ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 36 Выпуск 4

ISSN Online 2220-6426
Volume 36 Issue 4

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2024

ИСП РАН

ISSN 2079-8156 (Print), ISSN 2220-6426 (Online)

Труды Института системного программирования РАН
Proceedings of the Institute for System Programming of the RAS

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областям системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.
Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

- операционные системы;
- компиляторные технологии;
- базы данных и информационные системы;
- параллельные и распределенные системы;
- автоматизированная разработка программ;
- верификация, валидация и тестирование;
- статический и динамический анализ;
- защита и обеспечение безопасности ПО;
- компьютерные алгоритмы;
- искусственный интеллект.

Журнал издается по одному тому в год, шесть выпусков в каждом томе.
Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.
Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.
The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

- Operating Systems.
- Compiler Technology.
- Databases and Information Systems.
- Parallel and Distributed Systems.
- Software Engineering.
- Software Modeling and Design Tools.
- Verification, Validation, and Testing.
- Static and Dynamic Analysis.
- Software Safety and Security.
- Computer Algorithms.
- Artificial Intelligence.

The journal is published one volume per year, six issues in each volume.
Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

УДК004.45

Адрес: 109004, г. Москва, ул. А. Солженицына, дом 25.
Телефон: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Сайт: http://www.ispras.ru/proceedings/

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Web: http://www.ispras.ru/en/proceedings

Содержание

Table of Contents

DOI: 10.15514/ISPRAS-2024-36(4)-1

Comparison of Voice Cloning Algorithms in Zero-shot and Few-shot Scenarios

O. Hovhannisyan, ORCID: 0009-0000-9384-7945 <olga.hovhannisyan@student.rau.am>
D. Sargsyan, ORCID: 0009-0006-0349-6031 <d.sargsyan@ispras.ru>
A. Malajyan, ORCID: 0000-0002-3566-9316 <malajyanarthur@ispras.ru>

Russian-Armenian University,
123, Hovsep Emin st., Yerevan, 0051, Armenia.

Abstract. Voice cloning technology has made significant strides in recent years, with applications ranging from personalized virtual assistants to sophisticated entertainment systems. This study compares nine voice cloning models, focusing on both zero-shot and fine-tuned approaches. Zero-shot voice cloning models have gained attention for their ability to generate high-quality synthetic voices without requiring extensive training data for each new voice and for their capability to perform real-time inference online. In contrast, non-zero- shot models typically require additional data but can offer improved fidelity in voice reproduction. The study comprises two key experiments. The first experiment evaluates the performance of zero-shot voice cloning models, analyzing their ability to reproduce target voices without prior exposure accurately. The second experiment involves fine-tuning the models on target speakers to assess improvements in voice quality and adaptability. The models are evaluated based on key metrics assessing voice quality, speaker identity preservation, and subjective and objective performance measures. The findings indicate that while zero-shot models offer greater flexibility and ease of deployment, fine-tuned models can deliver superior performance.

Keywords: voice cloning; zero-shot cloning; fine-tuning; speech synthesis; speaker adaptation.

For citation: Hovhannisyan O., Sargsyan D., Malajyan A. Comparison of voice cloning algorithms in zero-shot and few-shot scenarios. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024, pp. 7-16. DOI: 10.15514/ISPRAS-2024-36(4)-1.

Acknowledgements. This work was supported by the Science Committee of RA (Research project № 23AA-1B006).

Сравнение алгоритмов клонирования голоса в условиях нулевого и малого количества примеров

О. Оганесян, ORCID: 0009-0000-9384-7945 <olga.hovhannisyan@student.rau.am>
Д. Саргсян, ORCID: 0009-0006-0349-6031 <d.sargsyan@ispras.ru>
А. Маладжян, ORCID: 0000-0002-3566-9316 <malajyanarthur@ispras.ru>

Российско-Армянский университет,
Армения, 0051, Ереван, ул. Овсепа Эмина, 123.

Аннотация. Технология клонирования голоса сделала значительные шаги вперед в последние годы, с применением от персонализированных виртуальных ассистентов до сложных развлекательных систем. В данном исследовании проводится сравнение девяти моделей клонирования голоса, сосредотачиваясь на подходах нулевого и тонкой настройки. Модели клонирования голоса с нулевым обучением привлекают внимание своей способностью генерировать высококачественные синтетические голоса без необходимости в больших объемах обучающих данных для каждого нового голоса, а также возможностью осуществлять онлайн выводы в режиме реального времени. В отличие от них, модели, не относящиеся к нулевому обучению, обычно требуют дополнительных данных, но могут обеспечить улучшенную точность воспроизведения голоса. Исследование включает два ключевых эксперимента. Первый эксперимент оценивает эффективность моделей клонирования голоса с нулевым обучением, анализируя их способность точно воспроизводить целевые голоса без предварительного ознакомления. Второй эксперимент включает тонкую настройку моделей на целевых спикеров для оценки улучшений в качестве голоса и адаптивности. Модели оцениваются на основе ключевых показателей, оценивающих качество голоса, сохранение идентичности спикера, а также субъективные и объективные показатели производительности. Результаты показывают, что, хотя модели с нулевым обучением предлагают большую гибкость и простоту использования, модели с тонкой настройкой могут обеспечить более высокую производительность.

Ключевые слова: клонирование голоса; клонирование с нулевым обучением; тонкая настройка; синтез речи; адаптация говорящего.

Для цитирования: Оганесян О., Саргсян Д., Маладжян А. Сравнение алгоритмов клонирования голоса в условиях нулевого и малого количества примеров. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 7–16 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(4)–1.

Благодарности. Работа выполнена при поддержке Комитета по науке Республики Армения (исследовательский проект № 23AA-1B006).

1. Introduction

Voice cloning technology has advanced rapidly, enabling the creation of synthetic voices that closely mimic human speech. This technology has significant applications in personalized virtual assistants, entertainment, and communication. The core challenge in voice cloning is to produce synthetic voices that are indistinguishable from human voices while preserving the unique characteristics of the target speaker.

Two primary methodologies have emerged in voice cloning: zero-shot cloning and fine-tuning. Zero-shot models, such as XTTS 2 [1], StyleTTS [2], YourTTS [3], OpenVoice [4], VoiceCraft [5], Vall-E-X [6], and Natural Speech 3 [7], can generate voices without extensive speaker-specific training data, offering flexibility and scalability. However, maintaining voice quality and identity without prior exposure to the target speaker remains challenging. Fine-tuning models, including VITS [8] and RVC [9], improve voice fidelity by adapting pre-trained models with additional data from the target speaker, although they require more data and computational resources.

This study focuses on models that excel in flexibility, scalability, and efficiency in zero-shot and few-shot scenarios. Older models like WaveNet [10], Deep Voice [11], SV2TTS [12], TortoiseTTS [13], Tacotron [14], and Glow-TTS [15] are excluded due to their high computational demands, extensive data requirements, and complexity.

This paper evaluates nine voice cloning models through two key experiments. The first experiment tests the zero-shot capabilities of the models, assessing their performance in replicating voices without prior exposure. The second experiment involves fine-tuning the models on target speakers to evaluate enhancements in voice quality and adaptability. Evaluation metrics include speaker embedding cosine similarity (SECS [16]) for identity preservation, Mel cepstral distortion (MCD[1] [17]) for spectral similarity, F0 mean absolute error (F0 MAE [17]) for pitch accuracy, F0 Pearson correlation coefficient (F0-PCC [18]) for pitch contour correlation, and universal target mean opinion score (UTMOS2[2] [1]) for subjective quality.

The paper is organized as follows: Section 2 provides a detailed description of each voice cloning model. Section 3 outlines the experimental setup and presents the experiments and results. Only original manuscripts that have not been previously published nor in other editions, neither in the Internet, are accepted for publication in Proceedings of ISP RAS. The authors of the articles can be ISP RAS staff or representatives of other organizations. Only manuscripts in Russian or English are allowed to be published. As a rule, the volume of published articles should not be less than 8-9 pages, and shouldn't exceed 20 pages.

2. Overview of Voice Cloning Models

In recent years, various voice cloning models have emerged, each offering unique approaches and capabilities. This section provides an overview of the voice cloning models discussed in the introduction, highlighting their key features and processes.

Variational Inference Text-to-Speech (VITS): The VITS model is designed to generate speech directly from text. It incorporates a stochastic duration predictor to capture natural speech rhythms, enabling the production of authentic and fluid voice waveforms. This end-to-end approach supports high-quality voice cloning by accurately translating text into speech with precise timing and natural intonation. However, VITS is not a zero-shot model and requires training on target voices beforehand.

Retrieval-based Voice Conversion (RVC): The RVC model is a system for converting one speaker's voice into another's. It leverages a retrieval-based approach to map and synthesize voice characteristics from a database of target voices. This method enables high-quality voice transformation by accurately capturing and replicating speaker-specific traits. Unlike zero-shot models, RVC needs to be trained on a set of target voices to effectively perform voice conversion.

OpenVoice: OpenVoice uses simply a short audio clip from the reference speaker to generate speech in multiple languages. It provides flexible control over voice styles and enables zero-shot cross-lingual cloning, though it requires a TTS model trained for the target language.

StyleTTS 2: StyleTTS 2 generates high-quality, natural-sounding speech using style diffusion techniques. It models voice styles as latent variables, enabling it to clone voices with no need for specific reference recordings. By leveraging large pre-trained speech language models and innovative training methods, StyleTTS 2 excels in producing expressive and accurate voice clones, including effective zero-shot speaker adaptation.

VoiceCraft: VoiceCraft excels in speech editing and zero-shot text-to-speech generating. It uses a Transformer decoder and an innovative token rearrangement method to generate high-quality, natural-sounding speech by efficiently reconstructing and infilling speech tokens. It analyzes and replicates the vocal characteristics of a target speaker, capturing emotional tone and subtle vocal nuances to produce realistic and engaging speech.

YourTTS: YourTTS builds on the VITS model, excelling in zero-shot voice cloning and multi-speaker text-to-speech with minimal data. It performs well across various languages and can adapt

[1] https://pypi.org/project/pymcd/

[2] https://github.com/sarulab-speech/UTMOSv2

to new voices with less than one minute of audio. However, it may occasionally face issues with speech duration and mispronunciations.

VALL-E-X: VALL-E X is a cross-lingual neural codec model that excels in zero-shot text-to-speech and speech-to-speech translation. It generates high-quality speech in a target language from a single utterance in a source language, preserving the speaker's voice and emotion. The model avoids the need for paired cross-lingual data and effectively addresses foreign accent issues, making it suitable for diverse multilingual applications.

XTTS-2: XTTS 2 is a multilingual zero-shot text-to-speech (TTS) model trained in 16 languages. Building on the Tortoise model, XTTS 2 enhances voice cloning, speed, and multilingual capabilities. It achieves high-quality results in prosody and style mimicking, including whispering, with minimal fine-tuning data. XTTS 2 is notably faster than previous models like VALL-E.

Natural Speech 3: NaturalSpeech 3 generates high-quality, natural-sounding speech by separating and controlling speech attributes like content, prosody, and timbre. Its novel factorized diffusion approach allows for detailed and accurate speech synthesis, achieving superior performance and human-level quality on diverse datasets.

WaveNet: WaveNet generates raw audio waveforms using an autoregressive model, achieving high naturalness. However, it requires substantial computational resources and has slow inference times.

Deep Voice: Deep Voice uses a modular pipeline to produce human-like speech but requires extensive speaker-specific training data. Modern models overcome this limitation by utilizing less data, enabling more flexible and scalable voice cloning.

SV2TTS: SV2TTS employs a three-stage pipeline for voice cloning but struggles with voice quality and identity preservation without extensive fine-tuning.

Tortoise TTS: Tortoise TTS excels in expressive speech synthesis but demands significant computational resources and data for adaptation. Its complexity and inefficiency make it impractical for zero-shot applications requiring minimal data.

Tacotron: Tacotron generates speech from text with high naturalness but relies on the Griffin-Lim [19] algorithm, which can introduce artifacts. It requires substantial training data, limiting its effectiveness in zero-shot learning scenarios that require rapid adaptation.

Glow-TTS: Glow-TTS offers efficient parallel synthesis but lacks built-in support for speaker adaptation, necessitating additional modifications and data. Its focus on general TTS tasks rather than speaker-specific scenarios reduces its suitability for robust zero-shot application.

Table 1 provides an overview of the voice cloning models discussed, highlighting their zero-shot capabilities and the number of parameters.

3. Experimental setup and results

The experiments aim to evaluate the performance of nine voice cloning models – XTTS 2, StyleTTS 2, YourTTS, VITS, OpenVoice, RVC, VoiceCraft, Vall-E-X, and Natural Speech 3 – using both zero-shot and fine-tuning approaches. The goal is to assess each model's ability to reproduce target voices with high quality and fidelity.

3.1 Experimental Setup

The experiments utilize the VCTK corpus [20], which includes speech data from 109 English speakers with various accents. The experiment is structured as follows:

- **Zero-Shot Experiment:** We select 30 speakers from the VCTK dataset, using 5 audio samples per speaker, each ranging in duration from 7 to 10 seconds. Models requiring text input are evaluated with additional sentences from the remaining speakers of the same dataset, consisting of 6-15 words on average. Audio-to-audio models are tested using audio samples instead sentences from the same speakers' set, containing audios in the range of 5–10 seconds.

- **Fine-Tuning Experiment:** Each model is fine-tuned on the same 30 speakers using 10 minutes of audio per speaker (excluding test samples). The fine-tuning process involves training for 100 epochs on a single NVIDIA RTX 3060 12GB GPU. After fine-tuning, the models are tested on the same data from the zero-shot experiment.

Table 1. Comparison of Voice Cloning Models with Zero-Shot Capability and Parameters.

Model	Zero-shot	Fine-tune ability	Params	Text Ref + Audio Ref	Promt + Text Ref + Audio Ref	Audio Orig + Audio Ref	Audio Ref + Speaker ID
XTTS 2	✓	✓	518M	✓			
StyleTTS 2	✓	✓	218M	✓			
YourTTS	✓	✓	94M	✓			
VITS	×	✓	39M				✓
OpenVoice	✓	×	32M	✓			
RVC	×	✓	27M				✓
VoiceCraft	✓	✓	830M		✓		
Vall-E-X	✓	✓	300M		✓		
Natural Speech 3	✓	×	1B			✓	

3.2 Evaluation Metrics

Models are evaluated using:

- **Speaker Embedding Cosine Similarity (SECS):** Measures the retention of the speaker's identity. Values close to 1 are better**,** as they indicate a greater similarity of the speaker's identity.
- **Mel Cepstral Distortion (MCD):** Assesses spectral similarity between synthesized and reference voices. Lower values are better**,** suggesting a closer match to the reference voice and thus better spectral quality.
- **F0 Mean Absolute Error (F0 MAE):** Measures the accuracy of pitch reproduction. Lower values are better**,** as they indicate a more accurate pitch reproduction.
- **F0 Pearson Correlation Coefficient (F0-PCC):** Assesses correlation between generated and reference pitch. Higher values are better, with a value of 1 indicating a perfect correlation, demonstrating that the model effectively captures and replicates the pitch dynamics of the original voice.
- **Universal Target Mean Opinion Score (UTMOS2):** Evaluates subjective voice quality.

Values closer to 4 are better, indicating good voice quality and naturalness.

3.3 Results

In this section, we present the results of our two experiments:

- **Zero-Shot Voice Cloning:** Table 2 shows the performance of 7 out of 9 models capable of zero-shot voice cloning.
- **Fine-Tuning:** Table 3 details the results for 7 out of 9 models fine-tuned with 10 minutes of audio per speaker, comparing their performance to the zero-shot results (OpenVoice and Natural Speech 3 do not have available implementations for fine-tuning).

We analyze the results for male (M) and female (F) voices separately, as presented in the tables. In the zero-shot experiments, the SECS scores indicate that most models effectively preserve speaker identity, with scores ranging from 0.75 to 0.78 for male voices and 0.72 to 0.8 for female voices. Natural Speech 3 and XTTS 2 perform particularly well in this regard. However, spectral fidelity, as measured by MCD (Mel Cepstral Distortion), shows significant disparities. Natural Speech 3 achieves the lowest MCD (9.7 dB for males), indicating better spectral accuracy, while VoiceCraft exhibits much higher distortion, with MCD values reaching 24 dB for males and 23.7 dB for females. Female voices generally suffer from higher MCD values across all models, indicating greater spectral distortion and less natural-sounding results compared to male voices. Pitch accuracy, reflected by F0, varies widely among the models. VoiceCraft shows the highest pitch at 196.4 Hz for males, indicating a significant difference of pitch, while Natural Speech 3 the lowest pitch at 55.9 Hz. For female voices, F0 values also vary, with XTTS 2 producing the highest pitch at 113.8 Hz and Natural Speech 3 the lowest at 72.5 Hz. The F0-PCC (F0 Pearson Correlation Coefficient) scores, which measure pitch contour accuracy, are moderate across the board, with values around 0.3 to 0.4 for both genders. It suggests that while some pitch dynamics are captured, the models struggle with accurate pitch reproduction. UTMOS2 scores reflect these trends, with Natural Speech 3, StyleTTS 2, and OpenVoice achieving the highest perceived quality for male voices (up to 3.6), while female voices generally score lower, reaching the highest result of 3.6 for StyleTTS 2.

In the fine-tuning experiments, SECS scores remain high, showing continued voice resemblance. XTTS 2 and YourTTS maintain good scores, but there is no significant improvement over the zero-shot scenario. Fine-tuning does lead to notable improvements in MCD for some models, especially XTTS 2, which reduces MCD from 16.6 dB to 9.1 dB for males, indicating better spectral fidelity. However, not all models benefit equally; VoiceCraft's MCD remains high, particularly for females, and Vall-E-X experiences a drastic increase in MCD for females, rising from 12.8 dB to 43.9 dB, indicating worsened spectral accuracy. Pitch accuracy shows mixed results post fine-tuning. XTTS 2 improves pitch consistency, reducing F0 from 134.2 Hz to 87.7 Hz for males, aligning better with typical pitch ranges. However, Vall-E-X exhibits worsened F0 accuracy, particularly for females. F0-PCC values remain stable, indicating little improvement in pitch contour accuracy, and UTMOS2 scores show minor gains, with XTTS 2 and YourTTS performing slightly better.

When comparing models across both experiments, XTTS 2 shows significant improvements in spectral fidelity, with MCD reducing by 7.5 dB (from 16.6 dB to 9.1 dB for males), and in pitch accuracy, with F0 improving by 46.5 Hz (from 134.2 Hz to 87.7 Hz for males). YourTTS exhibits moderate gains, reducing MCD by 6.9 dB (from 17.3 dB to 10.4 dB for males) and showing slight improvements in UTMOS2 scores, increasing by 0.1 points for males (from 2.98 to 3.08). After fine-tuning UTMOS2 decreased to 3.43 for males (down from 3.56) and 3.22 for females (down from 3.58) for StyleTTS 2. VoiceCraft performs worse post fine-tuning, with MCD reducing by 7.7 dB for males (from 24 dB to 16.3 dB) but no significant improvements in F0 accuracy, which changes by only 0.3 Hz for males (from 196.4 Hz to 196.1 Hz). UTMOS2 scores for VoiceCraft increase by 0.15 points for males (from 2.81 to 2.96), but there is still room for improvement. Vall- E-X experiences a drastic worsening in spectral fidelity for female voices, with MCD increasing by 31.1 dB (from 12.8 dB to 43.9 dB), and F0 accuracy declining by 25.8 Hz (from 85.3 Hz to 59.5 Hz).

Table 2. Results of voice cloning for zero-shot models.

Model	SECS(⬆)	MCD(⬇)	F0(⬇)	F0-PCC(⬆)	UTMOS2(⬆)
XTTS 2	**M**: 0.78 **F**: 0.77	**M:** 16.6 db **F:** 21 db	**M:** 134.2 Hz **F:** 113.8 Hz	**M:** 0.4 **F:** 0.4	**M**: 3.07 **F**: 2.76
StyleTTS 2	**M**: 0.75 **F**: 0.75	**M:** 10.7 db **F:** 12.9 db	**M:** 91.4 Hz **F:** 92.2 Hz	**M:** 0.34 **F:** 0.3	**M**: 3.56 **F**: 3.58
YourTTS	**M**: 0.77 **F**: 0.76	**M:** 17.3 db **F:** 17.8 db	**M:** 126.6 Hz **F:** 97.6 Hz	**M:** 0.4 **F:** 0.4	**M**: 2.98 **F**:2.56
OpenVoice	**M**: 0.75 **F**: 0.72	**M:** 19.1 db **F:** 15.4 db	**M:** 111.2 Hz **F:** 110.3 Hz	**M:** 0.4 **F:** 0.3	**M**: 3.51 **F**: 3.23
VoiceCraft	**M:** 0.77 **F**: 0.75	**M:** 24 db **F:** 23.7 db	**M:** 196.4 Hz **F:** 99.4 Hz	**M:** 0.3 **F:** 0.4	**M:** 2.81 **F:** 2.6
Vall-E-X	**M**: 0.75 **F**: 0.76	**M:** 16.2 db **F:** 12.8 db	**M:** 84.8 Hz **F:** 85.3 Hz	**M:** 0.3 **F:** 0.3	**M**: 3.06 **F**: 2.89
Natural Speech 3	**M**: 0.78 **F**: 0.8	**M:** 9.7 db **F:** 9.4 db	**M:** 55.9 Hz **F:** 72.5 Hz	**M:** 0.3 **F:** 0.3	**M**: 3.58 **F**: 3.41

Table 3. Results of voice cloning for fine-tuned models.

Model	SECS(⬆)	MCD(⬇)	F0(⬇)	F0-PCC(⬆)	UTMOS2(⬆)
XTTS 2	**M**: 0.77 **F**: 0.76	**M:** 9.1 db **F:** 12.2 db	**M:** 87.7 Hz **F:** 92.6 Hz	**M:** 0.3 **F:** 0.4	**M**: 3.31 **F**: 2.87
StyleTTS 2	**M**: 0.67 **F**: 0.7	**M:** 15.4 db **F:** 11.8 db	**M:** 34.1 **F:** 79.3	**M:** 0.4 **F:** 0.4	**M**: 3.43 **F**: 3.22
YourTTS	**M:** 0.77 **F:** 0.74	**M:** 10.4 db **F:** 14.1 db	**M:** 88.1 Hz **F:** 95.4 Hz	**M:** 0.3 **F:** 0.5	**M**: 3.08 **F**: 2.88
VITS	**M:** 0.53 **F:** 0.58	**M:** 25.8 db **F:** 21.8 db	**M:** 111.9 Hz **F:** 154.8 Hz	**M:** 0.4 **F:** 0.4	**M**: 3.02 **F**: 3.27
VoiceCraft	**M:** 0.76 **F:** 0.73	**M:** 16.3 db **F:** 13.5 db	**M:** 76.8 Hz **F:** 101.4 Hz	**M:** 0.3 **F:** 0.4	**M**: 2.96 **F**: 2.7
Vall-E-X	**M:** 0.67 **F:** 0.75	**M:** 33.2 db **F:** 43.9 db	**M:** 26.8 Hz **F:** 60 Hz	**M**: 0.3 **F:** 0.3	**M:** 2.26 **F:** 2.65
RVC	**M:** 0.72 **F:** 0.71	**M:** 9.9 db **F:** 10.7 db	**M:** 58.9 Hz **F:** 114.2 Hz	**M:** 0.3 **F:** 0.3	**M**: 2.87 **F**: 2.68

4. Conclusion

This study presents a comparison of nine voice cloning algorithms across zero-shot and fine-tuning scenarios. Zero-shot models demonstrate flexibility and satisfactory performance without the need for extensive data, making them highly suitable for rapid deployment. However, these models face challenges in maintaining spectral accuracy, as evidenced by elevated MCD values, particularly for female voices.

Fine-tuning introduces significant improvements in spectral fidelity and pitch accuracy for some models, notably XTTS 2 and YourTTS. XTTS 2 shows a reduction in MCD and an improvement in F0 for males, while YourTTS reduces MCD and slightly improves UTMOS2 scores. However, the impact of fine-tuning is mixed for other models. For instance, StyleTTS 2 experiences a mixed effect on perceived quality with a UTMOS2 increase for males but a slight decrease for females. Meanwhile, VoiceCraft and Vall-E-X exhibit worsened spectral fidelity and pitch accuracy post fine-tuning, especially for female voices.

Overall, fine-tuning successfully enhances certain aspects of voice cloning for specific models and presents opportunities for further refinement to extend these improvements to other models as well.

References

[1]. Edresson Casanova, Kelly Davis, Eren Gölge, Görkem Göknar, Iulian Gulea, Logan Hart, Aya Aljafari, Joshua Meyer, Reuben Morais, Samuel Olayemi, Julian Weber. XTTS: a Massively Multilingual Zero-Shot Text-to-Speech Model. arXiv:2406.04904, 2024.

[2]. Yinghao Aaron Li, Cong Han, Vinay S. Raghavan, Gavin Mischler, Nima Mesgarani. StyleTTS 2: Towards Human-Level Text-to-Speech through Style Diffusion and Adversarial Training with Large Speech Language Models. arXiv preprint arXiv:2306.07691, 2023

[3]. Edresson Casanova, Julian Weber, Christopher Shulby, Arnaldo Candido Junior, Eren Gölge, Moacir Antonelli Ponti. YourTTS: Towards Zero-Shot Multi-Speaker TTS and Zero-Shot Voice Conversion for everyone. arXiv preprint arXiv:2112.02418, 2023.

[4]. Zengyi Qin, Wenliang Zhao, Xumin Yu, Xin Sun. OpenVoice: Versatile Instant Voice Cloning. arXiv preprint arXiv:2312.01479v5, 2024.

[5]. Puyuan Peng, Po-Yao Huang, Daniel Li, Abdelrahman Mohamed, David Harwath. VoiceCraft: Zero-Shot Speech Editing and Text-to-Speech in the Wild. arXiv preprint arXiv:2403.16973v1, 2024.

[6]. Ziqiang Zhang, Long Zhou, Chengyi Wang, Sanyuan Chen, Yu Wu, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, Furu Wei. Speak Foreign Languages with Your Own Voice: Cross-Lingual Neural Codec Language Modeling. arXiv preprint arXiv:2303.03926v1, 2023.

[7]. Zeqian Ju, Yuancheng Wang, Kai Shen, Xu Tan, Detai Xin, Dongchao Yang, Yanqing Liu, Yichong Leng, Kaitao Song, Siliang Tang, Zhizheng Wu, Tao Qin, Xiang-Yang Li, Wei Ye, Shikun Zhang, Jiang Bian, Lei He, Jinyu Li, Sheng Zhao. NaturalSpeech 3: Zero-Shot Speech Synthesis with Factorized Codec and Diffusion Models. arXiv:2403.03100, 2024

[8]. Jaehyeon Kim, Jungil Kong, Juhee Son. Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech. arXiv:2106.06103

[9]. Retrieval based Voice Cloning. Available at the link: https://github.com/RVC-Project/Retrieval-based-Voice-Conversion-WebUI

[10]. Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. arXiv:1609.03499, 2016.

[11]. Sercan O. Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, Shubho Sengupta, Mohammad Shoeybi. Deep Voice: Real-time Neural Text-to-Speech. arXiv:1702.07825, 2017.

[12]. Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu. Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis. arXiv:1806.04558, 2018.

[13]. James Betker. Better speech synthesis through scaling. arXiv:2305.07243, 2023.

[14]. Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, Rif A. Saurous. Tacotron: Towards End-to-End Speech Synthesis. arXiv:1703.10135, 2017.

[15]. Jaehyeon Kim, Sungwon Kim, Jungil Kong, Sungroh Yoon. Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search. arXiv:2005.11129, 2020.
[16]. L. Wan, Q. Wang, A. Papir, and I. L. Moreno, “Generalized end-to-end loss for speaker verification” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 487–488.
[17]. Robert F. Kubichek. Mel-cepstral distance measure for objective speech quality assessment. Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing, 1:125–128 vol.1, 1993.
[18]. J. Benesty, J. Chen, Y. Huang, and I. Cohen, “Pearson correlation coefficient,” in Noise reduction in speech processing. Springer, pp. 1–4, 2009.
[19]. Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. IEEE Transactions on Acoustics, Speech, and Signal Processing, 32(2):236–243, 1984.
[20]. Veaux, Christophe; Yamagishi, Junichi; MacDonald, Kirsten. CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit, [sound]. University of Edinburgh. The Centre for Speech Technology Research (CSTR). https://doi.org/10.7488/ds/1994, 2017.

Информация об авторах / Information about authors

Ольга ОГАНЕСЯН – научный сотрудник Центра передовых программных технологий (CAST) и аспирант Российско-Армянского университета, специализируется на математическом и программном обеспечении вычислительных систем. Получила степень бакалавра по информатике и прикладной математике в Российско-Армянском Университете Армении (2021) и степень магистра по интеллектуальным системам и робототехнике в Российско-Армянском университете (2023). Её исследования сосредоточены на обработке речи и синтезе голоса, включая клонирование голоса, а также на развитии методов машинного обучения.

Olga HOVHANNISYAN is a researcher at the Center of Advanced Software Technologies (CAST) and a Ph.D. student at Russian-Armenian University, specializing in mathematical and software support for computing systems. She holds a B.Sc. in Informatics and Applied Mathematics from the Russian- Armenian University of Armenia (2021) and an M.Sc. in intellectual systems and robotics from Russian-Armenian University (2023). Her research centers on speech processing and voice synthesis, including voice cloning, as well as advancements in machine learning.

Давид САРГСЯН студент бакалавра в Российско-Армянского университете по направлению Прикладной Математики и Информатики. Он также является исследователем в Центре Передовых Программных Технологий (CAST). Его научные интересы включают автоматическое распознавание речи, технологии синтеза речи и большие языковые модели (LLM).

David SARGSYAN is a Bachelor’s student in Applied Mathematics and Informatics at the Russian-Armenian University. He is also a researcher at the Center of Advanced Software Technologies (CAST). His research interests include speech recognition, text-to-speech technologies, and large language models (LLMs).

Артур МАЛАДЖЯН – бакалавр в области информатики и прикладной математики в Российско-Армянском Университете Армении (2020). Магистр в области машинного обучения в Российско-Армянском Университете, Армения (2022). В настоящее время он является исследователем в Центре Передовых Программных Технологий (CAST). Сфера научных интересов: обработка естественного языка и голосовые технологии.

Artur MALAJYAN received his Bachelor’s degree in Informatics and Applied Mathematics from the Russian-Armenian University in 2020. In 2022, he earned a Master’s degree in Machine Learning from Russian-Armenian University, Armenia. He is currently a researcher at the Center of Advanced Software Technologies (CAST). His research interests include natural language processing (NLP) and voice technologies.

DOI: 10.15514/ISPRAS-2024-36(4)-2

Detecting Errors in the Pandas Software Module using the Svace Static Code Analyzer

M.A. Lapina, ORCID: 0000-0001-8117-9142 <mlapina@ncfu.ru>
M.I. Khodakov, ORCID: 0009-0007-6848-7184<mxkhodakov@yandex.ru>
S.K. Grobova, ORCID: 0009-0000-2608-7712 <sofya.grobova@yandex.ru>

North Caucasus Federal University,
1, Pushkina st., Stavropol, 355017, Russia.

Abstract. The article deals with the urgent problem of software security at the early stage of its development. Special attention is paid to static code analysis, which is a key tool for detecting vulnerabilities at early stages of the software development life cycle. The article emphasizes the importance of integrating static analysis tools into the development process in order to detect and eliminate vulnerabilities early. The methods of static analyzers' error search are considered, as well as the main components of the Svace static analyzer developed at the Institute of System Programming of the Russian Academy of Sciences. Classification of analyses used in the Svace static analyzer is presented. Static analysis of source code in Python programming language is considered in detail. As a practical example the analysis of the Pandas 2.2.1 project performed with the help of Svace is given. The result was the detection of 241 vulnerabilities for 590709 lines of code, which shows a high density of warnings per million lines of code and confirms the effectiveness of static analysis in ensuring software security.

Keywords: static code analysis; static analyzer; lexical analyzers; lightweight analyzers; abstract syntax tree; code fragment.

For citation: Lapina M.A., Khodakov M.I., Grobova S.K. Detecting errors in the Pandas software module using the Svace static code analyzer. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024. pp. 17-26. DOI: 10.15514/ISPRAS-2024-36(4)-2.

Выявление ошибок в программном модуле Pandas с помощью статического анализатора Svace

М.А. Лапина, ORCID: 0000-0001-8117-9142 <mlapina@ncfu.ru>
М.И. Ходаков, ORCID: 0009-0007-6848-7184 <mxkhodakov@yandex.ru>
С.К. Гробова, ORCID: 0009-0000-2608-7712 <sofya.grobova@yandex.ru>

Северо-Кавказский федеральный университет,
355017, Россия, г. Ставрополь, ул. Пушкина, д. 1.

Аннотация. В статье рассматривается актуальная проблема обеспечения безопасности программного обеспечения на раннем этапе его разработки. Особое внимание уделяется статическому анализу кода, который является ключевым инструментом для выявления уязвимостей на ранних этапах жизненного цикла разработки программного обеспечения. Статья подчеркивает важность интеграции инструментов статического анализа в процесс разработки с целью раннего обнаружения и устранения уязвимостей. Рассмотрены методы поиска ошибок статических анализаторов, а также основные компоненты статического анализатора Svace, разработанного в Институте системного программирования РАН. Представлена классификация анализов, используемых в статическом анализаторе Svace. Детально рассмотрен статический анализ исходного кода на языке программирования Python. В качестве практического примера приведен анализ проекта Pandas 2.2.1, выполненный с помощью Svace. Результатом послужило выявление 241 уязвимости на 590709 строк кода, что показывает высокую плотность предупреждений на миллион строк кода и подтверждает эффективность статического анализа в обеспечении безопасности программного обеспечения.

Ключевые слова: статический анализ кода; статический анализатор; лексические анализаторы; легковесные анализаторы; абстрактное синтаксическое дерево; фрагмент кода.

Для цитирования: Лапина М.А., Ходаков М.И., Гробова С.К. Выявление ошибок в программном модуле Pandas с помощью статического анализатора Svace. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 17–26 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(4)–2

1. Introduction

Nowadays, software is used everywhere. Statistics shows that as the number of programs increases, so does the number of vulnerabilities in them. That's why special attention is paid to software analysis.

At present there are two most popular methods of code analysis – dynamic analysis and static analysis. It is impossible to imagine software development without using static analysis tools. Static analysis is a type of analysis when the program is not executed but its whole code is analyzed [1]. This method is most often implemented at the initial stages of development, which helps to detect vulnerabilities earlier and eliminate them faster. Static analysis of source code can be performed in two ways - manually and with the help of special software tools.

In manual analysis of source code, checks such as code review or code inspection are performed. This approach is actively used because a human is able to detect defects in source code that software analyzers cannot do yet.

Software analysis is performed using various analyzers. Many methods of static error search in programming have developed from the sphere of program compilation and operate with abstractions. Depending on the abstractions used, error search methods [2] can be divided into several groups:

- Lexical analyzers are designed to break down the source code of a program into individual tokens or tokens. They can be used to find only the simplest types of defects [3].
- Lightweight analyzers, also known as first-level analyzers, check the text for compliance with some grammar and build a parse tree (an abstract syntactic tree) by a linear sequence of tokens of this text, which is well suited for further processing and analysis of the text [4].

- More sophisticated parsers (Level 2 parsers), which use more complex algorithms and methods for parsing related to phases after syntactic analysis [5].

The disadvantage [6] of this method is that the analyzer may consider absolutely safe code fragments suspicious. Excessive suspiciousness leads to an increase of the false/true alarms ratio [7].

In this article we will consider the Svace static analyzer developed at the Ivannikov Institute for System Programming of the Russian Academy of Sciences (ISP RAS) and analyze the Pandas 2.2.1 project using it. This analyzer detects a large number of vulnerabilities in code, has a high level of true positives, fine-tuning and a large number of supported programming languages were the reasons for choosing this product.

2. Description of Svace

Svace is a tool developed at the Institute of System Programming of the Russian Academy of Sciences that performs static error search using several types of analyzers. It combines the key qualities of foreign analogs (Synopsis Coverity Static Analysis, Perforce Klocwork Static Code Analysis, Fortify Static Code Analyzer) with the unique use of open industrial compilers in order to maximize support for new programming language standards [8]. Supported languages are C, C++, C#, Java, Kotlin, Go, Python, Scala.

2.1 Svace analysis scheme

The schematic in (Fig. 1) shows the general analysis scheme of the Svace static analyzer [9].

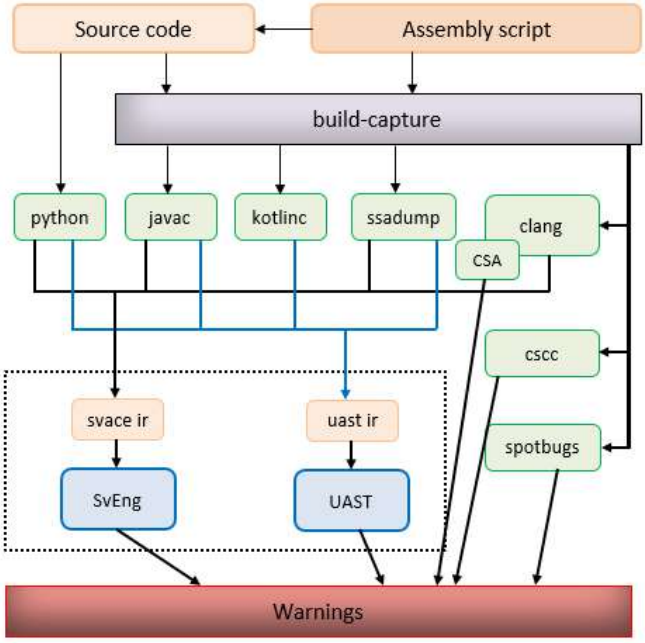


Fig.1. Schematic of the Svace static analyzer [9].

2.2 Svace components

The Svace static analyzer includes five components – SvEng, UAST, SpotBugs, CSA and cscc.

SvEng is the main component of Svace. It uses the following types of static analysis: pattern search in AST-trees, data flow analysis, interprocedural symbolic execution with state merging and using method summaries, static analysis of labeled data for security errors.

UAST – component in which errors are searched in a unified abstract syntax tree.

SpotBugs - component intended for analyzing programs in Java programming language.

CSA – intended for analyzing software in C/C++ languages.

cscc – intended for analyzing programs in C# language. This component is based on the Roslyn compiler. For searching errors, it uses the same types of analysis as SvEng.

2.3 Classification of analyses used at Svace

The analyses used in the Svace static analyzer [9] can be divided into the following groups:

- Analyses based on pattern search in the abstract syntax tree (ASD);
- Interprocedural analyses based on summaries;
- Statistical analyses;
- Special function analyses;
- Symbolic execution-based analyses;
- Labeled function analyses.

3. Analyzing a Python project with Svace

Analyzing the source code of a Python application does not require building the project because it is an interpreted language. Two components are involved in the analysis: SvEng and UAST.

Pandas was chosen as a project written in Python. 590709 lines of code were checked and 241 vulnerabilities were detected as a result of the analysis. The density of warnings per million code lines is 407.98. Table 1 shows the analysis results.

Fig. 2 shows a code fragment where the INVARIANT_RESULT warning was detected in the project Pandas pandas/tests/window/moments/test_moments_consistency_ewm.py. The essence of this error is that the value of the expression does not depend on the values of its parts, or a part of the expression can be omitted without changing the result.

Table 1. Results of the analysis.

Warning	Number of warnings
INVARIANT_RESULT	93
DIVISION_BY_ZERO.EX	23
BAD_COPY_PASTE	8
CATCH.NO_BODY	107
MUTABLE_DEFAULT_ARGUMENT	1
SIMILAR_BRANCHES	7
WRONG_ARGUMENTS_ORDER	2

```
35      if adjust:
36          count = 0
37          for i in range(len(s)):
38              if s.iat[i] == s.iat[i]:# INVARIANT_RESULT
39                  w.iat[i] = pow(1.0 / (1.0 - alpha), count)
40                  count += 1
41              elif not ignore_na:
42                  count += 1
```

Fig. 2. Error INVARIANT_RESULT.

Fig. 3 shows the DIVISION_BY_ZERO.EX error in the project Pandas pandas/tests/io/parser/test_skiprows.py. This error occurs whenever the program tries to divide an interval or a numeric value by 0.

The following types of errors were detected on the abstract syntax tree [10] using the component UAST (unified abstract syntax tree).

Fig. 4 shows a code fragment with BAD_COPY_PASTE error in the project Pandas pandas/tests/extension/base/constructors.py. An error occurs if the code has been copied and reproduced, but not all necessary changes have been made. Fig. 5 shows the CATCH.NO_BODY error in the project Pandas pandas/tests/plotting/test_converter.py. This error occurs if an exception was caught but not handled by the corresponding except block.

```
285  def test_skip_rows_bad_callable(all_parsers):
286      msg = "by zero"
287      parser = all_parsers
288      data = "a\n1\n2\n3\n4\n5"
289
290      with pytest.raises(ZeroDivisionError, match=msg):
291          parser.read_csv(StringIO(data), skiprows=lambda x: 1 / 0)
```

Fig. 3. Error DIVISION_BY_ZERO.EX.

```
28          if hasattr(result._mgr, "blocks"): # Original code
29              assert isinstance(result._mgr.blocks[0], EABackedBlock)
30          assert result._mgr.array is data
31
32
33          result2 = pd.Series(result)
34          assert result2.dtype == data.dtype
35          if hasattr(result._mgr, "blocks"): # Bad copy paste
36              assert isinstance(result2._mgr.blocks[0], EABackedBlock)
```

Fig. 4. Error BAD_COPY_PASTE.

```
37  try:
38      from pandas.plotting._matplotlib import converter
39  except ImportError: # CATCH.NO_BODY
40
41
42      pass
```

Fig. 5. Error CATCH.NO_BODY.

Fig. 6 shows a code fragment with the MUTABLE_DEFAULT_ARGUMENT error in the project Pandas pandas/io/formats/style_render.py. The error occurs if we use some modifiable object (list, dictionary) as a default value for a function argument. If we change the value of the argument inside the function, we will have to change the original value because they both refer to the same object, in this case it is the css_props argument. This can lead to unexpected results and errors in your code.

Fig. 7 shows a code fragment with SIMILAR_BRANCHES_ARGUMENT error in the project Pandas pandas/tests/indexes/datetimes/test_date_range.py. This error occurs when executing the same instructions regardless of the condition.

Fig. 8 shows a code fragment with WRONG_ARGUMENTS_ORDER ARGUMENT error in the project Pandas pandas/_testing/__init__.py. The error occurs when method arguments are passed in the wrong order. In this example, left and right are mixed up in the wrong order.

```
2154    def __init__(
2155        self,
2156        css_props: CSSProperties = [  #MUTABLE_DEFAULT_ARGUMENT
2157            ("visibility", "hidden"),
2158            ("position", "absolute"),
2159            ("z-index", 1),
2160            ("background-color", "black"),
2161            ("color", "white"),
2162            ("transform", "translate(-20px, -20px)"),
2163        ],
```

Fig. 6. Error MUTABLE_DEFAULT_ARGUMENT.

```
71    elif inclusive_endpoints == "both":
72        expected_range = both_range[:] # First branch
73    else:
74        expected_range = both_range[:] # Second branch
```

Fig. 7. Error SIMILAR_BRANCHES.

```
495 if isinstance(left, np.ndarray) and isinstance(right, np.ndarray):
496     return np.shares_memory(left, right)
497 elif isinstance(left, np.ndarray):
498
499     return shares_memory(right, left) # WRONG_ARGUMENTS_ORDER
```

Fig. 8. Error WRONG_ARGUMENTS_ORDER.

4. Classification of errors

Errors in software can be classified by the degree of their influence on the program operation [11]. Table 2 presents the table of error classification and their description. Errors obtained as a result of static analysis were classified by the level of their influence on the program code. Table 3 presents the error classification table.

The DIVISION_BY_ZERO.EX error belongs to the "Critical error" class because division by zero in the program code may cause unpredictable behavior of the command and eventually lead to its crash. Fig. 9 shows the scenario when this error occurs.

The MUTABLE_DEFAULT_ARGUMENT error belongs to the "Major error" class because it can cause unexpected results when using a variable data type as a default value for an argument. Fig.10 shows a scenario where this error occurs. In this code, the item function uses the variable data type (list) as the default value for the item_list argument. As a result, each time the add_item function is called, items are added to the same item_list instead of a new list. This is the MUTABLE_DEFAULT_ARGUMENT error.

The WRONG_ARGUMENTS_ORDER error is a "Major error" because if the order of arguments in a function call is incorrect, it can lead to unexpected program behavior. Fig. 11 shows the scenario of this error. In this example, the main function expects the first argument to be "x" and the second argument to be "y". However, when the function is called, these arguments are specified in reverse order, which causes an error.

The errors INVARIANT_RESULT, BAD_COPY_PASTE, SIMILAR_BRANCHES, CATCH.NO_BODY are not serious and belong to the "Minor error" class, but they require corrections to prevent unexpected program behavior.

Fig. 12 shows the SIMILAR_BRANCHES error scenario.

In this example, the if and else branches contain the same instructions y = x*2, which may affect the efficiency of the program.

Table 2. Classification of errors and their description.

Error class	Description
Critical errors	This class of errors leads to an emergency situation that makes the program operation impossible.
Major errors	Errors that lead the program to unexpected results.
Minor errors	Errors that reduce the efficiency of the program and its performance.

Table 3. Classification of errors.

Class	Error name
Critical error	DIVISION_BY_ZERO.EX
Major error	MUTABLE_DEFAULT_ARGUMENT
	WRONG_ARGUMENTS_ORDER
Minor error	INVARIANT_RESULT
	BAD_COPY_PASTE
	SIMILAR_BRANCHES
	CATCH.NO_BODY

```
1   def divide_numbers(num1, num2):
2       return num1 / num2
3
4   def main():
5       values = [10,0]
6       for i in range(len(values) - 1):
7           print(f"division {values[i]} by {values[i+1]}")
8           outcome = divide_numbers(values[i],values[i+1])
9           print(f"Result: {outcome}")
10
11          if __name__ == "__main__":
12              main()
```

Fig. 9. Error scenario DIVISION_BY_ZERO.EX.

```
def item(item, item_list=[]):
    item_list.append(item)
    return item_list


print(item('apple'))
print(item('banana'))
```

Fig.10. Error scenario MUTABLE_DEFAULT_ARGUMENT.

One of the examples of the BAD_COPY_PASTE error is shown in Fig. 13. In this example, the calculate difference function should calculate the difference between variables a and b, but due to code copying, an error was made that may cause the program to work incorrectly.

