мешка слов, FastText и BERT. В этой и следующих таблицах для каждой пары корпус-семейство моделей серым выделен лучший достигнутый результат и результаты, отличающиеся от него не более чем на 1 п. п.

Набор признаков	**FicRARS**	**FicChAd**	**RL**	**BR**
Метод случайного леса				
BoW	0.427 ± 0.004	0.743 ± 0.002	0.485 ± 0.010	0.298 ± 0.009
Topic	**0.450 ± 0.002**	**0.768 ± 0.003**	**0.538 ± 0.014**	0.311 ± 0.008
Keywords	0.422 ± 0.002	**0.768 ± 0.002**	0.448 ± 0.003	0.271 ± 0.004
BoW+Topic	0.437 ± 0.003	**0.764 ± 0.002**	0.489 ± 0.009	**0.338 ± 0.010**
BoW+Keywords	0.435 ± 0.003	**0.765 ± 0.001**	0.463 ± 0.009	0.309 ± 0.003
Метод опорных векторов				
BoW	0.424	0.751	0.618	0.322
Topic	**0.440**	**0.766**	**0.658**	0.308
Keywords	0.423	**0.763**	0.519	0.251
BoW+Topic	0.428	0.754	0.631	**0.333**
BoW+Keywords	0.420	**0.757**	0.645	**0.327**
Свёрточная нейронная сеть				
FT	**0.452 ± 0.011**	**0.792 ± 0.005**	0.531 ± 0.028	0.409 ± 0.029
FT+Topic	**0.460 ± 0.012**	**0.793 ± 0.003**	**0.552 ± 0.038**	**0.421 ± 0.006**
FT+Keywords	**0.461 ± 0.010**	**0.793 ± 0.007**	0.517 ± 0.040	**0.411 ± 0.024**
Многослойный перцептрон				
BERT	0.440 ± 0.032	0.666 ± 0.004	0.572 ± 0.040	0.333 ± 0.015
BERT+Topic	**0.541 ± 0.042**	**0.757 ± 0.025**	**0.594 ± 0.044**	0.351 ± 0.018
BERT+Keywords	0.467 ± 0.013	0.710 ± 0.008	0.541 ± 0.042	**0.374 ± 0.005**

Ни в одной из трёх серий экспериментов нам не удалось обнаружить полного превосходства одной группы признаков над другой.

Эксперименты с тематическими маркерами LDA и ключевыми словами в большинстве случаев показали превосходство первых. В постановке с обучением только на вычисленных характеристиках среднее качество метода случайного леса и метода опорных векторов, обученных только на маркерах LDA, превзошло хотя бы на 1 п. п. качество аналогичных моделей, обученных только на ключевых, в 6 экспериментах из 8. Более того, в 7 из 8 экспериментов эти модели превзошли и модели, обученные на векторном представлении текста против 2 из 8 для ключевых. В случае обучения на совокупности векторного представления и лингвистических характеристик ситуация менее выраженная: из 16 проведённых экспериментов превосходство хотя бы на 1 п. п. в 7 случаях было достигнуто моделями, обученными с добавлением маркеров LDA, в 2 — с ключевыми, в остальных 7 случаях разница составила менее 1 п. п.

В отличие от эксперимента с тематическими признаками, модели, обученные только на лексических признаках, почти всегда показывали качество хуже, чем аналогичные модели, обученные на векторном представлении текста. Только в одном случае (набор *Morpheme*, метод случайного леса, корпус RL) модель продемонстрировала значительное превосходство над базовой (почти на 10 п. п.). При этом в большинстве случаев использование любого из трёх наборов в совокупности с векторным

Table 4. The quality of the models trained using text representation, information about frequency (*Freq*), occurrence of word-formation patterns (*Pattern*) and features of morphemic word parsing (*Morpheme*). *BoW*, *FT*, *BERT* — text representation using a bag-of-words, FastText and BERT, respectively.