The CATCH.NO_BODY error scenario is shown in Fig. 14. In this example, the exception is not handled in the except block, which may cause the error to be ignored and the program to continue execution without handling the error. Fig. 15 shows the INVARIANT_RESULT error scenario. In this example, the value of result1 does not depend on the variables x, y, z, because multiplication by 0 will always result in 0. The value of result2 will always be x + z because y - y is 0, so this part of the expression can be omitted without changing the result. Similarly with the value of result3, it will always be equal to x + y because z - z is 0.

```
def main(x, y):
    return f"{y}, {x}!"

print(main("x", "y"))
```

Fig.11. Error scenario WRONG_ARGUMENTS_ORDER.

```
x = 10

if x > 5:
    y = x * 2
else:
    y = x * 2
```

Fig. 12. Error scenario SIMILAR_BRANCHES.

```
def calculate_sum(a, b):
    return a + b


def calculate_difference(a, b):
    return a + b
```

Fig.13. Error scenario BAD_COPY_PASTE.

```
try:
    x = 1 / 0
except ZeroDivisionError:

    pass
```

Fig.14. Error scenario CATCH.NO_BODY.

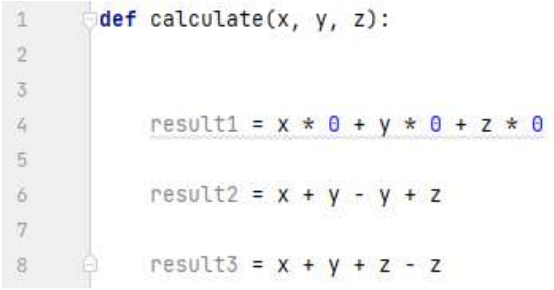


Fig.15. Error scenario INVARIANT_RESULT.

5. Conclusion

As it was shown in the article, static analysis tools help detect errors at the initial stages of application development. Different static analyzers use different algorithms for detecting vulnerabilities in application source code. Svace static analyzer allows you to analyze program code in different languages. We also analyzed the project in Python, Pandas 2.2.1. In the process of analysis, we found errors of different classes. Classification of errors by their level of influence on the program code, their full description and examples of scenarios when these errors can occur in the program code were carried out. The results of the research may be useful for improving the quality and reliability of the product.

References

[1]. Chernov D. Code analysis: problems, solutions, perspectives. 2022. URL: https://www.tadviser.ru/index.php/Статья:Анализ_кода:_проблемы,_решения,_перспективы#:~:text=У метода есть два недостатка,программы%2C который не всегда доступен, accessed 20.03.2024.

[2]. Borodin A.E., Belevantsev A.A. Static analyzer Svace as a collection of analyzers of different levels of complexity. Proceedings of ISP RAS, vol. 27, issue 6, 2015, pp. 111-134. (in Russian). DOI: 10.15514/ISPRAS-2015-27(6)-8.

[3]. Zaboleva-Zotova A.V., Orlova Y.A. MODELING OF LEXICAL ANALYSIS OF TECHNICAL TASK TEXT. URL: https://www.elibrary.ru/item.asp?id=9506673, accessed 21.03.2024.

[4]. Mavchun E.V. Comparison of algorithms of tabular bottom-up syntactic analysis. URL: https://oops.math.spbu.ru/SE/dip loma/2015/s/544-Mavchun-report.pdf, accessed 24.03.2024.

[5]. A. P. Syzranov, A. I. Nenakhov, A. Y. Bolotov, A. J. Osipov. Development of criteria for evaluation of means of automation of certification testing on the level of control of the absence of undeclared capabilities. URL: https://elibrary.ru/item.asp?id=47856981, accessed 24.03.2024.

[6]. Fadeev S.G. TECHNOLOGY OF STATIC ANALYSIS OF SOURCE CODE. URL: https://apni.ru/media/Sbornik-6-3.pdf#page=131, accessed 20.03.2024.

[7]. Kuchin I.Yu. Review of existing methods of program code analysis. URL: https://cyberleninka.ru/article/n/obzor-suschestvuyuschih-metodov-analiza-programmnogo-koda/viewe, accessed 21.03.2024.

[8]. Svace static analyzer. URL: https://www.ispras.ru/technologies/svace/, accessed 24.03.2024.

[9]. Svace components. URL: https://svace.pages.ispras.ru/svace-website/2023/10/04/analysis-types.html, accessed 24.03.2024.

[10]. Afanasyev V.O., Borodin A.E., Vikhlyantsev K.I., Belevantsev A.A. Static analysis based on generalized abstract syntactic tree. Proceedings of ISP RAS, vol. 35, issue 6, 2023, pp. 103-120. (in Russian). DOI: 10.15514/ISPRAS–2023–35(6)–6.

[11]. V.V. Bykova, G.E. Glukhov, A.N. Sharypov, P.E. Chernikov, S.V. Koval, A.Yu. Konkov. PROBLEMS OF VULNERABILITY OF INFORMATION SYSTEMS OF AVIATION INDUSTRY ENTERPRISES: ANALYSIS AND CLASSIFICATION OF ERRORS. URL: https://mlgvs.ru/files/iac/art-niiga-2019-27.pdf, accessed 01.04.2024.

Информация об авторах / Information about authors

Мария Анатольевна ЛАПИНА – кандидат физико-математических наук, доцент, доцент кафедры информационной безопасности автоматизированных систем Северо-Кавказского федерального университета. Сфера научных интересов: цифровые технологии, киберфизические системы, анализ данных, управление информационной безопасностью, доверенный искусственный интеллект, криптография, анализ программного кода.

Maria Anatolyevna LAPINA – Cand. Sci. (Phys.-Math.), Associate Professor, Associate Professor of the Department of Information Security of Automated Systems of the North Caucasus Federal University. Research interests: digital technologies, cyber-physical systems, data analysis, information security management, trusted artificial intelligence, cryptography, program code analysis.

Максим Иванович ХОДАКОВ – студент кафедры информационной безопасности автоматизированных систем Северо-Кавказского федерального университета. Сфера научных интересов: программирование, цифровые технологии, анализ программного кода.

Maxim Ivanovich KHODAKOV – student of the Department of Information Security of Automated Systems of the North Caucasus Federal University. Research interests: programming, digital technologies, program code analysis.

Софья Кирилловна ГРОБОВА – студентка кафедры инфокоммуникаций Северо-Кавказского федерального университета. Сфера научных интересов: программирование, цифровые технологии, анализ программного кода.

Sofya Kirillovna GROBOVA – student of the Department of Infocommunications of the North Caucasus Federal University. Her research interests: programming, digital technologies, program code analysis.

DOI: 10.15514/ISPRAS-2024-36(4)-3

Разработка безопасного компилятора на основе Clang

[1] П.Д. Дунаев, ORCID: 0000-0002-9142-0945, <p.dunaev@ispras.ru>
[1,2] А.А. Синкевич, ORCID: 0009-0002-3364-6468, <artsin666@gmail.com>
[1,3] А.М. Гранат, ORCID: 0009-0007-6589-3347, <a.granat@ispras.ru>
[2] И.А. Батраева, ORCID: 0000-0002-6539-8473, <batraevaia@info.sgu.ru>
[2] С.В. Миронов, ORCID: 0000-0003-3699-5006, <mironovsv@sgu.ru>
[1,4] Н.Ю. Шугалей, ORCID: 0009-0000-9310-8317, <shugaley@ispras.ru>

[1] Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
[2] Саратовский государственный университет имени Н.Г. Чернышевского,
410012, Россия, Саратов, ул. Астраханская, 83.
[3] Высшая школа экономики,
101000, Россия, г. Москва, ул. Мясницкая, д. 20.
[4] Московский физико-технический институт
(национальный исследовательский университет),
Россия, 117303, г. Москва, ул. Керченская, д.1А, корп. 1.

Аннотация. В связи с использованием современными компиляторами C/C++ агрессивных оптимизаций, эксплуатирующих неопределённое поведение, существует потребность в безопасном компиляторе, который не проводит подобные оптимизации, а также предотвращает использование разработчиком небезопасных конструкций. В ИСП РАН был реализован безопасный компилятор на основе GCC, однако часть разработчиков предпочитает GCC Clang, который не лишён проблемы эксплуатации неопределённого поведения. В этой работе рассматриваются возможности Clang по осуществлению безопасной компиляции и описывается реализация безопасного компилятора на его основе. Для созданного безопасного компилятора показывается применимость на практике и оценивается влияние на производительность программ.

Ключевые слова: компилятор; уязвимость; неопределенное поведение; Clang; LLVM; C; C++.

Для цитирования: Дунаев П.Д., Синкевич А.А., Гранат А.М., Батраева И.А., Миронов С.В., Шугалей Н.Ю. Разработка безопасного компилятора на основе Clang. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 27–40. DOI: 10.15514/ISPRAS–2024–36(4)–3.

Developing a Clang-Based Safe Compiler

[1] *P.D. Dunaev, ORCID: 0000-0002-9142-0945, <p.dunaev@ispras.ru>*
[1,2] *A.A. Sinkevich, ORCID: 0009-0002-3364-6468, <artsin666@gmail.com>*
[1,3] *A.M. Granat, ORCID: 0009-0007-6589-3347, <a.granat@ispras.ru>*
[2] *I.A. Batraeva, ORCID: 0000-0002-6539-8473, <batraevaia@info.sgu.ru>*
[2] *S.V. Mironov, ORCID: 0000-0003-3699-5006, <mironovsv@sgu.ru>*
[1,4] *N.U. Shugaley, ORCID: 0009-0000-9310-8317, <shugaley@ispras.ru>*

[1] *Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Saratov State University, 83 Astrakhanskaya Street, Saratov, 410012, Russia.*
[3] *Higher School of Economics, 20, Myasnitskaya Street, Moscow, 101000, Russia*
[4] *Moscow Institute of Physics and Technology (National Research University), 1 A Kerchenskaya st., Moscow, 117303, Russia.*

Abstract. Due to the use of aggressive optimizations by modern C/C++ compilers that exploit undefined behavior, there is a need for a safe compiler that does not perform such optimizations and prevents developers from using unsafe statements and expressions. Such a safe compiler based on GCC has been developed in ISP RAS, but some developers prefer Clang instead of GCC, which has mainly the same problems of exploiting undefined behavior. This paper examines the capabilities of Clang to perform safe compilation and describes the implementation of a safe compiler based on it. For the created safe compiler, the applicability in practice is shown and the impact on program performance is evaluated.

Keywords: compiler; vulnerability; undefined behavior; Clang; LLVM; C; C++.

For citation: Dunaev P.D., Sinkevich A.A., Granat A.M., Batraeva I.A., Mironov S.V., Shugaley N.U. Developing a Clang-Based Safe Compiler. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024. pp. 27-40 (in Russian). DOI: 10.15514/ISPRAS-2024-36(4)-3.

1. Введение

Последнее время большую популярность в качестве компилятора языка C++ завоевал Clang [1]. Согласно статистике JetBrains [2], в 2023 году компилятор использовали более трети опрошенных, что характеризовало его как второй по популярности компилятор C++. Clang имеет определённые преимущества перед конкурирующими разработками, в том числе перед самым популярным компилятором – GCC [3]. Среди таких преимуществ можно выделить лицензию на основе Apache 2.0 [4], которая позволяет использовать исходный код компилятора для проектов под бóльшим числом лицензий. Существенным преимуществом Clang также является то, что он построен на компиляторной инфраструктуре LLVM [5], ценность которой заключается в том, что при доработке компилятора нередко можно полностью сфокусироваться на преобразованиях промежуточного представления LLVM IR, что было использовано, например, в работах [6-9].

При этом Clang, как и GCC, обладает существенным недостатком – он осуществляет оптимизации, эксплуатирующие небезопасное поведение. Так, в работе [10] приводится ряд недостатков, многие из которых напрямую относятся к Clang. Примером оптимизации, выполняемой рассматриваемым компилятором, является удаление проверок вида `if(1<<X==0)`, где `X` – целое число. Результат выполнения инструкции побитового сдвига единицы влево, которое в подобных случаях обычно ожидается программистом, на некоторых архитектурах процессоров, таких как PowerPC, равен нулю. Компилятор же считает такую проверку избыточной, так как может доказать, что в отсутствие неопределённого поведения это утверждение всегда ложно, и потому при наличии такового снимает с себя всякие обязательства, если с помощью опций не задано иное. Отключение данной оптимизации с помощью опций невозможно. Таким образом, при компиляции

проекта с помощью Clang, необходима дополнительная защита от внесения программистом и компилятором уязвимостей в выходной код.

Потребность в избавлении кода программы от уязвимостей могут помочь удовлетворить различные методы, такие как, например, статический и динамический анализ, а также полноценное тестирование и использование стандартов безопасного кодирования. Авторы статьи [11] показывают, что существуют ситуации, когда ни один из данных способов не применим для решения проблемы устранения создаваемых компилятором уязвимостей. Авторы предлагают альтернативный путь решения – безопасный компилятор, в который встроена функциональность, предотвращающая уязвимости, вносимые агрессивными оптимизациями, и предупреждающая об уязвимостях, вносимых программистом. В статье описывается безопасный компилятор [12], основанный на GCC. Однако GCC не может выступать в качестве замены Clang, поскольку Clang не является полностью совместимым с GCC [13], и, по причине ряда вышеописанных преимуществ, вероятно, не все разработчики будут готовы отказаться от Clang. Таким образом, становится актуальной проблема безопасной компиляции посредством Clang.

2. Концепция безопасного компилятора

В статье [11] описана концепция безопасного компилятора. Безопасный компилятор, чтобы считаться таковым, должен удовлетворять следующим требованиям:

- Компилятор не может вносить во время выполнения оптимизаций уязвимости в генерируемый код;
- Компилятор обязан не удалять код, исходя из предположения об отсутствии неопределённого поведения, не сильно замедляя при этом работу выходной программы;
- Правки, требуемые для успешной компиляции исходного кода, минимально возможны;
- Компилятор не предоставляет возможность отключения опций, контролирующих выполнение первых двух требований.

Такой компилятор может работать в качестве замены ранее используемого, однако помимо основного своего преимущества – профилактики уязвимостей – компилятор будет наделён и недостатками в виде замедления работы компилируемого приложения и необходимости, пусть и минимальной, модификации исходного кода, что в ряде случаев может быть затруднительно или невозможно.

Детализация требований к безопасному компилятору приведена в стандарте [14]. Выделяется три класса требований, каждый из которых характеризуется строгостью механизмов предотвращения уязвимостей и скоростью работы генерируемых программ. Требованиям стандарта может соответствовать как написанный с нуля компилятор, так и доработанный либо правильно сконфигурированный существующий. Поскольку в данной работе в качестве целевой аудитории приняты пользователи Clang, вариант написания нового компилятора не рассматривается; возможности конфигурации Clang для выполнения требований к безопасному компилятору рассмотрены в разделе 3, в разделе 4 описаны работы, проведённые для доработки Clang до безопасного компилятора, а в разделе 5 приведены результаты тестирования безопасного Clang на реальных приложениях.

3. Соответствие возможностей Clang требованиям к безопасному компилятору

Безопасный компилятор третьего класса должен выполнять только безопасные преобразования исходного и машинного кода. Например, компилятор может преобразовать условие `N+1>N` в `true`, так как принимает за истину то, что неопределённого поведения

произойти не может (в данном случае – целочисленного переполнения), и третий класс безопасности должен гарантировать защиту от преобразований такого вида. Ограничение на преобразования и соответствующие им опции Clang представлены в табл. 1.

Помимо этого, компилятор третьего класса безопасности должен включать механизмы повышенной защищенности. Требуемые механизмы и соответствующие им опции Clang представлены в табл. 2.

Также одной из задач третьего класса безопасности является выдача предупреждений в ходе сборки программы в некоторых случаях неопределенного поведения. Опции Clang, включающие выдачу соответствующих предупреждений, представлены в табл. 3.

Табл. 1. Реализация требований 3 класса по отключению небезопасных преобразований в Clang.
Table 1. Implementation of class 3 requirements for disabling unsafe transformations in Clang.

Пункт стандарта	Требование	Опция
5.2.1 а	Отключение преобразований, связанных с целочисленным переполнением	`-fwrapv`
5.2.1 б	Отключение преобразований, связанных с тем фактом, что значения указателей разных типов могут совпадать	`-fno-strict-aliasing`
5.2.1 в	Отключение преобразований, связанных с разыменованием нулевых указателей	`-fno-delete-null-pointer-checks`
5.2.1 г	Отключение преобразований, связанных с делением на ноль и взятием нулевого остатка	—
5.2.1 д	Отключение преобразований, связанных со значениями аргументов побитового сдвига	—

Табл. 2. Реализация требований 3 класса по включению механизмов повышенной защищённости в Clang.
Table 2. Implementation of class 3 requirements for enabling increased safety mechanisms in Clang.

Пункт стандарта	Требование	Опция
5.2.2 а	Защита от переполнения буфера постоянного размера при вызове функций стандартной библиотеки	`-D_FORTIFY_SOURCE=2` с оговоркой: при возникновении ошибки, данные функции выводят избыточную информацию, в то время как, согласно требованиям, программа должна немедленно завершаться
5.2.2 б	Механизм контроля за целостностью стека	`-fstack-protector-strong`
5.2.2 в	Механизм рандомизации размещения кода в адресном пространстве	`-fpic/-fPIC/-fPIE`
5.2.2 г,д	Запрет на замену вызовов некоторых функций форматированного вывода и работы с памятью на эквивалентные последовательности машинных инструкций	`-fno-builtin-*`, при этом некоторые функции, работающие с widechar-строками, не имеют встроенных аналогов в Clang, поэтому для них это требование выполняется автоматически
5.2.6	Опциональные механизмы (например, контроль за целостностью потока управления)	Существует решение в виде `-fsanitize=cfi`, противоречащее, однако, логике третьего класса, поскольку способно существенно замедлить программу

Табл. 3. Реализация требований 3 класса по включению предупреждений о небезопасных конструкциях в Clang.
Table 3. Implementation of class 3 requirements for enabling warnings about unsafe statements in Clang.

Пункт стандарта	Требование	Опция
5.2.3 а	Затирание переменной, размещённой в автоматической памяти, при вызове `longjmp`	—
5.2.3 б	Чтение или запись по некорректному индексу элемента массива	`-Warray-bounds` и `-Warray-bounds-pointer-arithmetic`
5.2.3 в	Целочисленное деление или взятие остатка от целого числа с делителем, равным нулю	`-Wdivision-by-zero`
5.2.3 г	Операция побитового сдвига со вторым аргументом меньше нуля или больше ширины типа сдвигаемого значения	`-Wshift-count-negative`/`-Wshift-count-overflow`

Основными задачами безопасного компилятора второго класса являются предотвращение уязвимостей, связанных с некорректной работой с памятью, и недопущение использования неопределённых конструкций – тех же, о которых предупреждает компилятор третьего класса – путём остановки компиляции с ошибкой. Clang реализует лишь некоторые из них. Полный список требований и опций, с помощью которых они реализованы, указан в табл. 4.

Табл. 4. Реализация требований 2 класса в Clang.
Table 4. Implementation of class 2 requirements in Clang.

Пункт стандарта	Требование	Опция
5.3.1 а	Сохранение побочных эффектов записи в память	—
5.3.1 б	Автоматическая инициализация переменных нулями	`-ftrivial-auto-var-init=zero`
5.3.1 доп. а	Запрет оптимизаций побитовых сдвигов, если величина сдвига может оказаться отрицательной или больше или равна размеру типа	—
5.3.1 доп. б	Использование операций с векторными машинными регистрами, не требующих выравнивания данных	—
5.3.1 доп. в	Запрет оптимизаций адресной арифметики по информации о размерах объектов	—
5.3.2 а	Остановка компиляции с ошибкой для предупреждений 3 класса	`-Werror=*` вместо `-W*`, см. табл. 3
5.3.2 б	Запрет использования функции `gets`	`-Werror=deprecated-declarations`

Одной из основных задач безопасного компилятора первого класса является динамический контроль неопределённых конструкций. При этом от компилятора требуется включать в выходной файл машинный код, который предотвращает ошибочное выполнение неопределённых конструкций во время работы программы путём её аварийной остановки. Эта возможность в Clang реализована с помощью встроенного инструмента UndefinedBehaviourSanitizer (UBSan) [15], использование которого достигается с помощью указания в аргументах командной строки компиляции различных опций вида `-fsanitize=*`, где `*` – идентификатор проверки, определяющей тот или иной вид неопределённых

конструкций. UBSan выполняет большую часть видов проверок, которые должен осуществлять безопасный компилятор (см. табл. 5).

Табл. 5. Реализация требований 1 класса в Clang.
Table 5. Implementation of class 1 requirements in Clang.

Пункт стандарта	Идентификатор	Покрываемое требование: проверяет, что…
5.4.3 а	`bool`	Значение типа `bool` равно `0` или `1`.
5.4.3 б	`float-cast-overflow`	При преобразовании значения `float` в `int` не происходит переполнения. Не проверяются преобразования между вещественными типами.
5.4.3 в	`shift`	Правый операнд сдвига неотрицателен и меньше ширины типа.
5.4.3 г	`signed-integer-overflow`	Не происходит знакового переполнения.
5.4.3 д	`alignment`	Чтение или запись совершаются только по указателю, чей адрес выровнен по размеру операнда.
5.4.3 е	`null`	Нет использования нулевого указателя. Не проверяется непрямой вызов функции по нулевому указателю.
5.4.3 ж	`bounds`	Чтение или запись осуществляется по индексу, не выходящему за пределы массива.
5.4.3 и	`pointer-overflow`	Нет переполнения типа указателя.
5.4.3 к	`function`	При непрямом вызове сигнатура функции соответствует типу указателя на неё. Работает только для C++ и x86(-64).
5.4.3 л	`return`	Вызов функции завершается оператором `return`. Работает только для C++.
5.4.3 м	`builtin`	Во встроенные функции передаются корректные параметры.
5.4.3 н	`unreachable`	Не передаётся управление коду, помеченному как недостижимый.
5.4.3 п	`integer-divide-by-zero`	Нет деления на ноль или взятия остатка от нуля.
5.4.3 р	`vla-bound`	Массив, выделяемый в автоматической памяти, имеет положительный размер.

Другой задачей безопасного компилятора является управление распределением автоматической и статической памяти. В рамках этой задачи компилятор первого класса должен поддерживать возможность динамической компоновки программы, при которой при каждом запуске программы функции будут расположены в памяти в случайном порядке. Если полноценная реализация этого механизма невозможна (чему может препятствовать загрузчик операционной системы) или нецелесообразна (как в случае компиляции ядра операционной системы) распределение должно быть статическим – уникальным для каждого процесса компиляции. Clang не поддерживает статическое случайное распределение памяти, но может поддерживать динамическое распределение при использовании компоновщика и динамического загрузчика, в которых реализована эта возможность.

Необходимо заметить, что безопасный компилятор второго класса заимствует также часть требований третьего класса, а компилятор первого класса – все требования второго.

Таким образом, не существует конфигурации Clang, которая удовлетворяла бы требованиям хотя бы одного из классов безопасной компиляции, однако компилятор предоставляет ряд

возможностей для предотвращения небезопасных оптимизаций и выполнения небезопасных конструкций, что может быть использовано, в частности, при разработке безопасного компилятора на его основе.

4. Реализация безопасного компилятора

Реализованный безопасный компилятор разработан на базе Clang 16.0.6. Были добавлены опции `-Safe3`, `-Safe2`, `-Safe1`, включающие доступные в Clang и созданные в этой работе опции соответствующих классов безопасности, а для удобного управления функциональностью опций был разработан предметно-ориентированный язык на основе TableGen [16], позволяющий описать включение опций в следующем формате:

```
defm fno_strict_aliasing : Force<"-relaxed-aliasing", 1, 3>;
defm d_fortify_source_2 : Force<"-D_FORTIFY_SOURCE=2", 3, 3>;
```

Для 3 класса были реализованы следующие опции:

- `-fkeep-oversized-shifts`: предотвращает оптимизацию побитовых сдвигов в случаях, когда второй аргумент оператора сдвига меньше нуля или больше или равен ширине типа. Эта опция реализована заменой инструкций сдвига LLVM IR вызовами новых intrinsic-функций. Проходы `InstCombine` и `SCCP` дополнены оптимизациями этих вызовов, осуществляемыми только в тех случаях, когда компилятор способен доказать, что второй аргумент неотрицателен и строго меньше ширины типа. В противном случае вызовы intrinsic-функций раскрываются (заменяются на соответствующие инструкции) после всех оптимизационных проходов, избегая, таким образом, оптимизаций.
- `-fkeep-div-by-zero`: предотвращает оптимизацию деления на ноль и взятия остатка в случаях, когда второй аргумент равен нулю. Опция реализована аналогично `-fkeep-oversized-shifts`.
- Заголовочные файлы, содержащие fortified-версии функций стандартной библиотеки, и которые используются в Alpine Linux и безопасном компиляторе SAFEC, были доработаны для поддержки Clang. Благодаря использованию заголовочных файлов, включённых в компилятор, поддерживается и стандартная библиотека glibc, и musl. Кроме опции `-D_FORTIFY_SOURCE=2`, была добавлена поддержка `-D_FORTIFY_SOURCE=3` (включаемая в `-Safe2`), проверяющая не только вызовы с объектами константного размера, но и с теми, для которых можно во время компиляции составить выражение вычисления размера. Также с помощью этих заголовочных файлов было реализовано немедленное завершение программы при ошибке во время выполнения fortified-функций, предотвращена замена функций на встроенные в Clang аналоги, и добавлено предупреждение (ошибка в `-Safe2`) при использовании функции `gets`.
- Вместо предупреждения о затирании переменной, размещённой в автоматической памяти, при вызове `longjmp`, была реализована опция `-fforce-volatile-before-setjmp`, отмечающая все локальные переменные, доступные в момент вызова `setjmp`, как `volatile`, что не позволяет компилятору разместить эти переменные на регистрах и предотвращает их затирание после вызова `longjmp`. Эта опция реализована как проход в начале оптимизационного конвейера, работающего с промежуточным представлением LLVM IR. Этот проход обнаруживает уязвимые переменные (выделения на стеке) и помечает все использующие их инструкции как `volatile`.

Для 2 класса были реализованы следующие опции:

- Для сохранения побочных эффектов записи в память была создана опция `-fpreserve-memory-writes`, предотвращающая DSE (Dead Store Elimination) в

нескольких проходах оптимизационного конвейера. Благодаря этой опции сохраняется, например, очистка памяти, содержащей чувствительные данные. В проходе `EarlyCSE`, устраняющем тривиально избыточные инструкции, опция отключает удаление последовательных записей в тот же участок памяти без чтения между ними. В `InstCombine`, выполняющем объединение и удаление инструкций, отключается аналогичная оптимизация. В проходе `MemCpyOptimizer`, оптимизирующем такие инструкции работы с памятью, как `memset` и `memcpy`, отключается объединение пересекающихся записей в память в один `memset`. Наконец, в проходе `DeadStoreElimination`, выполняющем основную работу по оптимизации избыточных записей, вместо их удаления производится установка флага `volatile`, чтобы эти инструкции не могли быть оптимизированы следующими проходами.

- Опция `-fassume-unaligned` включает в начало оптимизационного конвейера проход, удаляющий выравнивание у инструкций `load` и `store`, а также у аргументов-указателей в вызовах функций. Благодаря этому вместо векторных инструкций, ожидающих выравненную память, генерируются инструкции для невыравненной памяти и предотвращаются аварийные завершения программ в случаях, когда используемый участок памяти имел некорректное выравнивание.
- Опция `-finbounds-aliasing` предотвращает оптимизации, использующие информацию о том, что указатели указывают на разные объекты (отсутствие алиасинга), если хотя бы один из них может указывать за границу объекта. Для этого были добавлены проверки в проход `BasicAliasAnalysis`, а также отключены оптимизации для выходящих за границу указателей в `SelectionDAGAddressAnalysis` и `InstCombine`.

Для 1 класса были реализованы следующие опции:

- Так как проверка `float-cast-overflow` в UndefinedBehaviorSanitizer проверяет наличие переполнения только при преобразовании из вещественных типов с плавающей запятой в целочисленные типы, была создана опция `-fsanitize=float-to-float-cast-overflow`, проверяющая преобразования между вещественными типами. Реализация основана на старой версии проверки `float-cast-overflow`, поведение которой было изменено в Clang 9 из-за того, что такой вид переполнения определён стандартом IEEE 754 [17].
- Опция `-fsanitize=null-call` проверяет, что при непрямом вызове функции не используется нулевой указатель. Такая проверка отсутствует в `-fsanitize=null` из UBSan.
- В Clang 16 опция `-fsanitize=function`, проверяющая соответствие формального типа функции и фактического типа указателя, поддерживает только C++ и архитектуру x86(-64), поэтому из Clang 17 были портированы улучшения, позволяющие использовать её для C и других архитектур. Основным изменением является использование хешей типов вместо RTTI (Run-Time Type Information), доступного только для C++.
- Стандартная опция `-fsanitize=return` проверяет наличие операции возврата при выходе из функции, имеющей возвращаемое значение, но только для C++, так как в C неопределённым поведением считается не отсутствие возврата, а использование значения, возвращаемого такой функцией. Чтобы сделать поведение проверки единообразным, была реализована опция `-fsanitize=return-c`, работающая для C так же, как и для C++.
- Для поддержки уникального распределения статической памяти программы на этапе компиляции были добавлены опции `-frandom-func-reorder` и `-frandom-func-`

`and-globals-reorder`, перемешивающие только функции или функции и глобальные переменные соответственно в каждой единице компиляции. Эти опции включают в конец оптимизационного конвейера, работающего с LLVM IR, проход, который задаёт случайный порядок функций и глобальных переменных на основе содержимого модуля и числа, задаваемого опцией `-mllvm -rng-seed`.

- Для рандомизации автоматической памяти создана опция `-floc-var-per`, перемешивающая локальные переменные (точнее, выделения на стеке константного размера) в случайном порядке. Эта функциональность реализована в том же проходе, который используется для предыдущих двух опций. Также поддерживается опция `-fadd-loc-var`, задающая количество локальных переменных, которые добавляются при перемешивании. Без неё при использовании `-floc-var-per` добавляется небольшое случайное количество переменных для большей случайности автоматической памяти.

5. Результаты

5.1. Корректность

Разработанный безопасный компилятор на основе Clang успешно проходит все тесты из набора, созданного для проверки корректности безопасного компилятора SAFEC [11] и его соответствия стандарту. Этот набор включает в себя тесты, проверяющие наличие вывода требуемых диагностик, тесты, проверяющие срабатывание динамических проверок во время выполнения, и тесты, проверяющие результат кодогенерации.

5.2. Исследование производительности

Производительность программ, скомпилированных безопасным компилятором, оценивалась с помощью тех же 5 тестов, что и для SAFEC:

- воспроизведение партии в го с помощью GNU Go 3.8;
- перекодирование файлов из формата WAV в MP3 с помощью LAME 3.100;
- выполнение теста fannkuch из The Computer Language Benchmarks Game;
- перекодирование файлов из формата YUV в MKV с помощью x264 (x264-snapshot-20190407-2245-stable);
- сжатие текстового файла с помощью zlib 1.2.11.

В табл. 6 приведены результаты измерения времени выполнения тестов на компьютере с процессором AMD Ryzen™ 5 4600H (архитектура x86-64) и ОС Manjaro Linux 23.1.4. Столбец Baseline соответствует запуску компилятора с опцией `-O2`, а столбцы `-Safe3`, `-Safe2`, `-Safe1` соответствуют запуску с уровнем оптимизации `-O2` и соответствующим классом защиты. Каждое значение вычислялось как округлённое до 0.01 с среднее по 5 запускам. Также для каждого уровня безопасности указано замедление относительно базового времени.

Табл. 6. Результаты измерения производительности.
Table 6. Performance evaluation results.

Тест	Baseline	`-Safe3`	`-Safe3` замедл.	`-Safe2`	`-Safe2` замедл.	`-Safe1`	`-Safe1` замедл.
GNU Go	3.93 с	4.03 с	2.54%	4.26 с	8.12%	6.66 с	69.04%
LAME	5.21 с	5.27 с	1.15%	4.90 с	-5.82%	13.19 с	153.40%
fannkuch	2.24 с	2.10 с	-6.25%	2.08 с	-7.14%	2.72 с	21.43%
x264	1.69 с	1.81 с	7.42%	1.79 с	5.20%	6.32 с	274.74%

Тест	Baseline	`-Safe3`	`-Safe3` замедл.	`-Safe2`	`-Safe2` замедл.	`-Safe1`	`-Safe1` замедл.
zlib	1.54 с	1.63 с	5.88%	1.62 с	5.87%	2.40 с	55.60%

Из представленных данных следует, что при использовании 3 или 2 класса защиты замедление не превышает 10%, при этом программы, скомпилированные с 2 уровнем безопасности иногда оказываются быстрее. Это может происходить из-за того, что в 2 классе, в отличие от 3 класса, отсутствует запрет на замену вызовов функций работы с памятью из стандартной библиотеки на эквивалентные последовательности машинных инструкций. При использовании 1 класса защиты замедление составляет от 21% до 275%, что в случае x264 превышает допустимые 200%.

Существенное замедление работы x264 можно объяснить добавлением санитайзером `pointer-overflow` большого количества проверок переполнения указателей — без него замедление составляет 165%. Также было обнаружено, что базовая версия x264, скомпилированная Clang, выполнялась на 64% быстрее чем та, что была скомпилирована SAFEC 11.4.0, при этом с 1 уровнем безопасности время работы программ приблизительно совпадало, а значит при сравнении 1 класса безопасности Clang с базовой версией GCC замедление составит менее 200%.

Кроме того, в результате выполнения тестов выяснилось, что санитайзер `pointer-overflow` в Clang, в отличие от GCC и основанного на нём SAFEC, находит в GNU Go переполнение при добавлении беззнакового числа к указателю. Это известное ограничение GCC [18].

5.3. Сборка дистрибутива Linux

Кроме тестирования нескольких приложений отдельно с помощью безопасного Clang, была произведена оценка применимости безопасного компилятора с помощью сборки дистрибутива Linux. Для этой задачи был выбран Alpine Linux 3.18 [19] — дистрибутив, ориентированный на легковесность и безопасность, использующий musl, BusyBox и OpenRC. Так как в дистрибутиве для сборки из исходных кодов по умолчанию используется GCC, то в сборочный мета-пакет `build-base` был добавлен Clang как зависимость и произведена замена `/usr/bin/gcc`, `/usr/bin/cc` и подобных файлов на символические ссылки на Clang. Также в пакеты `clang16` и `llvm16` были добавлены патчи безопасного компилятора. В результате была выполнена сборка каждого пакета из репозиториев `main` и `community` со всеми классами безопасности: от небезопасного режима до 1 класса.

В табл. 7 для каждого класса безопасности приведено количество пакетов, которые удалось собрать, и количество пакетов, чья сборка завершилась ошибкой. Большое количество пакетов, не собранных Clang в небезопасном режиме, во многих случаях объясняется невозможностью загрузить исходный код пакетов, а также несовместимостью Clang с GCC из-за включения по умолчанию некоторых предупреждений в качестве ошибок. Также стоит отметить, что приблизительно треть от всех пакетов (2382 из 6879) не использует компилятор C/C++, поэтому в работе они не рассматриваются.

Табл. 7. Результаты сборки пакетов Alpine Linux.
Table 7. Alpine Linux package build results.

Класс	Успешно собрано	Ошибки сборки
Baseline	3587	910
`-Safe3`	3581	6
`-Safe2`	3500	81
`-Safe1`	3456	44

На 3 уровне безопасности не удалось собрать всего 6 пакетов – в 2 случаях возникла ошибка из-за попытки установки `_FORTIFY_SOURCE` в 0 при наличии `-Werror`. Ещё в одном пакете

выполнялось удаление ключевого слова `const` с помощью `#define const`, что мешает компиляции заголовочных файлов fortified-функций. Также в одном пакете, компилируемом с `-Werror`, присутствовало предупреждение `-Warray-bounds-pointer-arithmetic`. Ещё два пакета оказались несовместимы с механизмом контроля целостности стека. Так как в 2 классе безопасности предупреждения из 3 класса становятся ошибками, то на этом уровне из-за `-Warray-bounds-pointer-arithmetic` не скомпилировались 68 пакетов, ещё 6 – из-за `-Warray-bounds`, и 7 из-за `-Wshift-count-overflow`.

В 1 класс безопасности включаются санитайзеры, из-за чего могут аварийно завершиться программы, скомпилированные и запущенные во время сборки пакетов. По этой причине не удалось собрать 39 пакетов. Также сборка 2 пакетов завершилась ошибками из-за несовместимости `-fsanitize=function` с WebAssembly. Ещё один пакет не удалось собрать из-за `-Werror=shift-count-overflow`. Кроме того, при сборке пакетов `community/cabextract` и `community/libu2f-server` была обнаружена ошибка в исходном Clang 16.0.6, приводящая к аварийному завершению компилятора. Выяснилось, что она была исправлена в Clang 17.

В этой работе проверялось только то, что пакеты дистрибутива успешно собираются, так как тестирование работоспособности всех программ потребовало бы слишком больших затрат времени. Было показано, что 96.3% пакетов, собираемых небезопасным Clang, могут быть собраны и безопасной версией с 1 классом безопасности. Но необходимо дальнейшее тестирование, так как может оказаться, что многие успешно собранные программы не будут работать, например, из-за наличия в них неопределённого поведения, обнаруживаемого санитайзерами.

6. Заключение

В рамках данной работы был реализован безопасный компилятор на основе Clang. В работе была рассмотрена концепция безопасного компилятора применительно к исследуемому. Были выделены компоненты, реализующие указанные возможности, а также рассмотрены требования, которым данный компилятор не удовлетворяет. Было обосновано решение о разработке безопасного компилятора на основе Clang. Были реализованы и описаны все недостающие компоненты. С помощью созданного безопасного компилятора удалось собрать большую часть пакетов дистрибутива ОС Alpine Linux. Было показано, что производительность программ, собранных безопасным компилятором, почти во всех случаях удовлетворяет требованиям.

Список литературы / References

[1]. Clang, Available at: https://clang.llvm.org/, accessed 24.04.2024.
[2]. Jetbrains Developer Ecosystem: C++, Available at: https://www.jetbrains.com/lp/devecosystem-2023/cpp/, accessed 24.04.2024.
[3]. GCC, Available at: https://gcc.gnu.org/, accessed 24.04.2024.
[4]. LLVM Developer Policy, Available at: https://llvm.org/docs/DeveloperPolicy.html, accessed 24.04.2024.
[5]. LLVM, Available at: https://llvm.org/, accessed 24.04.2024.
[6]. Скворцов Л.В., Баев Р.В., Долгорукова К.Ю., Шарыгин Е.Ю. Разработка компилятора для стековой процессорной архитектуры TF16 на основе LLVM. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 137-154. DOI: 10.15514/ISPRAS–2021–33(5)–8./ Skvortsov L.V., Baev R.V., Dolgorukova K.Y., Sharygin E.Y. Developing an LLVM-based compiler for stack based TF16 processor architecture. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 137-154 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–8.
[7]. Мельник Д., Курмангалеев Ш., Аветисян А., Белеванцев А., Плотников Д., Варданян М. Оптимизация приложений для заданных статических компиляторов и целевых архитектур: методы и инструменты. Труды ИСП РАН, том 26, вып. 1, 2014 г., стр. 343-356. DOI: 10.15514/ISPRAS-2014-26(1)-13./ Melnik D., Kurmangaleev S., Avetisyan A., Belevantsev A., Plotnikov D., Vardanyan M.

Optimizing programs for given hardware architectures with static compilation: methods and tools. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 1, 2014, pp. 343-356 (in Russian). DOI: 10.15514/ISPRAS-2014-26(1)-13.

[8]. Иванников В., Курмангалеев Ш., Белеванцев А., Нурмухаметов А., Савченко В., Матевосян Р., Аветисян А. Реализация запутывающих преобразований в компиляторной инфраструктуре LLVM. Труды ИСП РАН, том 26, вып. 1, 2014 г., стр. 327-342. DOI: 10.15514/ISPRAS-2014-26(1)-12./ Ivannikov V., Kurmangaleev S., Belevantsev A., Nurmukhametov A., Savchenko V., Matevosyan H., Avetisyan A. Implementing Obfuscating Transformations in the LLVM Compiler Infrastructure. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 1, 2014, pp. 327-342 (in Russian). DOI: 10.15514/ISPRAS-2014-26(1)-12.

[9]. Гайсарян С.С., Курмангалеев Ш.Ф., Долгорукова К.Ю., Савченко В.В., Саргсян С.С. Применение метода двухфазной компиляции на основе LLVM для распространения приложений с использованием облачного хранилища. Труды ИСП РАН, том 26, вып. 1, 2014 г., стр. 315-326. DOI: 10.15514/ISPRAS-2014-26(1)-11./ Gaissaryan S., Kurmangaleev S., Dolgorukova K., Savchenko V., Sargsyan S. Applying two-stage LLVM-based compilation approach to application deployment via cloud storage. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 1, 2014, pp. 315-326 (in Russian). DOI: 10.15514/ISPRAS-2014-26(1)-11.

[10]. Wang X., Chen H. et al. Undefined behavior: what happened to my code? In Proc. of the Asia-Pacific Workshop on Systems, 2012, pp. 1-7.

[11]. Баев Р.В., Скворцов Л.В., Кудряшов Е.А., Бучацкий Р.А., Жуйков Р.А. Предотвращение уязвимостей, возникающих в результате оптимизации кода с неопределенным поведением. Труды ИСП РАН, том 33, вып. 4, 2021 г., стр. 195-210. DOI: 10.15514/ISPRAS–2021–33(4)–14./ Baev R.V., Skvortsov L.V., Kudryashov E.A., Buchatskiy R.A., Zhuykov R.A. Prevention of vulnerabilities arising from optimization of code with Undefined Behavior. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 4, 2021. pp. 195-210 (in Russian). DOI: 10.15514/ISPRAS–2021–33(4)–14.

[12]. Безопасный компилятор SAFEC. Доступно по ссылке: https://www.ispras.ru/technologies/safecomp/, доступ осуществлён 24.04.2024.

[13]. Clang: Language Compatibility, Available at: https://clang.llvm.org/compatibility.html, accessed 24.04.2024.

[14]. ГОСТ Р 71206-2024 «Защита информации. Разработка безопасного программного обеспечения. Безопасный компилятор языков С/С++. Общие требования». М., Российский институт стандартизации, 2024, 20 с.

[15]. UndefinedBehaviourSanitizer, Available at: https://clang.llvm.org/docs/UndefinedBehaviorSanitizer.html, accessed 24.04.2024.

[16]. TableGen Overview, Available at: https://llvm.org/docs/TableGen/, accessed 24.04.2024.

[17]. IEEE 754-2019, Standard for Floating-Point Arithmetic, 2019. pp. 1-84. DOI: 10.1109/IEEESTD.2019.8766229.

[18]. Missing pointer overflow detection with -fsanitize=pointer-overflow, Available at: https://gcc.gnu.org/bugzilla/show_bug.cgi?id=82079, accessed 24.04.2024.

[19]. Alpine Linux, Available at: https://www.alpinelinux.org/, accessed 24.04.2024.

Информация об авторах / Information about authors

Павел Дмитриевич ДУНАЕВ - Аспирант Института системного программирования им. В.П. Иванникова Российской академии наук. Сфера научных интересов: компиляторы, операционные системы, дискретная математика.

Pavel Dmitrievich DUNAEV - PhD student at Ivannikov Institute for System Programming of the Russian Academy of Sciences. Research interests: compilers, operating systems, discrete mathematics.

Артем Александрович СИНКЕВИЧ - Старший лаборант Института системного программирования им. В.П. Иванникова Российской академии наук, студент 2 курса магистратуры Саратовского государственного университета. Сфера научных интересов: компиляторы, дискретная математика, нейронные сети.

Artem Aleksandrovich SINKEVICH - Senior Laboratory technician at Ivannikov Institute for System Programming of the Russian Academy of Sciences, 2nd year Master's student at Saratov State University. Research interests: compilers, discrete mathematics, neural networks.

Артемий Максимович ГРАНАТ - Старший лаборант Института системного программирования им. В.П. Иванникова Российской академии наук, студент 1 курса магистратуры Высшей школы экономики. Сфера научных интересов: компиляторы, операционные системы, компьютерные сети.

Artemiy Maksimovich GRANAT - Senior Laboratory technician at Ivannikov Institute for System Programming of the Russian Academy of Sciences, 1st year Master's student at Higher School of Economics. Research interests: compilers, operating systems, computer networks.

Инна Александровна БАТРАЕВА - кандидат физико-математических наук, доцент, заведующая кафедрой технологий программирования. Сфера научных интересов: дискретная математика, теория автоматов, теория формальных языков и грамматик, информационные системы в теоретической и прикладной лингвистике.

Inna Aleksandrovna BATRAEVA - Candidate of Science in Physics and Mathematics, Associate Professor, Head of the Department of Programming Technologies. Research interests: discrete mathematics, automata theory, theory of formal languages and grammars, information systems in theoretical and applied linguistics.

Сергей Владимирович МИРОНОВ - кандидат физико-математических наук, доцент, декан факультета компьютерных наук и информационных технологий. Сфера научных интересов: методы сокращения диагностической информации с использованием словарей неисправностей, формальные языки и грамматики, функциональное программирование.

Sergei Vladimirovich MIRONOV - Candidate of Science in Physics and Mathematics, Associate Professor, Dean of the Faculty of Computer Science and Information Technologies. Research interests: methods of diagnostic information compression using fault dictionaries, formal languages and grammars, functional programming.

Никита Юрьевич ШУГАЛЕЙ - Старший лаборант Института системного программирования им. В.П. Иванникова Российской академии наук, студент 2 курса магистратуры Московского Физико-технического Института Физтех-школы Радиотехники и Компьютерных технологий по направлению Прикладные Физика и Математика.

Nikita Yurievich SHUGALEY - Senior Laboratory technician at Ivannikov Institute for System Programming of the Russian Academy of Sciences, 2st year Master's student at the Moscow Institute of Physics and Technology, Phystech School of Radio Engineering and Computer Technology, field of Applied Physics and Mathematics.

DOI: 10.15514/ISPRAS-2024-36(4)-4

Реализация траекторного профилирования в компиляторе LCC для процессоров Эльбрус

В.Е. Шампаров, ORCID: 0000-0001-9900-5159 <shamparov_v@mcst.ru>
М.И. Нейман-заде, ORCID: *0000-0002-4250-9724 <muradnz@mcst.ru>*
АО "МЦСТ", 117437, Москва, ул. Профсоюзная, д. 108.
Московский физико-технический институт
(национальный исследовательский университет),
141701, Московская область, г. Долгопрудный, Институтский переулок, д. 9.

Аннотация. В работе предложена реализация траекторного профилирования методом инструментирования, реализованная в компиляторе LCC для архитектур «Эльбрус» и SPARC и предназначенная для улучшения работы специфических оптимизаций для процессоров с архитектурой типа VLIW.

Ключевые слова: компилятор; траекторное профилирование; библиотека поддержки профилирования; широкое командное слово VLIW.

Для цитирования: Шампаров В.Е., Нейман-заде М.И. Реализация траекторного профилирования в компиляторе LCC для процессоров Эльбрус. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 41–56. DOI: 10.15514/ISPRAS–2024–36(4)–4.

Path Profiling for LCC Compiler for Elbrus CPUs

V.E. Shamparov, *ORCID: 0000-0001-9900-5159* <shamparov_v@mcst.ru>
M.I. Neiman-zade, ORCID: *0000-0002-4250-9724 <muradnz@mcst.ru>*
AO "MCST", 108, Profsoyuznaya str., Moscow, 117437, Russia.
Moscow Institute of Physics and Technology,
9, per. Institutskij, Dolgoprudnyj, Moskovskaya oblast', 141701, Russia.

Abstract. This paper presents a new version of instrumentation-based path profiling, implemented for the LCC compiler for Elbrus and SPARC processors. This profiling is intended to be used for VLIW-specific compiler optimizations, where path information and path correlations are needed. It was optimized, so profiling overhead descreased to 5.5 times on average.

Keywords: compiler; path profiling; profiling support library; VLIW.

For citation: Shamparov V.E., Neiman-zade M.I. Path profiling for LCC compiler for Elbrus CPUs. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024. pp. 41-56 (in Russian). DOI: 10.15514/ISPRAS-2024-36(4)-4.

1. Введение

Для многих видов оптимизации, применяемых к программам во время компиляции, требуется информация о ходе исполнения программы. Существует множество способов сбора подобной информации, в числе которых находится инструментирование программы компилятором. В этом случае после запуска программы на тренировочных данных компилятору становится доступна информация о ходе исполнения программы.

Классическим является профилирование со сбором счётчиков дуг графа потока управления. Но данный способ не даёт информации про корреляцию между путями исполнения программы, которая может понадобиться для некоторых оптимизаций. Для решения данного вопроса применяются траекторные профили, в которых собрана статистика траекторий, которые проходила программа.

В данной работе предложена реализация траекторного профилирования, предназначенная для улучшения работы специфических оптимизаций для процессоров с архитектурой типа Very Long Instruction Word (VLIW). Работа состоит из нескольких частей. В разделе 2 приведена решаемая проблема. Связанные с темой данной работы публикации описаны в разделе 3. Созданные для решения проблемы алгоритмы описаны в разделе 4. В разделе 5 приведены полученные результаты. Наконец, в разделе 6 подведены итоги совершённой работы и описаны планы дальнейшей работы над траекторным профилированием.

2. Недостаточность информации классического профиля

Для архитектуры «Эльбрус», являющейся архитектурой типа VLIW с явным параллелизмом исполнения команд и имеющей предикатный режим исполнения команд, существенно важна оптимизация слияния кода (она же if-конверсия, она же оптимизация слияния альтернатив условий [15]).

Алгоритм оптимизации слияния кода предполагает следующие действия:

1. выбор региона – набора узлов графа потока управления для слияния; среди выбранных узлов есть голова региона, и все входы в регион должны быть входами в голову;
2. дублирование узлов с внешними входами в регион (не в голову) для обеспечения требования в предыдущем пункте;
3. слияние региона в один гиперузел с использованием предикатного режима исполнения кода; таким образом получается, что в гиперузле на исполнение поступают все слитые пути, а по предикатам происходит выборка операций, которые действительно начинают исполняться;
4. спекулятивный вынос части операций вверх из-под предикатов.

Из-за использования предикатного режима при исполнении на процессорах с последовательным исполнением команд (in-order), например, процессорах «Эльбрус», операции под отрицательным предикатом занимают место на устройствах исполнения команд, но не исполняются. Из-за этого существенным требованием является отсутствие в регионе маловероятных путей исполнения, так как они с высокой вероятностью бесполезно занимают место на устройствах и при этом негативно влияют на планирование кода.

Для качественного определения маловероятных узлов графа потока управления используются результаты проведённого профилирования либо предсказанный профиль. Но правильная коррекция профиля при дублировании узлов данным способом недостижима, так как требуется корреляция между траекториями до дублируемого узла и траекториями после него. Без правильной коррекции профиля при дублировании невозможно достичь правильного выбора регионов, так как нельзя корректно определить маловероятные узлы на траекториях после продублированного узла.

На рис. 1 изображён пример участка графа потока управления, где классический профиль по счётчикам дуг не всегда позволяет оптимальным образом провести слияние узлов. Пусть из-за зависимостей в коде получается так, что после слияния кода операции из узлов, связанных дугами до слияния, не будут смешиваться, а длительность узлов следующая:

- *A* – 10 тактов;
- *B* – 5 тактов;
- *C* – 15 тактов;
- *D* – 10 тактов;
- *E* – 15 тактов;
- *F* – 5 тактов;
- *G* – 10 тактов.

Также пусть любая пара узлов, которые по данным не зависимы друг от друга, при планировании параллельно друг другу не увеличивают свою длительность.

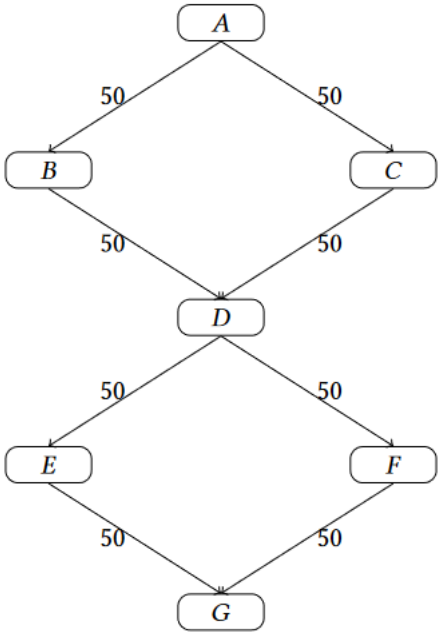


Рис. 1. Пример участка графа потока управления, где классический профиль по счётчикам дуг не всегда позволяет оптимальным образом провести слияние узлов.
Дуги маркированы своими счётчиками.
Fig. 1. Example of CFG part, where classic arcs profile does not always allow optimal if-conversion.
Edges are marked with their counters.

В ситуации, когда при исполнении выполняются только траектории *ABDEG* и *ACDFG* по 50 раз каждая, они формируют счётчики дуг, как показано на рисунке. В этом случае без информации о траекториях компилятор производит слияние кода в гиперузел как на рис. 2. Прерывистые дуги на рисунке обозначают зависимость по данным. Узел *A* генерирует предикат p_A, узел *D* – предикат p_D. Код узла *B* выполняется при условии истинности предиката p_A, код узла *C* выполняется при условии ложности предиката p_A. Аналогично и узлы *E* и *F* относительно предиката p_D. Длительность такого гиперузла составляет 60 тактов.

При наличии информации о траекториях оптимальное слияние в гиперузел изображено на рис. 3. В этом случае узел D и последующие узлы E, F и G дублируются, но маловероятные пути среди дублированных узлов выносятся из гиперузла в отдельные «холодные» гипе-рузлы. Длительность основного («горячего») гиперузла составляет уже 50 тактов.

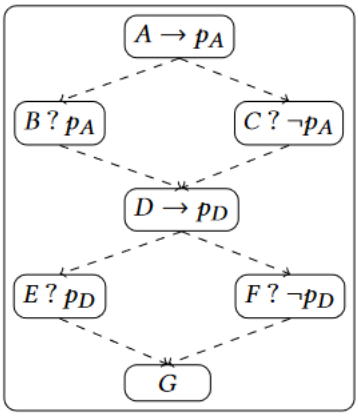


Рис. 2. Пример участка графа потока управления с рис. 1, слитого без учёта траекторий.
Fig. 2. Example of CFG part from fig. 1 after if-conversion without paths.

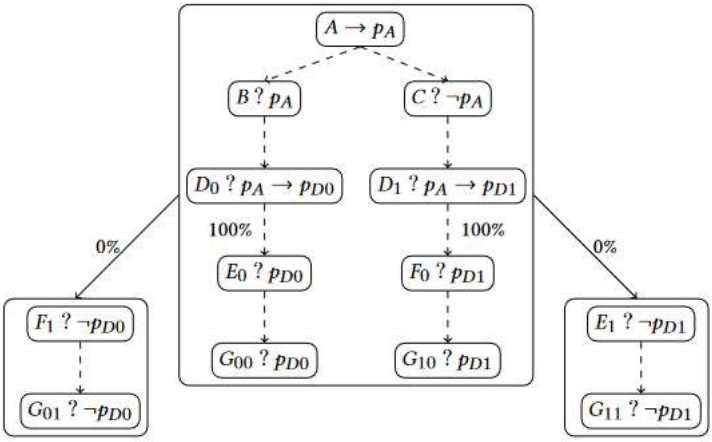


Рис. 3. Пример участка графа потока управления с рис. 1, слитого с учётом траекторий.
На дугах из дублированного узла D отображены вероятности переходов.
Fig. 3. Example of CFG part from fig. 1 after if-conversion with paths.
Edges from duplicated node D are marked with probabilities for control transfer towards them.

Таким образом, для коррекции профиля при дублировании узлов в ходе оптимизации слияния кода требуется траекторный профиль с информацией о корреляциях между траекториями до дублируемых узлов и траекториями после дублируемых узлов.

Также для иных цикловых оптимизаций требуется информация о корреляции траекторий в разных итерациях цикла, а для оптимизации inline – траектории, проходящие через операции `call` и `return`.

3. Работы по данной тематике

Известна работа [3], в которой предложен легковесный алгоритм сбора траекторий с присвоением каждой нециклической траектории в функции или в цикле уникального номера. Указанный алгоритм делит всю программу на такие ациклические участки по границам циклов и функций и позволяет получить счётчик каждой такой траектории. За счёт алгоритма нумерации траекторий номер траектории к концу ациклического участка получается из серии инкрементов счётчика на определённых пройденных дугах. Соответственно, на большинстве дуг программы либо не создаётся операций, либо создаётся единственная операция инкремента счётчика. И только на границах ациклических участков и функций добавляется код инкремента счётчика исполнений полученной траектории.

Легковесность приведённого алгоритма является его преимуществом, но из-за разделения кода на ациклические участки данный способ профилирования не позволяет использование для межпроцедурных и некоторых цикловых оптимизаций.

Для траекторного профилирования методом, описанным в [3], было сделано несколько улучшений. В работе [12] за счёт изменения алгоритма нумерации траекторий удалось ещё сильнее уменьшить накладные расходы на профилирование – в улучшенном алгоритме инкременты счётчиков оказываются на более редких траекториях.

Интересное улучшение было предложено в работе [4] для систем, в которых работающая программа находится под наблюдением работающего в другом потоке профилировщика либо иных подобных профилировщику программ. Например, в этой статье используется JIT-компилятор, занимающийся профилированием программы. В этом случае вместо увеличения счётчика для конкретного пути, что авторы считают дорогостоящей операцией, профилировщик по сигналу от программы в месте записи получает из программы номер пути и далее уже сам записывает полученную информацию. Таким образом, накладные расходы на запись номера траектории перекладываются на профилировщик.

Ещё один способ уменьшения накладных расходов на профилирование предложен в работе [2] для многоядерных систем, где предлагается выполнять несколько экземпляров профилируемой программы параллельно, и в каждой из них инструментировать разные части. Таким образом, накладные расходы на профилирование разделяются между разными экземплярами.

В работе [8] предложен способ обобщения траекторного профилирования для сбора в том числе межпроцедурных траекторий, но для этого строится суперграф (то есть граф потока управления для всех функций сразу с дугами по вызовам функций), а на дугах строятся более сложные вычисления, чем инкременты счётчиков. Этот профиль может быть использован для оптимизации частичной подстановки кода вызываемых функций. Но при инструментировании больших программ может не хватать оперативной памяти, так как строится суперграф.

В работе [13] предложена идея разделять процедуру на иерархию вложенных подграфов и профилировать траектории в подграфах независимо. Это позволяет в JIT-компиляторах гибко собирать только нужную информацию и в разумное время принимать решения о перекомпиляции участков кода на основании таких частичных траекторных профилей.

В работе [11] сделан подход к сбору траекторий разных итераций циклов (максимальной длиной k ациклических подтраекторий, по одной на итерацию) для определения корреляций между ними. В данной работе для каждого цикла предложено собирать собственные коллекции траекторий длиной k итераций. Но полноценно данный подход удалось развить в работе [6]. Там предложен более общий алгоритм, позволяющий для каждой функции собирать коллекции траекторий длиной k ациклических подтраекторий. Оба подхода позволяют выявить корреляции между разными итерациями циклов на расстоянии до $k - 1$ итераций.

Иной вид профиля предложен в статье [14]. Там авторы поставили цель улучшить расположение базовых блоков программы, в том числе с помощью их дублирования, для улучшения работы предсказателя переходов. Для этого они собирают профиль траекторий, подобный собираемому в предсказателе переходов [9]. Траектория состоит из данных для k предыдущих

операций `branch` перед текущей, в которых сохраняется номер базового блока, в котором была операция, и результат: произошёл или не произошёл переход. Для каждой такой траектории собираются два счётчика: сколько раз история произошла и сколько раз из них `branch` привёл к переходу (branch was taken). На основе этих данных, собранных в дерево для каждой операции `branch`, для каждого узла с операцией `branch` производится предсказание, какие варианты там возможны: переход происходит, не происходит или возможны оба варианта. Узлы, для которых предсказатель может давать оба варианта, дублируются так, чтобы предсказатель переходов редко делал неверные предсказания.

Ещё один вид траекторного профиля, предложенный в работе [10], требует широкой поддержки со стороны аппаратуры. В данной работе за основу профиля взята комбинация из битового вектора результатов операций `branch` на данном ациклическом участке кода и полученного аппаратно счётчика времени в тактах, которое заняло конкретное исполнение участка кода. На основе таких данных определяются вариации времени исполнения каждой траектории в программе. Если время выполнения конкретных траекторий конкретного участка кода сильно вариативно, то можно считать, что этот участок кода недооптимизирован. В статье не предложено способов автоматического использования результатов профилирования, кроме приоритизации участков кода для оптимизации.

Поддержка собираемого аппаратно траекторного профиля предложена компанией Intel и используется, например, в утилите BOLT [9]. Аппаратура собирает последние k произошедших переходов в формате (откуда – куда) в специальный буфер (регистр) LBR, и это позволяет программе или профилировщику получить эти данные и, переведя в вид узлов CFG, получить траектории в программе. В утилите BOLT данные возможности используются для расположения базовых блоков программы в оптимальном порядке.

Для архитектур без LBR предложен способ собирать трассы в том же формате в работе [7], хотя в целях уменьшения накладных расходов это сделано в формате сэмплирования. Во время остановки для сбора сэмпла сбор одной трассы длиной k происходит в несколько этапов. Сначала, начиная с адреса текущей команды, происходит поиск k последующих операций перехода. Если операция безусловная, то она декодируется и соответствующая запись добавляется в трассу. Если же операция находится под условием, то на неё ставится точка останова, и программа продолжает исполнение. Вскоре программа будет остановлена на этой точке останова, и тогда появится возможность выяснить, произойдёт там переход либо нет. В первом случае в трассу добавляется соответствующая запись. В обоих случаях можно продолжить собирать трассу аналогичным способом, пока в трассе не наберутся k записей.

4. Реализованное траекторное профилирование

В рамках данной работы реализовано инструментирование в оптимизирующем компиляторе LCC для процессорных архитектур «Эльбрус» и SPARC и библиотека поддержки для сбора профильных данных.

Вид собираемого профиля выбран на основе работы [14], но с существенными отличиями:

- так как в общем случае у узла графа потока управления может быть более 2 исходящих дуг, однобитный флаг произошедших или не произошедших переходов не подходит, поэтому для идентификации исходящей дуги выбран номер выбранной исходящей дуги;
- в ходе оптимизации слияния кода происходит дублирование узлов с множественными входными дугами, соответственно для коррекции профиля после дублирования потребуется различать траектории для разных входных дуг; из-за этого помимо узлов с множественными выходными дугами происходит сбор узлов с множественными входными дугами;

- для коррекции профиля нужна история после узла с множественными входными дугами, поэтому сохраняемые траектории имеют следующий вид:
- *n* узлов с множественными выходными дугами (чаще всего с операциями `branch`), для каждого указана конкретная исполненная дуга; будем называть эту часть *«предшествующей траекторией»*;
- узел с множественными входными дугами;
- *k* узлов с множественными выходными дугами, для каждого указана конкретная исполненная дуга; будем называть эту часть *«последующей траекторией»*.

Назовём *«узлами схождения»* (также просто *«схождениями»*) узлы с множественными входными дугами и исполненной конкретной дугой среди данных входных, а *«узлами расхождения»* (также просто *«расхождениями»*) − узлы с множественными выходными дугами и исполненной конкретной дугой среди данных выходных. Таким образом, сохраняются траектории из *n* расхождений до схождения, самого схождения и *k* расхождений после него.

В целях возможного будущего применения данного профиля для межпроцедурных оптимизаций, например, частичной подстановки кода вызываемых функций как в работе [8], на собираемые профили не накладывается никаких ограничений на вхождение составляющих их узлов графа потока управления в разные процедуры. Из-за этого для узлов разных функций строятся уникальные идентификаторы для однозначного определения узлов. Идентификатор узла с дугой (входящей либо исходящей) состоит из следующих частей:

- идентификатор единицы трансляции;
- идентификатор функции в единице трансляции;
- уникальный номер узла внутри графа потока управления функции (выбран номер по RPO-нумерации);
- уникальный номер дуги.

Созданные во время компиляции идентификаторы единиц трансляции и функций записываются в специальный файл метаданных для использования во время применения собранного траекторного профиля.

4.1 Неоптимизированное инструментирование

Алгоритм инструментирования во время компиляции одной единицы трансляции состоит из следующих этапов:

1) для каждой функции в единице трансляции сгенерировать уникальный идентификатор; для этого подходит простое назначение каждой новой функции последовательных целых неотрицательных чисел;
2) для каждой функции:
    1. сбор узлов схождения и расхождения с запоминанием их идентификаторов; это делается для того, чтобы созданные во время инструментирования узлы и дуги не влияли на собираемые в профиле данные;
    2. непосредственно инструментирование каждого узла схождения и расхождения.

В обычном случае инструментирование узла расхождения предполагает вставку на каждой исходящей дуге вызова функции `TraceDivNode` из библиотеки поддержки, куда подаются идентификаторы единицы трансляции, функции, узла и данной исходящей дуги. Аналогично производится инструментирование узла схождения − вызовы `TraceConvNode` с параметрами в виде идентификаторов единицы трансляции, функции, узла и данной исходящей дуги вставляются на каждой входящей дуге.

Исключение – случаи, когда на дуге нельзя вставлять никакого кода. Например, такое происходит на дугах, соединяющих узел с потенциально вызывающим исключения вызовом с узлом обработчика исключения. В таких случаях требуется специальная версия инструментирования. Для узлов расхождения инструментирование выглядит следующим образом:

1) в узел с множественными выходами добавляется запись идентификатора узла в специально созданную временную переменную;
2) вызов `TraceDivNode`, где в параметрах находится значение временной переменной, вставляется ниже по графу потока управления от дуги настолько рано, насколько это возможно.

Аналогично и для узлов схождения:

1) в узел перед дугой вставляется запись идентификатора дуги в специально созданную временную переменную;
2) вызов `TraceConvNode`, где в параметрах находится значение временной переменной, вставляется ниже по графу потока управления от дуги настолько рано, насколько это возможно.

Таким образом, на каждой входной дуге узлов схождения и на каждой выходной дуге узлов расхождения (либо рядом с ними) оказывается по вызову функции из библиотеки поддержки.

4.2 Неоптимизированная библиотека поддержки

В библиотеке поддержки находятся два буфера:

- буфер узлов расхождения;
- буфер узлов схождения.

Для каждого узла схождения хранится индекс следующего во времени узла расхождения для того, чтобы корректно определять нужные траектории.

Функции `TraceConvNode` и `TraceDivNode` заполняют эти буферы до тех пор, пока один из буферов не заполнится. Затем производится частичное заполнение статистики, при котором происходит обход буферов, построение всех находящихся там накопленных траекторий и инкременты счётчиков для каждой из них.

Подобный способ промежуточного сохранения информации для сбора траекторий позволяет иметь два буфера на всю профилируемую программу и собирать траектории, включающие схождения и расхождения из разныю функций и единиц трансляции.

Статистика хранится в виде вложенных хэш-таблиц:

- ключ первого уровня – схождение; таким образом, вложенная таблица имеет смысл статистики для данного схождения; например, для примера на рис. 1 возможные ключи:
  - узел *D* с дугой *BD*;
  - узел *D* с дугой *CD*;
  - узел *G* с дугой *EG*;
  - узел *G* с дугой *FG*;
- ключ второго уровня – предшествующая траектория; таким образом, вложенная таблица имеет смысл распределения счётчиков для данной предшествующей траектории; например, для примера на рис. 1 и схождения в виде узла *D* с дугой *BD* возможный ключ длины 1 единственен − узел *A* с дугой *AB*;
- ключ третьего уровня – последующая траектория; значение − счётчик; например, для примера на рис. 1 для схождения в виде узла *D* с дугой *BD* и предшествующей

траектории из одного расхождения в виде узла *D* с дугой *BD* возможно следующее состояние таблицы:

- ключ – расхождение в виде узла *D* с дугой *DE*, значение – счётчик 10;
- ключ – расхождение в виде узла *D* с дугой *DF*, значение – счётчик 40.

Соответственно, для очередной траектории для инкремента счётчика достаточно найти или создать его в трёхуровневой таблице.

Подобное представление статистики позволяет легко получить условные вероятности последующих траекторий в зависимости от предшествующих. Например, для указанного для третьего уровня таблицы примера условные вероятности исполнения дуг *DE* и *DF* при условии произошедшей предшествующей траектории из одного расхождения в виде узла *D* с дугой *BD* и схождения в виде узла *D* с дугой *BD* составляют:

- дуги *DE* – 20%;
- дуги *DF* – 80%.

Из-за буферизации могут быть потеряны несколько траекторий на краях буфера схождений или расхождений. Доля потерянных траекторий тем меньше, чем больше размер буферов. Также из соображений оптимальности работы кэша инструкций и цикловых оптимизаций для цикла обработки буферов предпочтительны большие буферы. Выбраны размеры буферов по 2^{25} элементов: они достаточно большие и в сумме занимают в памяти процесса около 0,5 Гб, что является приемлемой нагрузкой на память программы.

4.3 Оптимизация

Исследование накладных расходов на профилирование неоптимизированных версий инструментирования и обработки показало, что наибольшее замедление достигается из-за вызова функций библиотеки поддержки на каждой инструментированной дуге, тогда как внутри эти функции в обычном случае только пополняют соответствующие буферы. Соответственно, для ускорения работы требуется как можно сильнее уменьшить количество вызовов библиотеки поддержки.

Для этого в самой программе при инструментировании создаются буферы узлов схождения и расхождения. Но для того, чтобы в программе оказалось только по одному буферу каждого вида, программа должна собираться в режиме `-fwhole` или `-flto`.

На инструментируемых дугах вместо вызовов библиотеки поддержки строится специальный код для пополнения буферов и редкого вызова функции из библиотеки поддержки для обработки буферов и пополнения статистики. Эквивалентный код инструментирования на языке Си представлен на рис. 4 и рис. 5. В этих примерах буферы схождений и расхождений `conv_buffer` размером `max_conv_size` и `div_buffer` размером `max_div_size`. Также в данных примерах счётчики актуальных размеров буферов – `conv_size` и `div_size`.

Таким образом получается, что вместо вызовов функций библиотеки поддержки на каждой инструментируемой дуге вызовы происходят в среднем 1 раз на `max_conv_size` или `max_div_size` прохождений инструментируемых дуг за счёт переноса буферов схождений и расхождений из библиотеки поддержки в профилируемую программу.

Помимо этого, имеются и оптимизации в самой библиотеке поддержки:

- эксперименты показали, что количество частых траекторий до и после узла схождения мало; таким образом, хэш-таблицы второго и третьего уровня можно заменить на TNV-таблицы [5] – массивы фиксированного размера из ключей, значений и счётчиков, где при увеличении счётчика элемент может "всплыть" на одну позицию выше; при использовании периодической очистки нижней (более редкой) половины такой таблицы и при условии, что количество частых элементов не более

половины размера TNV-таблицы, вероятность упустить какой-то частый элемент становится крайне малой;

```
if ( conv_size >= max_conv_size ) {
    TraceBunch( conv_buf,
                conv_size,
                div_buf,
                div_size );
    conv_size = 0;
    div_size = 0;
}
conv_node.module = module_id;
conv_node.func = func_id;
conv_node.node = node_id;
conv_node.edge = edge_id;
conv_buf[conv_size].node     = conv_node;
conv_buf[conv_size].next_div = div_size;
conv_size++;
```

Рис. 4. Код инструментирования, создаваемый на дуге узла схождения.
Fig. 4. Instrumentation code, placed at every edge before CFG node with multiple predececcor edges.

```
if ( div_size >= max_div_size ) {
    TraceBunch( conv_buf,
                conv_size,
                div_buf,
                div_size );
    conv_size = 0;
    div_size = 0;
}
div_node.module = module_id;
div_node.func = func_id;
div_node.node = node_id;
div_node.edge = edge_id;
div_buf[div_size] = div_node;
div_size++;
```

Рис. 5. Код инструментирования, создаваемый на дуге узла расхождения.
Fig. 5. Instrumentation code, placed at every edge after CFG node with multiple successor edges.

- на первом уровне таблицы введена регулярная фильтрация редких узлов схождения таким образом, чтобы за всё время профилирования суммарный счётчик удалённых траекторий составил около 1%;
- наивная реализация добавления траектории предполагает активное создание копий данных, хотя они продолжают храниться в буферах; аккуратное использование передачи указателей (с копированием данных только в ключи таблицы) и раскладки ключей в TNV-таблицах в памяти подряд позволило снизить потери на аллокации, деаллокации и кэш-промахи до приемлемых;
- добавлена возможность разрежения получаемого профиля с фактором N: обрабатывается только один набор из N буферов; также добавлена возможность разрежения с фактором, растущим по простым числам;
- обработка буфера узлов расхождения сделана параллельной: каждый поток исполнения (всего их количество равно числу ядер процессора) получает задание на

обработку своей части буфера и хранит свою частичную статистику, при завершении работы программы статистики объединяются.

5. Результаты

Предложен и реализован вариант траекторного профиля, подходящий для коррекции профиля после дублирования узлов при слиянии кода.

Характеристики реализованного профилирования были измерены на компьютере с процессором «Эльбрус-8СВ» (архитектура «Эльбрус» типа VLIW [16], 8 ядер, частота 1,55 ГГц) на бенчмарках SPEC CPU1995, SPEC CPU2000 и SPEC CPU2006 [1]. Характеристики профилирования для измерений:

- фактор прореживания − растущий по простым числам;
- длина траекторий − 4 узла до и 2 после;
- размеры TNV-таблиц для траекторий до и после − 32 элемента для предшествующей траектории и 16 элементов для последующей траектории.

В табл. 1-3 показаны замедления задач наборов SPEC CPU1995, 2000 и 2006 и доля траекторий, чьи счётчики остались после фильтраций, для каждой задачи.

Получены следующие усреднённые характеристики профилирования:

- замедление из-за профилирования:
  - среднее геометрическое: 5,5 раз.
  - медианное: 6,3 раза.
  - худшее: 90,9 раз.
  - лучшее: на 2%.

Табл. 1. Замедление и доля траекторий, чьи счётчики остались после фильтраций, для задач бенчмарка SPEC CPU1995.
Table 1. Slowdown and saved path ratio (after all filtrations) for SPEC CPU1995 benchmark.

Задача	Доля сохранённых траекторий	Замедление, раз
099.go	97,01%	9,174
101.tomcatv	99,73%	1,250
103.su2cor	99,06%	2,632
104.hydro2d	99,50%	41,667
107.mgrid	98,92%	2,128
110.applu	99,99%	6,329
124.m88ksim	98,98%	1,894
125.turb3d	99,01%	3,096
126.gcc	95,63%	6,250
129.compress	99,92%	5,682
130.li	95,77%	10,204
132.ijpeg	98,82%	12,346
134.perl	99,41%	9,259
141.apsi	99,37%	20,000
145.fpppp	99,38%	12,658

146.wave5	99,59%	7,937
147.vortex	96,91%	5,525

Табл. 2. Замедление и доля траекторий, чьи счётчики остались после фильтраций, для задач бенчмарка SPEC CPU2000.

Table 2. Slowdown and saved path ratio (after all filtrations) for SPEC CPU2000 benchmark.

Задача	Доля сохранённых траекторий	Замедление, раз
164.gzip	98,25%	9,615
168.wupwise	99,73%	1,020
172.mgrid	99,76%	1,953
173.applu	100,00%	7,042
175.vpr	99,02%	3,610
176.gcc	96,96%	1,548
177.mesa	98,80%	1,439
178.galgel	98,81%	2,755
179.art	99,93%	1,292
181.mcf	99,10%	2,639
183.equake	99,89%	1,792
186.crafty	93,95%	4,651
187.facerec	99,19%	3,472
189.lucas	99,02%	8,197
191.fma3d	98,99%	6,024
197.parser	90,17%	10,101
200.sixtrack	98,99%	7,937
252.eon	99,58%	8,621
253.perlbmk	99,58%	3,906
254.gap	93,23%	13,333
255.vortex	97,62%	9,709
256.bzip2	98,25%	7,407
300.twolf	98,93%	7,752
301.apsi	99,11%	8,929

- доля траекторий, чьи счётчики остались после всех фильтраций:
  - среднее геометрическое: 97,8%.
  - медианное: 98,9%.
  - худшее: 82,1%.
  - лучшее: 100,0%.
- доля предшествующих траекторий, которые показывают существенную разницу в распределениях вероятности исходящих дуг ближайшего узла расхождения после:
  - среднее геометрическое: 9,6%.
  - медианное: 7,9%.
  - худшее: 1,6%.

- лучшее: 98,5%.

Анализ одного из худших случаев по замедлению (задачи 104.hydro2d, замедляющейся в 42 раза) показал, что значительную долю в замедление вносит сам инструментированный код: задача без библиотеки поддержки замедлилась примерно в 11 раз, а библиотека вносит ухудшение всего около 4 раз. Причина – в том, что многие цикловые оптимизации перестают работать при наличии в теле цикла операций вызова (пусть даже и редких), и иногда добавленное сложное управление (условный оператор с редким переходом по истинности условия) тоже мешает проведению цикловых оптимизаций.

Табл. 3. Замедление и доля траекторий, чьи счётчики остались после фильтраций, для задач бенчмарка SPEC CPU2006.

Table 3. Slowdown and saved path ratio (after all filtrations) for SPEC CPU2006 benchmark.

Задача	**Доля сохранённых траекторий**	**Замедление, раз**
400.perlbench	98,25%	7,092
401.bzip2	96,98%	5,747
403.gcc	95,98%	9,091
410.bwaves	99,91%	11,494
416.gamess	97,51%	1,188
429.mcf	98,39%	4,115
433.milc	99,16%	3,937
434.zeusmp	99,61%	1,661
435.gromacs	97,90%	9,009
436.cactusADM	99,02%	6,452
437.leslie3d	98,95%	5,952
444.namd	99,09%	6,061
445.gobmk	94,08%	2,967
447.dealII	91,21%	2,597
450.soplex	98,94%	3,356
454.calculix	98,19%	3,165
456.hmmer	99,11%	1,484
458.sjeng	91,15%	5,181
459.GemsFDTD	98,64%	2,994
462.libquantum	99,00%	6,897
464.h264ref	97,77%	50,000
465.tonto	82,08%	90,909
470.lbm	99,99%	5,917
471.omnetpp	98,01%	2,801
481.wrf	98,34%	7,519
482.sphinx3	97,95%	6,897
483.xalancbmk	97,06%	10,638

6. Заключение и направления развития

Разработанный вариант траекторного профилирования имеет приемлемые характеристики по накладным расходам, хотя в некоторых случаях имеет большое замедление, вызванное выключением работы оптимизаций из-за инструментирования. В дальнейшем (как развитие данной работы) предполагается вести исследования в следующих направлениях:

- применение профиля к оптимизации слияния кода;
- использование аппаратных возможностей процессоров «Эльбрус», схожих с LBR, для получения траекторий аппаратным способом; для этого потребуется приводить собранные трассы к описанному в данной статье виду;
- использование профилей по [6], представляющих определённый интерес в плане легковесности профилирования, для генерации профилей описанного в данной статье вида; для этого потребуется приводить ациклические номера траекторий к траекториям из узлов схождения и расхождения и проводить дополнительный анализ.

Для реализованной библиотеки поддержки траекторного профилирования получено свидетельство о государственной регистрации программы для ЭВМ №2023685378.

Список литературы / References

[1]. SPEC CPU Benchmarks, Available at: https://www.spec.org/benchmarks.html#cpu, accessed 23.05.2024.
[2]. Mohammed Afraz, Diptikalyan Saha, and Aditya Kanade (2015) P3: Partitioned Path Profiling. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015). Association for Computing Machinery, New York, NY, USA, 485–495. DOI: 10.1145/2786805.2786868.
[3]. Thomas Ball and James R. Larus (1996) Efficient Path Profiling. In Proceedings of the 29th Annual ACM/IEEE International Symposium on Microarchitecture (MICRO 29). IEEE Computer Society, USA, 46–57.
[4]. M.D. Bond and K.S. McKinley (2005) Continuous path and edge profiling. DOI: 10.1109/MICRO.2005.16.
[5]. Eustace A. Calder B., Feller P. Value Profiling and Optimization. Journal of Instruction-Level Parallelism 1 (1999), pp. 1–37.
[6]. Daniele Cono D'Elia and Camil Demetrescu. Ball-Larus Path Profiling across Multiple Loop Iterations. SIGPLAN Not. 48, 10 (oct 2013), pp. 373–390. DOI: 10.1145/2544173.2509521.
[7]. Gabriel Marin, Alexey Alexandrov, and Tipp Moseley (2021) Break Dancing: Low Overhead, Architecture Neutral Software Branch Tracing. In Proceedings of the 22nd ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2021). Association for Computing Machinery, New York, NY, USA, pp. 122–133. DOI: 10.1145/3461648. 3463853.
[8]. David Melski and Thomas Reps (1999) Interprocedural Path Profiling. In Compiler Construction, Stefan Jähnichen (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 47–62.
[9]. Maksim Panchenko, Rafael Auler, Bill Nell, and Guilherme Ottoni. BOLT: A Practical Binary Optimizer for Data Centers and Beyond. In Proceedings of the 2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO 2019). IEEE Press, pp. 2–14.
[10]. Erez Perelman, Trishul Chilimbi, and Brad Calder. Variational Path Profiling. In Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques (PACT '05). IEEE Computer Society, USA, pp. 7–16. DOI: 10.1109/PACT.2005.41
[11]. Subhajit Roy and Y.N. Srikant. Profiling k-Iteration Paths: A Generalization of the Ball-Larus Profiling Algorithm. International Symposium on Code Generation and Optimization (CGO'09), pp. 70–80. DOI: 10.1109/CGO.2009.11
[12]. Kapil Vaswani, Aditya V. Nori, and Trishul M. Chilimbi (2007) Preferential Path Profiling: Compactly Numbering Interesting Paths. In Proceedings of the 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '07). Association for Computing Machinery, New York, NY, USA, pp. 351–362. DOI: 10.1145/1190216.1190268
[13]. Toshiaki Yasue, Toshio Suganuma, Hideaki Komatsu, and Toshio Nakatani (2004) Structural Path Profiling: An Efficient Online Path Profiling Framework for JustIn-Time Compilers. J. Instruction-Level Parallelism 6.

[14]. Cliff Young and Michael D. Smith. Improving the Accuracy of Static Branch Prediction Using Branch Correlation. SIGOPS Oper. Syst. Rev. 28, 5 (Nov. 1994), 232–241. DOI: 10.1145/381792.195549

[15]. Нейман-заде М. И. and Королёв С. Д. 2020. Руководство по эффективному программированию на платформе «Эльбрус». АО «МЦСТ».

[16]. А.К. Ким, В.И. Перекатов, and С.Г. Ермаков. 2013. Микропроцессоры и вычислительные комплексы семейства «Эльбрус». Питер, СПб.

Информация об авторах / Information about authors

Виктор Евгеньевич ШАМПАРОВ – программист в АО «МЦСТ» с 2017 года, аспирант МФТИ. Сфера научных интересов: оптимизации в компиляторах, профилирование.

Viktor Evgenievich SHAMPAROV – software engineer in MCST Design Center since 2017 and post-graduate student in MIPT. Research interests: compiler optimizations, profiling.

Мурад Искендер оглы НЕЙМАН-ЗАДЕ – начальник отделения разработки систем программирования в АО «МЦСТ». Его научные интересы включают методы оптимизации кода, JIT-компиляцию, профилирование и библиотеки ускоренных вычислений.

Murad Iskender ogly NEIMAN-ZADE – head of the Department of Programming Systems of MCST Design Center. His research interests include compiler optimizations, JIT compilation, profiling and accelerated computation libraries.

DOI: 10.15514/ISPRAS-2024-36(4)-5

Creating Distributed Artificial Neural Networks Based on Orthogonal Transformations

N.A. Vershkov, ORCID: 0000-0001-5756-7612 <vernick61@yandex.ru>
M.G. Babenko, ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>
V.V. Lutsenko, ORCID: 0000-0003-4648-8286 <vvlutcenko@ncfu.ru>
N.N. Kuchukova, ORCID: 0000-0002-8070-0829 <nkuchukova@ncfu.ru>

North-Caucasus Federal University, Stavropol,
1, Pushkin st., Stavropol, 355017, Russia.

Abstract. The article addresses the issue of separating input information of artificial neural networks into modules using orthogonal transformations. This separation enables modular organization of neural networks with layer separation, facilitating the use of the proposed approach for distributed computing. Such an approach is required for organizing the operation of neural networks in fog and edge computing environments, as well as for high-performance computing across multiple low-performance computational nodes. The possibility of cross-layer separation of artificial neural networks using orthogonal transformations is theoretically substantiated, and practical examples of such an approach are provided. A comparison of the characteristics of modular neural networks using various types of orthogonal transformations, including the Haar wavelet transform, is conducted.

Keywords: orthogonal transformations; modular artificial neural networks; neural network optimization; wavelet transformations.

For citation: Vershkov N.A., Babenko M.G., Lutsenko V.V., Kuchukova N.N. Creating distributed artificial neural networks based on orthogonal transformations. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024. pp. 57-68. DOI: 10.15514/ISPRAS-2024-36(4)-5.

Acknowledgements. The research was supported by the Russian Science Foundation Grant No. 24-21-00149, https://rscf.ru/en/project/24-21-00149/.

Создание распределенных искусственных нейронных сетей на основе ортогональных преобразований

Н.А. Вершков, ORCID: 0000-0001-5756-7612 <vernick61@yandex.ru>
М.Г. Бабенко, ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>
В.В. Луценко, ORCID: 0000-0003-4648-8286 <vvlutcenko@ncfu.ru>
Н.Н. Кучукова, ORCID: 0000-0002-8070-0829 <nkuchukova@ncfu.ru>

Северо-Кавказский федеральный университет,
355017, Россия, г. Ставрополь, ул. Пушкина, д. 1.

Аннотация. В статье рассматривается вопрос разделения входной информации искусственных нейронных сетей на модули с помощью ортогональных преобразований. Благодаря такому разделению становится возможным модульная организация нейронных сетей с разделением слоев, что, в свою очередь, позволяет использовать предлагаемый подход для организации распределенных вычислений. Такой подход требуется при организации работы нейронных сетей в среде туманных и периферийных вычислений, организации высокопроизводительных вычислений на множестве вычислительных узлов невысокой производительности. Теоретически обоснована возможность поперечнослойного разделения искусственных нейронных сетей с помощью ортогональных преобразований и приведены примеры практической реализации такого подхода. Проведено сравнение характеристик модульных нейронных сетей с применением различных видов ортогональных преобразований, в том числе с помощью вейвлет-преобразования Хаара.

Ключевые слова: ортогональные преобразования; модульные искусственные нейронные сети; оптимизация нейронных сетей; вейвлет-преобразования.

Для цитирования: Вершков Н.А., Бабенко М.Г., Луценко В.В., Кучукова Н.Н. Создание распределенных искусственных нейронных сетей на основе ортогональных преобразований. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 57–68 (на английском языке). DOI: 10.15514/ISPRAS–20124–36(4)–5.

Благодарности. Исследование выполнено за счет гранта Российского научного фонда № 24-21-00149, https://rscf.ru/project/24-21-00149/.

1. Introduction

The practical need to represent functions of n variables as a superposition of functions from a smaller number of variables arose due to the development of the theory and practice of neural networks. The basis of Artificial Neural Networks (ANN) is the Kolmogorov-Arnold theorem [1, 2], which showed the possibility of representing a continuous real function of n variables $f(x_1, x_2, \dots, x_n)$ as a superposition of functions of a smaller number of variables.

A.N. Gorban [3] concludes that while the Kolmogorov-Arnold theorem guarantees the exact representation of functions of many variables in the class of continuous functions, the practical computation of most functions is only approximate even when exact formulas are available. The solution lies in approximating the function $f(x_1, x_2, \dots, x_n)$ on a compact set Q using a sequence of polynomials (theorems of Weierstrass, Stone). Furthermore, functions can be approximated through linear operations and superpositions of one-variable functions [3]. This approach gained popularity after the works of McCulloch and Pitts [4], which predicted the emergence of ANN. The Hecht-Nielsen theorem [5] was a significant advancement in ANN, demonstrating the possibility of approximating a multi-variable function with a single hidden layer ANN in a non-constructive manner. However, the single-layer perceptron based on the Hecht-Nielsen theorem demonstrated low efficiency.

The emergence of multilayer ANNs and the development of methods for their training have made it possible to solve problems of classification, extrapolation, feature extraction, etc. under conditions of high uncertainty, i.e. to obtain satisfactory results with a sufficiently small training sample size.

Due to the very rapid growth of the amount of data generated and the need to process it, ANNs also increase in size and require significant expenditure of computational resources. Natural Language Processing (NLP) is of great interest and also requires very large ANNs. For example, the popular GPT-3 ANN created in 2020 uses 175 billion parameters. It is clear that such ANNs require very high computational costs to run, which can only be achieved in cloud data centres.

Simultaneously, there is an increasing demand for data processing in close proximity to the equipment being used. This demand has led to the rise of edge computing and fog computing [6], which are becoming more popular due to their enhanced information security and the limited communication channels used for cloud computing. However, the computational nodes of fog computing typically lack the necessary power to run ANNs. Therefore, ANNs must be optimized by reducing the network size while only slightly degrading performance. Furthermore, modular artificial neural networks (ANNs) have been developed on multiple computational nodes [7, 8], creating distributed computing structures. However, the proposed ideas for modular ANNs are based on the assumption that network layer separation is impossible [9], and therefore rely on sequential separation into modules, one layer at a time. However, this approach does not address the main issue of resource estimation, as the number and performance of available fog computing nodes are typically unknown beforehand. The same problem arises when utilising a swarm of unmanned aerial vehicles (UAV). A single vehicle, equipped with a low-power computational node, cannot perform ANN operations. By leveraging the computing capabilities of a UAV swarm, it becomes feasible to operate ANNs of significant size. In this case, it is crucial to optimize the amount of information transferred between computational nodes. When separating the ANN layer by layer, the amount of information is significant due to the large number of parameters in each layer.

Ahmed and Rao [10] present their approach to building image recognition systems with optimal architecture. They suggest using orthogonal transformations to optimize image recognition algorithms, which reduces the number of significant features and the size of the classifier, a forward propagation ANN. The authors propose a concept of optimization and partitioning into ANN modules based on this approach.

2. Utilizing Orthogonal Transformations for Optimization of Neural Networks and Modularization

Dimensionality reduction is a transformation of data from a high-dimensional dataset to a lower-dimensional vector by eliminating uninformative features while preserving the structure and information contained within them to the maximum extent possible [12]. This transformation typically involves two steps: feature generation and selection [13]. In the first step, features that most comprehensively describe the research object are identified, while selection involves identifying features with the best classification properties for the given task. Commonly used methods for dimensionality reduction include Principal Component Analysis (PCA) [14], Factor Analysis (FA) [15], Linear Discriminant Analysis (LDA) [15], Singular Value Decomposition (SVD) [16], Kernel PCA [17], Independent Component Analysis (ISA) [18], Matrix Factorization [19], among others. However, they all have a significant drawback: dimensionality reduction requires preprocessing of information, which can sometimes demand considerable time and computational resources.

N. Ahmed and K. R. Rao [10] proposed optimizing the structure of the input signal through orthogonal transformations, rather than the ANN architecture. This approach is of interest because it allows for the realization of ANNs based on existing principles, approaches, and libraries. By reducing the amount of the input signal, it is possible to decrease the size of the ANN. If the input signal is divided into modules, processing can be carried out by several ANN modules. The orthogonal transformations discussed in the works of N. Ahmed and K. R. Rao can be considered as the first step, i.e., feature generation.

The method of orthogonal transformations is a well-known technique associated with the concept of orthogonal functions. A set of functions of a real variable, denoted by $\{d_i(t)\} = \{d_1, d_2, \dots, d_n\}$, is considered orthogonal on the segment $[0; T]$ if the following condition is satisfied:

$$\int_0^T d_i(t) d_j(t) dt = \begin{cases} k, \text{если } i = j, \\ 0, \text{если } i \neq j. \end{cases} \quad (1)$$

where k is the autocorrelation coefficient of the function $d_i(t)$.