Таблица 4. Сравнение качества моделей, обученных с использованием векторного представления текста, сведений о частотности (*Freq*), встречаемости словообразовательных паттернов (*Pattern*) и особенностях морфемных разборов слов (*Morpheme*). *BoW*, *FT*, *BERT* — векторное представление текста при помощи мешка слов, FastText и BERT, соответственно.

Набор признаков	FicRARS	FicChAd	TB	RL	BR
Метод случайного леса					
BoW	**0.427 ± 0.004**	**0.743 ± 0.002**	**0.722 ± 0.021**	0.485 ± 0.010	0.298 ± 0.009
Freq	0.369 ± 0.001	0.669 ± 0.002	0.680 ± 0.012	0.433 ± 0.003	0.265 ± 0.012
Pattern	0.345 ± 0.002	0.616 ± 0.001	0.606 ± 0.017	0.462 ± 0.006	0.248 ± 0.006
Morpheme	0.380 ± 0.002	0.666 ± 0.003	0.680 ± 0.009	**0.582 ± 0.010**	0.239 ± 0.004
BoW+Freq	**0.424 ± 0.003**	**0.746 ± 0.002**	0.711 ± 0.033	0.475 ± 0.007	0.286 ± 0.009
BoW+Pattern	**0.425 ± 0.002**	**0.747 ± 0.003**	**0.728 ± 0.026**	0.485 ± 0.013	0.301 ± 0.003
BoW+Morpheme	**0.431 ± 0.003**	**0.752 ± 0.002**	**0.718 ± 0.030**	0.484 ± 0.005	**0.313 ± 0.003**
Метод опорных векторов					
BoW	**0.424**	**0.751**	**0.813**	**0.618**	**0.322**
Freq	0.355	0.654	0.640	0.449	0.286
Pattern	0.351	0.620	0.633	0.421	0.225
Morpheme	0.376	0.691	0.662	0.467	0.214
BoW+Freq	**0.424**	**0.750**	**0.813**	**0.622**	**0.317**
BoW+Pattern	**0.426**	**0.752**	**0.813**	**0.623**	**0.328**
BoW+Morpheme	**0.426**	**0.756**	**0.807**	**0.628**	**0.325**
Свёрточная нейронная сеть					
FT	0.452 ± 0.011	**0.792 ± 0.005**	0.538 ± 0.037	0.531 ± 0.028	0.409 ± 0.029
FT+Freq	0.454 ± 0.020	**0.796 ± 0.003**	0.560 ± 0.026	**0.563 ± 0.022**	**0.424 ± 0.014**
FT+Pattern	**0.474 ± 0.037**	**0.794 ± 0.004**	0.550 ± 0.053	0.533 ± 0.021	**0.422 ± 0.031**
FT+Morpheme	0.455 ± 0.011	**0.793 ± 0.002**	**0.615 ± 0.068**	**0.573 ± 0.035**	0.413 ± 0.010
Многослойный перцептрон					
BERT	0.440 ± 0.032	0.666 ± 0.004	0.513 ± 0.030	0.572 ± 0.040	0.333 ± 0.015
BERT+Freq	**0.477 ± 0.016**	**0.697 ± 0.005**	0.659 ± 0.021	0.528 ± 0.031	0.323 ± 0.015
BERT+Pattern	0.441 ± 0.023	0.677 ± 0.008	0.629 ± 0.010	0.563 ± 0.074	0.340 ± 0.005
BERT+Morpheme	**0.478 ± 0.020**	**0.690 ± 0.010**	**0.719 ± 0.007**	**0.611 ± 0.035**	**0.358 ± 0.014**

представлением для метода случайного леса и метода опорных векторов не давало значительного изменения качества. Улучшение хотя бы на 1 п. п. было достигнуто только один раз. Иначе обстоят дела для нейросетевых моделей. Для всех пар корпус-модель, кроме пары FicChAd+CNN, использование дополнительной информации позволило добиться прироста качества на 1 п. п. хотя бы для одного из наборов признаков. Попарное сравнение трёх наборов показывает значительную неоднородность, однако в 5 из 10 случаев лучший результат с отрывом более, чем на 1 п. п., достигнут с использованием набора признаков, связанных с особенностями морфемного строения, а ещё в трёх случаях эта модель оказалась в числе лучших (разница менее 1 п. п.).