If $x(t)$ is a function of a real variable on the interval $[0; T]$, it can be represented as a series

$$x(t) = \sum_{n=1}^{\infty} a_n d_n(t). \quad (2)$$

In (2), a_n represents the expansion coefficients, which can be determined by

$$a_n = \frac{1}{k} \int_0^T x(t) d_n(t) dt.$$

From the definition of a closed (complete) orthogonal set, it can be inferred that a function $x(t)$ can be represented as a finite set of expansion coefficients $\{a_0, a_1, \dots, a_n\}$ by decomposing it into orthogonal functions. Even if the set of orthogonal functions is not closed, a finite set of coefficients can still be used. In this case, the representation of the function $x(t)$ is not exact, but rather an interpolation based on a certain criterion. The most common criterion used is the least squares principle, which is defined by the functional.

$$\Phi = \int \left(x(t) - \sum_{n=0}^{l} a_n d_n(t) \right)^2. \quad (3)$$

Once the value of the small ε, has been determined, the number of members of the series can be calculated so that the condition $\Phi \leq \varepsilon$ is satisfied.

The Fourier transform is a well-known orthogonal transformation that is widely used in the theory of information processing and transmission. It allows for the transition from the time representation of a signal to the frequency representation and vice versa. In this context, we will consider the application of an orthogonal transformation based on the Fourier transform [10, 11]. In some cases, functions such as Lagger, Lejandre, Hermite, Walsh, Chebyshev, Adamar, etc. may be more appropriate than trigonometric functions as a kernel. As the input and output signals are represented discretely, it is possible to use variants of the discrete Fourier transform, including the Discrete Cosine Transform (DCT) [10]. The DCT is commonly used in the JPEG format for lossy image compression and operates exclusively with real numbers. The DCT employs a set of basis vectors in the form of [10], which is explored on the interval $[0, \pi]$:

$$\left\{ \frac{1}{\sqrt{n}}, \sqrt{\frac{2}{n}} cos \frac{(2m+1)k\pi}{2n} \right\}. \quad (4)$$

Here, k represents the harmonic (coefficient) number, and $m = 0,1,2, \dots, n-1$, representing the size of the initial data array. Equation (4) represents a class of discrete Chebyshev polynomials [10]. The DCT has a notable property where the basis vectors approximate the eigenvectors of the Toeplitz matrices, allowing for effective compression of the original signal using DCT. Therefore, applying an orthogonal transform of the form (1) generates a set of coefficients. In the case of the Fourier transform and its variant DCT, these coefficients represent the amplitudes of frequency harmonics. In this case, the signal can be completely restored or restored with a certain level of accuracy by summing the harmonic components (inverse Fourier transform), depending on the number of components summed.

For the second step (feature selection), N. Ahmed and K. R. Rao [10] utilized the root-mean-square deviation (RMSD) of coefficients in their studies. The higher the RMSD of a coefficient, the more pronounced its classification capability. However, this measure is inconvenient as it requires additional processing, similar to other dimensionality reduction methods. While the issue of orthogonal transformation in the first layer of neural networks was addressed [23], a fundamentally different approach is needed for real-time feature selection.

When transitioning to the investigation of a function presented in both time and frequency domains, it is necessary to delineate the distinction in its reconstruction in each case. The representation of a continuous function by discrete samples in time is defined by the Whittaker-Kotelnikov-Shannon theorem [20], which states that the sampling rate should exceed the maximum frequency of the signal by a factor of two or more. Each sample characterizes the instantaneous value of the function at time t_i. The expansion coefficient a_i represents the projection of the function onto the i-th orthogonal function of the chosen basis over the interval $[0, T]$. Moreover, a higher value of a_i indicates a closer affinity of the investigated function to the i-th component. We shall employ the concept of approximation accuracy for closed systems of orthogonal functions [21]. Based on the Lyapunov-Steklov (Parseval) equality:

$$B = \int_0^T x^2(t)dt = \sum_{n=0}^{\infty} a_n^2,$$

Then the relative integral accuracy of the approximation can be estimated as

$$\gamma = \frac{\sum_{n=0}^{k} a_n^2}{B} = \frac{a_0^2}{B} + \frac{a_1^2}{B} + \cdots + \frac{a_n^2}{B}. \quad (5)$$

In (5), each term characterizes the contribution of each projection to the formation of the original function. Based on this, one can speak about the accuracy of approximating a function obtained from a limited number of components (coefficients).

Thus, an approximate function ($\gamma < 1$) can be used for training ANNs, consequently reducing the size of the ANN. From the course of mathematical analysis, there is a known relationship between the smoothness of a function and the rate of decrease of Fourier coefficients: if a function on an interval has piecewise continuous derivatives of the first and higher orders, then it converges to the original function absolutely and uniformly [22]. Hence, it follows that the partial sums of the coefficients decrease as the component harmonic numbers increase. Therefore, dividing the spectrum can be divided into two parts, based on equal number of frequencies, then we obtain the following values of integral approximation accuracy for each half of the spectrum:

$$\gamma_1 = \frac{\sum_{n=0}^{k/2} a_n^2}{B}, \quad (6)$$

$$\gamma_2 = \frac{\sum_{n=k/2+1}^{k} a_n^2}{B}. \quad (7)$$

Since the spectral density in (7) will be smaller than in (6), the accuracy of approximation by the neural network trained on the first part of the spectrum will be higher than that of the second. This will be demonstrated experimentally in Section 3. Depending on the problem being solved, the number of parts into which the spectrum of the input function is divided may vary. Additionally, the parts of the spectrum do not necessarily have to be identical. If there are several computational nodes of higher power, a larger portion of the spectrum can be allocated to them using a larger size ANN.

By using wavelet transforms instead of Fourier-like transforms, we can combine the operations of feature generation and selection and obtain a gain in the number of relevant features without additional research [24]. In general, wavelets are a system of functions of the following form:

$$\varphi_{a,b}(x) = \sqrt{2^a}\varphi(2^a x - b). \quad (8)$$

If V_a is the space spanned by the system of functions (8), then the following inclusions hold [25]: $V_0 \subset V_1 \subset \cdots \subset V_a$. Thus, we obtain a sequence of nested subspaces $V_i \subset L^2(R)$, each equipped with an orthonormal basis $\{\varphi_{i,b}(x)\}$. This sequence of subspaces can be used to approximate a function $f(x)$ from $L^2(R)$ by its projection operator

$$P_a: L^2(R) \to V_a, P_a(f) = \sum_{b \in Z} (f, \varphi_{a,b}) \varphi_{a,b}(x).$$

The projections P_i become increasingly accurate approximations of $f(x)$ as i increases. Returning to neural networks, the following analogy can be drawn. A set of input vectors $\{f_i(x_j)\}$ in the $L^2(R)$ space can be projected onto a set of subspaces $V_0 \subset V_1 \subset \cdots \subset V_a$ such that each projection P_i is an approximation of the input data. By training a neural network on the projections P_0, we obtain the coarsest approximation of the expected outcome. However, due to decimation, this will be the most "compact" approximation, requiring minimal computational resources for operation.

The widely known Haar wavelet [10, 25, 26] allows for partitioning the $L^2(R)$ space into two subspaces, V_0 and V_1. By using V_0 as the base subspace, a modular architecture of neural networks can be constructed, enabling the use of a basic module for low-power devices [24]. Essentially, the Haar wavelet performs a partitioning of the coefficient space into two equal parts [25], as previously proposed. This allows for solving the problem as described above, where the basic module facilitates the application of neural networks on low-computational-power devices with minimal loss of accuracy and without additional training. In practical applications of the proposed theoretical principles, it is important to consider that such transformation can be repeated for each half of the coefficients. In this case, the entire space can be partitioned into 4 parts and a 4-module structure can be formed, and so forth.

Wavelet transformation divides the coefficient space into several equal parts. However, if there exist nodes with high computational capabilities in a distributed neural network, multiple modules can be assigned to such a node. Consequently, the computational load can be distributed more evenly.

The following conclusion can be drawn from this: orthogonal transformations allow you to divide the input signal space into segments (modules). At the same time, due to (1) modules can be processed independently of each other on different nodes of a distributed computing system. Combining the information processed on different nodes can be realised due to the possibility of inverse orthogonal transformation. However, due to the nonlinearity of ANN, the application of the inverse transformation core is usually impossible and requires training of the layer that combines the results of the ANN modules. In this case, the training of the unifying layer is possible together with the training of the modules.

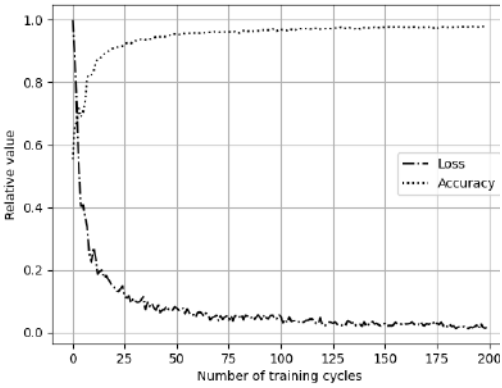
Therefore, the implementation of distributed (modular) ANN requires a number of computational nodes that corresponds to the number of modules to be organised, as well as two additional nodes: one to perform orthogonal transformation and another to combine the results of the modules. This organization optimizes the amount of information transmitted through communication channels. The amount of information transmitted from the orthogonal transformation module to the ANN modules is not greater than the amount of the input signal. Additionally, the amount of information transmitted from the modules to the unifying layer is l times greater than the amount of output data of ANN, where l is the number of modules in the ANN. All other information is transmitted within the layers of the modules and does not require communication channels.

3. Practical Demonstration of Optimization Potential for ANNs and Cross-Layer Network Partitioning for Deployment on Edge and Fog Computing Nodes

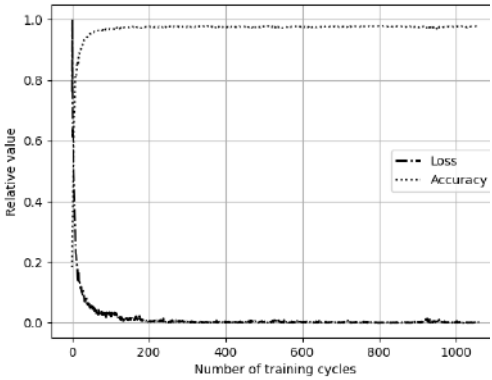
Thus, by performing orthogonal transformation and dividing the coefficients into modules, for example, with the help of ideal digital filters, we obtain a group of feature modules, which can be used to train several ANN modules. In this case, each layer in the ANN modules is reduced

proportionally to the amount of the coefficient modules and does not require from the computational node such a high performance that is required for the ANN as a whole. The real discrete Fourier transform and DCT were chosen for the experiment. The first step is to evaluate the quality of ANN training in time and frequency domains.

The PyTorch library [28] was used to implement the ANN, with the MNIST database [30] serving as the training data. The experiment was conducted on a personal computer running the Windows 10 Home operating system, equipped with an Intel Core i7-10510U processor and 16 GB of RAM. Fig. 1 shows that training of ANN in the frequency domain using Fourier transform takes 5 times more cycles than in the time domain.



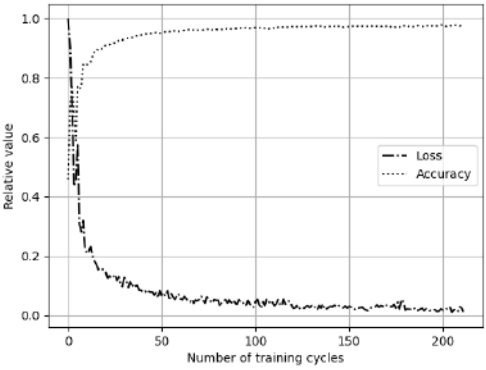
(a) ANN training in the time domain



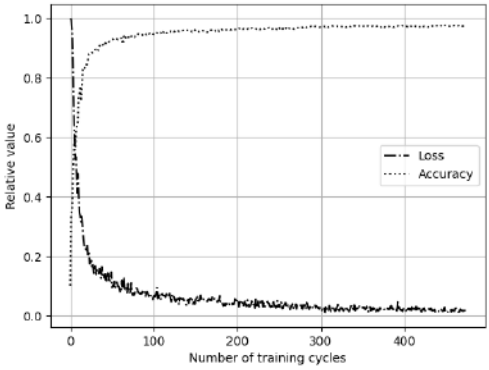
(b) Training of ANN in the frequency domain

Fig. 1. ANN learning processes in time and frequency domains using the Fourier transform.

However, the application of DCT showed a slightly different picture (Fig. 2). ANN training in the frequency domain outperformed training in the time domain by only a factor of 2. This can be explained by the compression properties of DCT mentioned earlier.



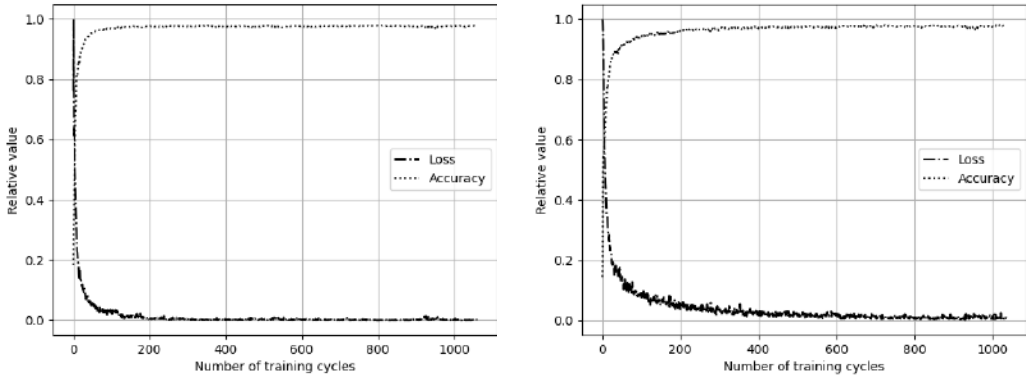
(a) ANN training in the time domain



(b) Training of ANN in the frequency domain

Fig. 2. ANN learning processes in time and frequency domains using DCT.

The problems of optimisation and design of distributed ANNs have a common solution: in optimisation, as many modules are placed on a computing node as the node's computing power allows. And in the design of distributed ANNs, the modules are placed on different nodes connected by communication channels.



(a) Training of 2-module ANN (Fourier transform)

(b) Training of the 2-module ANN (DCT)

Fig. 3. Learning processes of 2-module ANNs.

Consider the construction of modular ANNs. The first layer performs an orthogonal transformation based on the weights set and fixed in the neurons of the first layer [24]. Furthermore, the transformation coefficients obtained after passing through the first layer are divided into two equal parts and sent for processing to different ANNs with layer sizes reduced by a factor of 2. The outputs of the two ANNs are combined in a layer with 20 inputs and 10 outputs. Fig. 3 shows the training processes of the modular ANNs on MNIST data presented in the frequency domain using Fourier transform (Fig. 3a) and using DCT (Fig. 3b). The learning rate is approximately the same and comparable to the learning rate of the monolithic ANN in the frequency domain.

The characteristics of the modular ANNs obtained for optimisation and construction of distributed neural networks are considered. Table 1 presents the characteristics of 2-modular ANN.

Table 1. Characteristics of the 2-module ANN.

	Module 1	Module 2
Recognition quality, %	79.63	69.02
Average time of 1 cycle, sec	0.029	0.024

Table 1 confirms that the recognition quality of the second module is lower than that of the first module. This is due to the fact that the integral energy spectrum used to train the first module was larger than that used for the second module.

Table 2 presents the characteristics of the four-module ANN.

Table 2. Characteristics of a 4-module ANN.

	Module 1	Module 2	Module 3	Module 4
Recognition quality, %	78.76	37.39	17.49	20.86
Average time of 1 cycle, sec	0.013	0.012	0.013	0.012

Table 2 shows that the average execution time per cycle for all modules is approximately the same, while the recognition quality decreases as the module number increases. Module 4 is the exception, with recognition quality exceeding that of module 3. This can be explained by the high-frequency oscillations at the object boundaries, which cause the input signal to the ANN to be not completely smooth.

Table 3 presents the characteristics of the 4-module ANN when the modules are connected in turn.

Table 3. Characteristics of ANN when several modules are connected.

	Module 1	Module 2	Module 3	Module 4
Recognition quality, %	78.76	94.01	97.22	98.05
Average time of 1 cycle, sec	0.013	0.017	0.022	0.028

Table 3 shows the improvement in ANN recognition quality with the addition of modules. The average cycle time increases at a slower rate than the number of connected modules. Therefore, the time required for four modules is only twice that of one module. This is due to the parallelism of the modules working simultaneously.

The Haar wavelet transform is used as an orthogonal transform for constructing a modular ANN. The procedure for wavelet transformation involves passing the input signal through a half-band digital filter with frequency response $h(n)$ (high-pass filter) or $g(n)$ (low-pass filter) [25, 27]:

$$\begin{cases} x(n) * h(n) = \sum_k x(k)h(n-k), \\ x(n) * g(n) = \sum_k x(k)g(n-k). \end{cases}$$

If the input signal of the ANN is a one-dimensional series of numbers with a length of n, we can obtain wavelet transform coefficients at the output by using a one-dimensional convolution layer with either kernel $h(n)$ or $g(n)$. To reduce the number of ANN layers, we can use one layer with two or more different kernels. To achieve this, we create a convolution layer with one input, two or more outputs, and a step equal to the dimensionality of the wavelet kernel. In this case, a convolution layer will be created with two kernels, where the values of $h(n)$ and $g(n)$ are inputted [25].

The Haar wavelet transform, as shown in Table 4, distributes significant features in the frequency-time matrix more strictly. This allows for the separate use of the first module with $\approx 1\%$ loss in accuracy. Additionally, the module's speed is slightly increased due to the use of the convolutional layer as an orthogonal transformer. The use of wavelets for constructing modular ANNs is discussed in more detail in [27].

Table 4. Characteristics of 2-module ANN using wavelet transform.

	Module 1	Module 2
Recognition quality, %	97.01	53.12
Average time of 1 cycle, sec	0.019	0.019

4. Conclusion

The article discusses the use of orthogonal transformations, specifically the Fourier transform, discrete cosine transform, and Haar wavelet transform, for constructing distributed modular ANNs. The approaches outlined in detail in [10] enable the use of these transformations for training ANNs and dividing the input vector into parts for modular network application. The examples provided demonstrate the potential for optimising ANNs for use on low-performance computing devices. They also enable the creation of distributed computing systems with performance equal to that of a monolithic network. The proposed approach is considered advantageous due to its independence from the ANN architecture. This allows for the separation of input information and a reduction in the size of modules, which can be applied to any neural network architecture using standard libraries.

References

[1]. Колмогоров А. Н. О представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного переменного и сложения, Докл. АН СССР, 1957, том 114, номер 5, с. 953–956 / Kolmogorov A. N. On the representation of continuous functions of several

variables as superpositions of continuous functions of one variable and addition. Reports of the Academy of Sciences, 1957, vol. 5, pp. 953-956 (In Russian).

[2]. Арнольд В. И. О представлении функций нескольких переменных в виде суперпозиции функций меньшего числа переменных. Мат. Просвещение, 1958, Вып. 3, с. 41-61 / Arnol'd V. I. On the representation of functions of several variables as a superposition of functions of a smaller number of variables. Mathematical education, 1958, vol. 3, pp. 41-61 (In Russian).

[3]. Горбань А. Н. Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей, Сиб. журн. вычисл. матем., 1998, 1:1, с. 11-24 / Gorban' A. N. Generalised approximation theorem and computational capabilities of neural networks. Siberian Journal of Computational Mathematics, 1998, vol. 1, pp. 11-24 (In Russian).

[4]. Мак-Каллок У., Питтс У. Логическое исчисление идей, относящихся к нервной активности. Автоматы. М.: Изд. иностр. лит., 1956, вып. 5, с. 115-133 / McCulloch W. S., Pitts W. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 1943, vol. 5, issue.4, pp. 115-133.

[5]. Hecht-Nielsen R. Neurocomputing. Addison-Wesley, 1989.

[6]. Кирсанова А.А., Радченко Г.И., Черных А.Н. Обзор технологий организации туманных вычислений. Вестник ЮУрГУ. Серия: Вычислительная математика и информатика, 2020, т. 9, № 3. с. 35–63. doi: 10.14529/cmse200303. / Kirsanova A.A., Radchenko G.I., Chernykh A.N. Review of technologies of fog computing organization. Bulletin of the South Ural State University, 2020, vol. 9, no. 3, pp. 35-63, doi: 10.14529/cmse200303 (In Russian).

[7]. Mao, J., Chen, X., Nixon, K.W., Krieger, C., Chen. MoDNN: Local distributed mobile computing system for deep neural network. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, Lausanne, Switzerland, 2017, pp. 1396-1401, doi: 10.23919/DATE.2017.7927211.

[8]. Бахтин В.В. Алгоритм разделения монолитной нейронной сети для реализации туманных вычислений в устройствах на программируемой логике. Вестник ПНИПУ. Серия: Электротехника, информационные технологии, системы управления, 2022, № 41, с. 123-145. doi: 10.15593/2224-9397/2022.1.06 / Bakhtin V.V. Algoritm razdeleniya monolitnoj nejronnoj seti dlya realizacii tumannyh vychislenij v ustrojstvah na programmiruemoj logike [Separation algorithm of the monolithic neural network for realization of fog computing in devices on programmable logic], Vestnik PNIPU. Elektrotekhnika, informacionnye tekhnologii, sistemy upravleniya [Bulletin of PNIPU. Series: Electrical engineering, information technologies, control systems, 2022, vol. 41, pp. 123-145, doi: 10.15593/2224-9397/2022.1.06 (In Russian).

[9]. Ushakov Y.A., Polezhaev P.N., Shukhman A.E., Ushakova M.V. Distribution of the neural network between mobile device and cloud infrastructure services. Research and development in the field of new IT and their applications, 2018, vol. 14, no.4. pp. 903-910, doi: 10.25559/SITITO.14.201804.903-910.

[10]. Ахмед Н., Рао К.Р. Ортогональные преобразования при обработке цифровых сигналов: Пер. с англ. Под ред. И.Б. Фоменко. М.: Связь, 1980 / Ahmed N., Rao K.R. Orthogonal transforms for digital signal processing. Moscow, Svjaz' publ., 1980. (In Russian).

[11]. А.В. Солодов. Теория информации и ее применение к задачам автоматического управления и контроля. – М.: Издательство «Наука» Главная редакция физико-математической литературы, 1967 / Solodov A.V. Information theory and its application to the tasks of automatic control and monitoring. Publishing house "Science", 1967 (In Russian).

[12]. Burges C.J.C. Dimension reduction: A guided tour. Foundetions and Trends in Machine Learning, 2010, vol.2, no. 4, pp. 275-365, DOI: 10.1561/2200000002.

[13]. Ерохин С.Д., Борисенко Б.Б., Мартишин И.Д., Фадеев А.С. Анализ существующих методов снижения размерности входных данных. // T-Comm: Телекоммуникации и транспорт. 2022. том 16. № 1. с. 30-37 / Erohin S.D., Borisenko B.B., Martishin I.D., Fadeev A.S. Analysis of existing methods to reduce the dimensionality of input data. T-Comm: Telecommunications and Transport, 2022, vol. 16, no. 1. pp. 30-37 (In Russian).

[14]. Jolliffe I.T. Principal component analisis // Second Edition, Springer, 2007, 487 p.

[15]. Mardia K.V., Kent J.T., Bibby J.M. Multivariate analysis (Probability and mathematical statistics). Academic Press Limited, 1995, 521 p.

[16]. Stewart G.W. On the early history on the singular value decomposition. SIAM Review, 1993, vol. 35, no. 4, pp. 551-566. DOI: 10.1137//1035134.

[17]. Van Der Maaten L., Postm, E. O., Van den Herik H. J. Dimensionality reduction: A comparative review. Journal of Machine Learning Research, 2009, vol. 10, 13 p.

[18]. Hyvarinen A., Karhunen J., Oja E. Independent component analysis. John Wiley and Sons, 2001, 504 p.

[19]. Snasel V., Horak Z., Kocibova J., Abraham A. Reducing social network dimensions using matrix factorization analysis. Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining, 2009, pp. 348-351. DOI: 10.1109/ASONAM.2009.48
[20]. Джерри А. Дж. Теория отсчетов Шеннона, ее различные приложения и обобщения. Обзор. ТИИЭР, 1977, т.65, № 11, с.53 - 89. / Jerry A. J. Shannon's reference theory, its various applications and generalisations. Review TIIER, 1977, vol. 65, no. 11, pp. 53-89 (In Russian).
[21]. Дедус Ф.Ф., Куликова Л.И., Панкратов А.Н., Тетуев Р.К. Классические ортогональные базисы в задачах аналитического описания и обработки информационных сигналов: учеб. пособие к спецкурсу. Фак. вычисл. математики и кибернетики (МГУ, ВМК). М.: Издат. отд. Фак. вычисл. математики и кибернетики МГУ им. М.В. Ломоносова, 2004, с. 168 / Dedus F.F., Kulikova L.I., Pankratov A.N., Tetuev R.K. Classical orthogonal bases in problems of analytical description and processing of information signals. FCMIC OF MSU, 2004, 168 p. (In Russian).
[22]. Зорич В. А. Математический анализ: Учебник. Ч. II. М.: Наука Главная редакция физико-математической литературы, 1984. / Zorich V. A. Mathematical analysis. Nauka Publishing House, Main Editorial Office of Physical and Mathematical Literature, 1984, pp. 637 (In Russian).
[23]. Vershkov N.A., Kuchukov, V.A., Kuchukova, N.N., Babenko M., The Wave Model of Artificial Neural Network. Proceedings of the 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, EIConRus, 2020, pp. 542-547, DOI: 10.1109/EIConRus49466.2020.9039172
[24]. Vershkov, N., Babenko, M., Tchernykh, A., Kuchukov, V., Kucherov, N., Kuchukova, N., Drozdov, A. Y. Optimization of Artificial Neural Networks using Wavelet Transforms. Programming and Computer Software, vol. 48, no. 6, pp. 376-384, https://doi.org/10.1134/S036176882206007X
[25]. Смоленцев Н.К. Основы теории вейвлетов. Вейвлеты в MATLAB. М.: ДМК Пресс, 2019 / Smolencev N.K. Fundamentals of wavelet theory. Wavelets in MATLAB. DMK Press, 2019. (In Russian).
[26]. Haar A. Zur theorie der orthogonalen funktionensysteme. Georg-August-Universitat, Gottingen, 1909.
[27]. Vershkov N.A., Babenko M.G., Kuchukova N.N., Kuchukov V.A., Kucherov N.N. Transverse-layer partitioning of artificial neural networks for image classification. Computer Optics, 2024, vol. 48, no. 2, pp. 312-320. DOI: 10.18287/2412-6179-CO-1278.
[28]. PyTorch. Source: https://pytorch.org/
[29]. PyWavelets. Source: https://pypi.org/project/PyWavelets/
[30]. Qiao Y. THE MNIST DATABASE of handwritten digits. 2007. Source: http://www.gavo.t.utokyo.ac.jp/qiao/database.html.

Информация об авторах / Information about authors

Николай Анатольевич ВЕРШКОВ, кандидат технических наук, старший научный сотрудник ФГАОУ ВО «Северо-Кавказский федеральный университет».

Nikolay Anatolievitch VERSHKOV – Cand. Sci. (Tech.), Senior Researcher at the North Caucasus Federal University.

Михаил Григорьевич БАБЕНКО – доктор физико-математических наук, заведующий кафедры вычислительной математики и кибернетики факультета математики и компьютерных наук имени профессора Н.И. Червякова ФГАОУ ВО «Северо-Кавказский федеральный университет». Сфера научных интересов: облачные вычисления, высокопроизводительные вычисления, система остаточных классов, нейронные сети, криптография.

Mikhail Grigoryevich BABENKO – Dr. Sci. (Phys.-Math.), Head of the Department of Computational Mathematics and Cybernetics, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov, North Caucasus Federal University. His research interests include cloud computing, high-performance computing, residue number systems, neural networks, cryptography.

Владислав Вячеславович ЛУЦЕНКО – аспирант, кафедры вычислительной математики и кибернетики факультета математики и компьютерных наук имени профессора Н.И. Червякова ФГАОУ ВО «Северо-Кавказский федеральный университет». Сфера научных

интересов: высокопроизводительные вычисления, система остаточных классов, умный город, нейронные сети, интернет вещей.

Vladislav Vyacheslavovich LUTSENKO – postgraduate student, Department of Computational Mathematics and Cybernetics, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov, North Caucasus Federal University. Research interests: high-performance computing, residue number system, smart city, neural networks, Internet of Things.

Наталья Николаевна КУЧУКОВА – ведущий специалист Центра перспективных исследований и разработок технологий Северо-Кавказского федерального университета.

Natalya Nikolaevna KUCHUKOVA – Leading Specialist, Center for Advanced Research and Technology Development, North Caucasus Federal University.

DOI: 10.15514/ISPRAS-2024-36(4)-6

GraphTyper: Neural Types Inference from Code Represented as Graph

G.A. Arutyunov, ORCID: 00000-0003-4537-4332 <gaarutyunov@edu.hse.ru>
S.M. Avdoshin, ORCID: 0000-0001-8473-8077 <savdoshin@hse.ru>

HSE University
20, Myasnitskaya st., Moscow, 101000, Russia.

Abstract. Although software development is mostly a creative process, there are many scrutiny tasks. As in other industries, there is a trend for automation of routine work. In many cases, machine learning and neural networks have become a useful assistant in that matter. Programming is not an exception: GitHub has stated that Copilot is already used to write up to 30% of code in the company. Copilot is based on Codex, a Transformer model trained on code as a sequence. However, a sequence is not a perfect representation for programming languages. In this work, we claim and demonstrate that by combining the advantages of Transformers and graph representations of code, it is possible to achieve excellent results even with comparably small models.

Keywords: neural networks; transformers; graphs; abstract syntax tree.

For citation: Arutyunov G.A., Avdoshin S.M. GraphTyper: Neural types inference from code represented as graph. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 4, 2024. pp. 69-80. DOI: 10.15514/ISPRAS-2024-36(4)-6.

Acknowledgements. This research was supported in part through computational resources of HPC facilities at HSE University.

GraphTyper: Вывод типов из графовой репрезентации кода посредством нейронных сетей

Г.А. Арутюнов, ORCID: 0000-0003-4537-4332 <gaarutyunov@edu.hse.ru>
С.М. Авдошин, ORCID: 0000-0001-8473-8077 <savdoshin@hse.ru>

Национальный исследовательский университет «Высшая школа экономики» (НИУ ВШЭ), 101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Несмотря на то, что программирование – это творческий процесс, достаточно много времени уходит на решение рутинных задач. Как и в других индустриях в сфере информационных технологий стремятся автоматизировать рутинные задачи. Во многих случаях применяются нейронные сети. Программирование не является исключением: Github заверяют, что уже около 30% кода написано при помощи Copilot. Этот инструмент основан на модели Codex – трансформере, обученном на исходном коде программ. Однако представление кода в виде последовательности, как это сделано в Copilot, не так эффективно. В данной работе мы показали, что использование трансформеров и графового представления кода приводит к очень хорошим результатам даже для маленьких моделей.

Ключевые слова: нейронные сети; трансформеры; графы; абстрактное синтаксическое дерево.

Для цитирования: Арутюнов Г.А., Авдошин С.М. GraphTyper: Вывод типов из графовой репрезентации кода посредством нейронных сетей. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 69–80 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(4)–6.

Благодарности. Исследование выполнено с использованием суперкомпьютерного комплекса НИУ ВШЭ.

1. Introduction

Application of Transformers yet again has managed to break the deadlock: this time in the task of code generation [1–4]. Nevertheless, the versatile Transformer architecture has displayed good results on several benchmarks, in the recent work [5] it was shown that increasing the size of the model doesn't result in a better performance. Moreover, it is evident that context matters a lot to produce a working code. However, it is not practical to relentlessly increase the length of context sequence in a Transformer. Therefore, a different approach is needed to boost the efficiency in machine programming tasks [6].

First of all, an expressive code representation has to be selected. Several ways, including token-based, structured and graph-based approaches, have been reviewed [7]. For instance, graph representation using abstract syntax tree (AST), data- flow graph (DFG) and control-flow graph (CFG) yield good results in such tasks as variable misuse detection and correction [8]. Such graph representation can capture an extensive amount of information about the program's code.

Secondly, a versatile model architecture that supports learning on graphs must be used. Multiple models such as RNN [9], LSTM [10] and CNN [11] with flattened graphs have been used. However, graph-aware model architecture is more suitable for the graph representation of code. For this reason, Graph Neural Networks (GNN) are a more reasonable choice of architecture, namely message-passing neural networks [8].

Nonetheless, in this work we aim to make the most of both worlds: the advantages of Transformer architecture and graph representation of code. For instance, we will use Transformer architecture optimizations [12] and graph code representation created from AST and DFG. To make this possible, we will use Pure Transformers [13] instead of models that have some architectural alterations to support graph structure [14–16].

2. Problem Statement

In this work, we test the ability of Pure Transformers to add types to Python source code based on its graph structure. This task was selected as a starting point for future research due to its practical relevance.

Firstly, dynamically typed languages, such as Python and JavaScript, have gained quite some traction during the last years [17]. However, it doesn't mean they're easier [18–20] or less error-prone than statically typed languages [21]. Moreover, lack of type hints in libraries might lead to expensive errors in fields such as Data Science [22].

There are some tools outside the neural networks domain that perform static type checking and inferencing type annotations [23,24]. Nonetheless, these utilities do not work without type hints in the source code of the dependencies, which is pretty common. To alleviate this, there are proposals about Domain-Specific Languages (DSL) for Data Science [22]. However, it wouldn't work on existing code base and massive adoption is not very likely.

On the other hand, absence of type hints is not a restriction for neural networks (see. Section 5.2). In addition, they don't only find erroneous types in existing codebase [25] but can also be used during development to annotate code on the fly [26].

Most importantly, inferring types requires a model to learn a lot about the source code. Therefore, developing a model with versatile architecture to infer types allows it to be later applied for other tasks.

2.1 Metrics

To test the model, we use two metrics from the Typilus paper [25]:

- Exact Match: Predicted and ground truth types match exactly.
- Match up to Parametric Type: Exact match when ignoring all type parameters.

3. Previous Work

3.1 Graph Representation of Code

AST and DFG have already been used with Transformers in the code generation and summarization tasks [27–29]. In addition, some joint graph structure representations that include different code graphs have been developed, namely code property graph (CPG) [30], that incorporates AST, CFG and PDG (program dependency graph). This graph representation has already been used for vulnerability detection [30] and similarity detection [31].

3.2 Graph Transformers

Graph Transformers is a novel architecture that has been developing in the past few years. They have been applied for several tasks, mostly in the field of molecule generation, node classification and node feature regression [13–16]. Apart from models with alterations to Transformer base architecture [15,16] researchers have recently developed simpler models [13] that are compatible with many popular techniques developed for standard Transformers [12].

3.3 Type Inference with Neural Networks

The task of type inference has been also extensively covered in recent research. Many different architectures have been used for this task: GNNs [25], RNNs [26,32] and Transformers [33,34] among others. Moreover, graph representation of code has been used for the task of type inference in dynamically typed programming languages such as Python [25] and Javascript [35].

However, the power of Graph Transformers and Graph Representation of code hasn't been combined yet to solve the task of type inference in source code. This is the gap our model aims to fill. The results of our model compared to previous work [25,26,32–34] are displayed in Table 1.

Table 1. Quantitative evaluation of models measuring their ability to predict ground truth type annotations. EM – exact match, UTPT – Match up to parametric type.

Top-n	**Top-1**		**Top-3**		**Top-5**	
Metric	**EM**	**UTPT**	**EM**	**UTPT**	**EM**	**UTPT**
GraphTyper	34.7	36.42	45.47	55.01	50.69	64.58
TypeBERT	45.4	48.1	51.4	53.5	54.1	56.5
TypeWriter	56.1	58.3	63.7	67.3	65.9	70.4
Typilus	66.1	74.2	71.6	79.8	72.7	80.9
Type4Py	75.8	80.6	78.1	83.8	78.7	84.7
TypeGen	79.2	87.3	85.6	91	87	91.7

4. Proposed Solution

4.1 Dataset

To train and test the model we gathered 600 Python repositories from GitHub containing type annotations from Typilus [25]. We clone these repositories and use pytype [24] for static analysis, augmenting the corpus with inferred type annotations. The top 175 most downloaded libraries are added to the Python environment for type inference. Through deduplication, we remove over 133 thousand code duplicates to prevent bias.

The resulting dataset comprises 118,440 files with 5,997,459 symbols, of which 252,470 have non-Any non-None type annotations. The annotations exhibit diversity with a heavy tailed distribution, where the top 10 types cover half of the dataset, primarily including str, bool, and int. Only 158 types have over 100 annotations, while the majority of types are used fewer than 100 times each, forming 32% of the dataset. This distribution underscores the importance of accurately predicting annotations, especially for less common types. The long-tail of types consists of user-defined and generic types with various type arguments.

The source files are processed to generate graphs that contain AST, DFG, as well as lexical and syntactical information. An example of such a graph is shown on Fig. 1.

In addition to extracting graphs from source code AST, we split them by setting a maximum node and edges number in one graph. For this, we prune the graphs around nodes that have annotations that are later used as targets during training and testing. Finally, we split the data into train-validation-test sets with proportions of 70-10-20, respectively.

4.2 Model Architecture

We base our model architecture on TokenGT [13]. The main advantage of this model is that standard Transformer architecture is not altered to support graph data. It allows us to use some advantages developed specifically for Transformers. For instance, Performer [12] is used to speed up training by using linear time as space complexity.

The main idea of the authors is that combining appropriate token-wise embeddings and self-attention over the node and edge tokens is expressive enough to accurately encode graph structure to make

graph and node-wise predictions. The embeddings in the model are composed of orthonormal node identifiers, namely Laplacian eigenvectors obtained from eigendecomposition of graph Laplacian matrix. In addition, type identifiers are used to encode types of tokens (nodes or edges).

In our model, we use node and edge types extracted from code as token features. Node ground truth annotations are added to the features and randomly masked during training. The overall architecture of the model is displayed at Fig. 2.

Predicting type annotations in graph domain is a node classification task. However, since we are using a Pure Transformer with graphs represented as a sequence of tokens, the task can be reduced to token classification. In the Natural Language Processing (NLP) domain, this is a ubiquitous task, also known as Named Entity Recognition (NER).

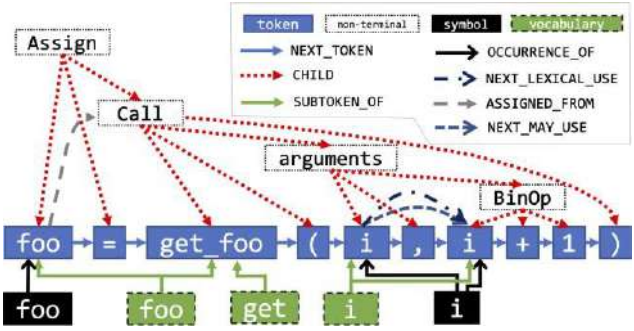


Fig. 1. Sample graph for foo=get foo(i, i+1) showing different node and edge types implemented by Allamanis et al. [25].

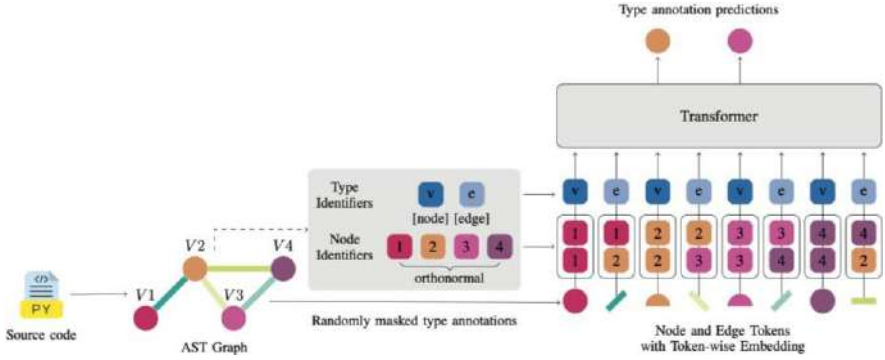


Fig. 2. GraphTyper Architecture. The source code is first transformed into AST graph, then type annotations are randomly masked. The graph is enriched by token type identifiers (node or edge) and orthonormal node identifiers obtained from eigendecomposition of Laplacian matrix. The resulting graph is fed through a Transformer Encoder to obtain type annotations for masked nodes.

Encoder-only architecture has been widely used for the NER task, namely BERT is one of the most popular models [36,37]. We adapt similar architecture by randomly masking type annotations. We then apply an MLP layer to the output of TokenGT [13] to get logits of type annotations.

Masked model architecture is very versatile, and the pre- trained model can be later easily fine-tuned for other tasks, similar to the approaches from the NLP-domain [36]. For example, error [38] and vulnerability [39] data can be added to the code graph to detect and fix them [40–44].

5. Experiment and Ablation Results

To select the final model architecture, we test different models. For our experiments and ablation analysis, we train and test the models using one sample repository. We also limit the number of types

in the vocabulary to one hundred to speed up training and use less resources. To test the models, we calculate Top-n predictions similar to the previous work [26]. Table 2 depicts the results of the experiments and ablation. The model was trained on 1 NVIDIA Tesla V100 32 GB NVLink [45] for 10 epochs with the batch size of 32 graphs.

Table 2. Experiment results of top-n predictions for different model variants.
EM – exact match, UTPT – Match up to parametric type.

Top-n	**Top-1**		**Top-3**		**Top-5**	
Metric	**EM**	**UTPT**	**EM**	**UTPT**	**EM**	**UTPT**
Plane Transformer	10.15	19.46	15.06	29.40	16.81	37.91
+ Node & Type Identifiers	30.88	36.55	40.33	50.37	42.82	56.01
+ Type Annotations	33.36	**42.28**	41.71	52.90	43.62	57.00
+ Decoder (Autoencoder)	15.90	16.65	28.26	32.81	44.17	56.33
or Longer Context	**38.49**	39.80	**53.14**	**57.41**	**58.80**	**67.38**
or More Parameters	29.39	31.82	44.85	49.72	49.74	56.14

5.1 Validating the necessity of node and type identifiers that encode graph structure

First of all, we remove the node and type identifiers introduced by Kim et. al [13]. Our ablation analysis demonstrates that indeed, the graph structure embeddings play a key role in model quality. By removing them from the model, we are left with a simple Transformer that makes predictions only based on AST nodes and edges types without any information about graph structure. Such a model outputs the worst results among all the experiments.

5.2 Using the model without node type annotations

In addition, we try to remove the type annotations from the model completely. This alteration turns our training into a masked NER task. Surprisingly, our model performs well in such conditions. This means that the selected graph representation of code contains a lot of necessary information to infer types.

5.3 Increasing the number of parameters

As we can see, increasing the number of parameters also increases the predictive power of the model. However, increasing the parameters indefinitely is not very practical and requires a lot of computational resources [6]. Moreover, keeping the low number of parameters allows us to use longer context length (more node and edges in graph) during inference with same resource capabilities. Therefore, we don't change the parameter number of the final model, so it remains compact.

5.4 Testing different context length

As for the context length, i.e., maximum number of nodes in graph (512 vs. 1024), our findings are aligned with the conclusions from previous work [6]: longer context increases the performance of the model. However, the AST representation of source code is very bloated and even having a lot of nodes in the graph might not capture enough useful information to make quality predictions. In addition, increasing the context length drastically slows down the training process. Thus, in future research, we will be working on finding a better and more compact graph representation of code.

5.5 Testing different Transformer architectures

Recently, Masked Graph Autoencoders have been applied for the tasks of link prediction and node classification [46], as well as feature reconstruction [47,48]. To validate the robustness of the Encoder-only Model, we also implement a Masked Autoencoder Model. For this, we adapt the approach of Hou et. al [48] for our model. We introduce a learnable mask token and a decoder based on the encoder layers. We reconstruct the type annotations by re-masking the target nodes before feeding them into the decoder. However, we do not observe as good results as with a simple Encoder-only model.

6. Known Limitations

6.1 Size of Type Vocabulary

Since we define our task as node (token) classification, we feed our transformer output into a classifier linear head. Therefore, our type vocabulary is limited. Because of the computational resources' constraints, we limit it to one thousand types.

This issue is addressable by formulating the task as Deep Similarity Learning Problem [49,50]. In this way, the model will output vector representations of types that can be grouped into cluster of similar types. After that, an algorithm such as KNN [51] is used to transform vector representation into a probability of each type [25,26]. Illustration for such an approach is depicted on Fig. 3. The approach follows the methodology described in Typilus paper [25].

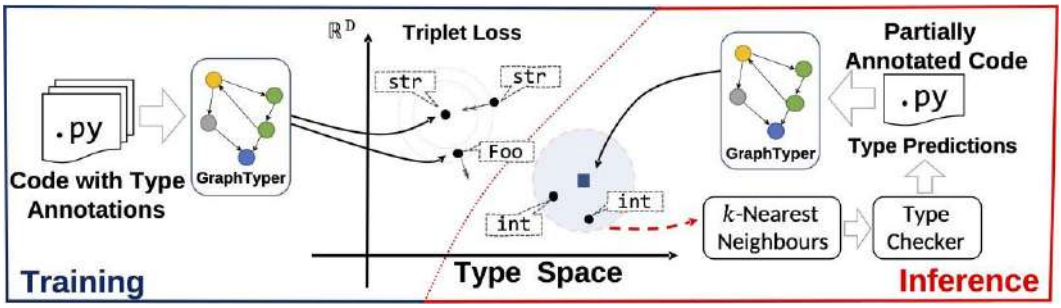


Fig. 3. Solution to the problem using Deep Similarity Learning [25].

6.2 Absence of Natural Language information

In our work, we use only categorical features of nodes and edges of code graph, e.g., AST node types and Python type annotations. Therefore, it would be challenging to apply it directly for tasks such as code generation, because the representation doesn't encode any information about variable names.

There are several approaches that would help address this issue. Firstly, it is possible to use the model output as graph encoding that would be later fed into another model along with tokenized code [52]. This approach could also address the issue from the previous section, since types would be treated as a set of text tokens [34]. Secondly, it is possible to use neural networks to infer variables' names from the context they are used in [53].

7. Future Work

In this work, we explored the application of Graph Transformers for type inference. The versatile architecture of the proposed solution lets us explore other tasks.

7.1 Universal code graph representation

If a universal version of graph code representation is used, similar to CPG [30], we can train the model for multiple programming languages [29]. However, because of the differences of type systems, separate models would be trained for each programming language for better results.

7.2 Detecting duplicates

It is crucial to address the issue of duplicates in source code to train neural networks for code. Several architectures have already been used for such task: Transformers [54], GNNs [55] and RNNs [56]. We believe that the graph representation obtained with our model can be successfully used for code clone detection.