Для метода случайного леса и метода опорных векторов, обученных с использованием синтаксических признаков, ситуация схожа с аналогичными экспериментами с лексическими признаками. В большинстве ситуаций базовая модель, обученная на векторном представлении текста, либо превосходит модели с добавлением информации, либо лишь незначительно уступает им. Обратная ситуация наблюдается только в трёх парах корпус-модель: TB+RF (лучшая модель использует

Table 5. The quality of the models trained using text representation and the syntactic feature sets Syn_{old}, Syn_{rich}, and Syn_{best} defined in subsection 2.3. *BoW*, *FT*, *BERT* — text representation using a bag-of-words, FastText and BERT, respectively.

Таблица 5. Сравнение качества моделей, обученных с использованием векторного представления текста и наборов синтаксических признаков Syn_{old}, Syn_{rich} и Syn_{best}, определённых в подразделе 2.3. *BoW*, *FT*, *BERT* — векторное представление текста при помощи мешка слов, FastText и BERT, соответственно.

Набор признаков	FicRARS	FicChAd	TB	RL	BR
Метод случайного леса					
BoW	**0.427 ± 0.004**	**0.743 ± 0.002**	0.722 ± 0.021	0.485 ± 0.010	**0.298 ± 0.009**
Syn_{old}	0.375 ± 0.001	0.675 ± 0.001	0.680 ± 0.007	**0.605 ± 0.007**	0.256 ± 0.007
Syn_{rich}	0.386 ± 0.003	0.695 ± 0.001	0.703 ± 0.007	0.573 ± 0.004	0.286 ± 0.010
Syn_{best}	0.366 ± 0.003	0.670 ± 0.002	0.687 ± 0.005	0.566 ± 0.003	0.271 ± 0.007
BoW + Syn_{old}	**0.432 ± 0.001**	0.731 ± 0.004	0.720 ± 0.015	0.485 ± 0.008	**0.292 ± 0.010**
BoW + Syn_{rich}	0.415 ± 0.004	0.731 ± 0.004	0.707 ± 0.006	0.483 ± 0.010	0.262 ± 0.008
BoW + Syn_{best}	**0.427 ± 0.004**	**0.738 ± 0.002**	**0.736 ± 0.026**	0.484 ± 0.005	0.262 ± 0.005
Метод опорных векторов					
BoW	**0.424**	**0.751**	**0.813**	0.618	**0.322**
Syn_{old}	0.373	0.680	0.704	0.606	0.227
Syn_{rich}	0.394	0.714	0.655	0.572	0.306
Syn_{best}	**0.424**	0.686	0.661	0.579	0.266
BoW + Syn_{old}	**0.425**	**0.752**	**0.804**	**0.629**	**0.323**
BoW + Syn_{rich}	**0.421**	**0.752**	**0.807**	**0.627**	**0.315**
BoW + Syn_{best}	**0.424**	**0.752**	0.800	**0.621**	**0.317**
Свёрточная нейронная сеть					
FT	0.452 ± 0.011	**0.792 ± 0.005**	0.538 ± 0.037	0.531 ± 0.028	0.409 ± 0.029
FT + Syn_{old}	**0.464 ± 0.023**	**0.793 ± 0.004**	0.598 ± 0.112	0.569 ± 0.016	**0.425 ± 0.018**
FT + Syn_{rich}	**0.454 ± 0.011**	**0.793 ± 0.003**	**0.684 ± 0.050**	**0.584 ± 0.045**	0.416 ± 0.022
FT + Syn_{best}	0.448 ± 0.012	**0.793 ± 0.003**	**0.680 ± 0.078**	0.552 ± 0.044	**0.427 ± 0.032**
Многослойный перцептрон					
BERT	0.440 ± 0.032	0.666 ± 0.004	0.513 ± 0.030	0.572 ± 0.040	**0.333 ± 0.015**
BERT + Syn_{old}	0.481 ± 0.018	**0.704 ± 0.003**	0.727 ± 0.010	**0.615 ± 0.050**	**0.337 ± 0.017**
BERT + Syn_{rich}	**0.497 ± 0.013**	**0.714 ± 0.009**	**0.771 ± 0.012**	**0.625 ± 0.035**	0.315 ± 0.017
BERT + Syn_{best}	0.464 ± 0.018	0.701 ± 0.012	0.741 ± 0.007	**0.618 ± 0.020**	**0.336 ± 0.017**

комбинацию векторного представления и набора признаков Syn_{best}), RL+RF (лучшая модель использует только набор признаков Syn_{old}) и RL+LSVC (в этом случае лучший результат показала модель, использующая комбинацию векторного представления и набора признаков Syn_{old}, а модели с использованием Syn_{rich} и Syn_{best} отстали менее, чем на 1 п. п.). При этом модели, обученные на комбинации векторного представления и набора признаков Syn_{best} ни разу не отстали более, чем на 1 п. п., от моделей на комбинации векторного представления и набора признаков Syn_{rich}, а в двух случаях (с методом случайного леса) обошли их более, чем на 1 п. п.. В случае нейросетевых алгоритмов в большинстве пар корпус-модель лучший или один из лучших результатов показали модели, обученные с использованием набора признаков Syn_{rich} В особенности это заметно для многослойного перцептрона, где лучший результат достигнут при помощи этих моделей в 4 из 5 случаев. Попарное сравнение наборов Syn_{old} и Syn_{best} не позволяет выделить превосходящий: Syn_{old} лучше в трёх случаях, Syn_{best} — в двух, в остальных пяти разница составляет менее 1 п. п.

Заключение

В данной работе рассматривается задача формирования и расширения признакового описания при оценке сложности текста. Мы провели три серии экспериментов, в ходе которых сравнили новые наборы тематических, лексических и синтаксических признаков с ранее описанными. В качестве тематических признаков были изучены автоматически генерируемые ключевые слова, в качестве лексических — признаки, связанные с морфемным строением слов, в качестве синтаксических — обширный набор признаков, включающий информацию о типах рёбер, разветвлённости и глубине деревьев. Мы провели сравнение с использованием четырёх различных алгоритмов машинного обучения на материале четырёх корпусов (в одном из которых использовано два типа возрастных меток), что позволило получить более обширное представление о применимости признаков.

Проведённый анализ показал, что в большинстве ситуаций ключевые слова показали худший результат в сравнении с тематическими маркерами, полученными при помощи латентного размещения Дирихле. Это может быть связано как с недостаточным качеством генерации ключевых, так и с малой применимостью ключевых слов для домена художественных текстов. Для установления точных причин необходимо проведение дополнительных исследований. В то же время использование двух других новых наборов признаков (набора признаков, связанных с морфемным разнообразием, и обширного набора синтаксических признаков) позволило улучшить качество работы в сравнении с ранее изученными способами формирования признакового описания. При этом формирование этих наборов алгоритмически сложнее (в особенности это касается набора признаков, связанных с морфемным разнообразием, где для формирования признаков текст предварительно токенизируется, лемматизируется, а затем каждой лемме сопоставляются морфемные разборы, в том числе, автоматически генерируемые при помощи свёрточной нейронной сети).

В дальнейшем мы планируем сконцентрироваться на поиске других лингвистически мотивированных признаков и наиболее эффективных комбинаций из числа ранее изученных признаков, в частности, планируется провести более подробное исследование синтаксических признаков и интерпретировать полученные результаты с точки зрения лингвистики. Кроме того, будет проведён поиск устойчивых относительно смены корпуса и типа разметки моделей. Также интересной для изучения представляется оценка алгоритмической сложности вычисления тех или иных признаков и сравнение различных моделей с точки зрения быстродействия.