7.3 Code and docstring generation

Firstly, we can train the model using a technique similar to generative pretrained models [57,58] or masked language models [52] to generate code. Secondly, our model can be used to generate code summarization or docstring generation [59,60]. This could only be possible if we adapt some of the approach discussed in the previous section.

7.4 Vulnerability and error detection

Another useful task is to detect errors and generate fixes [61,62]. This is possible by simply adding features that contain error indication or types. Similar approach can be used to scan for vulnerabilities [40,41,44]. Fixing bugs and vulnerabilities, however, would imply that the graph structure could change. Therefore, solving this task would require the model to be modified for graph generation [63].

7.5 Refactoring

Finally, we can extend our model with information about changes to analyze them and propose refactoring possibilities [64]. This goal could be achieved by using the model from the previous section.

8. Conclusion

As for the conclusion, we were able to create a universal model based on TokenGT [13] and code represented as graphs. One of the most important advantages of this model is that it uses the code graph directly. Secondly, the model can be modified to fit other tasks, such as code generation and summarization, docstring generation, refactoring and many more. The code graph can also be extended by different features and node types, since the representation does not differ depending on graph structure. The source code is available at this https URL [65].

References / Список литературы

[1]. M. Chen, J. Tworek, H. Jun, Q. Yuan, H.P. de O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, Evaluating large language models trained on code, ArXiv Prepr. ArXiv210703374 (2021).

[2]. D. Hendrycks, S. Basart, S. Kadavath, M. Mazeika, A. Arora, E. Guo, C. Burns, S. Puranik, H. He, D. Song, Measuring coding challenge competence with apps, ArXiv Prepr. ArXiv210509938 (2021).

[3]. Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, J. Keeling, F. Gimeno, A.D. Lago, T. Hubert, P. Choy, C. de, Competition-Level Code Generation with AlphaCode, (n.d.) 74.

[4]. E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, C. Xiong, A Conversational Paradigm for Program Synthesis, ArXiv Prepr. ArXiv220313474 (2022).

[5]. F.F. Xu, U. Alon, G. Neubig, V.J. Hellendoorn, A Systematic Evaluation of Large Language Models of Code, ArXiv Prepr. ArXiv220213169 (2022).

[6]. G.A. Arutyunov, S.M. Avdoshin, Big Transformers for Code Generation, Proc. Inst. Syst. Program. RAS 34 (2022) 79–88. https://doi.org/10.15514/ispras-2022-34(4)-6.
[7]. S.M. Avdoshin, G.A. Arutyunov, Code Analysis and Generation Methods Using Neural Networks: an Overview, Inf. Technol. 28 (2022) 378–391. https://doi.org/10.17587/it.28.378-391.
[8]. M. Allamanis, M. Brockschmidt, M. Khademi, Learning to represent programs with graphs, ArXiv Prepr. ArXiv171100740 (2017).
[9]. M. White, M. Tufano, C. Vendome, D. Poshyvanyk, Deep learning code fragments for code clone detection, in: 2016 31st IEEEACM Int. Conf. Autom. Softw. Eng. ASE, IEEE, 2016: pp. 87–98.
[10]. H. Wei, M. Li, Supervised Deep Features for Software Functional Clone Detection by Exploiting Lexical and Syntactical Information in Source Code., in: IJCAI, 2017: pp. 3034–3040.
[11]. L. Mou, G. Li, L. Zhang, T. Wang, Z. Jin, Convolutional neural networks over tree structures for programming language processing, in: Thirtieth AAAI Conf. Artif. Intell., 2016.
[12]. K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, A. Weller, Rethinking Attention with Performers, (2020).
[13]. J. Kim, T.D. Nguyen, S. Min, S. Cho, M. Lee, H. Lee, S. Hong, Pure Transformers are Powerful Graph Learners, (2022). https://doi.org/10.48550/arXiv.2207.02505.
[14]. V.P. Dwivedi, X. Bresson, A Generalization of Transformer Networks to Graphs, (2021). https://doi.org/10.48550/arXiv.2012.09699.
[15]. D. Kreuzer, D. Beaini, W.L. Hamilton, V. Létourneau, P. Tossou, Rethinking Graph Transformers with Spectral Attention, (2021). https://doi.org/10.48550/arXiv.2106.03893.
[16]. C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, T.-Y. Liu, Do Transformers Really Perform Bad for Graph Representation?, (2021). https://doi.org/10.48550/arXiv.2106.05234.
[17]. P.M. Julia Elliott, 2021 Kaggle Machine Learning & Data Science Survey, (2021). https://kaggle.com/competitions/kaggle-survey-2021.
[18]. M.P. Robillard, What Makes APIs Hard to Learn? Answers from Developers, IEEE Softw. 26 (2009) 27–34. https://doi.org/10.1109/MS.2009.193.
[19]. M.P. Robillard, R. Deline, A field study of API learning obstacles, Empir. Softw Engg 16 (2011) 703–732. https://doi.org/10.1007/s10664-010-9150-8.
[20]. M.F. Zibran, F.Z. Eishita, C.K. Roy, Useful, But Usable? Factors Affecting the Usability of APIs, in: 2011 18th Work. Conf. Reverse Eng., 2011: pp. 151–155. https://doi.org/10.1109/WCRE.2011.26.
[21]. N. Alzahrani, F. Vahid, A. Edgcomb, K. Nguyen, R. Lysecky, Python Versus C++: An Analysis of Student Struggle on Small Coding Exercises in Introductory Programming Courses, in: Proc. 49th ACM Tech. Symp. Comput. Sci. Educ., Association for Computing Machinery, New York, NY, USA, 2018: pp. 86–91. https://doi.org/10.1145/3159450.3160586.
[22]. L. Reimann, G. Kniesel-Wünsche, Safe-DS: A Domain Specific Language to Make Data Science Safe, (2023).
[23]. Pyre: A performant type-checker for Python 3, (n.d.). https://pyre-check.org (accessed May 12, 2024).
[24]. PyType: A static type analyzer for Python code, (n.d.). https://github.com/google/pytype (accessed May 12, 2024).
[25]. M. Allamanis, E.T. Barr, S. Ducousso, Z. Gao, Typilus: Neural Type Hints, in: PLDI, 2020.
[26]. A.M. Mir, E. Latoskinas, S. Proksch, G. Gousios, Type4py: Deep similarity learning-based type inference for python, ArXiv Prepr. ArXiv210104470 (2021).
[27]. Z. Sun, Q. Zhu, Y. Xiong, Y. Sun, L. Mou, L. Zhang, Treegen: A tree-based transformer architecture for code generation, in: Proc. AAAI Conf. Artif. Intell., 2020: pp. 8984–8991.
[28]. Z. Tang, C. Li, J. Ge, X. Shen, Z. Zhu, B. Luo, AST-Transformer: Encoding Abstract Syntax Trees Efficiently for Code Summarization, (2021). https://doi.org/10.48550/arXiv.2112.01184.
[29]. K. Wang, M. Yan, H. Zhang, H. Hu, Unified Abstract Syntax Tree Representation Learning for Cross-Language Program Classification, in: Proc. 30th IEEEACM Int. Conf. Program Comprehension, 2022: pp. 390–400. https://doi.org/10.1145/3524610.3527915.
[30]. F. Yamaguchi, N. Golde, D. Arp, K. Rieck, Modeling and Discovering Vulnerabilities with Code Property Graphs, in: 2014 IEEE Symp. Secur. Priv., 2014: pp. 590–604. https://doi.org/10.1109/SP.2014.44.
[31]. J. Liu, J. Zeng, X. Wang, Z. Liang, Learning Graph-based Code Representations for Source-level Functional Similarity Detection, in: 2023 IEEEACM 45th Int. Conf. Softw. Eng. ICSE, 2023: pp. 345–357. https://doi.org/10.1109/ICSE48619.2023.00040.
[32]. M. Pradel, G. Gousios, J. Liu, S. Chandra, TypeWriter: neural type prediction with search-based validation, in: Proc. 28th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng., Association

for Computing Machinery, New York, NY, USA, 2020: pp. 209–220. https://doi.org/10.1145/3368089.3409715.
[33]. K. Jesse, P.T. Devanbu, T. Ahmed, Learning type annotation: is big data enough?, in: Proc. 29th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng., Association for Computing Machinery, New York, NY, USA, 2021: pp. 1483–1486. https://doi.org/10.1145/3468264.3473135.
[34]. Y. Peng, C. Wang, W. Wang, C. Gao, M.R. Lyu, Generative Type Inference for Python, (2023).
[35]. J. Schrouff, K. Wohlfahrt, B. Marnette, L. Atkinson, Inferring javascript types using graph neural networks, ArXiv Prepr. ArXiv190506707 (2019).
[36]. Z. Liu, F. Jiang, Y. Hu, C. Shi, P. Fung, NER-BERT: A Pre-trained Model for Low-Resource Entity Tagging, (2021).
[37]. H. Darji, J. Mitrović, M. Granitzer, German BERT Model for Legal Named Entity Recognition, in: Proc. 15th Int. Conf. Agents Artif. Intell., SCITEPRESS - Science and Technology Publications, 2023. https://doi.org/10.5220/0011749400003393.
[38]. D. Bieber, R. Goel, D. Zheng, H. Larochelle, D. Tarlow, Static Prediction of Runtime Errors by Learning to Execute Programs with External Resource Descriptions, (2022).
[39]. S. Sun, S. Wang, X. Wang, Y. Xing, E. Zhang, K. Sun, Exploring Security Commits in Python, (2023).
[40]. V.-A. Nguyen, D.Q. Nguyen, V. Nguyen, T. Le, Q.H. Tran, D. Phung, ReGVD: Revisiting Graph Neural Networks for Vulnerability Detection, ArXiv Prepr. ArXiv211007317 (2021).
[41]. Z. Li, D. Zou, S. Xu, X. Ou, H. Jin, S. Wang, Z. Deng, Y. Zhong, Vuldeepecker: A deep learning-based system for vulnerability detection, ArXiv Prepr. ArXiv180101681 (2018).
[42]. S. Cao, X. Sun, L. Bo, Y. Wei, B. Li, Bgnn4vd: constructing bidirectional graph neural-network for vulnerability detection, Inf. Softw. Technol. 136 (2021) 106576.
[43]. Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu, Z. Chen, SySeVR: A Framework for Using Deep Learning to Detect Software Vulnerabilities, IEEE Trans. Dependable Secure Comput. (2021) 1–1. https://doi.org/10.1109/TDSC.2021.3051525.
[44]. R. Russell, L. Kim, L. Hamilton, T. Lazovich, J. Harer, O. Ozdemir, P. Ellingwood, M. McConley, Automated vulnerability detection in source code using deep representation learning, in: 2018 17th IEEE Int. Conf. Mach. Learn. Appl. ICMLA, IEEE, 2018: pp. 757–762.
[45]. P.S. Kostenetskiy, R.A. Chulkevich, V.I. Kozyrev, HPC Resources of the Higher School of Economics, J. Phys. Conf. Ser. 1740 (2021) 012050. https://doi.org/10.1088/1742-6596/1740/1/012050.
[46]. Q. Tan, N. Liu, X. Huang, R. Chen, S.-H. Choi, X. Hu, MGAE: Masked Autoencoders for Self- Supervised Learning on Graphs, (2022).
[47]. S. Zhang, H. Chen, H. Yang, X. Sun, P.S. Yu, G. Xu, Graph Masked Autoencoders with Transformers, (2022).
[48]. Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, J. Tang, GraphMAE: Self-Supervised Masked Graph Autoencoders, (2022).
[49]. S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: 2005 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. CVPR05, 2005: pp. 539–546 vol. 1. https://doi.org/10.1109/CVPR.2005.202.
[50]. W. Liao, M.Y. Yang, N. Zhan, B. Rosenhahn, Triplet-based Deep Similarity Learning for Person Re-Identification, (2018).
[51]. T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1967) 21–27. https://doi.org/10.1109/TIT.1967.1053964.
[52]. S. Tipirneni, M. Zhu, C.K. Reddy, StructCoder: Structure-Aware Transformer for Code Generation, (2022). https://doi.org/10.48550/arXiv.2206.05239.
[53]. R. Bavishi, M. Pradel, K. Sen, Context2Name: A Deep Learning-Based Approach to Infer Natural Variable Names from Usage Contexts, (2018).
[54]. Zhang, L. Fang, C. Ge, P. Li, Z. Liu, Efficient transformer with code token learner for code clone detection, J Syst Softw 197 (2023). https://doi.org/10.1016/j.jss.2022.111557.
[55]. W. Wang, G. Li, B. Ma, X. Xia, Z. Jin, Detecting code clones with graph neural network and flow-augmented abstract syntax tree, in: 2020 IEEE 27th Int. Conf. Softw. Anal. Evol. Reengineering SANER, IEEE, 2020: pp. 261–271.
[56]. J. Yasaswi, S. Purini, C.V. Jawahar, Plagiarism Detection in Programming Assignments Using Deep Features, in: 2017 4th IAPR Asian Conf. Pattern Recognit. ACPR, 2017: pp. 652–657. https://doi.org/10.1109/ACPR.2017.146.
[57]. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, OpenAI Blog 1 (2019) 9.

[58]. T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, Language models are few-shot learners, Adv. Neural Inf. Process. Syst. 33 (2020) 1877–1901.
[59]. A.V.M. Barone, R. Sennrich, A parallel corpus of python functions and documentation strings for automated code documentation and code generation, ArXiv Prepr. ArXiv170702275 (2017).
[60]. X. Liu, D. Wang, A. Wang, Y. Hou, L. Wu, HAConvGNN: Hierarchical attention based convolutional graph neural network for code documentation generation in jupyter notebooks, ArXiv Prepr. ArXiv210401002 (2021).
[61]. S. Bhatia, R. Singh, Automated correction for syntax errors in programming assignments using recurrent neural networks, ArXiv Prepr. ArXiv160306129 (2016).
[62]. Marginean, J. Bader, S. Chandra, M. Harman, Y. Jia, K. Mao, A. Mols, A. Scott, Sapfix: Automated end-to-end repair at scale, in: 2019 IEEEACM 41st Int. Conf. Softw. Eng. Softw. Eng. Pract. ICSE-SEIP, IEEE, 2019: pp. 269–278.
[63]. Khajenezhad, S.A. Osia, M. Karimian, H. Beigy, Gransformer: Transformer-based Graph Generation, (2022).
[64]. R. Cabrera Lozoya, A. Baumann, A. Sabetta, M. Bezzi, Commit2vec: Learning distributed representations of code changes, SN Comput. Sci. 2 (2021) 1–16.
[65]. G. Arutyunov, gaarutyunov/graph-typer, (2024). https://github.com/gaarutyunov/graph-typer (accessed August 12, 2024).

Информация об авторах / Information about authors

Герман Аренович АРУТЮНОВ – магистр факультета компьютерных наук НИУ ВШЭ. Сфера научных интересов: генерация и анализ языков программирования посредством машинного обучения и глубоких нейронных сетей.

German Arsenovich ARUTYUNOV – Master's graduate at the Faculty of Computer Science at HSE University. Research interests include programming language generation and programming language understanding using machine learning and deep neural networks.

Сергей Михайлович АВДОШИН – кандидат технических наук, профессор департамента компьютерной инженерии Московского института электроники и математики им. А.Н. Тихонова НИУ ВШЭ. Сфера научных интересов: разработка и анализ компьютерных алгоритмов, имитация и моделирование, параллельные и распределенные процессы, машинное обучение.

Sergey Mikchailovitch AVDOSHIN – Cand. Sci. (Tech.), Professor of the School of Computer Engineering at Tikhonov Moscow Institute of Electronics and Mathematics HSE University. Research interests include design and analysis of computer algorithms, simulation and modeling, parallel and distributed processing, machine learning.

DOI: 10.15514/ISPRAS-2024-36(4)-7

О времени реализации распределенных вычислений в синхронном режиме при ограниченном числе копий программного ресурса

П.А. Павлов, *ORCID: 0009-0008-1233-7387* <pavlov.p@polessu.by>

Полесский государственный университет,
Республика Беларусь, 225710, г. Пинск, ул. Днепровской флотилии, д. 23.

Аннотация. В статье: построена математическая модель распределенных вычислений при ограниченном числе копий структурированного программного ресурса; в случаях неограниченного и ограниченного параллелизма по числу процессоров многопроцессорной распределенной вычислительной системы решены задачи нахождения минимального времени выполнения неоднородных, однородных и одинаково распределенных конкурирующих процессов в синхронном режиме, обеспечивающем непрерывное выполнение каждого блока программного ресурса всеми процессами.

Ключевые слова: программный ресурс; синхронный режим; неограниченный (ограниченный) параллелизм; структурирование; конвейеризация; диаграмма Ганта; функционал Беллмана-Джонсона.

Для цитирования: Павлов П.А. О времени реализации распределенных вычислений в синхронном режиме при ограниченном числе копий программного ресурса. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 81–98. DOI: 10.15514/ISPRAS–2024–36(4)–7.

On the Implementation Time of Distributed Computing in Synchronous Mode with a Limited Number of Copies of a Software Resource

P.A. Pavlov *ORCID: 0009-0008-1233-7387* <pavlov.p@polessu.by>

Polessky state university,
23, Dneprovskoy flotilii st., Pinsk, 225710, Belarus.

Abstract. The article constructs a mathematical model of distributed computing with a limited number of copies of a structured software resource; in cases of unlimited and limited parallelism by the number of processors of a multiprocessor distributed computing system, the problems of finding the minimum execution time of heterogeneous, homogeneous and identically distributed competing processes in a synchronous mode, ensuring the continuous execution of each block of software resource by all processes.

Keywords: software resource; synchronous mode; unlimited (limited) parallelism; structuring; conveyorization; Gantt chart; Bellman-Johnson functional.

For citation: Pavlov P.A. On the implementation time of distributed computing in synchronous mode with a limited number of copies of a software resource. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024. pp. 81-98 (in Russian). DOI: 10.15514/ISPRAS-2024-36(4)-7.

1. Введение

Повышение производительности вычислительных систем всегда было и остается актуальной проблемой. Но никакая вычислительная система по своей мощности не может сравниться с теми суммарными ресурсами, которые сосредоточены в локальных и глобальных компьютерных сетях. Быстрое развитие информационно-коммуникационных и сетевых технологий привело к интенсивному использованию географически распределенных вычислительных ресурсов и созданию на их основе динамически-масштабируемых высокопроизводительных *распределенных вычислительных систем* (РВС). В литературе отсутствует каноническое определение того, что такое "*распределенная вычислительная система*". Например, профессор Эндрю Стюарт Таненбаум определяет распределенную систему как "набор независимых компьютеров, представляющиеся их пользователям единой объединенной системой" [1]. В книге [2] сказано, что "под распределенной системой мы понимаем всякую вычислительную систему, в которой несколько компьютеров или процессоров, так или иначе, вступают во взаимодействие". Можно также считать, что распределенная система – это система, компоненты которой расположены на разных сетевых компьютерах, которые обмениваются данными и координируют свои действия путем передачи сообщений друг другу [3].

На сегодняшний день существуют различные типы распределенных вычислительных систем – это вычислительные кластеры, симметричные мультипроцессоры (SMP), системы с распределенной разделяемой памятью (DSM), массово–параллельные системы (MPP), мультикомпьютеры, системы облачных, параллельных и грид–вычислений [4]. Из вышесказанного следует, что *распределенные вычислительные системы – это сложные сетевые системы (системы систем), предназначенные для обработки больших объемов данных, одним из основных преимуществ которых является возможность параллельной обработки процессов* [5].

В настоящее время в области распределенных вычислений ведутся интенсивные исследования. Но до сих пор открытыми остаются проблемы оптимальной организации параллельных процессов *в условиях ограниченных ресурсов*. Это порождает сложные в математическом отношении задачи эффективной (оптимальной) реализации заданных объемов вычислений в различных режимах асинхронного и синхронного взаимодействия процессов при неограниченном и ограниченном параллелизме, задачи синхронизации множества параллельных конкурирующих процессов, задачи определения критериев эффективности и оптимальности заданных объемов вычислений, задачи эффективного отображения алгоритмов и соответствующих программных реализаций с учетом особенностей многопроцессорных систем (МС) и вычислительных комплексов (ВК) и др. Данные задачи имеют как прямой, так и обратный характер. При постановке прямых задач условиями являются значения параметров распределенной вычислительной системы, а решением минимальное общее время реализации заданных объемов вычислений. Постановка обратных задач сводится к поиску критериев эффективности и оптимальности организации выполнения множества распределенных процессов. В виду дискретного и комбинаторного характера вышеуказанных задач определенный прогресс на пути их решения может быть достигнут за счет применения математического аппарата и методов дискретных систем и дискретной оптимизации, теории расписаний и сетевых графов, теории алгоритмов и множеств, алгебры матриц и др. [6-8].

Случай, когда в общей памяти РВС размещена одна копия программного ресурса (ПР), с различных точек зрения был изучен в работах [9-15] и др. Но, к сожалению, мало работ по математическому моделированию функционирования распределенных вычислительных систем, в которых в общей памяти одновременно может находиться не одна, а несколько копий программного ресурса [16-19]. Такое обобщение носит принципиальный характер в

виду того, что отражает основные черты мультиконвейерной обработки, а также позволяет сравнить эффективность конвейерной и параллельной обработки.

Исследования в указанном направлении являются чрезвычайно актуальными. Особая активность характерна для Института системного программирования Российской Академии наук, факультета вычислительной математики и кибернетики Московского государственного университета имени М. В. Ломоносова, Объединенного института проблем информатики Национальной Академии наук Беларуси и др.

2. Математическая модель системы распределенных вычислений при ограниченном числе копий программного ресурса

Конструктивными элементами для построения математических моделей функционирования распределенных вычислительных систем являются понятия *процесса* и *программного ресурса*. Как и в работах [9-19] *процесс* будем рассматривается как выполнение последовательности наборов блоков $I_s = (1,\ 2,\ \ldots,\ s)$. Для ускорения выполнения процессы могут выполняться параллельно на разных вычислительных устройствах или псевдопараллельно на одном вычислительном устройстве взаимодействуя между собой. Процессы, которые влияют на поведение друг друга путем обмена информацией, называют *взаимодействующими* процессами. Многократно выполняемую в многопроцессорной системе программу или ее часть будем называть *программным ресурсом* (ПР), а множество процессов его выполняющим – *конкурирующими.*

При решении задач организации взаимодействия в МС и ВК конкурирующих процессов используется концепция *структурирования*. Структурирование предполагает разбиение программы решения сложной задачи на составные ее части (подпрограммы, процедуры, блоки) с последующей организацией линейного или частичного порядка выполнения на множестве этих частей. Как правильно осуществить разбиение сложной программы или системы? На сколько составных частей производить разбиение?

Пусть ПР – программный ресурс, $n \geq 2$ число взаимодействующих конкурирующих процессов. Требуется организовать вычислительный процесс таким образом, чтобы общее время выполнения n процессов, использующих ПР, было минимальным. Одной из стратегий решения данной задачи на $p \geq 2$ процессорах является предоставление каждому процессу отдельной копии ПР. Но этот путь не всегда осуществим из-за ограниченного объема ресурсов вычислительной системы и тем более трудно достижим в случае больших программ, используемых в качестве программных ресурсов. Поэтому при решении данной задачи применяется стратегия последовательного обслуживания n процессов с использованием различных механизмов их синхронизации. В этом случае суммарное время выполнения процессов составит величину $T_{sum} = nT$, где T – время выполнения каждым из процессов программного ресурса.

Время T_{sum} можно существенно сократить, если обеспечить структурирование программного ресурса на блоки Q_1, Q_2, ..., Q_s с последующей конвейеризацией блоков по процессам, а процессов по процессорам многопроцессорной РВС. Структурирование программного ресурса на блоки осуществляется, как правило, либо исходя из физического смысла задачи на этапах создания математической модели и алгоритмов её решения, либо путём анализа готовой, последовательной программы с целью её сегментирования. Число блоков, на которое осуществляется структурирование программного ресурса, зависит от количества процессов и процессоров, длительности выполнения программного ресурса, накладных расходов и других параметров [10-13].

Один из возможных способов (механизмов) взаимодействия процессов, процессоров и блоков следующий. Блоки, процессы и процессоры ВК или МС нумеруются в порядке

1, 2, …, s, 1, 2, …, n и 1, 2, …, p соответственно. Предположим, что все n процессов используют одну копию структурированного программного ресурса. В дальнейшем под процессом будем понимать выполнение всех блоков ПР в порядке 1, 2, …, s. При этом процесс называется *распределённым*, если все блоки или часть из них выполняются на разных процессорах. Операционная система или специально выделенный организующий процесс предоставляет блоки Q_1, Q_2, ..., Q_s каждому из процессов в порядке 1, 2, …, n. В случае распределённой обработки монополизация процессоров процессами не происходит, а блоки одного и того же процесса выполняются на разных процессорах. Очевидно, что при наличии в многопроцессорной системе p процессоров возможно совмещённое во времени выполнение процессов. Запоминание и восстановление промежуточных состояний процессов, запуск процессов на выполнение и их завершение, выбор режимов взаимодействия процессов, процессоров и блоков осуществляет специальная подсистема операционной системы или организующий процесс.

Математическая модель системы распределенной обработки взаимодействующих процессов, конкурирующих за использование ограниченного числа копий структурированного программного ресурса, включает в себя: $p \ge 2$, процессоров многопроцессорной системы, которые имеют доступ к общей памяти; $n \ge 2$, распределенных взаимодействующих конкурирующих процессов; $s \ge 2$, блоков структурированного на блоки программного ресурса; матрицу $T = [t_{ij}]$, $i = \overline{1,n}$, $j = \overline{1,s}$, времен выполнения блоков программного ресурса распределенными взаимодействующими конкурирующими процессами; $2 \le c \le p$, число копий структурированного на блоки программного ресурса, которые могут одновременно находиться в оперативной памяти, доступной для всех p процессоров; $\theta > 0$ – параметр, характеризующий время дополнительных системных расходов, связанных с организацией конвейерного режима использования блоков структурированного программного ресурса множеством взаимодействующих конкурирующих процессов при распределенной обработке.

Будем также предполагать, что число процессов n кратно числу копий c структурированного программного ресурса, т. е. $n = mc$, где $m = n/c \ge 2$, и что взаимодействие процессов, процессоров и блоков программного ресурса подчинено следующим условиям: 1) ни один из процессоров не может обрабатывать одновременно более одного блока; 2) процессы выполняются в параллельно–конвейерном режиме группами, т. е. осуществляется одновременное (параллельное) выполнение c копий каждого блока в сочетании с конвейеризацией группы из c копий Q_j–го блока, $j = \overline{1,s}$, по процессам и процессорам; 3) обработка каждого блока программного ресурса осуществляется без прерываний; 4) в случае, когда число блоков программного ресурса $s \le \left[\frac{p}{c}\right]$, где $[x]$ – целая часть числа, для каждого i–го процесса, где $i = c(l-1)+q$, $l = \overline{1,m}$, $q = \overline{1,c}$, распределение блоков Q_j, $j = \overline{1,s}$, программного ресурса по процессорам осуществляется по правилу: блок с номером j распределяется на процессор с номером $c(j-1)+q$.

В работах [16,17] исследован **асинхронный режим** конвейерной реализации взаимодействия процессов, процессоров и блоков с учетом наличия c копий структурированного программного ресурса, который предполагает, что начало выполнения c копий очередного Q_j–го блока, $j = \overline{1,s}$, определяется наличием c процессоров и готовностью копий блока к

выполнению, при этом программный блок считается готовым к выполнению, если он не выполняется ни на одном из процессоров.

На рис. 1 представлена линейная диаграмма Ганта, иллюстрирующая асинхронный режим выполнения $n=4$ распределенных конкурирующих процессов, использующих $c=2$ копии структурированного на $s=3$ блока программного ресурса в многопроцессорной системе с $p=7$ процессорами и с заданной матрицей времен выполнения блоков с учетом дополнительных системных расходов $T^{\theta}=[t_{ij}^{\theta}]_{4\times3}=[t_{ij}+\theta]_{4\times3}$.

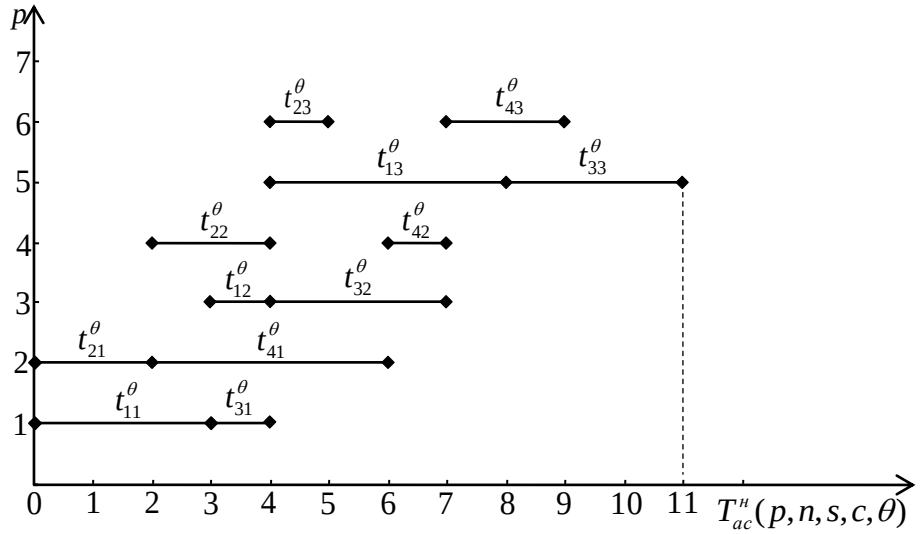


Рис. 1. Асинхронный режим взаимодействия процессов, процессоров и блоков структурированного программного ресурса.
Fig. 1. Asynchronous mode of interaction between processes and processors and blocks of structured software resource.

В работе [20] изучен **первый синхронный режим,** обеспечивающий линейный порядок выполнения блоков программного ресурса внутри каждого из процессов без задержек, т. е. в случае, когда $2\le s\le\left[\frac{p}{c}\right]$, момент завершения выполнения Q_j–го блока, $j=\overline{1,s-1}$, процессом с номером $i=(l-1)c+q$, $l=\overline{1,m}$, $m=\frac{n}{c}$, $q=\overline{1,c}$, на $((j-1)c+q)$–м процессоре совпадает с моментом начала выполнения следующего Q_{j+1}–го блока на процессоре с номером $(jc+q)$ (рис. 2).

Далее, как и в случае одной копии программного ресурса [7], введем следующие определения.

О п р е д е л е н и е 1. Система n распределенных конкурирующих процессов называется **неоднородной**, если времена выполнения блоков программного ресурса Q_1, Q_2, ..., Q_s зависят от объемов обрабатываемых данных и/или их структуры, т. е. разные для разных процессов.

О п р е д е л е н и е 2. Систему распределенных конкурирующих процессов будем называть **однородной**, если времена выполнения Q_j–го блока каждым из i–х процессов равны, т. е. $t_{ij}^{\theta}=t_j^{\theta}$, $i=\overline{1,n}$, $j=\overline{1,s}$.

О п р е д е л е н и е 3. Систему распределенных конкурирующих процессов будем называть **одинаково распределенной**, если времена выполнения всех блоков ПР каждым из процессов совпадают и равны t_i^{θ}, т.е. справедлива цепочка равенств $t_{i1}^{\theta} = t_{i2}^{\theta} = \ldots = t_{is}^{\theta} = t_i^{\theta}$, для всех $i = \overline{1,n}$.

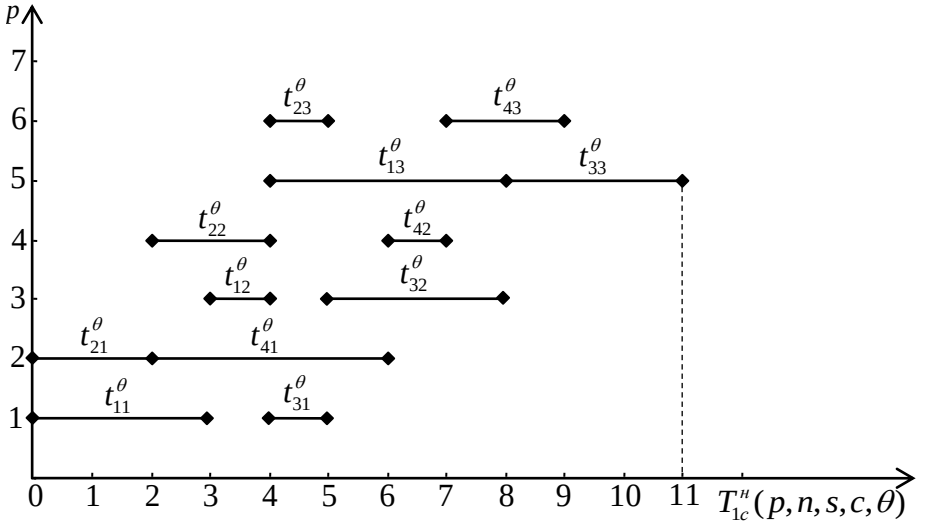


Рис. 2. Первый синхронный режим взаимодействия процессов, процессоров и блоков.
Fig. 2. The first synchronous mode of interaction between processes, processors and blocks.

3. Время реализации неоднородных распределенных процессов

В рамках математической модели организации распределенной обработки конкурирующих процессов с учетом ограниченного числа копий структурированного программного ресурса, введенной в п. 2, исследуем **второй синхронный режим** взаимодействия процессов, процессоров и блоков, который обеспечивает непрерывное выполнение каждого блока программного ресурса всеми процессами, т.е. в случае, когда $2 \le s \le \left[\frac{p}{c}\right]$, момент завершения выполнения i–м процессом, где $i = (l-1)c + q$, $l = \overline{1, m-1}$, $q = \overline{1,c}$, j–го блока, $j = \overline{1,s}$, на $((j-1)c+q)$–м процессоре совпадает с моментом начала выполнения j–го блока процессом с номером $(i+c)$ на этом же процессоре (рис. 3).

Через $T_{2c}^{н}(p,n,s,c,\theta)$ обозначим минимальное общее время выполнения во втором синхронном режиме на p процессорах n *неоднородных* распределенных конкурирующих процессов использующих c копий структурированного на s блоков программного ресурса с матрицей времен выполнения с учетом дополнительных системных расходов $T^{\theta} = [t_{ij}^{\theta}]_{n \times s}$.

Для дальнейшего исследования множество из n процессов разобьем на c подмножеств по m процессов в каждом. В каждое q–е подмножество, $q = \overline{1,c}$, будут включены процессы с номерами $i = c(l-1)+q$, $l = \overline{1,m}$.

При $s = \left[\frac{p}{c} \right]$ рассматриваемый синхронный режим взаимодействия процессов, процессоров и блоков, обеспечивающий непрерывное выполнение каждого из блоков программного ресурса по процессам, совпадает с технологией выполнения операций в многостадийной задаче теории расписаний с m требованиями и s приборами, когда каждый прибор обслуживает требования непрерывно одно за другим [7].

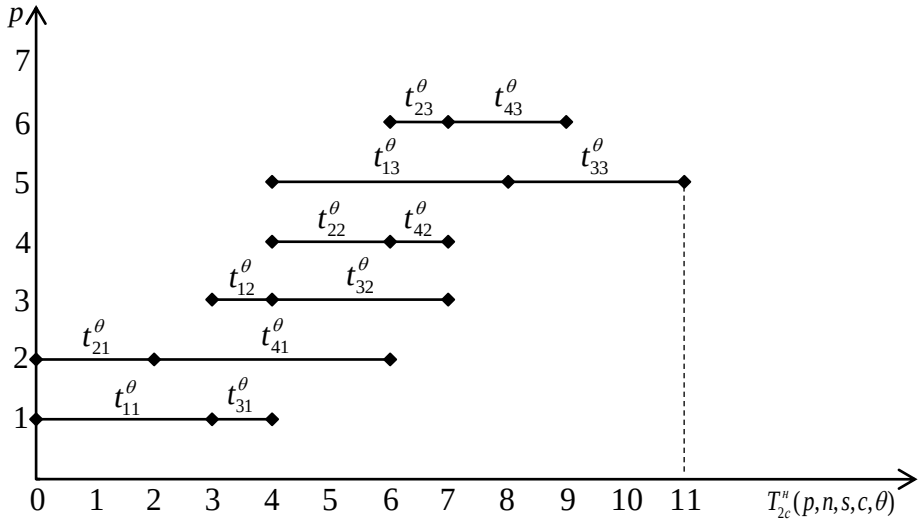


Рис. 3. Второй синхронный режим взаимодействия процессов, процессоров и блоков.
Fig. 3. The second synchronous mode of interaction between processes, processors and blocks.

Следовательно, общее время выполнения n неоднородных конкурирующих распределенных процессов будет определяться при $s = \left[\frac{p}{c} \right]$ функционалом этой задачи. Обозначив через $t^{q}_{ij} = t^{\theta}_{c(i-1)+q,\,j}$, $q = \overline{1,c}$, $i = \overline{1,m}$, $j = \overline{1,\left[\frac{p}{c} \right]}$, время выполнения i–м процессом из q–го подмножества процессов j–го блока с учетом параметра θ , получим:

$$T^{н}_{2c}(p,n,s,c,\theta) = T^{н}_{2c}\left(p,n,\left[\frac{p}{c} \right],c,\theta \right) =$$

$$= \max_{1 \le q \le c} \left(\sum_{j=1}^{[p/c]-1} \max_{1 \le v \le m} \left[\sum_{i=1}^{v} t^{\theta}_{c(i-1)+q,\,j} - \sum_{i=1}^{v-1} t^{\theta}_{c(i-1)+q,\,j+1} \right] + \sum_{i=1}^{m} t^{\theta}_{c(i-1)+q,\left[\frac{p}{c} \right]} \right). \qquad (1)$$

Здесь $\max\limits_{1 \le v \le m} \left[\sum\limits_{i=1}^{v} t^{\theta}_{c(i-1)+q,\,j} - \sum\limits_{i=1}^{v-1} t^{\theta}_{c(i-1)+q,\,j+1} \right]$, $j = \overline{1,\left[\frac{p}{c} \right]-1}$ – начало выполнения j–го блока, начиная со второго, первого процесса из q–го подмножества процессов, а $\sum\limits_{i=1}^{m} t^{\theta}_{c(i-1)+q,\left[\frac{p}{c} \right]}$ – общее время выполнения последнего блока всеми m процессами q–го подмножества, где $q = \overline{1,c}$.

Очевидно, что при $2 \le s < \left\lceil \frac{p}{c} \right\rceil$ формула (1) будет иметь вид:

$$T_{2c}^{u}(p,n,s,c,\theta) = \max_{1\le q\le c}\left(\sum_{j=1}^{s-1} \max_{1\le v\le m}\left[\sum_{i=1}^{v} t_{c(i-1)+q,j}^{\theta} - \sum_{i=1}^{v-1} t_{c(i-1)+q,j+1}^{\theta} \right] + \sum_{i=1}^{m} t_{c(i-1)+q,s}^{\theta} \right).$$

Рассмотрим случай, когда $s = k\left\lceil \frac{p}{c} \right\rceil$, $k > 1$ (рис. 4).

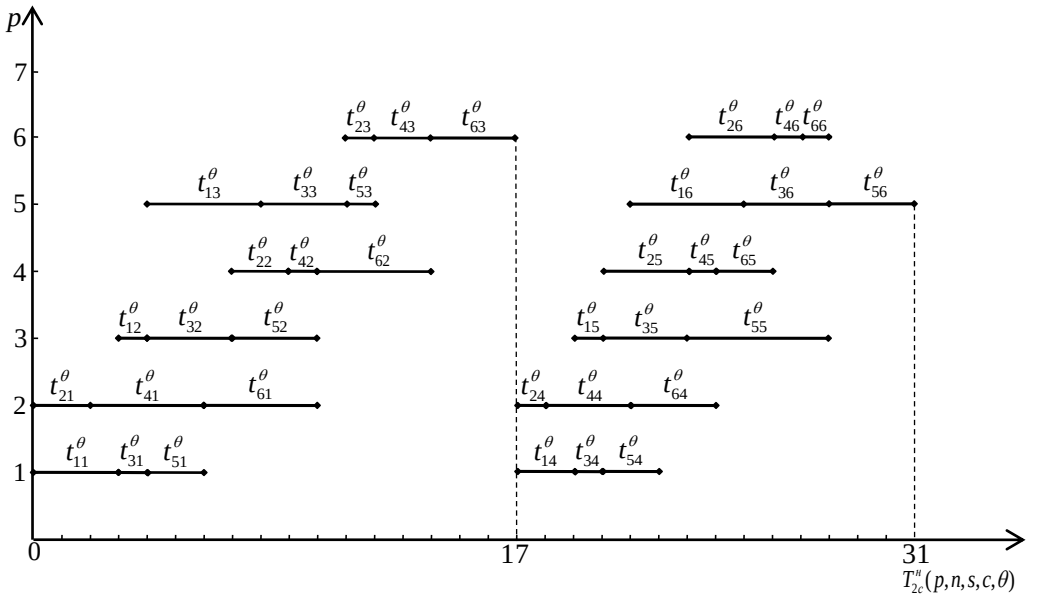


Рис. 4. Несовмещенные диаграммы Ганта при $s = k\left\lceil \frac{p}{c} \right\rceil$, $k > 1$.

Fig. 4. Unaligned Gantt charts when $s = k\left\lceil \frac{p}{c} \right\rceil$, $k > 1$.

Для дальнейшего исследования, как и при достаточном числе процессоров, все множество процессов разобьем на подмножества, причем каждое q–е подмножество, $q = \overline{1,c}$, будет состоять из m процессов с номерами $i = c(l-1)+q$, $l = \overline{1,m}$, которые будут выполняться на процессорах с номерами $(c(j-1)+q)$, $j = \overline{1,\left\lceil \frac{p}{c} \right\rceil}$. Введем следующие обозначения:

- $t_{ij}^{\varphi,q}$ – время выполнения в φ–ой группе блоков i–м процессом из q–го подмножества процессов j–го блока с учетом параметра θ:

$$t_{ij}^{\varphi,q} = t^{\theta}_{c(i-1)+q,(\varphi-1)\left\lceil \frac{p}{c} \right\rceil + j},\ \varphi = \overline{1,k},\ q = \overline{1,c},\ i = \overline{1,m},\ j = \overline{1,\left\lceil \frac{p}{c} \right\rceil};$$

- T_q^{φ} – время выполнения в φ–ой группе блоков q–го подмножества процессов:

$$T_q^{\varphi} = \sum_{i=1}^{m-1} \max_{1 \le u \le \left[\frac{p}{c}\right]} \left[\sum_{j=1}^{u} t^{\theta}_{c(i-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+j} - \sum_{j=1}^{u-1} t^{\theta}_{ci+q,(\varphi-1)\left[\frac{p}{c}\right]+j} \right] + \sum_{j=1}^{[p/c]} t^{\theta}_{c(m-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+j},$$

$$\varphi = \overline{1,k},\ q = \overline{1,c};$$

- T_{φ}^{θ} – общее время выполнения φ–ой группы блоков всеми n процессами на $c\left[\frac{p}{c}\right]$ процессорах с учетом параметра θ:

$$T_{\varphi}^{\theta} = \max_{1 \le q \le c} \left(\sum_{i=1}^{m-1} \max_{1 \le u \le \left[\frac{p}{c}\right]} \left[\sum_{j=1}^{u} t^{\theta}_{c(i-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+j} - \sum_{j=1}^{u-1} t^{\theta}_{ci+q,(\varphi-1)\left[\frac{p}{c}\right]+j} \right] + \right.$$

$$\left. + \sum_{j=1}^{[p/c]} t^{\theta}_{c(m-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+j} \right) = \max_{1 \le q \le c} T_q^{\varphi},\ \varphi = \overline{1,k};$$

- $E_{ij}^{\varphi,q}$ – время завершения обработки в φ–ой группе блоков i–м процессом из q–го подмножества j–го блока:

$$E_{ij}^{\varphi,q} = E^{\varphi}_{c(i-1)+q,j} = \sum_{\mu=1}^{i-1} \max_{1 \le u \le \left[\frac{p}{c}\right]} \left[\sum_{w=1}^{u} t^{\theta}_{c(\mu-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+w} - \sum_{w=1}^{u-1} t^{\theta}_{c\mu+q,(\varphi-1)\left[\frac{p}{c}\right]+w} \right] +$$

$$+ \sum_{w=1}^{j} t^{\theta}_{c(i-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+w} \quad \varphi = \overline{1,k},\ q = \overline{1,c},\ i = \overline{1,m},\ j = \overline{1,\left[\frac{p}{c}\right]}.$$

С учетом обозначений $t_{ij}^{\varphi,q}$, T_{φ}^{θ} и $E_{ij}^{\varphi,q}$ и в силу формулы (1) для нахождения времени выполнения φ–й группы блоков всеми n процессами T_{φ}^{θ} и времени завершения обработки в φ–ой группе блоков i–м процессом из q–го подмножества j–го блока $E_{ij}^{\varphi,q}$, получим следующие соотношения:

$$T_{\varphi}^{\theta} = \max_{1 \le q \le c} \left(\sum_{j=1}^{[p/c]-1} \max_{1 \le v \le m} \left[\sum_{i=1}^{v} t^{\theta}_{c(i-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+j} - \sum_{i=1}^{v-1} t^{\theta}_{c(i-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+j+1} \right] + \sum_{i=1}^{m} t^{\theta}_{c(i-1)+q,\varphi\left[\frac{p}{c}\right]} \right), \quad (2)$$

$$\varphi = \overline{1,k};$$

$$E_{ij}^{\varphi,q} = \sum_{w=1}^{j-1} \max_{1 \le v \le m} \left[\sum_{\mu=1}^{v} t^{\theta}_{c(\mu-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+w} - \sum_{\mu=1}^{v-1} t^{\theta}_{c(\mu-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+w+1} \right] +$$

$$+ \sum_{\mu=1}^{i} t^{\theta}_{c(\mu-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+j},\ \varphi = \overline{1,k},\ q = \overline{1,c},\ i = \overline{1,m},\ j = \overline{1,\left[\frac{p}{c}\right]}.$$

Через $B_{1j}^{\varphi,q}$ обозначим время начала выполнения в φ–ой группе блоков первым процессом из q–го подмножества процессов j–го блока:

$$B_{1j}^{\varphi,q}=\sum_{w=1}^{j-1}\max_{1\le v\le m}\left[\sum_{\mu=1}^{v}t^{\theta}_{c(\mu-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+w}-\sum_{\mu=1}^{v-1}t^{\theta}_{c(\mu-1)+q,(\varphi-1)\left[\frac{p}{c}\right]+w+1}\right], \quad (3)$$

$$\varphi=\overline{1,k},\ q=\overline{1,c},\ j=\overline{1,\left[\frac{p}{c}\right]}.$$

Из анализа последовательных диаграмм Ганта следует, что

$$T_{2c}^{н}(p,n,s,c,\theta)=T_{2c}^{н}\left(p,n,k\left[\frac{p}{c}\right],c,\theta\right)=\sum_{\varphi=1}^{k}T_{\varphi}^{\theta}-\Omega,$$

где $\Omega=\sum_{\varphi=1}^{k-1}\min\{\omega_{\varphi}^{'},\omega_{\varphi}^{''}\}$ является величиной максимально допустимого суммарного совмещения соседних диаграмм по оси времени.