References

[1] R. Flesch, “A new readability yardstick.”, *Journal of Applied Psychology*, vol. 32, no. 3, p. 221, 1948.

[2] E. Dale and J. S. Chall, “A formula for predicting readability: Instructions”, *Educational Research Bulletin*, vol. 27, pp. 37–54, 1948.

[3] R. Senter and E. A. Smith, “Automated readability index”, AMRL TR, Tech. Rep. 5302480, 1967.

[4] M. Solnyshkina, V. Ivanov, and V. Solovyev, “Readability formula for Russian texts: A modified version”, in *Proceedings of the 17th Mexican International Conference on Artificial Intelligence, Part II*, 2018, pp. 132–145. DOI: 10.1007/978-3-030-04497-8_11.

[5] A. Churunina, M. Solnyshkina, E. Gafiyatova, and A. Zaikin, “Lexical features of text complexity: The case of Russian academic texts”, *SHS Web of Conferences*, vol. 88, no. 1, p. 01 009, 2020. DOI: 10.1051/shsconf/20208801009.

[6] D. A. Morozov, A. V. Glazkova, and B. L. Iomdin, “Text complexity and linguistic features: Their correlation in English and Russian”, *Russian Journal of Linguistics*, vol. 26, no. 2, pp. 426–448, 2022. DOI: 10.22363/2687-0088-30132.

[7] N. Karpov, J. Baranova, and F. Vitugin, "Single-sentence readability prediction in Russian", in *Analysis of Images, Social Networks and Texts*, Cham: Springer International Publishing, 2014, pp. 91–100. DOI: 10.1007/978-3-319-12580-0_9.

[8] V. V. Ivanov, M. I. Solnyshkina, and V. D. Solovyev, "Efficiency of text readability features in Russian academic texts", *Komp'juternaja Lingvistika I Intellektual'nye Tehnologii*, vol. 17, pp. 267–283, 2018.

[9] O. Blinova and N. Tarasov, "A hybrid model of complexity estimation: Evidence from Russian legal texts", *Frontiers in Artificial Intelligence*, vol. 5, p. 1 008 530, 2022. DOI: 10.3389/frai.2022.1008530.

[10] U. Isaeva and A. Sorokin, "Investigating the robustness of reading difficulty models for Russian educational texts", in *Recent Trends in Analysis of Images, Social Networks and Texts*, Cham: Springer International Publishing, 2021, pp. 65–77. DOI: 10.1007/978-3-030-71214-3_6.

[11] A. N. Laposhina, T. S. Veselovskaya, M. U. Lebedeva, and O. F. Kupreshchenko, "Lexical analysis of the Russian language textbooks for primary school: Corpus study", in *Komp'juternaja Lingvistika I Intellektual'nye Tehnologii*, vol. 18, 2019, pp. 351–363.

[12] V. Solovyev, V. Ivanov, and M. Solnyshkina, "Readability formulas for three levels of Russian school textbooks", *Investigations on Applied Mathematics and Informatics. Part II–1*, vol. 529, pp. 140–156, 2023.

[13] A. N. Laposhina, M. Y. Lebedeva, and A. A. Berlin Khenis, "Word frequency and text complexity: An eye-tracking study of young Russian readers", *Russian Journal of Linguistics*, vol. 26, no. 2, pp. 493–514, 2022. DOI: 10.22363/2687-0088-30084.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation", *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[15] A. Glazkova, Y. Egorov, and M. Glazkov, "A comparative study of feature types for age-based text classification", in *Analysis of Images, Social Networks and Texts*, Cham: Springer International Publishing, 2021, pp. 120–134, ISBN: 978-3-030-72610-2. DOI: 10.1007/978-3-030-72610-2_9.

[16] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python", *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[17] A. Kutuzov and E. Kuzmenko, "WebVectors: A toolkit for building web interfaces for vector semantic models", in *Analysis of Images, Social Networks and Texts*, Springer, 2017, pp. 155–161. DOI: 10.1007/978-3-319-52920-2_15.