Здесь каждое значение $\min\{\omega_{\varphi}^{'},\omega_{\varphi}^{''}\}$, $\varphi=\overline{1,k-1}$, определяет величину максимально допустимого совмещения по оси времени между парами соседних диаграмм Ганта:

$$\omega_{\varphi}^{'}=\min_{1\le q\le c}\min_{1\le j\le\left[\frac{p}{c}\right]}\{T_{\varphi}^{\theta}-E_{mj}^{\varphi,q}+B_{1j}^{\varphi+1,q}\}; \quad (4)$$

$$\omega_{\varphi}^{''}=\min_{1\le q\le c}\min_{1\le i\le m}\left\{T_{\varphi}^{\theta}-E^{\varphi,q}_{m\left[\frac{p}{c}\right]}+\sum_{\mu=i+1}^{m}t^{\theta}_{c(\mu-1)+q,\varphi\left[\frac{p}{c}\right]}+\sum_{\mu=1}^{i-1}t^{\theta}_{c(\mu-1)+q,\varphi\left[\frac{p}{c}\right]+1}\right\},\ \varphi=\overline{1,k-1}. \quad (5)$$

Тогда для РВС на рис. 4 получим, что $T_{2c}^{н}(7,6,6,2,\theta)=17+14-\min\{6,4\}=27$ (рис. 5).

Для общего случая, когда $s=k\left[\frac{p}{c}\right]+r$, $k\ge 1$, $1\le r<\left[\frac{p}{c}\right]$ (рис. 6), общее время $T_{2c}^{н}\left(p,n,k\left[\frac{p}{c}\right]+r,\theta\right)$ будет определяться по формуле:

$$T_{2c}^{н}\left(p,n,k\left[\frac{p}{c}\right]+r,c,\theta\right)=\sum_{\varphi=1}^{k}T_{\varphi}^{\theta}+T_{k+1}^{\theta}-\sum_{l=1}^{k-1}\min\{\omega_{\varphi}^{'},\omega_{\varphi}^{''}\}-\min\{\omega_{k}^{'},\omega_{k}^{''}\},\ \text{где}$$

$$T_{k+1}^{\theta}=\max_{1\le q\le c}\left(\sum_{j=1}^{r-1}\max_{1\le v\le m}\left[\sum_{i=1}^{v}t^{\theta}_{c(i-1)+q,k\left[\frac{p}{c}\right]+j}-\sum_{i=1}^{v-1}t^{\theta}_{c(i-1)+q,k\left[\frac{p}{c}\right]+j+1}\right]+\sum_{i=1}^{m}t^{\theta}_{c(i-1)+q,\varphi\left[\frac{p}{c}\right]+r}\right), \quad (6)$$

$$\omega_{k}^{'}=\min_{1\le q\le c}\min_{1\le j\le r}\{T_{k}^{\theta}-E_{mj}^{k,q}+B_{1j}^{k+1,q}\}, \quad (7)$$

$$\omega_{k}^{''}=\min_{1\le q\le c}\min_{1\le i\le m}\left\{T_{k}^{\theta}-E^{k,q}_{m\left[\frac{p}{c}\right]}+\sum_{\mu=i+1}^{m}t^{\theta}_{c(\mu-1)+q,k\left[\frac{p}{c}\right]}+\sum_{\mu=1}^{i-1}t^{\theta}_{c(\mu-1)+q,k\left[\frac{p}{c}\right]+1}\right\}. \quad (8)$$

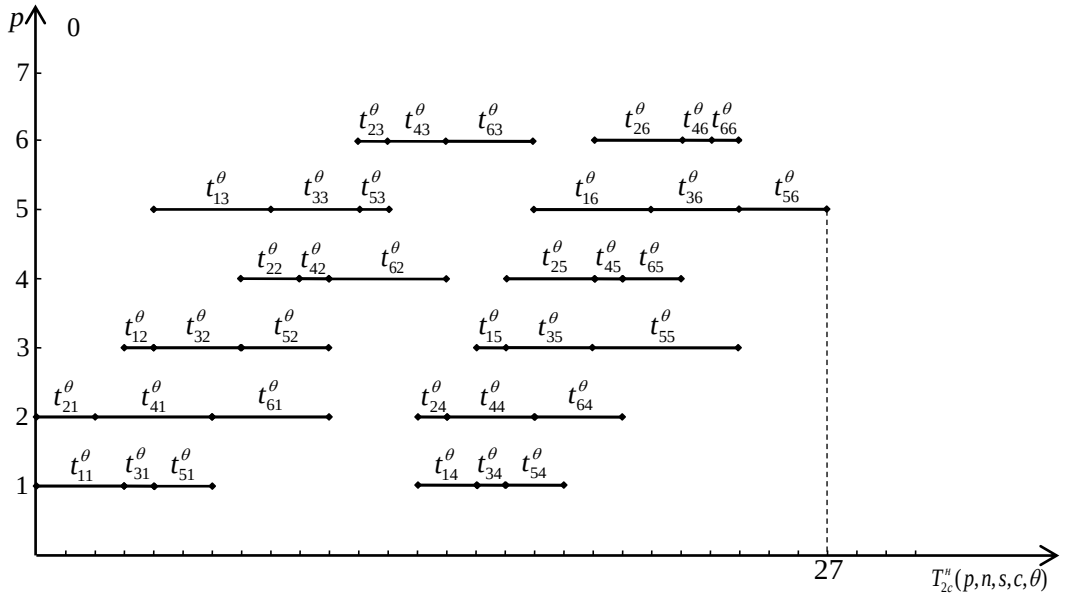


Рис. 5. Совмещенные диаграммы Ганта при $s = k\left[\frac{p}{c}\right],\ k > 1$.

Fig. 5. Combined Gantt charts when $s = k\left[\frac{p}{c}\right],\ k > 1$.

В свою очередь, значения $B_{1j}^{k+1,q}$ с учетом формулы (3) будут найдены из выражения:

$$B_{1j}^{k+1,q} = \sum_{w=1}^{j-1} \max_{1\le v\le m}\left[\sum_{\mu=1}^{v} t^{\theta}_{c(\mu-1)+q,k\left[\frac{p}{c}\right]+w} - \sum_{\mu=1}^{v-1} t^{\theta}_{c(\mu-1)+q,k\left[\frac{p}{c}\right]+w+1}\right],\ q=\overline{1,c},\ j=\overline{1,r}.$$

Таким образом, имеет место следующая

Т е о р е м а 1. *Если взаимодействие процессов, процессоров и блоков структурированного программного ресурса подчинено условиям второго синхронного режима, то для любых* $p \ge 2$, $n \ge 2$, $s \ge 2$, $2 \le c \le p$, $\theta > 0$ *минимальное общее время выполнения неоднородных распределенных конкурирующих процессов определяется следующим образом:*

$$T_{2c}^{н}(p,n,s,c,\theta) = \max_{1\le q\le c}\left(\sum_{j=1}^{s-1} \max_{1\le v\le m}\left[\sum_{i=1}^{v} t^{\theta}_{c(i-1)+q,j} - \sum_{i=1}^{v-1} t^{\theta}_{c(i-1)+q,j+1}\right] + \sum_{i=1}^{m} t^{\theta}_{c(i-1)+q,s}\right),$$

$$\text{при } 2 \le s < \left[\frac{p}{c}\right];$$

$$T_{2c}^{u}(p,n,s,c,\theta)=\begin{cases}\sum\limits_{\varphi=1}^{k}T_{\varphi}^{\theta}-\sum\limits_{l=1}^{k-1}\min\{\omega_{\varphi}^{'},\omega_{\varphi}^{''}\},\ \textit{при}\ \ s=k\left[\dfrac{p}{c}\right],\ \ k>1;\\ \sum\limits_{\varphi=1}^{k}T_{\varphi}^{\theta}+T_{k+1}^{\theta}-\sum\limits_{\varphi=1}^{k-1}\min\{\omega_{\varphi}^{'},\omega_{\varphi}^{''}\}-\min\{\omega_{k}^{'},\omega_{k}^{''}\},\\ \textit{при}\ \ s=k\left[\dfrac{p}{c}\right]+r,\ \ k\geq 1,\ 1\leq r<\left[\dfrac{p}{c}\right].\end{cases}$$

Здесь T_{φ}^{θ} и T_{k+1}^{θ} вычисляются по формулам (2) и (6), а $\omega_{\varphi}^{'}$, $\omega_{\varphi}^{''}$, $\varphi=\overline{1,k}$, по формулы (4), (5), (7), (8).

4. Время выполнения однородных распределенных процессов

Через $T_{2c}^{o}(p,n,s,c,\theta)$ обозначим минимальное общее время выполнения на p процессорах системы n *однородных* распределенных конкурирующих процессов, использующих c копий структурированного на s блоков программного ресурса с учетом дополнительных системных расходов $\theta>0$.

При $s\leq\left[\dfrac{p}{c}\right]$ подставив в формулу (1) значения $t_{ij}^{\theta}=t_{j}^{\theta}$, $i=\overline{1,n}$, $j=\overline{1,s}$, и проведя несложные преобразования, получим:

$$T_{2c}^{o}(p,n,s,c,\theta)=\sum_{j=1}^{s-1}\max_{1\leq v\leq m}\left[\sum_{i=1}^{v}t_{j}^{\theta}-\sum_{i=1}^{v-1}t_{j+1}^{\theta}\right]+\sum_{i=1}^{m}t_{s}^{\theta}=\sum_{j=1}^{s}t_{j}^{\theta}+(m-1)\left(t_{s}^{\theta}+\sum_{j=2}^{s}\max\{t_{j-1}^{\theta}-t_{j}^{\theta},0\}\right).$$

Итак, если обозначить длительность выполнения программного ресурса каждым из процессов через $T_{s}^{\theta}=\sum\limits_{j=1}^{s}t_{j}^{\theta}$, то при $s\leq\left[\dfrac{p}{c}\right]$, имеем:

$$T_{2c}^{o}(p,n,s,c,\theta)=T_{s}^{\theta}+(m-1)\left(t_{s}^{\theta}+\sum_{j=2}^{s}\max\{t_{j-1}^{\theta}-t_{j}^{\theta},0\}\right).$$

Рассмотрим случай, когда $s=k\left[\dfrac{p}{c}\right],\ k>1$.

Используя прием совмещения последовательных диаграмм Ганта, получим следующую формулу для вычисления $T_{2c}^{o}\left(p,n,k\left[\dfrac{p}{c}\right],c,\theta\right)$:

$$T_{2c}^{o}\left(p,n,k\left[\frac{p}{c}\right],c,\theta\right)=\sum_{\varphi=1}^{k}T_{\varphi}^{\theta}-\sum_{\varphi=1}^{k-1}\min\{\psi_{\varphi}^{'},\psi_{\varphi}^{''}\},$$

где T_{φ}^{θ}, $\varphi=\overline{1,k}$ – общее время выполнения φ–х $\left[\dfrac{p}{c}\right]$ блоков структурированного программного ресурса всеми n процессами на $c\left[\dfrac{p}{c}\right]$ процессорах, а $\psi_{\varphi}^{'}$ и $\psi_{\varphi}^{''}$ представляют

собой отрезки возможного совмещения двух последовательных диаграмм по оси времени, $\varphi = \overline{1,k-1}$. Значения T_φ^θ, $\psi_\varphi^{'}$ и $\psi_\varphi^{''}$ находятся по следующим формулам:

$$T_\varphi^\theta = \sum_{j=1}^{[p/c]} t^\theta_{(\varphi-1)\left[\frac{p}{c}\right]+j} + (m-1)\left(t^\theta_{\varphi\left[\frac{p}{c}\right]} + \sum_{j=2}^{[p/c]} \max\{t^\theta_{(\varphi-1)\left[\frac{p}{c}\right]+j-1} - t^\theta_{(\varphi-1)\left[\frac{p}{c}\right]+j}, 0\} \right),\ \varphi = \overline{1,k}; \quad (9)$$

$$\psi_\varphi^{'} = \min_{1\le j\le\left[\frac{p}{c}\right]} \{T_\varphi^\theta - E_{mj}^\varphi + B_{1j}^{\varphi+1}\}, \quad (10)$$

$$\psi_\varphi^{''} = (m-1)\min\{t^\theta_{\varphi\left[\frac{p}{c}\right]}, t^\theta_{\varphi\left[\frac{p}{c}\right]+1}\},\ \varphi = \overline{1,k-1}. \quad (11)$$

Здесь $t^\theta_{(\varphi-1)\left[\frac{p}{c}\right]+j}$ – время выполнения в φ–ой группе j–го блока каждым из m процессов, E_{ij}^φ – времена завершения обработки в φ–ой группе блоков i–м процессом j–го блока, а B_{1j}^φ – время начала выполнения в φ–ой группе блоков первым процессом j–го блока:

$$B_{1j}^\varphi = \sum_{w=1}^{j-1} \max_{1\le v\le m} \left[vt^\theta_{(\varphi-1)\left[\frac{p}{c}\right]+w} - (v-1)t^\theta_{(\varphi-1)\left[\frac{p}{c}\right]+w+1} \right],$$

$$E_{ij}^\varphi = \sum_{w=1}^{j-1} \max_{1\le v\le m} \left[vt^\theta_{(\varphi-1)\left[\frac{p}{c}\right]+w} - (v-1)t^\theta_{(\varphi-1)\left[\frac{p}{c}\right]+w+1} \right] + it^\theta_{(\varphi-1)\left[\frac{p}{c}\right]+j},\ \varphi = \overline{1,k},\ i = \overline{1,m},\ j = \overline{1,\left[\frac{p}{c}\right]}.$$

Для случая $s = k\left[\frac{p}{c}\right] + r$, $k \ge 1$, $1 \le r < \left[\frac{p}{c}\right]$, общее время выполнения множества однородных распределенных конкурирующих процессов определяется по формуле:

$$T_{2c}^o(p,n,kp,c,\theta) = \sum_{\varphi=1}^{k} T_\varphi^\theta + T_{k+1}^\theta - \sum_{\varphi=1}^{k-1} \min\{\psi_\varphi^{'}, \psi_\varphi^{''}\} - \min\{\psi_k^{'}, \psi_k^{''}\}.$$

Здесь

$$T_{k+1}^\theta = \sum_{j=1}^{r} t^\theta_{k\left[\frac{p}{c}\right]+j} + (m-1)\left(t^\theta_{k\left[\frac{p}{c}\right]+r} + \sum_{j=2}^{r} \max\{t^\theta_{k\left[\frac{p}{c}\right]+j-1} - t^\theta_{k\left[\frac{p}{c}\right]+j}, 0\} \right) \quad (12)$$

и обозначает время выполнения последних r блоков всеми n процессами, а $\min\{\psi_k^{'}, \psi_k^{''}\}$ – величина максимального совмещения по оси времени k–й и $(k+1)$–й диаграмм:

$$\psi_k^{'} = \min_{1\le j\le r}\{T_k^\theta - E_{mj}^k + B_{1j}^{k+1}\},\ \psi_k^{''} = (m-1)\min\{t^\theta_{k\left[\frac{p}{c}\right]}, t^\theta_{k\left[\frac{p}{c}\right]+1}\}, \quad (13)$$

$$B_{1j}^{k+1} = \sum_{w=1}^{j-1} \max_{1\le v\le m} \left[vt^\theta_{k\left[\frac{p}{c}\right]+w} - (v-1)t^\theta_{k\left[\frac{p}{c}\right]+w+1} \right],\ j = \overline{1,r}.$$

Таким образом, имеет место

Т е о р е м а 2. *Если взаимодействие процессов, процессоров и блоков структурированного программного ресурса подчинено условиям второго синхронного режима, то для любых*

$p \ge 2$, $n \ge 2$, $s \ge 2$, $2 \le c \le p$, $\theta > 0$ *минимальное общее время выполнения однородных распределенных конкурирующих процессов определяется по формулам:*

$$T_{2c}^{o}(p,n,s,c,\theta) = T_s^{\theta} + (m-1)\left(t_s^{\theta} + \sum_{j=2}^{s} \max\{t_{j-1}^{\theta} - t_j^{\theta}, 0\} \right), \text{ при } 2 \le s \le \left[\frac{p}{c} \right];$$

$$T_{2c}^{o}(p,n,s,c,\theta) = \begin{cases} \sum_{\varphi=1}^{k} T_{\varphi}^{\theta} - \sum_{\varphi=1}^{k-1} \min\{\psi_{\varphi}^{'}, \psi_{\varphi}^{''}\}, \ \textit{при } s = k\left[\frac{p}{c} \right], \ k > 1; \\ \sum_{\varphi=1}^{k+1} T_{\varphi}^{\theta} - \sum_{\varphi=1}^{k} \min\{\psi_{\varphi}^{'}, \psi_{\varphi}^{''}\}, \ \textit{при } s = k\left[\frac{p}{c} \right] + r, \ k \ge 1, \ 1 \le r < \left[\frac{p}{c} \right]. \end{cases}$$

Здесь T_{φ}^{θ}, $\varphi = \overline{1, k+1}$, вычисляются по формулам (9) и (12), а $\psi_{\varphi}^{'}$, $\psi_{\varphi}^{''}$, $\varphi = \overline{1,k}$, по формулы (10), (11), (13).

5. Время выполнения одинаково распределенных процессов

Обозначим через $T_{2c}^{op}(p,n,s,c,\theta)$ минимальное общее время выполнения n *одинаково распределенных* конкурирующих процессов в многопроцессорной системе с p процессорами во втором синхронном режиме взаимодействия процессов, процессоров и блоков с учетом параметра $\theta > 0$, характеризующего время дополнительных системных расходов, связанных с организацией параллельного использования s блоков структурированного программного ресурса множеством конкурирующих процессов при распределенной обработке.

Т е о р е м а 3. *Минимальное общее время выполнения множества конкурирующих одинаково распределенных процессов во втором синхронном режиме при любых* $p \ge 2$, $n \ge 2$, $s \ge 2$, $2 \le c \le p$, $\theta > 0$ *определяется по формулам:*

$$T_{2c}^{op}(p,n,s,c,\theta) =$$

$$= \max_{1 \le q \le c} \begin{cases} T_q^{\theta} + (s-1)t_{\max}^{q}, \ \textit{при } 2 \le s \le \left[\frac{p}{c} \right], \ \textit{или } s > \left[\frac{p}{c} \right], \ \textit{но } T_q^{\theta} \le \left[\frac{p}{c} \right] t_{\max}^{q}, \\ kT_q^{\theta} + \left(\left[\frac{p}{c} \right] - 1 \right) t_{\max}^{q}, \ \textit{при } s = k\left[\frac{p}{c} \right], \ k > 1, \ T_q^{\theta} > \left[\frac{p}{c} \right] t_{\max}^{q}, \\ (k+1)T_q^{\theta} + (r-1)t_{\max}^{q}, \ \textit{при } s = k\left[\frac{p}{c} \right] + r, \ k \ge 1, \ 1 \le r < \left[\frac{p}{c} \right], \ T_q^{\theta} > \left[\frac{p}{c} \right] t_{\max}^{q}. \end{cases}$$

Д о к а з а т е л ь с т в о . При $s \le \left[\frac{p}{c} \right]$ для неоднородных систем распределенных конкурирующих процессов имеет место формула:

$$T_{2c}^{u}(p,n,s,c,\theta) = \max_{1 \le q \le c} \left(\sum_{j=1}^{s-1} \max_{1 \le v \le m} \left[\sum_{i=1}^{v} t_{c(i-1)+q,j}^{\theta} - \sum_{i=1}^{v-1} t_{c(i-1)+q,j+1}^{\theta} \right] + \sum_{i=1}^{m} t_{c(i-1)+q,s}^{\theta} \right).$$

Для одинаково распределенных систем конкурирующих процессов времена t_{ij}^{θ} выполнения блоков Q_j, $j = \overline{1,s}$, программного ресурса каждым из i –х процессов совпадают и равны t_i^{θ} для всех $i = \overline{1,n}$, т. е. справедлива цепочка равенств $t_{i1}^{\theta} = t_{i2}^{\theta} = \ldots = t_{is}^{\theta} = t_i^{\theta}$, $i = \overline{1,n}$, следовательно:

$$T_{2c}^{op}(p,n,s,c,\theta)=\max_{1\le q\le c}\left(\sum_{j=1}^{s-1}\max_{1\le v\le m}\left[\sum_{i=1}^{v}t_{c(i-1)+q}^{\theta}-\sum_{i=1}^{v-1}t_{c(i-1)+q}^{\theta}\right]+\sum_{i=1}^{m}t_{c(i-1)+q}^{\theta}\right)=$$

$$=\max_{1\le q\le c}(T_q^{\theta}+(s-1)t_{\max}^{q}). \qquad (14)$$

Здесь $T_q^{\theta}=\sum_{i=1}^{m}t_{c(i-1)+q}^{\theta}$ – суммарное время выполнения q–й группы блоков всеми m процессами, а $t_{\max}^{q}=\max_{1\le i\le m}t_{c(i-1)+q}^{\theta}$ – максимальное время выполнения блока из этой группы, $q=\overline{1,c}$.

Рассмотрим случай, когда $s=k\left[\frac{p}{c}\right]$, $k>1$ и $T_q^{\theta}>\left[\frac{p}{c}\right]t_{\max}^{q}$, $q=\overline{1,c}$, тогда

$$T_{2c}^{op}(p,n,s,c,\theta)=T_{2c}^{op}\left(p,n,k\left[\frac{p}{c}\right],c,\theta\right)=\sum_{\varphi=1}^{k}T_{\varphi}^{\theta}-\sum_{\varphi=1}^{k-1}\min\{\xi_{\varphi}^{'},\xi_{\varphi}^{''}\}. \qquad (15)$$

С учетом формулы (14) время выполнения φ–й группы блоков T_{φ}^{θ} будет определяться по формуле:

$$T_{\varphi}^{\theta}=\max_{1\le q\le c}\left(T_q^{\theta}+\left(\left[\frac{p}{c}\right]-1\right)t_{\max}^{q}\right),\ \varphi=\overline{1,k}. \qquad (16)$$

Согласно (4) и (5) формулы для величин максимально допустимого совмещения по оси времени между парами соседних диаграмм Ганта $\xi_{\varphi}^{'}$ и $\xi_{\varphi}^{''}$, $\varphi=\overline{1,k-1}$, для одинаково распределенных процессов будут иметь вид:

$$\xi_{\varphi}^{'}=\min_{1\le q\le c}\min_{1\le j\le\left[\frac{p}{c}\right]}\{T_{\varphi}^{\theta}-E_{mj}^{\varphi,q}+B_{1j}^{\varphi+1,q}\}; \qquad (17)$$

$$\xi_{\varphi}^{''}=\min_{1\le q\le c}\min_{1\le i\le m}\left\{T_{\varphi}^{\theta}-E_{m\left[\frac{p}{c}\right]}^{\varphi,q}+\sum_{\mu=i+1}^{m}t_{c(\mu-1)+q}^{\theta}+\sum_{\mu=1}^{i-1}t_{c(\mu-1)+q}^{\theta}\right\},\ \varphi=\overline{1,k-1}. \qquad (18)$$

Так как $t_{ij}^{\theta}=t_i^{\theta}$, $i=\overline{1,n}$, $j=\overline{1,s}$, следовательно, времена начала выполнения в φ–ой группе блоков первым процессом из q–го подмножества процессов j–го блока $B_{1j}^{\varphi,q}$ и времена завершения обработки в φ–ой группе блоков i–м процессом из q–го подмножества j–го блока $E_{ij}^{\varphi,q}$, будут определяться по формулам:

$$B_{1j}^{\varphi,q}=(j-1)t_{\max}^{q},\ \varphi=\overline{1,k},\ q=\overline{1,c},\ j=\overline{1,\left[\frac{p}{c}\right]}; \qquad (19)$$

$$E_{ij}^{\varphi,q}=(j-1)t_{\max}^{q}+\sum_{\mu=1}^{i}t_{c(\mu-1)+q}^{\theta},\ \varphi=\overline{1,k},\ q=\overline{1,c},\ i=\overline{1,m},\ j=\overline{1,\left[\frac{p}{c}\right]}. \qquad (20)$$

Подставим (16), (19), (20) в (17) и (18), получим:

$$\xi'_{\varphi} = \min_{1\le q\le c}\min_{1\le j\le\left[\frac{p}{c}\right]}\left\{T_q^{\theta} + \left(\left[\frac{p}{c}\right]-1\right)t_{\max}^{q} - (j-1)t_{\max}^{q} - \sum_{\mu=1}^{m} t_{c(\mu-1)+q}^{\theta} + (j-1)t_{\max}^{q}\right\} =$$

$$= \min_{1\le q\le c}\left\{\left(\left[\frac{p}{c}\right]-1\right)t_{\max}^{q}\right\}; \qquad (21)$$

$$\xi''_{\varphi} = \min_{1\le q\le c}\min_{1\le i\le m}\left\{T_q^{\theta} + \left(\left[\frac{p}{c}\right]-1\right)t_{\max}^{q} - \left(\left[\frac{p}{c}\right]-1\right)t_{\max}^{q} - \sum_{\mu=1}^{m} t_{c(\mu-1)+q}^{\theta} + \sum_{\mu=i+1}^{m} t_{c(\mu-1)+q}^{\theta} + \sum_{\mu=1}^{i-1} t_{c(\mu-1)+q}^{\theta}\right\} =$$

$$= \min_{1\le q\le c}\min_{1\le i\le m}\left\{\sum_{\mu=i+1}^{m} t_{c(\mu-1)+q}^{\theta} + \sum_{\mu=1}^{i-1} t_{c(\mu-1)+q}^{\theta}\right\} \le \min_{1\le q\le c}\{(m-1)t_{\max}^{q}\}. \qquad (22)$$

По условию $\left[\frac{p}{c}\right]t_{\max}^{q} < T_q^{\theta} = \sum_{i=1}^{m} t_{c(i-1)+q}^{\theta} \le mt_{\max}^{q}$, следовательно:

$$\min_{1\le q\le c}\left\{\left(\left[\frac{p}{c}\right]-1\right)t_{\max}^{q}, (m-1)t_{\max}^{q}\right\} = \min_{1\le q\le c}\left\{\left(\left[\frac{p}{c}\right]-1\right)t_{\max}^{q}\right\}.$$

Получили, что (15) после преобразования будет иметь вид второй формулы теоремы 3:

$$T_{2c}^{op}\left(p,n,k\left[\frac{p}{c}\right],c,\theta\right) = \max_{1\le q\le c}\left\{\sum_{\varphi=1}^{k}\left(T_q^{\theta} + \left(\left[\frac{p}{c}\right]-1\right)t_{\max}^{q}\right) - \sum_{\varphi=1}^{k-1}\left(\left[\frac{p}{c}\right]-1\right)t_{\max}^{q}\right\} =$$

$$= \max_{1\le q\le c}\left\{kT_q^{\theta} + \left(\left[\frac{p}{c}\right]-1\right)t_{\max}^{q}\right\}.$$

Для случая $s = k\left[\frac{p}{c}\right] + r$, $k \ge 1$, $1 \le r < \left[\frac{p}{c}\right]$, $T_q^{\theta} > \left[\frac{p}{c}\right]t_{\max}^{q}$, общее время $T_{2c}^{op}\left(p,n,k\left[\frac{p}{c}\right]+r,c,\theta\right)$ аналогично формуле (15) будет определяться следующим образом:

$$T_{2c}^{op}\left(p,n,k\left[\frac{p}{c}\right]+r,c,\theta\right) = \sum_{\varphi=1}^{k} T_{\varphi}^{\theta} - \sum_{\varphi=1}^{k-1}\min\{\xi'_{\varphi},\xi''_{\varphi}\} + T_{k+1}^{\theta} - \min\{\xi'_{k},\xi''_{k}\}. \qquad (23)$$

Здесь время выполнения $(k+1)$–ой группы блоков T_{k+1}^{θ} с учетом (16) будет определяться по формуле:

$$T_{k+1}^{\theta} = \max_{1\le q\le c}(T_q^{\theta} + (r-1)t_{\max}^{q}). \qquad (24)$$

В $(k+1)$–ой группе блоков времена начала выполнения первым процессом из q–го подмножества процессов и времена завершения обработки i–м процессом из q–го подмножества j–го блока с учетом (19) и (20) будут определяться по формулам:

$$B_{1j}^{k+1,q} = (j-1)t_{\max}^{q},\ q = \overline{1,c},\ j = \overline{1,r}; \qquad (25)$$

$$E_{ij}^{k+1,q} = (j-1)t_{\max}^{q} + \sum_{\mu=1}^{i} t_{c(\mu-1)+q}^{\theta},\ q = \overline{1,c},\ i = \overline{1,m},\ j = \overline{1,r}.$$

Подставив формулы (16), (20) и (25) в (17) и (18), для нахождения $\xi_k^{'}$ и $\xi_k^{''}$ получим формулы аналогичные (21) и (22), следовательно,

$$T_{2c}^{op}\left(p,n,k\left[\frac{p}{c}\right]+r,c,\theta\right)=\max_{1\le q\le c}((k+1)T_q^{\theta}+(r-1)t_{\max}^{q}),$$

что доказывает третью формулу теоремы 3.

6. Заключение

В статье проведено исследование базового второго синхронного режима взаимодействия процессов, процессоров и блоков структурированного программного ресурса при котором обеспечивается непрерывное выполнение каждого блока всеми процессами. В случаях неограниченного и ограниченного параллелизма по числу процессоров распределенной вычислительной системы получены формулы для вычисления минимального времени выполнения множества неоднородных, однородных и одинаково распределенных процессов конкурирующих за использование ограниченного числа копий программного ресурса. Полученные математические соотношения позволят в дальнейшем выполнить сравнительный анализ введенных режимов, провести математическое исследование эффективности и оптимальности организации распределенных вычислений, решить задачи по расчету оптимальных характеристик многопроцессорных распределенных вычислительных систем.

Список литературы / References

[1]. Andrew S. Tanenbaum, Maarten Van Steen. Distributed Systems. Amazon Digital Services LLC, 2023. 684 p.

[2]. Robey R., Zamora Y. Parallel and High Performance Computing. Manning, 2021. 800 p.

[3]. Бабичев С.Л., Коньков К.А. Распределенные системы. М.: Юрайт, 2019. 507 с. / Babichev S., Konkov K. Distributed Systems. Moscow, Juright, 2019, 507 p. (in Russian)

[4]. Топорков В.В., Емельянов Д.М. Модели, методы и алгоритмы планирования в грид и облачных вычислениях. Вестник Московского энергетического института. 2018, №6. С. 75-86. / Toporkov V., Emelyanov D. Models, methods and algorithms for planning in grid and cloud computing. Bulletin of the Moscow Energy Institute. 2018, №6. pp. 75-86. (in Russian)

[5]. Антонов А.С., Афанасьев И.В., Воеводин Вл.В. Высокопроизводительные вычислительные платформы: текущий статус и тенденция развития. Вычислительные методы и программирование. 2021, Том 22. С. 135-177. / Antonov A., Afanasyev I., Voevodin Vl.V. High-performance computing platforms: current status and development trends. Computational methods and programming. 2021, Volume 22. pp. 135-177. (in Russian)

[6]. Емеличев В.А., Ковалев М.М., Кравцов М.К. Многогранники. Графы. Оптимизация. М.: Наука, 1981. 344 с. / Emelichev V., Kovalev M., Kravtsov M. Polyhedra. Graphs. Optimization. Moscow, Nauka, 1981, 344 p. (in Russian)

[7]. Танаев В.С., Сотсков Ю.Н., Струсевич В.А. Теория расписаний. Многостадийные системы. М.: Наука, 1989. 327 с. / Tanaev V., Sotskov Yu., Strusevich V. Polyhedra. Graphs. Optimization. Moscow, Nauka, 1989, 327 p. (in Russian)

[8]. Лазарев А.А. Теория расписаний. Методы и алгоритмы. М.: ИПУ РАН, 2019. 408 с. / Lazarev A. Scheduling theory. Methods and algorithms. Moscow, ICS RAS, 2019, 408 p. (in Russian)

[9]. Павлов П.А. Организация однородных конкурирующих процессов при распределенной конвейерной обработке. Проблемы управления. 2010, №3. С. 66-75. / Pavlov P. Organization of homogeneous competing processes in distributed pipeline processing. Control Sciences. 2010, №3. pp. 66-75. (in Russian)

[10]. Павлов П.А. Оптимальность структурирования программных ресурсов при конвейерной распределенной обработке. Программные продукты и системы. 2010, №3. С. 79-85. / Pavlov P. Optimal structuring of software resources during pipeline distributed processing. Software products and systems. 2010, №3. pp. 79-85. (in Russian)

[11]. Павлов П.А. Задача оптимизации числа процессоров при распределенной обработке. Вестник государственного Самарского аэрокосмического университета имени академика С.П. Королева. 2011, №4. С. 230-240. / Pavlov P. The problem of optimizing the number of processors in distributed processing. Bulletin of the State Samara Aerospace University named after Academician S.P. Koroleva. 2011, №4. pp. 230-240. (in Russian)
[12]. Pavlov P.A. The optimality of software resources structuring through the pipeline distributed processing of competitive cooperative processes. International Journal of Multimedia Technology (IJMT). 2012, Vol.2, №1. pp. 5–10.
[13]. Kovalenko N.S., Pavlov P.A. Optimal Grouping Algorithm of Identically Distributed Systems. Programming and Computer Software. 2012, №3. pp. 143-150.
[14]. Павлов П.А. Время реализации асинхронных параллельных процессов при макроконвейерной сосредоточенной обработке. Проблемы информатики. 2014, №3. С. 37–52. / Pavlov P. Implementation time of asynchronous parallel processes in macro-pipeline concentrated processing. Problems of computer science. 2014, №3. pp. 37-52. (in Russian)
[15]. Zaiets N., Shtepa V., Pavlov P., Elperin I., Hachkovska M. Development of a resource-process approach to increasing the efficiency of electrical equipment for food production. Eastern-European Journal of Enterprise Nechnologies. 2019, №8 (101). pp. 59-65.
[16]. Павлов П.А., Коваленко Н.С. Распределенные вычисления при ограниченном числе копий программного ресурса. Программные продукты и системы. 2011, №4. С. 155-163. / Pavlov P., Kovalenko N. Distributed computing with a limited number of copies of a software resource. Software products and systems. 2011, №4. pp. 155-163. (in Russian)
[17]. Kovalenko N.S., Pavlov P.A., Ovseec M.I. Asynchronous distributed computations with a limited number of copies of a structured program resource. Cybernetics and systems analysis. 2012, №1. pp. 86-98.
[18]. Pavlov P.A. Resource-process model of distributed computing with a limited number of software resource copies. Challenger and problems of modern science: proceedings of the IX international scientific conference. London, Great Britain. 2023, pp. 13-21.
[19]. Pavlov P.A. Asynchronous mode of distributed computing with a limited number of copies of a program resource. Theoretical and practical perspectives of modern science: proceedings of the IV international scientific and practical conference, Stockholm, Sweden. 2023, pp. 10-20.
[20]. Павлов П.А., Коваленко Н.С. Синхронный режим распределенных вычислений при непрерывном выполнении блоков ограниченного числа копий программного ресурса. Программные продукты и системы. 2024, №1. С. 43-53. / Pavlov P., Kovalenko N. Synchronous mode of distributed computing with continuous execution of blocks of a limited number of copies of a software resource. Software products and systems. 2024, №1. pp. 43-53. (in Russian)
[21]. Павлов П.А., Коваленко Н.С. Синхронный режим распределенных вычислений при непрерывном выполнении блоков ограниченного числа копий программного ресурса. Программные продукты и системы. 2024, №1. С. 43-53. / Pavlov P., Kovalenko N. Synchronous mode of distributed computing with continuous execution of blocks of a limited number of copies of a software resource. Software products and systems. 2024, №1. pp. 43-53. (in Russian)

Информация об авторах / Information about authors

Павел Александрович ПАВЛОВ – кандидат физико-математических наук, доцент, доцент кафедры информационных технологий и интеллектуальных систем Полесского государственного университета. Сфера научных интересов: математическое моделирование распределенных вычислительных систем конкурирующих процессов, исследование операций, математическое программирование.

Pavel Alexsandrovich PAVLOV – PhD, Associate Professor of the department of information technologies and intelligent systems (Polessky state university). Research interests: mathematical modeling of distributed computing systems of competing processes, operations research, mathematical programming.

DOI: 10.15514/ISPRAS-2024-36(4)-8

Intelligent Algorithms for Detecting Attacks in the Web Environment

[1] *M.A. Lapina, ORCID: 0000-0001-8117-9142 <mlapina@ncfu.ru>*
[1] *V.V. Movzalevskaya, ORCID: 0009-0007-7540-3110 <vitaliya1306@gmail.com>*
[1] *M.E. Tokmakova, ORCID: 0009-0000-2608-7712 <marinatokmakova175@mail.ru>*
[1] *M.G. Babenko, ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>*
[2] *V.P. Kochin, ORCID: 0000-0001-7782-9536 <kochyn@bsu.by>*

[1] *North Caucasus Federal University,*
1, Pushkina st., Stavropol, 355017, Russia.
[2] *Belarusian State University,*
4, Nezavisimosti ave., Minsk, 220030, Belarus.

Abstract. The article is devoted to the analysis of the use of machine learning algorithms to detect attacks using a custom web environment or the functionality of user applications. Learning with a teacher and clustering algorithms are considered. The dataset uses a sample of online shopping transactions collected by an e-commerce retailer. The dataset contains 39,221 transactions. To detect attacks in the web environment, the most optimal implementations of machine learning algorithms were selected after their review and comparative analysis. The most effective algorithm for detecting fraudulent transactions has been determined. We use the accuracy and running time of the algorithm as criteria. The accuracy of detecting fraudulent transactions for Random Forest, GB (Scikit-learn), GB (CatBoost) algorithms is 100%, and the KD-trees algorithm is 99,9%. The gradient boosting algorithm in the CatBoos implementation is 4,2 times faster than Random Forest, 2,4 times faster than GB Scikit-learn, 1,2 times faster than GB without using the cat_features parameter, 41,9 times faster than k-dimensional trees, 66,8 times faster than DBSCAN. The data obtained for each method is presented in the form of tables. Within the framework of this work, the parameters for evaluating the effectiveness of the algorithms under study are learning time indicators, as well as characteristics from the Confusion matrix and Classification Report for classification algorithms, and fowlkes_mallows_score, rand_score, adjusted_rand_score, Homogeneity, Completeness, V-measure for clustering algorithms.

Keywords: machine learning, web environment, consumer websites, cybersecurity, classification algorithms with a teacher, clustering.

For citation: Lapina M.A., Movzalevskaya V.V., Tokmakova M.E., Babenko M.G., Kochin V.P. Intelligent algorithms for detecting attacks in the web environment. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 4, 2024. pp. 99-116. DOI: 10.15514/ISPRAS–2024–36(4)–8.

Интеллектуальные алгоритмы обнаружения атак в веб-среде

[1] *М.А. Лапина, ORCID: 0000-0001-8117-9142 <mlapina@ncfu.ru>*
[1] *В.В. Мовзалевская, ORCID: 0009-0007-7540-3110 <vitaliya1306@gmail.com>*
[1] *М.Е. Токмакова, ORCID: 0009-0000-2608-7712 <marinatokmakova175@mail.ru>*
[1] *М.Г. Бабенко, ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>*
[2] *В.П. Кочин, ORCID: 0000-0001-7782-9536 <kochyn@bsu.by>*

[1] *Северо-Кавказский федеральный университет,*
355017, Россия, г. Ставрополь, ул. Пушкина, д. 1.
[2] *Белорусский государственный университет,*
220030, Белоруссия, г. Минск, пр-к. Независимости, д. 4

Аннотация. Статья посвящена анализу использования алгоритмов машинного обучения для обнаружения атак с использованием пользовательской веб-среды или функциональности пользовательских приложений. Рассматриваются алгоритмы обучения с преподавателем и кластеризации. В наборе данных используется выборка транзакций онлайн-покупок, собранная розничным продавцом электронной коммерции. Набор данных содержит 39 221 транзакцию. Для обнаружения атак в веб-среде были выбраны наиболее оптимальные реализации алгоритмов машинного обучения после их обзора и сравнительного анализа. Был определен и реализован наиболее эффективный по времени и качеству алгоритм для рассматриваемой выборки данных. Данные, полученные по каждому методу, представлены в виде таблиц. В рамках данной работы параметрами для оценки эффективности исследуемых алгоритмов являются показатели времени обучения, а также характеристики из матрицы путаницы и отчета о классификации для алгоритмов классификации, а также fowlkes_mallows_score, rand_score, adjusted_rand_score, Однородность, полнота, V-мера для алгоритмов кластеризации.

Ключевые слова: машинное обучение, веб-среда, потребительские веб-сайты, кибербезопасность, алгоритмы классификации с преподавателем, кластеризация.

Для цитирования: Лапина М.А., Мовзалевская В.В., Токмакова М.Е., Бабенко М.Г., Кочин В.П. Интеллектуальные алгоритмы обнаружения атак в веб-среде. Труды ИСП РАН том 36, вып. 4, 2024 г., стр. 99–116 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(4)–8.

1. Introduction

The urgent need to ensure information security on the Internet is easily explained by several factors, including the massive unification of heterogeneous and distributed systems, the presence of large amounts of confidential information in end systems maintained by corporations and government agencies, the easy distribution of automated malicious software by attackers, and the ease with which computer crimes can be committed anonymously.

With the development of web technologies and a significant increase in the volume of information, this problem is only getting worse. Attackers often exploit vulnerabilities in the system or web applications to upload a malicious file or malicious code to a web server. This is often used as a so-called backdoor for working with and managing a web server, because such a file can provide attackers with remote access to the server management interface, including executing commands, manipulating files and connecting to a database. Therefore, an accurate determination of whether files stored on a web server are malicious is of great importance for the security of the web server.

With the constant operation of web browsers, the security and privacy of users may be at risk, because browser vulnerabilities can lead to unprotected use [1]. Important user data, such as login, can be collected and used for profiling, which raises serious privacy concerns.

In this regard, the development of new, more advanced access control mechanisms and the optimization of existing ones is very relevant [1].

Machine learning algorithms allow you to detect and prevent attacks, which significantly increases the effectiveness of site protection [2].

At its core, machine learning can be represented as the process of inferring algorithms for predicting unknown data using previously collected information.

As part of the study of the effectiveness of several of the most widely used machine learning models, several algorithms have been implemented and analyzed, classification and clustering methods have been considered. In this regard, it is necessary to define a training sample. The article uses a sample from the work [2]. The paper does not provide a complete description of all the algorithms, as such information is too extensive. A detailed description of each method can be found in the book [3].

The article presents the following sections: section 1 – introduction; section 2 – relevance; section 3 – theoretical description of models; section 4 – modeling; section 5 – analysis of the results; section 6 – conclusions.

2. Relevance

The security issues of the web environment and user applications are very relevant, since web applications are accessible over the network and very often contain vulnerabilities, which is why they regularly become targets of cyber-attacks [4]. In today's digital environment, the protection of web space plays an important role in ensuring the integrity of user data and confidentiality, as well as in the safe and continuous use of web applications. Unfortunately, the rapid spread of digital technologies is always accompanied by the same rapid spread of threats and attacks in the web environment, which constantly requires the creation of new security methods, as well as the improvement of existing ones [5].

To date, there are no universal means of protection, much less those that could identify new types of threats in the web environment. Machine learning algorithms can be used to prevent attacks. They allow you to automate the process of detecting and preventing attacks.

The Bayesian networks. A Bayesian network is a graphical model that encodes probabilistic relationships between variables of interest [6, 7]. These networks are a combination of two different mathematical fields: graph theory and probability theory [8].

The Bayesian network is used to study cause-and-effect relationships and, therefore, to gain an understanding of the problem area and predict the consequences of intervention. Because the model has both causal and probabilistic semantics, it is ideally suited to represent a combination of prior knowledge and data.

Artificial neural network. The neural network includes many neurons used for processing and analyzing information, its structure is like the nervous system of a living organism [9, 10]. Neural networks consist of basic blocks like neurons. These blocks interact with each other based on connections, the strength of which can be changed because of the learning process or modification of the algorithm [11].

The use of neural networks demonstrates some of the best results in classification problems with limited input parameters, in such tasks, this method is much more effective than other machine learning methods [12].

The support vector machine. This method refers to learning algorithms with a teacher and is often used in solving problems related to detecting attacks in a web environment.

The method of reference vectors is based on linear classification [13-15]. It is one of the classic machine learning methods that can help solve big data classification problems. This method is especially effective in multi-domain applications in a big data environment [16]. However, the support vector machine is mathematically complex and computationally expensive.

Common attacks in the web environment. In this part of the study, the five most common attacks committed in 2020-2021 are presented and described [4]. The most common attacks include the following: attacks on clients (98%), data leakage (91%) and unauthorized access to the application (84%).

If we talk about the distribution of attacks by industry, the most popular for 2022 are: the public sector of the economy (30%), financial technology (23%), education (16%) [5] (Fig. 1).

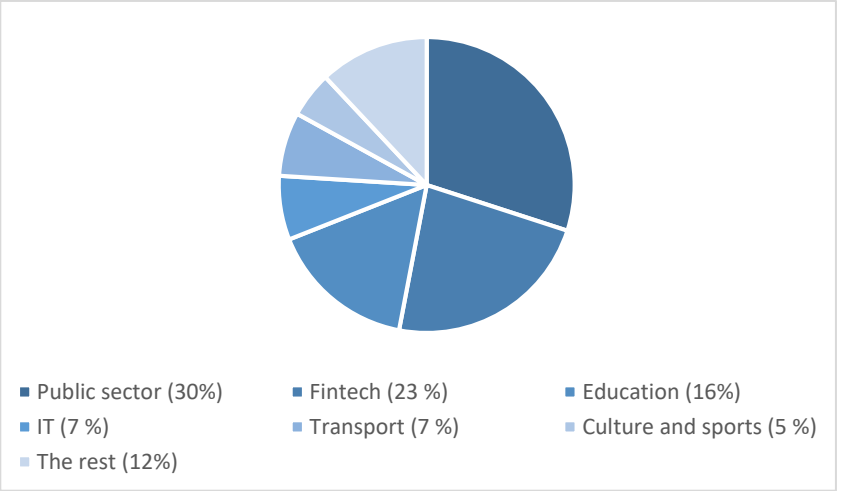


Fig. 1. Distribution of web attacks by industry.