[18] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[19] N. Reimers and I. Gurevych, "Making monolingual sentence embeddings multilingual using knowledge distillation", in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4512–4525. DOI: 10.18653/v1/2020.emnlp-main.365.

[20] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks", in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019. DOI: 10.18653/v1/D19-1410.

[21] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Association for Computational Linguistics, 2020, pp. 101–108. DOI: 10.18653/v1/2020.acl-demos.14.

[22] M. Korobov, "Morphological analyzer and generator for Russian and Ukrainian languages", in *International Conference on Analysis of Images, Social Networks and Texts*, Springer, 2015, pp. 320–332. DOI: 10.1007/978-3-319-26123-2_31.

[23] E. Loper and S. Bird, "NLTK: The natural language toolkit", in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, 2002, pp. 63–70.

[24] A. Glazkova, D. Morozov, M. Vorobeva, and A. Stupnikov, "Keyphrase generation for the Russian-language scientific texts using mT5", *Modeling and Analysis of Information Systems*, vol. 30, no. 4, pp. 418–428, 2023. DOI: 10.18255/1818-1015-2023-4-418-428.

[25] L. Xue *et al.*, "mT5: A massively multilingual pre-trained text-to-text transformer", in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 483–498. DOI: 10.18653/v1/2021.naacl-main.41.

[26] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer", *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[27] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing", in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.

[28] O. Lyashevskaya and S. Sharov, *Chastotnyj slovar' sovremennogo russkogo yazyka: na materialah Nacional'nogo korpusa russkogo yazyka*. Azbukovnik, 2009, in Russian, ISBN: 9785911720247.

[29] B. L. Iomdin, "How to define words with the same root?", *Russian Speech = Russkaya Rech'*, vol. 1, pp. 109–115, 2019, in Russian. DOI: 10.31857/S013161170003980-7.

[30] A. Sorokin and A. Kravtsova, "Deep convolutional networks for supervised morpheme segmentation of Russian language", in *Artificial Intelligence and Natural Language*, Cham: Springer International Publishing, 2018, pp. 3–10. DOI: 10.1007/978-3-030-01204-5_1.

[31] E. I. Bolshakova and A. S. Sapin, "Comparing models of morpheme analysis for Russian words based on machine learning", in *Komp'juternaja Lingvistika I Intellektual'nye Tehnologii*, vol. 18, 2019, pp. 104–113.

[32] E. Bolshakova and A. Sapin, "Bi-LSTM model for morpheme segmentation of Russian words", in *Artificial Intelligence and Natural Language*, Cham: Springer International Publishing, 2019, pp. 151–160. DOI: 10.1007/978-3-030-34518-1_11.

[33] A. N. Tikhonov, *Slovoobrazovatel'nyi slovar' russkogo yazyka*. Moscow: Russkiy yazyk, 1990, in Russian.

[34] T. Garipov, D. Morozov, and A. Glazkova, "Generalization ability of CNN-based Morpheme Segmentation", in *2023 Ivannikov Ispras Open Conference (ISPRAS)*, 2024, pp. 58–62.

[35] A. I. Kuznetsova and T. F. Efremova, *Dictionary of Morphemes of the Russian Language*. Firebird Publications, Incorporated, 1986, 1136 pp.

[36] T. Cover and A. Joy, "Entropy, relative entropy, and mutual information", in *Elements of Information Theory*. John Wiley & Sons, Ltd, 2005, ch. 2, pp. 13–55, ISBN: 9780471748823. DOI: 10.1002/047174882X.ch2.

[37] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen, *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[38] A. Altmann, L. Tolosi, O. Sander, and T. Lengauer, "Permutation importance: A corrected feature importance measure", *Bioinformatics (Oxford, England)*, vol. 26, no. 10, pp. 1340–1347, 2010. DOI: 10.1093/bioinformatics/btq134.