Attacks on customers. These attacks are the most popular in the statistics given, they exploit weaknesses in applications running on a computer controlled directly by a user, often referred to as a client [17]. In 84% of the studied applications, threats of unauthorized access to the personal accounts of users, including administrators, were identified. In 72% of web applications, an attacker can gain access to functionality or content that should not be available to him, for example, to view the personal accounts of other users.

Attackers can harm users or compromise them using information and data extracted in various ways, for example, from cookies.

The most deplorable consequences of the actions of intruders include the disclosure of important confidential information, obtaining unauthorized access to application codes and local network resources, which leads to the spread of malicious actions on the infrastructure.

Data leak. Data leaks are the second most pressing security threat in web application research. The results of the security analysis showed that more than three quarters of web applications were exposed to the disclosure of user IDs. Personal data was disclosed in 60% of applications, and user credentials in 47%, which is 13 and 16 percent more, respectively, than in 2019. Personal and credentials are desirable targets for intruders, which is confirmed by the data of the final analysis of current cyber threats in 2021 [18].

A data leak is the intentional or unintentional disclosure of confidential information to unauthorized persons. Data leakage poses a serious threat to organizations, including significant reputational damage and financial losses.

As the volume of data grows exponentially and data leaks occur more frequently than ever before, data loss detection and prevention has become one of the most pressing security concerns for enterprises. Despite a lot of research on protecting confidential information from leakage, this remains an urgent research problem.

Unauthorized access to the application. Unauthorized access to the application (UAA) was detected in 84% of web applications. This is an attack in which an attacker gains access to an application without the permission or consent of the owner. This can happen in a variety of ways, including password hacking, exploiting security vulnerabilities, and authentication.

Unauthorized access to the application may result in viewing, changing, or deleting confidential information, disrupting the operation of the application, and gaining full control over the system.

To prevent UAA, it is necessary to take measures to protect applications and systems, such as regular software updates, the use of complex passwords and two-factor authentication, as well as monitoring user activity and intrusion detection.

Denial of service. Distributed Denial of Service (DDoS) attacks consist of streams of packets from various sources. These streams consume some key resource, making it inaccessible to legitimate users.

The interaction of distributed machines generating attack streams makes tracking and mitigation a very difficult task. Some protection mechanisms focus on detecting an attack near a computer, characterizing it, and filtering attack packets. Although the detection accuracy of these mechanisms is high, traffic is usually so aggregated that it is difficult to distinguish legitimate packets from attack packets. More importantly, the volume of the attack may be more than the system can withstand.

DDoS attacks can be used to conceal other network attacks. When a website is under attack, an internal and external group of information security specialists usually focuses on closing the ports of these sites, clearing traffic, and resuming its operation.

Implementation of Operating system commands. OS command injection is a vulnerability in websites that allows an attacker to inject malicious code or commands into the system through the user interface or using scripts that are processed by the operating system. This entails obtaining confidential data by an attacker, as well as spreading attacks to other systems.

3. Theoretical description of the models

This section provides the mathematical construction of algorithms, description, and analysis of their implementations, as well as justification for choosing the most optimal one for the data sample used.

3.1 Random Forest

Random forests are formed as simple ensembles of several decision trees, usually containing tens to thousands of such trees.

After training each individual decision tree, general predictions of random forests are made by choosing the statistical mode of forecasts of individual trees for classification trees (that is, each tree "votes") and the average statistical value of forecasts of individual trees for regression trees. However, the increased complexity of random forests makes it difficult to analyze predictions compared to single decision trees. When building a random forest tree, the following happens:

- A subset of training objects is randomly selected from the entire dataset. This subset may contain duplicate objects.
- A subset of features is randomly selected (usually the square root of the total number of features). This reduces the correlation between the trees in the ensemble and improves their diversity.
- A decision tree is built based on the selected subset of data and features. When building a tree, it is necessary to use an information criterion (for example, the entropy criterion), which allows you to choose the best feature for dividing data at each available level of the tree.
- Repeat steps 1-3 for each tree in the ensemble.

In general, the algorithm for constructing a random forest consisting of N trees can be represented as follows:

For each n = 1, ..., N do

- Generate a selection of X_n bootstraps.
- Build a decision tree b_n from a sample of X_n.

To solve the classification problem, a majority vote is used, and for the regression problem, an average one is used. The final classifier looks like this:

$$a(x) = \frac{1}{N}\sum_{i=1}^{N} b_i(x), \#(1)$$

in classification tasks, it is recommended to take:

$$m = \sqrt{n}, \#(2)$$

and in regression tasks:

$$m = \frac{n}{3}, \#(3)$$

where n is the number of features.

3.2 Gradient boosting

Gradient boosting (GB) is an optimization algorithm used to minimize errors in a machine learning model.

This method is based on a vector called a gradient, which represents the largest increase in the function in its direction, its coordinates are partial derivatives of the function [19]. In other words, if the function f(x, y, z) is given, then the gradient is calculated using the formula:

$$gradf = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right). \#(4)$$

The idea of the gradient approach is to iteratively adjust the model parameters in the direction of the negative gradient of the loss function (which represents the error) to reduce the error and find the optimal parameters that give the best prediction results. Therefore, this method can be used to obtain the smallest error value or to find weights when training neural networks. Weights are values that indicate important information when training neural networks using a «teacher» [20].

Because at each iteration of the algorithm, the model parameters are updated in the direction opposite to the gradient of the loss function. The size of the step that the algorithm takes in this direction is determined by the learning rate. The key parameter is the optimal learning rate of the model, this is since too large a step can lead to skipping the minimum, and too small can affect the optimization process, slowing it down.

The formula for updating the parameter θ at each iteration is as follows:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta), \#(5)$$

where η is the learning rate, $\nabla_\theta J(\theta)$ is the gradient of the loss function J.

There are three main implementations of GB: LightGBM, XGBoost and CatBoost, below are the main characteristics of these implementations (Table 1).

Two implementations were chosen for the study: CatBoost and Scikit-learn. The implementation of CatBoost, because it provides high performance and prevents overfitting, it is very simple, but at the same time it is not inferior in efficiency to LightGBM and XGBoost (Table 1), also this implementation is stated as the fastest and optimized [21], therefore, the study compares it with Scikit-learn, the most frequently implemented.

3.3 k-dimensional trees

A k-dimensional tree (KD-tree) is a binary tree that stores data in a format optimized for multidimensional spatial analysis. The formation of a KD-tree can be considered as a preliminary stage of classification algorithms using the k-nearest neighbor (k-NN) method, but this technique can also be considered an independent clustering algorithm. The algorithm creates a tree structure, and clusters are stored in leaves.

A typical algorithm for the formation of KD-trees is given below. For each node that differs from the sheet, the following actions are performed:

- Choosing the dimension for separation.
- Select the separation point.
- Division of the subspace according to the selected dimension and the separation point.
- Termination of subspace separation when the current subspace contains fewer elements than a certain number of sample elements for a separate subspace.

Table 1. Results for different diode insertion strategies.

Parameter	LightGBM	XGBoost	CatBoost	Scikit-learn
The main purpose	Optimized for working with categorical data	Focused on efficiency and productivity	It is focused on speed when working with large amounts of data	A simple and effective way to implement (classical implementation)
Categorical data	Built-in processing without pre-coding	Pre-coding is required	It has optimizations for categorical features	Pre-coding is required
Learning rate	High, with GPU support	High, with GPU support	Very high	High
Preventing over-training	Uses a variety of regularization strategies	Supports L1 and L2 regularization	Uses mechanisms such as EFB	Uses different methods to prevent overfitting
Big Data	Optimized to work efficiently with large datasets	It may be ineffective on very large datasets	Optimized to work with large amounts of data with low memory requirements	It can handle large amounts of data, but there are limitations related to the amount of memory
Programming languages	Support for major languages, including Python, R, Java	Extensive language support, including Python, R, Java, Scala	Supports Python, R, Java and other languages	Supports Python
The complexity of the models	Generates more complex models with retraining control	Allows you to adjust the complexity of the model through hyperparameters	Builds lightweight models with a histogram approach	Uses regularization to reduce complexity
Interpretability	Provides good interpretability	Provides an average level of interpretability	Interpretability can be difficult due to optimizations	Provides good interpretability

The result of this procedure is a binary search tree in feature subspaces, while combining all subspaces of leaf nodes forms a complete feature space. When creating KD-tree models to search for nearest neighbors, a binary tree with a split space must be saved as an addition to the training data points. Moreover, additional data on which sample items belong to specific leaf nodes should also be stored in the model. Thus, even more space is required to store such a model, which makes it less economical (in terms of memory resource consumption) than the original k-NN models.

In an inhomogeneous KD-tree:

$$H_i(t) = (x_1, x_2, \ldots, x_{i-1}, t, x_{i+1}, \ldots, x_k), \#(6)$$

for $1 \le i \le k$, parallel to the axis *(k-1)* of the dimensional hyperplane at point *t*. For the root, you need to divide the points through the hyperplane $H_1(t)$ into two, if possible, equally large sets of points and write *t* to the root, to the left of this, all points with $x_1 < t$ are saved, on the right are those with $x_1 > t$.

For the left subtree, you need to divide the points again into a new «split plane» $H_2(t)$, and *t* is stored in the inner node. To the left of this, all points with $x_2 < t$ are saved. This continues recursively over all spaces. Then everything starts again from the first space until each point can be clearly identified through the hyperplane.

KD-trees are usually not suitable for high-dimensional data. For feature spaces with high dimensionality, the efficiency of KD-trees is comparable to the efficiency of linear search by simple iteration. Nevertheless, KD-trees are very convenient for quickly searching for nearest neighbors with an average time complexity of *O(log n).*

3.4 DBSCAN

DBSCAN (DensityBased Spatial Clustering of Applications with Noise) is a density–based spatial clustering algorithm for applications with noise. It increases clusters according to density-based connectivity analysis. Density-based clustering algorithms are used to detect clusters in datasets of arbitrary shape and large size. These algorithms are usually grouped as dense regions of points in the data space, separated by low-density regions.

The key idea of DBSCAN is that for each cluster object of a neighborhood of a given radius ε, there must be at least a minimum number of Pts_{min} objects, which means that the power of the neighborhood must exceed a certain threshold. The neighborhood of an arbitrary point p is defined by:

$$N_\varepsilon = q \in D/dist(p,q) < \varepsilon, \#(7)$$

where D is the database of objects. If the neighborhood of a point P contains at least the minimum number of points, then this point is called the main point. The main point is defined as:

$$N_\varepsilon(P) > Pts_{min}, \#(8)$$

where ε and Pts_{min} are user-defined parameters, which mean the radius of the neighborhood and the minimum number of points in the neighborhood of the base point, respectively.

4. Modeling

This section describes the implementation, training, and testing of the machine learning algorithms described above.

The dataset uses a sample of online shopping transactions collected by an e-commerce retailer. The dataset contains 39,221 transactions, each of which contains 5 properties or characteristics that can be used to describe the transaction: accountAgeDays (age of the account from which the payment was made), numItems (number of purchased items), localTime (local time of purchase), PaymentMethod (payment method), paymentMethodAgeDays (number of days since the addition of this payment method), as well as a binary label that determines whether the transaction is fraudulent: the label «1» indicates a fraudulent transaction, The label «0» is assigned to the legal transaction (label column, Table 2). The dataset is taken from [2]. The task of the trained model is to accurately determine by the properties of the transaction whether it is fraudulent or not.

Below are 3 random rows from the selected dataset (Table 2).

Table 2. Model training time.

line number	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays	lable
37813	64	1	4,748314	creditcard	0,0	0
15585	2000	1	4,524580	creditcard	0,0	0
24030	653	1	4,748314	creditcard	0,0	0

The values in the columns accountAgeDays, numItems, localTime, payment Method Age Days are integers (accountAgeDays, numItems) and non-integers (localTime, paymentMethodAgeDays). The PaymentMethod column contains the payment method as a string, which can take the following values: «creditcard», «paypal», «storecredit».

In machine learning, there are quite a few different metrics that allow you to determine the accuracy and efficiency of a trained model. Within the framework of this study, the parameters for evaluating

the effectiveness of the algorithms under study are learning time indicators (Section 5), as well as characteristics from the Confusion matrix and Classification Report for classification algorithms.

Support is the number of actual occurrences of the class in the dataset.

4.1 Random Forest

First, you need to import the Random Forest Classifier and create a model. The main input parameters of the classifier are the number of trees in the forest (n_estimators), the maximum depth (max_depth) and the number of functions that should be considered when searching for the best separation (max_features). By default, these characteristics are 100, None, and sqrt (n_features), respectively, where n_features is the number of functions in the data.

The results of the model trained using the random forest algorithm are presented in Table 8 (for the methods of random forest, GB (Scikit-learn), GB (CatBoost), a general table is given, since their results turned out to be identical).

The algorithm identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 legal ones. There are no objects identified as falsely negative or falsely positive. The precision, recall and F1-Score values for each class are close to their best value.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). The average operating time of the model is 0,08 sec.

4.2 Gradient boosting

One of the implementations of the GB algorithm, which is considered in the study, is the implementation of Scikit-learn. One of the undoubted advantages of the chosen implementation is ease of use and accessibility [22].

First, you need to import the GradientboostingClassifier and create a model. The main input parameters of the classifier are the number of boosting stages to perform (n_estimators), maximum depth (max_depth). GB is usually resistant to overfitting, therefore, with large values of the n_estimators parameter, the algorithm shows better results. By default, these characteristics are 100 and 3, respectively.

The results of the model trained using the GB algorithm in the implementation of Scikit-learn are presented in Table 2 (for the methods of random forest, GB (Scikit-learn), GB (CatBoost), a general table is given, since their results turned out to be identical).

The algorithm identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 legal ones. There are no objects identified as falsely negative or falsely positive. The precision, recall and F1-Score values for each class are close to their best value.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). The average operating time of the model is 0.046629 sec.

CatBoost is another implementation of the GB algorithm. The main parameters used by the algorithm in model training and prediction are the index of the first used tree (ntree_start), the index of the first unused tree (ntree_end) and the list of categorical features (cat_features).

The results of the model trained using the GB algorithm for the implementation of CatBoost are presented in Table 8 (for the methods of random forest, GB (Scikit-learn), GB (CatBoost), a general table is given, since their results turned out to be identical).

The algorithm identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 legal ones. There are no objects identified as falsely negative or falsely positive. The precision, recall and F1-Score values for each class are close to their best value.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). The average operating time of the model is 0,024148 sec.

By default, categorical features are not considered in the process of training a model using CatBoost, however, one of the key features of this implementation is precisely the ability to train a model without first preparing categorical data.

Below is the importance of each feature in the process of training the model without preliminary data preparation (Table 3) and with it, i.e. using the cat_features parameter (Table 4).

Table 3. The importance of features in training a model without the cat_features parameter.

line number	Feature Id	Importances
0	accountAgeDays	78,826096
1	localTime	6,487206
2	numItems	4,092288
3	paymentMethodAgeDays	3,228341
4	paymentMethoz_storecredit	3,123910
5	paymentMethod_creditcard	2,912722
6	paymentMethod_paypai	1,329436

Table 4. The importance of features in training a model with the cat_features parameter.

line number	Feature Id	Importances
0	accountAgeDays	84,247069
1	paymentMethodAgeDays	8,237712
2	localTime	3,882424
3	numItems	3,632795
4	paymentMethod	0,000000

According to the data from the tables above, it can be seen that the importance of features in the learning process for models with the cat_features parameter and without this parameter is quite different.

The results of the model trained using the GB algorithm when implementing CatBoost using the cat_features parameter is presented in Table 8 (for the methods of random forest, GB (Scikit-learn), GB (CatBoost), a general table is given, since their results turned out to be identical).

The algorithm identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 legal ones. There are no objects identified as falsely negative or falsely positive. The precision, recall and F1-Score values for each class are close to their best value.

The accuracy of the algorithm with the cat_features parameter has not changed compared to the algorithm without this parameter and remains the same high.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). the operating time of the model has improved slightly compared to the model with preprocessing of categorical features, the only exception is the «best operating time», which has deteriorated by several thousandths of a second.

The results of the model trained using the GB algorithm are presented in Table 5.

Table 5. Classification report for random forest, GB (Scikit-learn), GB (CatBoost) methods.

Parameter	correct	error	accuracy	macro avg	weighted avg
Precision	1	1	–	1	1
Recall	1	1	–	1	1
F1-Score	1	1	1	1	1
Support	12753	190	12943	12943	12943

As can be seen from the above data, the algorithm correctly identified all 190 fraudulent transactions and all 12753 legal transactions. There are no falsely negative and falsely positive objects. The precision, recall and F1-Score values for each class show their best value.

4.3 k-dimensional

The KD-trees algorithm is most often used as part of the (preliminary stage) k-NN algorithm. The study implemented a classifier based on the k-NN algorithm using KD-trees.

The main parameters of the algorithm are the number of points at which the transition to iteration occurs (leaf_size), the metric for calculating the distance (metric), as well as the number of neighbors that the algorithm considers during operation (n_neighbors), and the algorithm by which the k-NN are searched (algorithm).

By default, leaf_size = 40, metric = «minkowski», n_neighbors = 5, algorithm = «auto». The algorithm parameter will take the value «kd_tree» because it is the KD-trees that will be the basis of the algorithm.

The results of the model trained using the k-NN algorithm based on KD-trees are presented in Table 6.

Table 6. Classification Report for the k-NN method based on KD-trees.

Parameter	correct	error	accuracy	macro avg	weighted avg
Precision	1	1	–	1	1
Recall	1	0,99	–	1	1
F1-Score	1	1	1	1	1
Support	12753	190	12943	12943	12943

As can be seen from the above data, the algorithm correctly identified 189 out of 190 fraudulent transactions and all 12753 legal transactions. One object is identified as falsely negative, there are no false positive objects. The precision, recall and F1-Score values for each class are close to their best value.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). The average operating time of the model is 0,798228 seconds.

4.4 DBSCAN

The DBSCAN algorithm treats clusters as high-density areas in divided low-density areas. Because of this rather general view, the detected clusters can have any shape. The central component of DBSCAN is the concept of core samples, i.e., samples located in high density areas. Thus, a cluster is a set of core samples, each of which is close to each other, and a set of non-core samples that are close to the core sample but are not core samples themselves. This algorithm has two input parameters that should be selected based on the dataset: \epsilon – the maximum distance between two samples so that they are considered neighbors; min_samples – the number of samples in the vicinity of the point so that it is considered the base/central. These two parameters formally define «density». Higher min_samples or lower \epsilon indicate a higher density required to form a cluster. During the study, various values of the above parameters were sorted out.

A cluster is a set of core samples that can be constructed by recursively taking a core sample, searching for all its neighbors that are core samples, searching for all their neighbors that are core samples, etc. The cluster also has a set of non-core samples that are neighbors of the main sample in the cluster, but are not the main samples. Intuitively, these samples are located on the periphery of the cluster.

To assess the accuracy and efficiency of the model, the indicators of operating time, FMI, Homogeneity, number of clusters, degree of noise (Noise points) are used.

The operating time of the DBSCAN algorithm is shown below (Fig. 2). We can see that as the eps parameter increases, the running time increases, while the min_samples parameter does not significantly affect the running time of the algorithm.

The values of the FMI parameter are shown below (Fig. 3). We can see that when the eps parameter is increased, the FMI values increase. The min_samples parameter has no significant effect.

Below are the indicators of the value of the homogeneity parameter (Fig. 4). We can see that fraudulent transactions clearly fall into a separate class only with small eps values. The min_samples parameter has no significant effect.

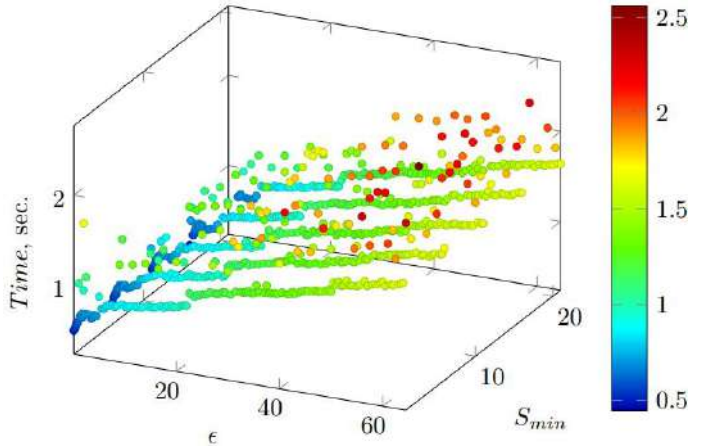


Fig. 2. The operating time of the DBSCAN algorithm.

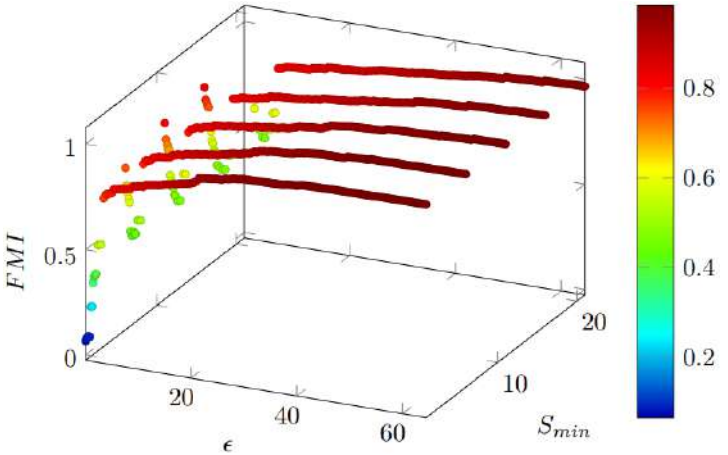


Fig. 3. FMI for the DBSCAN algorithm.

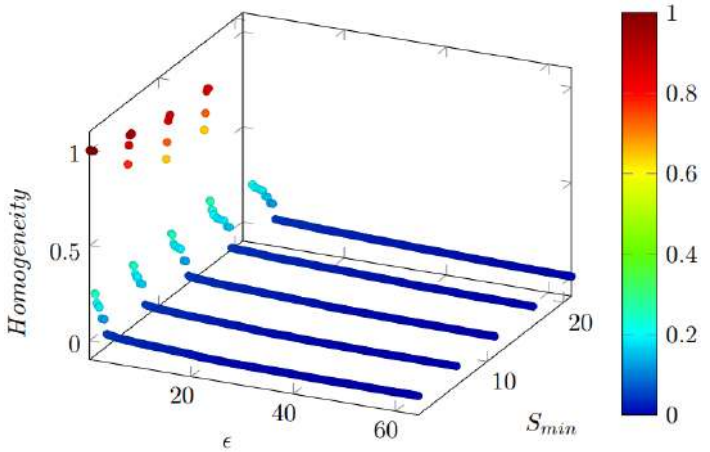


Fig. 4. Homogeneity for the DBSCAN algorithm.

The clusters of the DBSCAN algorithm are shown below (Fig. 5). We can see that with small values of min_samples, the number of clusters can vary significantly depending on eps. However, with an increase in the min_samples parameter, the number of clusters fluctuates significantly less.

The noise indicators are shown below (Fig. 6). We can see that with an increase in the eps parameter, the number of noise points decreases significantly. However, with small eps values, almost all points are identified as noise.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). The average running time of the model is 1,2715 sec., which is an order of magnitude slower than classification algorithms, for comparison, the running time of the slowest classification algorithm (KD-trees) is 0,798228 sec.

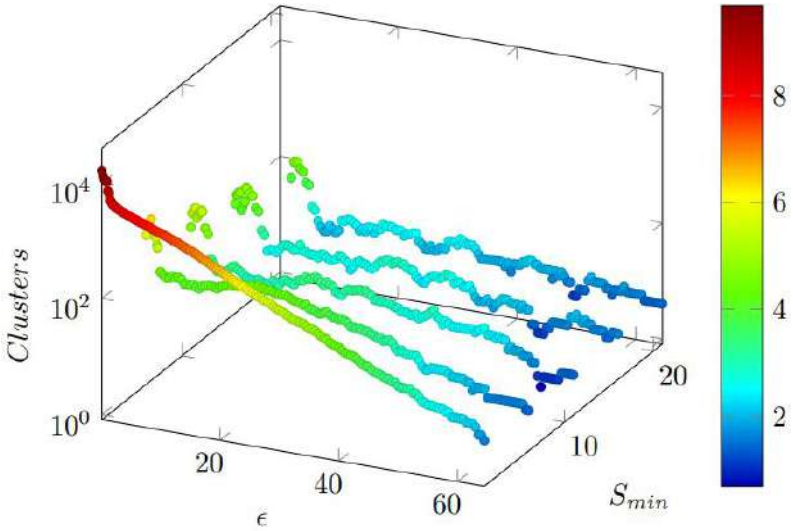


Fig. 5. Clusters of the DBSCAN algorithm.

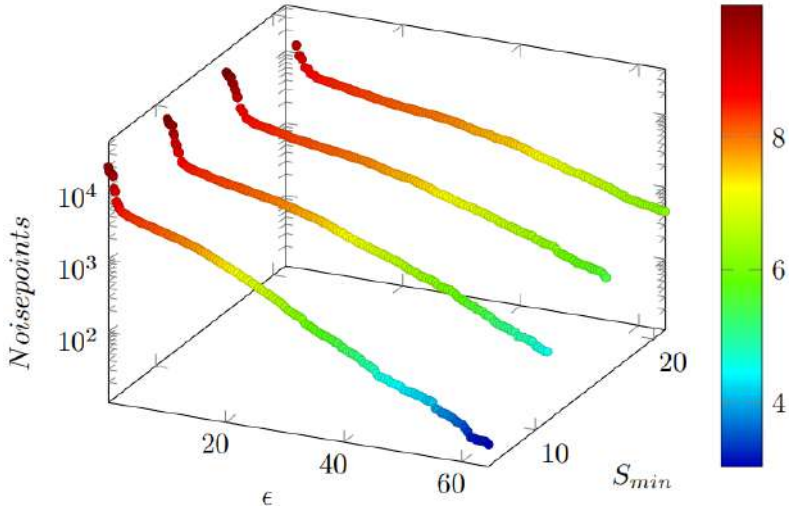


Fig. 6. Noise indicators for the DBSCAN algorithm.

5. Analysis of the results obtained

A comparative analysis of machine learning algorithms is carried out in relation to a sample of transaction data for online purchases. The study demonstrated a different degree of effectiveness of the models and the speed of their work in solving a specific task of searching for fraudulent monetary transactions. The following are the training time indicators for each model (Table 7).

Table 7. The working time of the algorithms, sec.

Method	Average, sec	T_{max}, sec	T_{min}, sec	σ
Random Forest	0,08	0,091674	0,074065	0,003564
GB Scikit-learn	0,046629	0,068366	0,043099	0,005155
GB CatBoost Without the use of cat_features	0,024148	0,052511	0,015452	0,009639
GB CatBoost With the use of cat_features	0,019016	0,035456	0,014613	0,002828
k-мерные деревья	0,798228	1,235431	0,628746	0,140186
DBSCAN	1,271589	4,436863	0,460014	0,445787

From the data presented in Table 7, we can draw the following conclusions:

- the average running time of the GB (cat_features) algorithm is 4.2 times faster than Random Forest, GB (Scikit-learn) is 2.4 times, GB (without cat_features) parameter is 1.2 times, KD-trees are 41.9 times, DBSCAN is 66.8 times.
- the maximum operating time of the GB (cat_features) algorithm is 2,5 times faster than Random Forest, 1,8 times faster than GB (Scikit-learn), 1,5 times faster than GB (without cat_features), 34,7 times faster than KD-trees, 125,1 times faster than DBSCAN.
- the minimum time algorithm GB (cat_features) is faster than Random Forest by 5,06 times, GB (Scikit-learn) by 2,9 times, GB (without cat_features) by 1,05 times, KD-trees by 43,02 times, DBSCAN by 31,5 times.
- the standard deviation algorithm GB (cat_features) works better than Random Forest by 1,3 times, GB (Scikit-learn) by 1,8 times, GB (without cat_features) by 3,4 times, KD-trees by 49,6 times, DBSCAN by 157,6 times.

For the convenience of comparing the operating time, a graph is presented below, which shows the average time of each algorithm (Fig. 7).

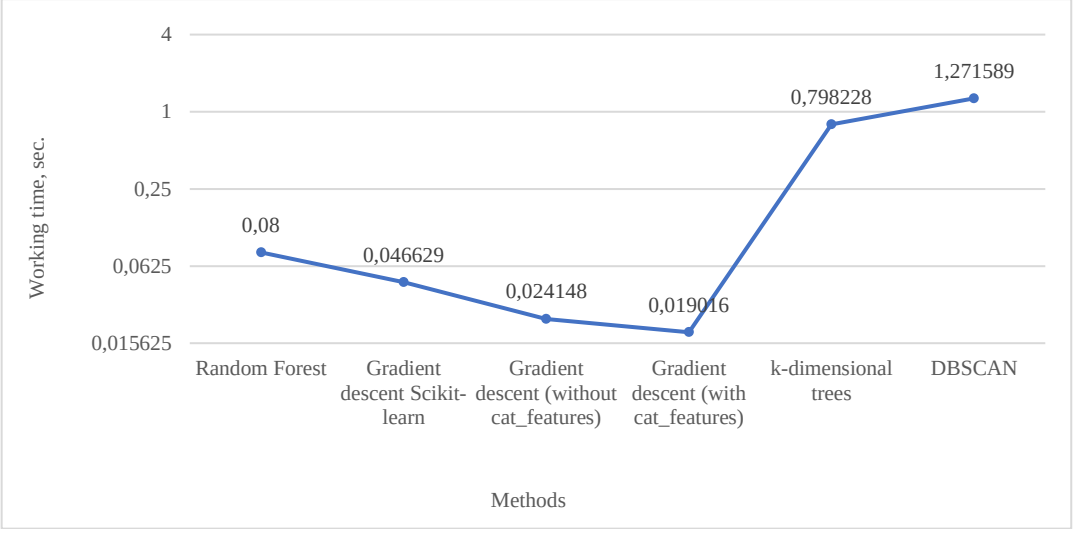


Fig. 7. Model training time.

Based on the indicators of the training time of the models, they can be divided into those that learn faster (random forest (0,08), decision trees with a GB (0,046629), GB (without cat_features)

(0,024148), GB (with cat_features) (0,019016)) and those that learn slower (KD-trees (0,798228), DBSCAN (1,271589)).

The indicators from the Confusion matrix for classification algorithms are also given (Table 8).

Table 8. Confusion matrix for classification algorithms.

Method	Accuracy	TN	FN	FP	TP
Random Forest	1	12753	0	0	190
GB Scikit-learn	1	12753	0	0	190
GB CatBoost Without the use of cat_features	1	12753	0	0	190
GB CatBoost With the use of cat_features	1	12753	0	0	190
k-мерные деревья	0,999	12753	1	0	189

As can be seen from the above data, almost all algorithms, not counting KD-trees, identified all classes without errors, that is, they identified 190 fraudulent transactions and 12753 cor legal rect ones. There are no objects identified as falsely negative or falsely positive. The KD-trees method correctly identified 189 out of 190 fraudulent transactions and all 12,753 legal transactions. One object is identified as falsely negative, there are no false positive objects.

6. Conclusions

The accuracy of detecting fraudulent transactions using Random Forest, GB (Scikit-learn), GB (CatBoost) algorithms is 100%, and the KD-trees algorithm is 99,9%. Thus, we used the transaction classification time as a comparison criterion.

Taking into account the minimum execution time, the GB method was chosen as the main model using the cat_features parameter. It is also worth noting that the implementation of the GB method without the cat_features parameter learns almost as quickly as with it, so it can also be considered within the processing used in the study of a data sample. The average running time of the algorithms is 0,019016 sec. and 0,024148 sec. respectively.

The GB algorithm using the cat_features parameter works 4,2 times faster than Random Forest, GB (Scikit-learn) 2,4 times, GB (without cat_features) 1,2 times, KD-trees 41,9 times, DBSCAN 66,8 times.

The main parameters used by the algorithm in model training and prediction are the index of the first used tree (ntree_start), the index of the first unused tree (ntree_end) and the list of categorical features (cat_features).

The operating time of these models turned out to be minimal, so it can be assumed that these are the most optimal models for processing the data sample under consideration.

References

[1]. T. Hastie, R. Tibshirani, J. Friedman, “The Elements of Statistical Learning”, p. 745, August 2009.
[2]. C. Chio, D. Freeman, “Machine Learning and Security”, p. 386, February 2017.
[3]. Power, R. “Tangled Web: Tales of Digital Crime from the Shadows of Cyberspace”, pp. 396-397, 2000.
[4]. Uyazvimosti i ugrozy veb-prilozhenij v 2020-2021 gg. Official website – URL: https://www.ptsecurity.com/ru-ru/research/analytics/web-vulnerabilities-2020-2021/#id5.
[5]. Andress, J., “The Basics of Information Security”, Second Edition, Chapter 3, Authorization and Access Control, p. 190, 2014.
[6]. R. Hamsa Veni, A. Hariprasad Reddy, C. Kesavulu, “Identifying Malicious Web Links and Their Attack Types in Social Networks”, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp.1060-1066, March-April 2018.
[7]. R. F. Fouladi, C. E. Kayatas, E. Anarim, “Frequency based DDoS attack detection approach using naive Bayes classification”, International Conference on Telecommunications and Signal Processing (TSP), June 2016.
[8]. Todd A. Stephenson, “An Introduction to Bayesian Network Theory and Usage”, p. 29, 2000.

[9]. D. Atienza, A. Herrero, E. Corchado, “Neural analysis of http traffic for web attack detection”, Computational Intelligence in Security for Information Systems Conference, pp.201-212, January 2015.
[10]. B. Goyal, M. Bansal, “Competent Approach for Type of Phishing Attack Detection Using Multi-Layer Neural Network”, International Journal of Advanced Engineering Research and Science, pp. 210-215, January 2017.
[11]. Hervé Abdi, “A neural network primer”, Journal of Biological Systems, pp. 247-281, 1994.
[12]. Sivak M. A., Timofeev V. S., “Configuring robust neural networks to solve the classification problem”, Reports of Tomsk State University of Control Systems and Radioelectronics, pp.26-32, 2021.
[13]. N. Florian Epp, R. Funk, C. R. Cappo, “Anomaly-based web application firewall using HTTP-specific features and one-class SVM”, September 2017.
[14]. Z. Tian, “Distributed Deep Learning System for Web Attack Detection on Edge Devices”, IEEE Transactions on Industrial Informatics, November 2019.
[15]. Ye Jin, “A DdoS attack detection method based on SVM in software defined network”, Security and Communication Networks, April 2018.
[16]. S. Suthaharan, “Support Vector Machine”, Machine Learning Models and Algorithms for Big Data Classification, pp. 207-235, January 2016.
[17]. Otchet ob atakah na onlajn-resursy rossijskih kompanij. official website [https://www.ptsecurity.com/ru-ru/] – URL: https://rt-solar.ru/upload/iblock/34a/5w4h9o57axovdbv3ng7givrz271ykir3/Ataki-na-onlayn_resursy-rossiyskikh-kompaniy-v-2022-godu.pdf?ysclid=lubdnvft2p622633541.
[18]. Aktual'nye kiberugrozy: itogi 2021 goda. official website [https://www.ptsecurity.com/ru-ru/] – URL: https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2021/.
[19]. Garfinkel, S. and Spafford, E.H., “Web Security and Commerce”, O'Reilly and Associates, pp. 450-470, February 1997.
[20]. Martynov A., Kandybla V., “Metod gradientnogo spuska v mashinnom obuchenii”, Zhurnal «Shag v nauku», pp. 4-8, 2022.
[21]. CatBoost is a high-performance open-source library for gradient boosting on decision trees. official website [https://catboost.ai/?ysclid=lwdjifxlqs185272725] – URL: https://catboost.ai/en/docs/concepts/speed-up-training?ysclid=lwdk46v95o460729700.
[22]. Gradient Boosting. official website [https://scikit-learn.org/stable/] – URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html.

Информация об авторах / Information about authors

Мария Анатольевна ЛАПИНА – кандидат физико-математических наук, доцент кафедры информационной безопасности автоматизированных систем Северо-Кавказского федерального университета. Сфера научных интересов: цифровые технологии, управление информационной безопасностью, процессный подход, образовательный процесс, криптография.

Maria Anatolyevna LAPINA – Cand. Sci. (Phys.-Math.), Associate Professor of the Department of Information Security of Automated Systems of the North Caucasus Federal University. Research interests: digital technologies, information security management, process approach, educational process, cryptography.

Виталия Валентиновна МОВЗАЛЕВСКАЯ – студентка кафедры информационной безопасности автоматизированных систем Северо-Кавказского федерального университета. Сфера научных интересов: программирование, машинное обучение.

Vitaliya Valentinovna MOVZALEVSKAYA – student of the Department of Information Security of Automated Systems of the North Caucasus Federal University. Research interests: programming, machine learning.

Марина Евгеньевна ТОКМАКОВА – студентка кафедры информационной безопасности автоматизированных систем Северо-Кавказского федерального университета. Сфера научных интересов: криптография, машинное обучение.

Marina Evgenievna TOKMAKOVA – student of the Department of Information Security of Automated Systems of the North Caucasus Federal University. Research interests: cryptography, machine learning.

Михаил Григорьевич БАБЕНКО – доктор физико-математических наук, заведующий кафедрой вычислительной математики и кибернетики Северо-Кавказского федерального университета. Сфера научных интересов: алгебраические структуры в полях Галуа, модулярная арифметика, нейрокомпьютерные технологии, цифровая обработка сигналов, криптографические методы защиты информации.

Mikhail Grigorievich BABENKO – Dr. Sci. (Phys.-Math.), Head of the Department of Computational Mathematics and Cybernetics of the North Caucasus Federal University. Research interests: algebraic structures in Galois fields, modular arithmetic, neurocomputer technologies, digital signal processing, cryptographic methods of information protection.

Виктор Павлович КОЧИН – кандидат технических наук, проректор по учебной работе и интернационализации образования Белорусского государственного университета. Сфера научных интересов: комплексные системы защиты информации, Информационно-коммуникационные технологии.

Viktor Pavlovich KOCHIN – Cand. Sci. (Tech.), Vice-Rector for Academic Affairs and Internationalization of Education of the Belarusian State University. Research interests: integrated information security systems, Information and communication technologies.

DOI: 10.15514/ISPRAS-2024-36(4)-9

High Speed Method of Conversion Numbers from Residue Number System to Positional Notation

[1] *V.V. Lutsenko, ORCID: 0000-0003-4648-8286 <vvlutcenko@ncfu.ru>*
[1] *M.G. Babenko, ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>*
[2] *M.M. Khamidov, ORCID: 0009-0002-9884-5716 <samsungmunis@gmail.com>*

[1] *North-Caucasus Federal University, Stavropol,*
1, Pushkin st., Stavropol, 355017, Russia.
[2] *Samarkand State University named after Sharof Rashidov,*
15, University blv., Samarkand, 703004, Uzbekistan.

Abstract. The Residue Number System is a widely used non-positional number system. Residue Number System can be effectively used in applications and systems with a predominant proportion of addition, subtraction and multiplication operations, due to the parallel execution of operations and the absence of inter-bit carries. The reverse conversion of a number from Residue Number System to positional notation requires the use of special algorithms. The main focus of this article lies in introducing the new conversion method, which incorporates Chinese Remainder Theorem, Akushsky Core Function and rank of number. The step-by-step procedure of the conversion process is detailed, accompanied by numerical examples. The proof of the relationship between the ranks of positional characteristics using the Chinese Remainder Theorem is presented. Through careful analysis and comparison with existing transformation methods, it is concluded that the presented approach takes on average 8 % less time than the Approximate Method.

Keywords: residue number system; Chinese remainder theorem, approximate method; Akushsky core functions; non-modular operations.

For citation: Lutsenko V.V., Babenko M.G., Khamidov M.M. High speed method of conversion numbers from residue number system to positional notation. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024. pp. 117-132. DOI: 10.15514/ISPRAS-2024-36(4)-9.

Acknowledgements. The research was supported by the Russian Science Foundation Grant No. 19-71-10033, https://rscf.ru/en/project/19-71-10033/.

Высокоскоростной метод перевода чисел из системы остаточных классов в позиционную систему счисления

[1] *В.В. Луценко*, ORCID: 0000-0003-4648-8286 <vvlutcenko@ncfu.ru>
[1] *М.Г. Бабенко*, ORCID: *0000-0001-7066-0061 <mgbabenko@ncfu.ru>*
[2] *М.М. Хамидов*, ORCID: *0009-0002-9884-5716 <samsungmunis@gmail.com>*
[1] *Северо-Кавказский федеральный университет,*
355017, Россия, г. Ставрополь, ул. Пушкина, д. 1.
[2] *Самаркандский государственный университет имени Шарофа Рашидова,*
703004, Узбекистан, г. Самарканд, Университетский бульвар, д. 15.

Аннотация. Система остаточных классов – это распространенная непозиционная система счисления. Система остаточных классов может эффективно использоваться в приложениях с преобладающей долей операций сложения, вычитания и умножения благодаря параллельному выполнению операций и отсутствию битовых сдвигов. Обратное преобразование числа из системы остаточных классов в позиционную систему счисления требует использования специальных алгоритмов. Основное внимание в данной статье уделено представлению нового метода преобразования, который использует Китайскую теорему об остатках, функцию ядра Акушского и ранг числа. Подробно описан алгоритм преобразования, представлены числовые примеры. Представлено доказательство связи между рангами позиционных характеристик с помощью Китайской теоремы об остатках. В результате тщательного анализа и сравнения с существующими методами преобразования сделан вывод, что представленный подход занимает в среднем на 8 % меньше времени, чем приближенный метод.

Ключевые слова: система остаточных классов; Китайская теорема об остатках, приближенный метод; функция ядра Акушского; немодулярные операции.

Для цитирования: Луценко В.В., Бабенко М.Г., Хамидов М.М. Высокоскоростной метод перевода чисел из системы остаточных классов в позиционную систему счисления. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 117–132 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(4)–9.

Благодарности. Исследование выполнено за счет гранта Российского научного фонда № 19-71-10033, https://rscf.ru/project/19-71-10033/.

1. Introduction

In today's world, where computational systems play an increasingly significant role in various fields of activity, the question of efficiently converting numbers between different numeral systems becomes particularly relevant. One such system is the Residue Number System (RNS), which provides unique capabilities for handling large numbers through parallel computations [1]. RNS is applied in the following areas: blockchain [2], homomorphic encryption [3], digital signal and image processing [4], neural networks [5].

However, there are cases where it is necessary to translate numbers from RNS to positional notation, which is commonly used in most computational devices [6]. Efficient methods for converting numbers from RNS to positional notation must be developed.

In [7] the authors introduced a technique based on the Chinese Remainder Theorem (CRT) and employed optimized modular arithmetic operations to achieve faster conversions. The algorithm was evaluated on a variety of RNS moduli sets and demonstrated significant improvements in conversion time compared to previously known methods.

Chervyakov et al. [8] focused on developing a hybrid conversion method that combines RNS and binary arithmetic to achieve more efficient conversions. The proposed method utilized operand scanning techniques to identify patterns in the RNS representation and optimize the conversion process. The authors demonstrated that their hybrid approach outperforms conventional conversion methods in terms of both speed and hardware resource utilization.

The article [9] focuses on hardware acceleration for RNS-to-decimal conversion using Field-Programmable Gate Arrays (FPGAs). The authors designed a specialized hardware accelerator capable of handling large-scale RNS numbers and converting them efficiently to decimal format. The proposed FPGA-based implementation demonstrated substantial speedup compared to software-based conversion methods, making it suitable for real-time applications.

In [10-11] proposed energy efficient conversion algorithms which minimizes the energy consumption in the process of number conversion and number sign determination. By optimizing the use of resources and considering the power constraints of the base equipment, the proposed methods provide significant energy savings compared to conventional conversion methods.

Advances in this area have paved the way for improved performance, reduced power consumption and increased fault tolerance, making RNS a more attractive option in various domains [12]. However, further research is still warranted to explore new techniques and optimizations that can further enhance the conversion process and maximize the potential of RNS in modern computing systems.

This paper researches methods of converting numbers from RNS to the positional notation. The main methods are the CRT based method, the Interval Method, the Mixed Radix Conversion (MRC) method, the Diagonal Function (DF) method and the Approximate Method.

The purpose of this paper is to present a high-speed method for converting numbers from RNS to positional notation based on the use of Akushsky core function and number rank.

The paper is organized as follows. In Section 2, we give a brief overview of the Residue Number System. In Section 3, various techniques for converting numbers from RNS to positional notation are described. Section 4 deals with performance evaluation. Finally, Section 5 concludes with some final thoughts and considerations, including possible directions for future research.

2. Residue Number System

In RNS, numbers are expressed as sets of residues obtained by performing modular arithmetic operations on those numbers with respect to a set of coprime moduli. The use of coprime moduli ensures that there is no overlap or interference between the residues, allowing for parallel computation of operations on individual residues [12].

The numerical representation in RNS utilizes the Chinese Remainder Theorem. Let $\{p_1, p_2, \dots, p_n\}$ be mutually prime moduli, and $P = p_1 \cdot p_2 \cdot \dots \cdot p_n$ be their product. For each number X, there exists a set of remainders $x_1, x_2, \dots, x_n$, where $0 \le x_i < p_i$, and these remainders form the RNS representation of X. Put differently, X exhibits congruence with the residues x_i modulo p_i.

Mathematically, this can be expressed as:

$$x_i \equiv X(mod\ p_i). \tag{1}$$

Thus, the number X is written in the RNS in the following form:

$$X = (x_1, x_2, \dots, x_n), \tag{2}$$

The computations for the reductions x_i can be derived through the application of the following equation:

$$x_i = X - \left\lfloor \frac{X}{p_i} \right\rfloor \cdot p_i, \tag{3}$$

To perform operations on numbers in RNS, such as addition and multiplication, operations are carried out independently on the remainders of each modulo. For example, calculations in RNS are performed according to equation:

$$X * Y = (x_1 * y_1, x_2 * y_2, \dots, x_n * y_n).$$

Here, the symbol $*$ represents arithmetic operations, encompassing addition $(+)$, subtraction $(-)$, or multiplication $(\cdot)$. Note that each modulo within the RNS is coprime with every other modulo, satisfying the condition: $(p_i, p_j) = 1$, where $i \neq j$.

3. Methods for Conversion Numbers from RNS to Positional Notation

3.1 Chinese Remainder Theorem

If the number X is given as residues $(x_1, x_2, \dots, x_n)$ from division by moduli $\{p_1, p_2, \dots, p_n\}$, the number X can be obtained from the equation based on the CRT [9]:

$$X = \left| \sum_{i=1}^{n} P_i \cdot x_i \cdot |P_i^{-1}|_{p_i} \right|_P = \sum_{i=1}^{n} P_i \cdot x_i \cdot |P_i^{-1}|_{p_i} - r(X) \cdot P, \quad (4)$$

where P is the dynamic range, $P_i = \frac{P}{p_i}$, $|P_i^{-1}|_{p_i}$ is the multiplicative inversion of P_i modulo p_i, and the operator $|X|_{p_i}$ denotes the remainder of division X by p_i, that is $X \; mod \; p_i$ and $r(X)$ is the rank of the number indicating how many times the range value must be subtracted from the resulting number to bring it back into the range. Let us consider the process of number reconstruction as an example.

Example 1. Given a system of bases $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11$ the volume of the dynamic range $P = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 2310$. Convert the number $X = (1, 2, 1, 4, 7)$ to a positional system.

For this purpose, find the values of P_i:

$$P_1 = \frac{P}{p_1} = 1155, P_2 = \frac{P}{p_2} = 770, P_3 = \frac{P}{p_3} = 462,$$
$$P_4 = \frac{P}{p_4} = 330, P_5 = \frac{P}{p_5} = 320.$$

Subsequently, our focus turns to the computation of multiplicative inversion, a process entailing the determination of α such that $\alpha \cdot P_i \equiv 1 (mod \; p_i)$. Thus:

$$|P_1^{-1}|_{p_1} = 1, |P_2^{-1}|_{p_2} = 2, |P_3^{-1}|_{p_3} = 3, |P_4^{-1}|_{p_4} = 1, |P_5^{-1}|_{p_5} = 1.$$

With these values, we can calculate the value of the number X, according to the (4):

$$X = |8411|_{2310} = 1481.$$

3.2 Approximate Method Based on CRT

In [10, 13] a fractional, approximate representation of numbers based on CRT is proposed. Let us divide (4) by P and obtain

$$\frac{X}{P} = \left| \sum_{i=1}^{n} x_i \cdot \frac{|P_i^{-1}|_{p_i}}{p_i} \right|_1 = \left| \sum_{i=1}^{n} x_i \cdot k_i \right|_1. \quad (5)$$

where $k_i = \frac{|P_i^{-1}|_{p_i}}{p_i}$ constants of the chosen system, and the (5) gives a result within the interval $[0, 1)$. In this context, the process of determining the remainder with a larger modulo is replaced by simply discarding the integer part, a simple operation to implement. To get the exact value, the fractional part is multiplied by P. Consider a similar example.

Example 2. Given a system of bases $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11$ and the number $X = (1, 2, 1, 4, 7)$. Find the constants k_i:

$$k_1 = \frac{1}{2}, \; k_2 = \frac{2}{3}, \; k_3 = \frac{3}{5}, k_4 = \frac{1}{7}, \; k_5 = \frac{1}{11}.$$

Then by (5) it is easy to find:

$$\frac{X}{P} = \left| 1 \cdot \frac{1}{2} + 2 \cdot \frac{2}{3} + 1 \cdot \frac{3}{5} + 4 \cdot \frac{1}{7} + 7 \cdot \frac{1}{11} \right|_1 = \left| 1 \frac{52}{105} \right|_1 = \frac{52}{105},$$

Hence

$$X = \frac{52}{105} \cdot 2310 = 1481.$$

Obviously, these calculations are simpler than in the CRT-based method, but in hardware calculations the fractional coefficients k_i can rarely be represented as finite fractions, so there is a question of rounding accuracy. To perform approximate calculations the fractional coefficients k_i are multiplied by 2^N, where N signifies the count of binary digits located beyond the decimal point, which provides the required level of calculation accuracy, each resulting number is rounded up to the next integer and then all calculations are performed modulo 2^N.

3.3 Mixed Radix Conversion Method

The Mixed Radix Conversion technique involves systematically translating a numerical representation from RNS to Weighted Number System (WNS) through a sequential process [14].This method involves subtracting moduli and multiplying by the multiplicative inversion of a modulo. In WNS the translated number has the following form:

$$X = d_1 + d_2 p_1 + d_3 p_1 p_2 + \cdots + d_n p_1 p_2 \dots p_{n-1}, \tag{6}$$

where $0 \le d_1 \le (p_{i+1} - 1)$. The parameters d_i are known as WNS digits.
The WNS digits can be obtained from the ratios:

$$\begin{gathered} d_1 = X - X_1 \cdot p_1, X_1 = \left\lfloor \frac{X}{p_1} \right\rfloor, \\ d_2 = X_1 - X_2 \cdot p_2, X_2 = \left\lfloor \frac{X_1}{p_2} \right\rfloor, \\ \vdots \\ d_n = X_{n-1} - X_n \cdot p_n, X_n = \left\lfloor \frac{X_{n-1}}{p_n} \right\rfloor. \end{gathered} \tag{7}$$

The conversion carried out according to the algorithm (7) contains $2(n-1)$ only residual arithmetic operations of subtraction and division without remainder, where is the number of moduli of the system. Some modification of the considered algorithm can be proposed in the sense that the division operation is replaced by the multiplication operation. For this purpose we pre-calculate constants τ_{kj} that satisfy the condition

$$\tau_{kj} p_k \equiv 1 \left(mod\ p_j\right), (1 \le k < j \le n) \tag{8}$$

It is noteworthy to highlight that the constants τ_{kj} are entirely dictated by the selected system of bases, rendering them computable beforehand and amenable to storage in a designated table.
If the constants τ_{kj} are calculated, the calculation of the digits d_i WNS by the algorithm (6) can be rewritten in the form:

$$\begin{gathered} d_1 \equiv x_1\ (mod\ p_1), \\ d_2 \equiv (x_2 - d_1)\tau_{12}\ (mod\ p_2), \\ d_3 \equiv \left((x_3 - d_1)\tau_{13} - d_2\right)\tau_{23}\ (mod\ p_3) \\ \vdots \\ d_n \equiv (\dots (x_n - d_1)\tau_{1n} - \cdots d_{n-1})\tau_{n-1n}\ (mod\ p_n). \end{gathered} \tag{9}$$

The constants τ_{kj} are multiplication inverses for the numbers p_k modulo p_j
Consider the algorithm (9) with an example.

Example 3. Let a system of bases $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11$ be given. The volume of the dynamic range $P = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 2310$. Convert the number $X = (1, 2, 1, 4, 7)$ to WNS.
We first find the constants τ_{kj}. For convenience, we write the constants τ_{kj} as a matrix $k \times j$:

$$\begin{pmatrix} 0 & 2 & 3 & 4 & 6 \\ 0 & 0 & 2 & 5 & 4 \\ 0 & 0 & 0 & 3 & 9 \\ 0 & 0 & 0 & 0 & 8 \end{pmatrix}$$

Now run the algorithm (9) and write the results in Tab. 1.

Table 1. Algorithm of the MRC method.

Actions	Moduli					Digits
	$p_1 = 2$	$p_2 = 3$	$p_3 = 5$	$p_4 = 7$	$p_5 = 11$	
$X - d_1$	1 1	2 1	1 1	4 1	7 1	$d_1 = 1$
$(X - d_1)\tau_{1j}$	0	1 2	0 3	3 4	6 6	
$X_1 - d_2$		2 2	0 2	5 2	3 2	$d_2 = 2$
$(X_1 - d_2)\tau_{2j}$		0	3 2	3 5	1 4	
$X_2 - d_3$			1 1	1 1	4 1	$d_3 = 1$
$(X_2 - d_3)\tau_{3j}$			0	0 3	3 9	
$X_3 - d_4$				0 0	5 0	$d_4 = 1$
$(X_3 - d_4)\tau_{4j}$				0	5 8	
X_4						$d_5 = 7$

Thus,

$$X = d_1 + d_2p_1 + d_3p_1p_2 + d_4p_1p_2p_3 + d_4p_1p_2p_3 + d_5p_1p_2p_3p_4 =$$
$$= 1 + 2 \cdot 2 + 1 \cdot 2 \cdot 3 + 0 \cdot 2 \cdot 3 \cdot 5 + 7 \cdot 2 \cdot 3 \cdot 5 \cdot 7 = 1481.$$

3.4 Interval Method

Sufficiently effective methods of converting numbers from RNS to positional representation is the interval method, based on the interval characteristics of numbers. One of these characteristics is the interval number [15].

Let RNS is given by a system of bases $\{p_1, p_2, \dots, p_n\}$, with the volume of the range $P = \prod_{i=1}^{n} p_i$. Choose a splitting modulo p_i and split the given range into intervals by dividing P by the modulo p_i. Then the number of intervals is $m = P_i = \frac{P}{p_i}$, and the length of an interval is determined by the modulo value. As a result, the value of any number X given in RNS on the chosen bases can be determined by the interval number:

$$l_X = \left\lceil \frac{X}{p_i} \right\rceil. \quad (10)$$

which contains the number X and by digit x_i of the number X in the RNS modulo p_i, i.e.

$$X = p_i l_X + x_i. \quad (11)$$

Since $(p_i, P_i) = 1$, by Euler's theorem:

$$P_i^{\varphi(p_i)} \equiv 1 \ (mod\ p_i), \quad (12)$$

where $\varphi(p_i)$ is an Euler function. If p_i is a prime number, then $\varphi(p_i) = p_i - 1$.

Substituting (12) into (4) the number X can be written as

$$X = \left| \sum_{i=1}^{n} P_i^{\varphi(p_i)} x_i \right|_P . \tag{13}$$

To determine the interval number l_X, substitute (13) into (10):

$$l_X = \left\lceil \frac{\sum_{i=1}^{n} P_i^{\varphi(p_i)} x_i - r(X)P}{p_i} \right\rceil. \tag{14}$$

Since p_i is a divisor of the numbers $P_j^{\varphi(p_j)}$ $(i \neq j)$, $P_i^{\varphi(p_i)} - 1$, P then

$$l_X = l_{X_1} x_1 + l_{X_2} x_2 + \cdots + l_{X_n} x_n - r_X P. \tag{15}$$

where $l_{X_j} = \frac{P_j^{\varphi(p_j)}}{p_i}$, $(i \neq j)$ and $l_{X_j} = \frac{P_i^{\varphi(p_i)} - 1}{p_i}$ are constant coefficients defined by the base system. Thus we have,

$$l_X = \left| \sum_{i=1}^{n} \left| l_{X_i} x_i \right|_{P_i}^{+} \right|_{P_i}^{+} . \tag{16}$$

Substituting (16) into (11), we obtain a positional notation of the number X:

$$l_X = \left(\left| \sum_{i=1}^{n} \left| l_{X_i} x_i \right|_{P_i}^{+} \right|_{P_i}^{+} \right) p_i + x_i. \tag{17}$$

It may be noted here that it is more appropriate to choose the largest modulo in the system as the split modulo. In this case, modular operations are performed with a smaller modulo value.

We will illustrate this method with an example.

Example 4. Let a system of bases $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11$ be given. Convert the number $X = (1, 2, 1, 4, 7)$ to a positional notation. Let us choose $p_5 = 11$ as the splitting modulo, then $P_5 = \frac{P}{p_5} = 210$, the interval number

$$l_X = \left| \sum_{i=1}^{5} \left| l_{X_i} x_i \right|_{210}^{+} \right|_{210}^{+} .$$

and the number $X = p_5 l_X + x_5$. Define l_{X_i}. Since $\varphi(p_1) = 2 - 1 = 1$, $\varphi(p_2) = 3 - 1 = 2$, $\varphi(p_3) = 5 - 1 = 4$, $\varphi(p_4) = 7 - 1 = 6$, $\varphi(p_4) = 11 - 1 = 10$, then

$$l_{X_1} = \left| \frac{1155}{11} \right|_{210}^{+} = 105,\ l_{X_2} = \left| \frac{770^2}{11} \right|_{210}^{+} = 140,\ l_{X_3} = \left| \frac{462^4}{11} \right|_{210}^{+} = 126,$$

$$l_{X_4} = \left| \frac{330^6}{11} \right|_{210}^{+} = 30,\ l_{X_5} = \left| \frac{210^{10} - 1}{11} \right|_{210}^{+} = 19.$$

Then $l_X = |764|_{210}^{+} = 210$.

Thus, $X = 134 \cdot 11 + 7 = 1481$.

3.4 Diagonal Function

There is another way of reconstructing the numbers in the literature [16, 17]. For RNS $\{p_1, p_2, \ldots, p_n\}$ define the Sum of Quotients (SQ) parameter as

$$SQ = P_1 + P_2 + \cdots + P_n, \tag{18}$$

and the constants

$$k_i = |-p_i^{-1}|_{SQ}. \tag{19}$$

The diagonal function for a given number $X = (x_1, x_2, \dots, x_n)$ is defined as

$$D(X) = |x_1 k_1 + x_2 k_2 + \cdots + x_n k_n|_{SQ}. \quad (20)$$

If (4) is multiplied by $\frac{SQ}{P}$, we get the scaled value of X:

$$\frac{X \cdot SQ}{P} = \left| \sum_{i=1}^{n} SQ \cdot \frac{x_i}{p_i} \cdot |P_i^{-1}|_{p_i} \right|_{SQ}. \quad (21)$$

From the definition of k_i (19) we can derive $\beta_i \cdot SQ - k_i p_i = 1$, where $\beta_i = |P_i^{-1}|_{p_i}$, which is equivalent to $\beta_i = |P_i^{-1}|_{p_i}$. Thus, $k_i = \frac{SQ}{p_i} \cdot |P_i|_{p_i} - \frac{1}{p_i}$, where $\frac{SQ}{p_i} \cdot |P_i^{-1}|_{p_i} = k_i + \frac{1}{p_i}$. Then substituting $k_i + \frac{1}{p_i}$ in (20) instead of k_i we get the scaled value of $D'(X)$. Thus, to obtain the value of X, substitute the calculated values in (21) and multiply by $\frac{P}{SQ}$.

$$X = \frac{SQ}{P} \cdot \left| \sum_{i=1}^{n} x_i \left(k_i + \frac{1}{p_i} \right) \right|_{SQ} = \frac{P \cdot D(X) + \sum_{i=1}^{n} x_i \cdot P_i}{SQ}. \quad (22)$$

Consider this method with an example.

Example 5. Similarly, we are given RNS $\{2, 3, 5, 7, 11\}$ and a number $X = 1481 = (1, 2, 1, 4, 7)$. From the previous examples we know $P = 2310$, $P_1 = 1155$, $P_2 = 770$, $P_3 = 462$, $P_4 = 330$, $P_5 = 210$. Then $SQ = 2927$ and from (19) $k_1 = 1463$, $k_2 = 1951$, $k_3 = 1756$, $k_4 = 418$, $k_5 = 266$. Find the diagonal function

$$D(X) = |10655|_{2927} = 1874,$$

From (22) find the required value:

$$X = \frac{4334887}{2927} = 1481.$$

4. The Akushsky Core Function Method Based on the Rank of Number

We present a fast technique for conversion numerical values from the RNS to positional notation. This approach involves using the Akushsky Core Function to find the rank of a number. The Akushsky Core Function [18] is defined by the following equation

$$C(X) = \sum_{i=1}^{n} w_i \left\lfloor \frac{X}{p_i} \right\rfloor. \quad (23)$$

where integers w_i are constants determined by the choice of the interpolation point. The numbers w_i in equation (23) can be arbitrary in a certain sense. It is they that define each particular core function and can vary depending on the problem to be solved. An algorithm for determining the optimal weights for the Akushsky core function is presented in [19].

Core function range value is calculated as

$$C(P) = C_P = \sum_{i=1}^{n} w_i P_i. \quad (24)$$

We define the so-called orthogonal bases B_i as

$$B_i = P_i \cdot |P_i^{-1}|_{p_i},$$

We also define the coefficients c_i as

$$c_i = C(B_i),$$

Rewrite (23) as

$$C(X) = \left|\sum_{i=1}^{n} c_i x_i\right|_{C_P} = \sum_{i=1}^{n} c_i \cdot x_i - \check{r}(X) \cdot C_P, \tag{25}$$

Then the rank of the Akushsky core function number can be defined as

$$\check{r}(X) = \left\lfloor \frac{\sum_{i=1}^{n} c_i \cdot x_i}{C_P} \right\rfloor. \tag{26}$$

There are three forms of representation of the CRT, each of them corresponds to a positional characteristic of the number represented in RNS.

The first form was represented in (4), the rank of a number in this representation can be calculated as follows

$$r(X) = \left\lfloor \sum_{i=1}^{n} \frac{1}{p_i} \cdot |P_i^{-1}|_{p_i} \cdot x_i \right\rfloor. \tag{27}$$

Second form

$$X = \left|\sum_{i=1}^{n} P_i \cdot \left||P_i^{-1}|_{p_i} \cdot x_i\right|_{p_i}\right|_P = \sum_{i=1}^{n} P_i \cdot \left||P_i^{-1}|_{p_i} \cdot x_i\right|_{p_i} - \hat{r}(X) \cdot P, \tag{28}$$

where $\hat{r}(X)$ is the normalised rank of the number, which can be calculated as

$$\hat{r}(X) = \left\lfloor \sum_{i=1}^{n} \frac{1}{p_i} \cdot \left||P_i^{-1}|_{p_i} \cdot x_i\right|_{p_i} \right\rfloor, \tag{29}$$

The third form is proposed and its rank is represented respectively in (25) and (26).

Consider the following properties.

Theorem 1.

$$\hat{r}(X) = -\frac{X}{P} + \sum_{i=1}^{n} \frac{\left||P_i^{-1}|_{p_i} \cdot x_i\right|_{p_i}}{p_i}.$$

Proof:

According to the definition

$$\hat{r}(X) = \left\lfloor \sum_{i=1}^{n} \frac{\left||P_i^{-1}|_{p_i} \cdot x_i\right|_{p_i}}{p_i} \right\rfloor = \left\lfloor \frac{1}{P} \sum_{i=1}^{n} \left||P_i^{-1}|_{p_i} \cdot x_i\right|_{p_i} \cdot P_i \right\rfloor. \tag{30}$$

Since $\left\lfloor \frac{X}{P} \right\rfloor = \frac{X}{P} - \frac{|X|_P}{P}$, then

$$\hat{r}(X) = \frac{1}{P} \sum_{i=1}^{n} \left||P_i^{-1}|_{p_i} \cdot x_i\right|_{p_i} \cdot P_i - \frac{1}{P} \cdot \left| \sum_{i=1}^{n} \left||P_i^{-1}|_{p_i} \cdot x_i\right|_{p_i} \cdot P_i \right|_P.$$

According to the CRT, $\left|\sum_{i=1}^{n} \left||P_i^{-1}|_{p_i} \cdot x_i\right|_{p_i} \cdot P_i\right|_P = X$, consequently,

$$\hat{r}(X) = \frac{1}{P} \sum_{i=1}^{n} \left||P_i^{-1}|_{p_i} \cdot x_i\right|_{p_i} \cdot P_i - \frac{X}{P}.$$

The theorem is proved.

Theorem 2.

$$\hat{r}(1) = -\frac{1}{P} + \sum_{i=1}^{n} \frac{|P_i^{-1}|_{p_i}}{p_i}.$$

Proof:

It follows directly from Theorem 1 that $\hat{r}(1) = -\frac{1}{P} + \sum_{i=1}^{n} \frac{|P_i^{-1}|_{p_i}}{p_i}$.

The theorem is proved.

Let us examine the correlation between the ranks of positional characteristics.

Theorem 3.

Let $p_1 < p_2 < \cdots < p_n$, the number $X \xrightarrow{RNS} (x_1, x_2, \dots, x_n)$ and the weights of the Akushsky core function $w_1, w_2, \dots, w_n$ satisfying the condition $0 \le X < P$, then

$$\check{r}(X) = r(X) + \left\lfloor \frac{C(X)}{C_P} \right\rfloor. \tag{31}$$

Proof:

Let us calculate c_i, we get

$$c_i = C(B_i) = \sum_{j=1}^{n} w_j \left\lfloor \frac{|P_i^{-1}|_{p_i} \cdot P_i}{p_j} \right\rfloor. \tag{32}$$

Since $\forall i \ne j$: $|P_i^{-1}|_{p_i} \cdot P_i \equiv 0 \bmod p_j$ and $\forall i$: $|P_i^{-1}|_{p_i} \cdot P_i \equiv 1 \bmod p_i$, then for $i \ne j$: $\left\lfloor |P_i^{-1}|_{p_i} \cdot P_i / p_i \right\rfloor = \frac{|P_i^{-1}|_{p_i} \cdot P_i}{p_i}$, and for $i = j$: $\left\lfloor |P_i^{-1}|_{p_i} \cdot P_i / p_i \right\rfloor = \frac{|P_i^{-1}|_{p_i} \cdot P_i - 1}{p_i}$, hence the coefficient c_i can be represented as follows

$$c_i = |P_i^{-1}|_{p_i} \cdot P_i \cdot \sum_{j=1}^{n} \frac{w_j}{p_j} - \frac{w_i}{p_i}. \tag{33}$$

Given that $\sum_{j=1}^{n} \frac{w_j}{p_j} = \frac{C_P}{P}$, then (33) is transformed to the form

$$\check{r}(X) = \left\lfloor \frac{\sum_{i=1}^{n} c_i \cdot x_i}{C_P} \right\rfloor = \left\lfloor \frac{1}{P} \cdot \sum_{i=1}^{n} |P_i^{-1}|_{p_i} \cdot P_i \cdot x_i - \frac{1}{C_P} \cdot \sum_{i=1}^{n} \frac{x_i \cdot w_i}{p_i} \right\rfloor. \tag{34}$$

Substituting (34) into (28), we obtain

$$\check{r}(X) = \left\lfloor r(X) + \frac{X}{P} - \frac{1}{C_P} \cdot \sum_{i=1}^{n} \frac{x_i \cdot w_i}{p_i} \right\rfloor. \tag{35}$$

Considering that

$$\sum_{i=1}^{n} \frac{x_i \cdot w_i}{p_i} = \sum_{i=1}^{n} \frac{\left(X - p_i \cdot \left\lfloor \frac{X}{p_i} \right\rfloor\right) \cdot w_i}{p_i} = X \cdot \sum_{i=1}^{n} \frac{w_i}{p_i} - \sum_{i=1}^{n} \left\lfloor \frac{X}{p_i} \right\rfloor \cdot w_i = X \cdot \frac{C_P}{P} - C(X). \tag{36}$$

Substituting (36) into (35), we obtain

$$\check{r}(X) = \left\lfloor r(X) + \frac{C(X)}{C_P} \right\rfloor. \tag{37}$$

Since as $r(X) \in \mathbb{Z}$, and $\forall a \in \mathbb{R}, n \in \mathbb{Z}$: $\lfloor a + n \rfloor = \lfloor a \rfloor + n$, then

$$\check{r}(X) = r(X) + \left\lfloor \frac{C(X)}{C_P} \right\rfloor.$$

Theorem 4. Let $p_1 < p_2 < \cdots < p_n$, a number $X \in \mathbb{Z}_P$ and an Akushsky core function with with all positive weights w_i be given, then $\check{r}(X) = r(X)$.

Proof:

According to Theorem 3, $\check{r}(X) = r(X) + \left\lfloor \frac{C(X)}{C_P} \right\rfloor$. Given that the Akushsky core function contains no critical cores, $\forall X \in [0, P)$: $0 \le C(X) < C_P$. Hence $\left\lfloor \frac{C(X)}{C_P} \right\rfloor = 0$, and hence $\check{r}(X) = r(X)$.

The theorem is proved.

Let us consider our proposed method with an example.

Example 6. Similarly, we are given RNS $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11$ and a number $X = 1481 = (1,2,1,4,7)$. $P = 2310, P_1 = 1155, P_2 = 770, P_3 = 462, P_4 = 330, P_5 = 210$. Let us use a set of weights $w_1 = 0, w_2 = 0, w_3 = 0, w_4 = 0, w_5 = 1$.

Let us calculate the values of B_i:

$$B_1 = P_1 \cdot |P_1^{-1}| = 1155, B_2 = P_2 \cdot |P_2^{-1}| = 1540, B_3 = P_3 \cdot |P_3^{-1}| = 1386,$$
$$B_4 = P_4 \cdot |P_4^{-1}| = 330, B_5 = P_5 \cdot |P_5^{-1}| = 210.$$

Then we find the value of the core function range by (24)

$$C(P) = C_P = 210.$$

Find the value of coefficients c_i:

$$c_1 = 105, c_2 = 140, c_3 = 126, c_4 = 30, c_5 = 19.$$

Then the rank of the number is

$$\check{r}(X) = \left\lfloor \frac{105 \cdot 1 + 140 \cdot 2 + 126 \cdot 1 + 30 \cdot 4 + 19 \cdot 7}{210} \right\rfloor = 3.$$

Thus,

$$X = 1155 \cdot 1 + 1540 \cdot 2 + 1386 \cdot 1 + 330 \cdot 4 + 210 \cdot 7 - 3 \cdot 2310 = 1481.$$

5. Performance Evaluation

The methodology expounded in Section 4 evinces an indisputable advantage over the approaches outlined in Section 3.

To validate the properties of each approach, every algorithm was carefully implemented in Python, and a comprehensive performance analysis was executed on a computer equipped with an Intel Core i7-7700HQ processor running at 2.80 GHz, 8 GB DDR4 RAM at 1196 MHz, and a 512 GB SSD, operating on Windows 10 Home Edition. The study involves two significant phases:

Stage A examines the performance of three moduli by processing data sets of 50000, 100000, 200000, 350000, and 500000 using each of the proposed methods.

In Stage B, we expanded our analysis to cover 19 sets, varying from 3 to 21 moduli, with each modulo having an 8-bit dimensionality. We processed a data set of 100000 numbers.

Throughout the two-stage simulation, we measured the time characteristics of each method with attention to detail. To guarantee precision and dependability, we reiterated each measurement one hundred times and recorded the average time for evaluation. The findings of these experiments are presented concisely in Tables 2 and 3, with time values depicted in seconds.

Let us conduct a detailed examination of the ensuing tables, delving deeper into the tabulated data with a scientific scrutiny. The provided information discusses two stages: Stage A and Stage B, focusing on their time characteristics and importance. Stage A is crucial for tracking method behavior with increasing data size. Analysis of the data shows a linear growth, which indicates the stability of the obtained method using the core function. To enhance understanding, graphs will be presented.

Table 2 provides insights into the time-related features observed during Stage B, underscoring the significance of this phase akin to Stage A. In a practical system comprising the control system may encompass various configurations, such as two, four, six, or more moduli. Consequently, exploring the behavior of methods in relation to the number of moduli within the system becomes imperative.

The acquired data not only facilitates an understanding of methodological performance but also allows for inferences regarding the stability of the methods.

Table 2. The result of the study of stage A.

Amount	CRT Method	Approximate Method	MRC Method	Interval Method	DF Method	Rank Core Method
50000	0.17184758	0.15310092	0.17178512	0.87428927	0.28112435	0.13308518
100000	0.37491608	0.33490013	1.61959763	1.9496377	0.48421788	0.29072099
200000	0.70988417	0.57174363	2.60662079	3.31276298	1.19526315	0.45420003
350000	1.19558525	1.14163585	4.49329138	5.65635467	1.68907547	0.87942809
500000	1.68847227	1.59833269	6.57562566	8.14383984	2.54651117	1.25240469

Table 3. The result of stage B study: dimension of modulo set p[n] where n represents modulo count in the ensemble.

p[n]	CRT Method	Approximate Method	MRC Method	Interval Method	DF Method	Rank Core Method
3	0.04886961	0.04188609	0.16806006	0.19650173	0.07034159	0.0388873
4	0.04986429	0.04387736	0.21841407	0.25733018	0.08476949	0.03990845
5	0.05085826	0.04582379	0.34164977	0.30221629	0.09973574	0.04392817
6	0.06984472	0.07378912	0.34810019	0.33629251	0.10273242	0.0588873
7	0.07682538	0.08676386	0.41370296	0.39549708	0.11066651	0.06990845
8	0.07836747	0.09275365	0.51267076	0.43252301	0.12862134	0.07392817
9	0.08676624	0.09826112	0.59272242	0.45488119	0.13066811	0.09067698
10	0.09473872	0.10372066	0.62236333	0.53865314	0.13461476	0.09264201
11	0.11070347	0.11968732	0.72314477	0.55988812	0.15419126	0.09558553
12	0.11466908	0.12268424	0.77975607	0.6223812	0.17807412	0.11655969
13	0.12469697	0.12665558	0.88087988	0.63008047	0.19151998	0.12052173
14	0.13267827	0.12510133	1.01868820	0.66010213	0.19850206	0.12311763
15	0.13463926	0.12766194	1.04511428	0.75057304	0.20049644	0.12251919
16	0.16458368	0.12769699	1.15422750	0.81782241	0.20647359	0.11738696
17	0.16755462	0.13862944	1.29933691	0.86782241	0.23633909	0.12436595
18	0.16951680	0.14361358	1.35722113	0.87273455	0.24734974	0.13237681
19	0.17810869	0.15760803	1.40875983	0.93827939	0.26701593	0.13935819
20	0.18051696	0.15960505	1.51713409	0.98166609	0.26928353	0.15541186
21	0.18350887	0.16758013	1.63444066	1.02923965	0.27526116	0.16631331

The data from the given table were utilized to create visual representations in the form of figures. In addition, a more comprehensive analysis was enabled by extrapolating the acquired values through polynomial methods, extending the perspective on the outcomes.

Upon scrutinizing the acquired outcomes, we can extrapolate the following deductions. Examining the graphical representation in Fig. 1, it becomes apparent that conventional methodologies demonstrate efficacy particularly when handling a limited quantity of numerical inputs. However, starting from the processing of two hundred thousand numbers, MRC method and interval method begin to lose.

A similar situation is apparent in the graph presented in Fig. 2. On average, our approach displays a time efficiency that is roughly 8 % superior to that of the Approximate Method.

The comparative analysis conducted on methods for translating numbers from RNS to positional notation revealed that the method utilizing Akushsky core function and number rank offers certain advantages. This is due to the performance of addition and multiplication operations in positional notation within the mentioned approach. When performing calculations using MRC, each RNS modulo corresponds to a separate channel in which calculations are completed using modular arithmetic. However, these calculations are not performed in parallel. When using the interval method, it is necessary to complete operations such as addition, multiplication, and degree expansion in the positional system. Degree expansion can result in rather large values. One positive aspect of the interval method is the ability to process data in a conveyor-like manner.

6. Conclusion

In this paper, we have presented a high speed method for converting numbers from RNS to positional notation. The proposed method offers a novel approach to achieve rapid and accurate conversions. By leveraging the inherent properties of the RNS and optimizing algorithms, our method streamlines the conversion process, minimizing computational complexities, and significantly reducing conversion times. Experiments demonstrate its superiority over conventional methods, showcasing notable improvements in speed.

While our proposed method represents a significant advancement, there is still room for further exploration and optimization. Future studies may investigate hybrid conversion techniques that combine the strengths of different algorithms, aiming to achieve even greater efficiency. Additionally, evaluating the proposed method's performance in large-scale systems and exploring its potential application in emerging technologies will be exciting avenues for future research.

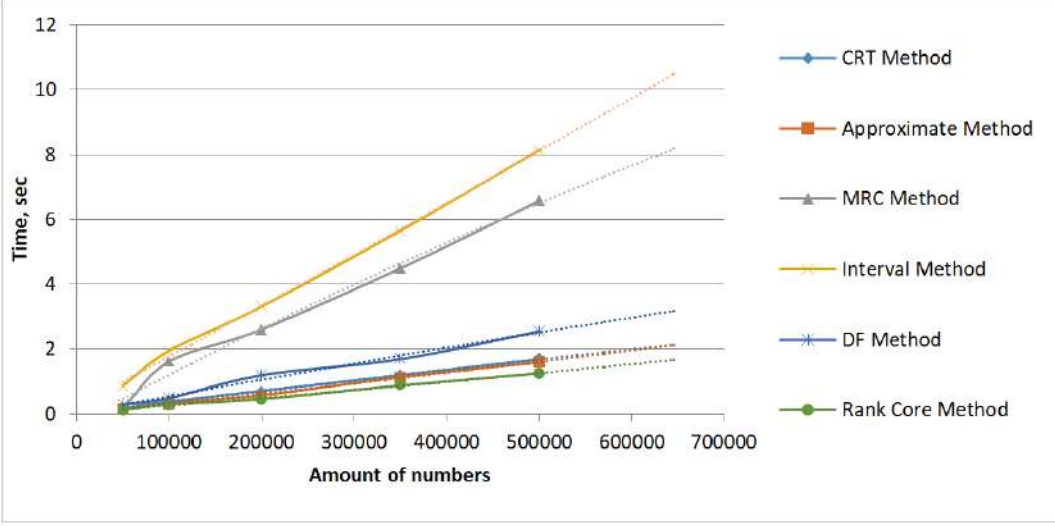


Fig. 1. Findings from stage A analysis.

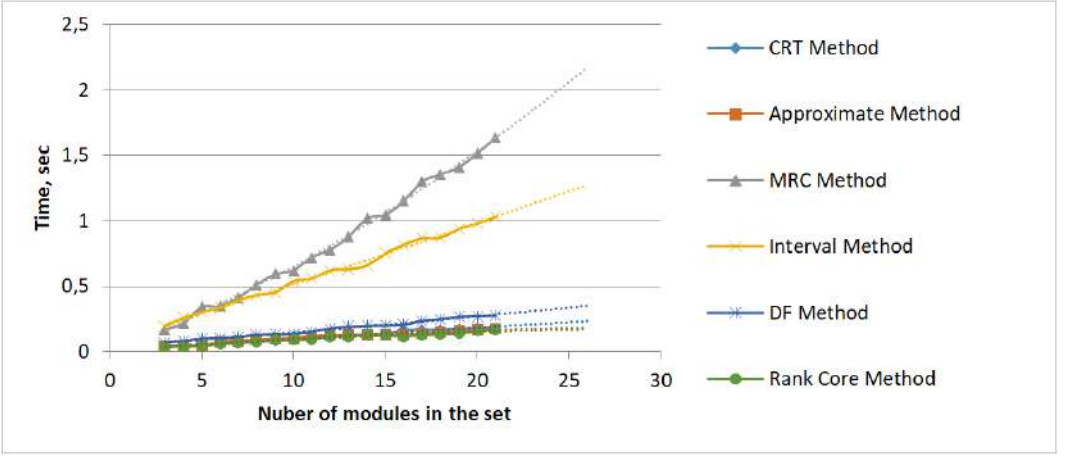


Fig. 2. Findings from stage B analysis.

References

[1]. Mohan P.A., Ananda Mohan P.V. Error Detection, Correction and Fault Tolerance in RNS-Based Designs. Residue Number Systems: Theory and Applications, Birkhäuser: Cham, Switzerland, 2016, pp. 163-175.
[2]. Guo Z., Gao Z., Mei H., Zhao M., Yang J. Design and optimization for storage mechanism of the public blockchain based on redundant residual number system. IEEE Access, 2019, vol. 7, pp. 98546-98554.
[3]. Al Badawi A., Polyakov Y., Aung K. M. M., Veeravalli B., Rohloff K. Implementation and performance evaluation of RNS variants of the BFV homomorphic encryption scheme. IEEE Transactions on Emerging Topics in Computing, 2019, vol. 9, no. 2, pp. 941-956.
[4]. Molahosseini A. S., De Sousa L. S., Chang C. H. Embedded systems design with special arithmetic and number systems. Cham, Switzerland: Springer, 2017, p. 390.
[5]. Valueva M. V., Nagornov N. N., Lyakhov P. A., Valuev G. V., Chervyakov N. I. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and computers in simulation, 2020, vol. 177, pp. 232-243.
[6]. Dong X. A multi-secret sharing scheme based on the CRT and RSA. International Journal of Electronics and Information Engineering, 2015, vol. 2, no. 1, pp. 47-51.
[7]. Van Vu T. Efficient implementations of the Chinese remainder theorem for sign detection and residue decoding. IEEE Transactions on Computers, 1985, vol. 100, no. 7, pp. 645-651.
[8]. Chervyakov N. I., Molahosseini A. S., Lyakhov P. A., Babenko M. G., Deryabin M. A. Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem. International journal of computer mathematics, 2017, vol. 94, no. 9, pp. 1833-1849.
[9]. Chervyakov N. I., Babenko M. G., Kuchukov V. A. Research of effective methods of conversion from positional notation to RNS on FPGA. In Proceedings of the 2011 45th Annual Conference on Information Sciences and Systems, 1-3 February 2017, IEEE: St. Petersburg/Moscow, Russia, 2017, pp. 371-375.
[10]. Soderstrand M., Vernia C., Chang J. H. An improved residue number system digital-to-analog converter. IEEE transactions on circuits and systems, 1983, vol. 30, no. 12, pp. 903-907.
[11]. Babenko M., Golimblevskaia E. About One Property of Number Rank in RNS. In Proceedings of the 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, 26-29 January 2021, St. Petersburg, FL, USA, 2021, pp. 212-216.
[12]. Isupov K. High-Performance Computation in Residue Number System Using Floating-Point Arithmetic. Computation, 2021, vol. 9, no. 2, 9.
[13]. Chervyakov N.I., Averbukh V.M., Babenko M.G., Lyakhov P.A., Gladkov A.V., Gapochkin A.V. An Approximate Method for Performing Non-Modular Operations in a System of Residual Classes. Fundam. Res., 2012, vol. 6, pp. 189-193.
[14]. Yassine H. M., Moore W. R. Improved mixed-radix conversion for residue number system architectures. IEE Proceedings G (Circuits, Devices and Systems), 1991, vol. 138, no. 1, pp. 120-124.
[15]. Gladkov A., Gladkova N., Kucherov N. Analytical Review of Methods for Detection, Localization and Error Correction in the Residue Number System. In Proceedings of the International Conference on

Mathematics and its Applications in New Computer Systems, 25-28 October 2022, Munich, Germany, Springer: Berlin/Heidelberg, 2022, pp. 507-514.

[16]. Babenko M., Piestrak S. J., Chervyakov N., Deryabin M. The study of monotonic core functions and their use to build RNS number comparators. Electronics, 2021, vol. 10, no. 9, p. 1041.

[17]. Piestrak, S. J. A note on RNS architectures for the implementation of the diagonal function. Information Processing Letters, 2015, vol. 115, no. 4, pp. 453-457.

[18]. Акушский И. Я., Бурцев В. М., Пак И. Т. О новой позиционной характеристике непозиционного кода и ее приложении. Теория кодирования и оптимизация сложных систем. Алма-Ата, Наука, КазССР, 1977, стр. 8–16. / Akushsky, I.Y., Burtsev, V.M., Pak, I.T. (1977) About the New Positional Characteristic of the Non-Positional Code and Its Application. In Theory of Coding and Optimization of Complex Systems, Alma-Ata, Nauka, KazSSR, 1977, pp. 8-16 (in Russian).

[19]. Shiriaev E., Kucherov N., Babenko M., Lutsenko V., Al-Galda S. Algorithm for Determining the Optimal Weights for the Akushsky Core Function with an Approximate Rank. Applied Sciences, 2023, vol. 13, no. 18, p. 10495.

Информация об авторах / Information about authors

Владислав Вячеславович ЛУЦЕНКО – аспирант кафедры вычислительной математики и кибернетики факультета математики и компьютерных наук имени профессора Н.И. Червякова ФГАОУ ВПО «Северо-Кавказский федеральный университет». Сфера научных интересов: высокопроизводительные вычисления, система остаточных классов, умный город, нейронные сети, интернет вещей.

Vladislav Vyacheslavovich LUTSENKO – postgraduate student, Department of Computational Mathematics and Cybernetics, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov, North Caucasus Federal University. Research interests: high-performance computing, residue number system, smart city, neural networks, Internet of Things.

Михаил Григорьевич БАБЕНКО – доктор физико-математических наук, заведующий кафедры вычислительной математики и кибернетики факультета математики и компьютерных наук имени профессора Н.И. Червякова ФГАОУ ВПО «Северо-Кавказский федеральный университет». Сфера научных интересов: облачные вычисления, высокопроизводительные вычисления, система остаточных классов, нейронные сети, криптография.

Mikhail Grigoryevich BABENKO – Dr. Sci. (Phys.-Math.), Head of the Department of Computational Mathematics and Cybernetics, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov, North Caucasus Federal University. His research interests include cloud computing, high-performance computing, residue number systems, neural networks, cryptography.

Мунис Мусинович ХАМИДОВ – доцент Самаркандского государственного университета имени Шарофа Рашидова.

Munis Musinovich KHAMIDOV – Assistant Professor at Samarkand State University named after Sharof Rashidov.

DOI: 10.15514/ISPRAS-2024-36(4)-10

Experimental Comparison of Logic Circuit Synthesis Methods

M.D. Vershkov, ORCID: 0009-0001-5389-6117 <mdvershkov@edu.hse.ru>
A.A. Yagzhov, ORCID: 0009-0007-8522-3391 <aayagzhov@edu.hse.ru>
N.S. Romanov, ORCID: 0009-0009-7440-1839 <nsromanov_1@edu.hse.ru>
A.A. Fedotova, ORCID: 0009-0001-5613-8696 <aafedotova_6@edu.hse.ru>
E.P. Znatnov, ORCID: 0009-0002-4260-2803 <epznatnov@edu.hse.ru>

Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
National Research University Higher School of Economics,
20, Myasnitskaya st., Moscow, 101000, Russia.

Abstract. This paper presents the results of an experimental comparison of methods for the synthesis of combinational logic circuits that implement specified Boolean functions. The following methods were considered: the method of Akers, bi-decomposition, the methods of cascades, Minato-Morreale, Reed-Muller and DSD-decomposition. The comparison was based on an estimate of power, delay and area of synthesized logic circuits. The evaluation was carried out without the process of technology mapping of the circuits. These parameters were chosen because they are the main criteria for technology-independent optimization, where these methods are widely used. Boolean functions with the number of arguments from 4 to 10 were used as input data. They were generated on the basis of information on the frequency of occurrence of various NPN-equivalence classes of Boolean functions of 4 variables. As a result of the study, it was found that the Minato-Morreale method is the most universal in solving technology-independent optimization problems and can be used for different criteria.

Keywords: Boolean functions, logic optimization, electronic design automation.

For citation: Vershkov M.D., Yagzhov A.A., Romanov N.S., Fedotova A.A., Znatnov E.P. Experimental Comparison of Logic Circuit Synthesis Methods. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024. pp. 133-142. DOI: 10.15514/ISPRAS-2024-36(4)-10.

Экспериментальное сравнение методов синтеза логических схем

М.Д. Вершков, ORCID: 0009-0001-5389-6117 <mdvershkov@edu.hse.ru>
А.А. Ягжов, ORCID: 0009-0007-8522-3391 <aayagzhov@edu.hse.ru>
Н.С. Романов, ORCID: 0009-0009-7440-1839 <nsromanov_1@edu.hse.ru>
А.А. Федотова, ORCID: 0009-0001-5613-8696 <aafedotova_6@edu.hse.ru>
Е.П. Знатнов, ORCID: 0009-0002-4260-2803 <epznatnov@edu.hse.ru>

Институт системного программирования РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
Национальный исследовательский университет «Высшая школа экономики»,
Россия, 101000, г. Москва, ул. Мясницкая, д. 20

Аннотация. В работе описаны результаты экспериментального сравнения методов синтеза комбинационных логических схем, реализующих заданные булевы функции. Рассмотрены следующие методы: Акерса, би-декомпозиции, каскадов, Минато-Морреале, Рида-Маллера и DSD-разложения. Сравнение основывалось на оценке энергопотребления, задержки и площади синтезируемых логических схем. Оценка осуществлялась без процесса технологического отображения схем. Выбор данных параметров обусловлен тем, что они являются основными критериями технологически независимой оптимизации, в которой данные методы находят широкое применение. В качестве исходных данных использовались булевы функции с числом аргументов от 4 до 10, которые были сгенерированы на основе информации о частоте встречаемости различных NPN-классов эквивалентности булевых функций от 4 переменных. В результате исследования было установлено, что метод Минато-Морреале является наиболее универсальным при решении задач технологически независимой оптимизации и может быть использован для различных критериев.

Ключевые слова: булевы функции, логическая оптимизация, автоматизация проектирования микроэлектронной аппаратуры.

Для цитирования: Вершков М.Д., Ягжов А.А., Романов Н.С., Федотова А.А., Знатнов Е.П. Экспериментальное сравнение методов синтеза логических схем. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 133–142 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(4)–10.

1. Introduction

The technology-independent optimization (optimization) of logic circuits is one of the most important steps in the logic synthesis process with a significant impact on the quality of the digital systems being designed.

Different methods of synthesis of logic circuits implementing given Boolean functions (synthesis methods) are widely used in logic optimization approaches. For instance, these methods can be used in optimization that is based on the rewriting that is the replacement of subcircuits of the original logic circuit with other subcircuits that more effectively implement the same functions [1].

There are three main criteria for logic optimization (metrics for evaluating these criteria in a logic net are given in parentheses):

1) power (switching activity);
2) delay (the longest path from input to output);
3) area (the number of logic gates).

The aim of this study was to perform an experimental comparison of logic circuit synthesis methods and to identify the best one for each optimization criterion.

Other published papers also present the results of comparisons between different synthesis methods. In [2], the considered methods are as follows: sum-of-products (SOP) and product-of-sums (POS) two-level expressions, MUX-based expressions, the Quine-McCluskey [3] method and different

types of XOR expressions. The comparison was made for all Boolean functions of 3 and 4 variables and for a thousand random generated functions of 5 variables. The comparison was based on estimating the delay and area of synthesized logic circuits without technology mapping.

In [4], a new approach to bi-decomposition of Boolean functions is described and compared with the best algorithms in logic synthesis tools such as FBDD [5], SIS [6], ABC [7]. Sixteen circuits from the MCNC [8], ISCAS [9] and IWLS [10] benchmark suites were used for the comparison by power, delay and area after the process of technology mapping.

In [11], a comparison was made between the size (number of products in the SOP) of the minimal SOP of a Boolean function and the size of the SOP obtained by Minato-Morreale [12] method. The study was considered for different Boolean functions with the number of variables ranging from 3 to 20.

In this article an experimental comparison of methods of synthesis of logic circuits implementing given Boolean functions was carried out. The following methods were considered: the method of Akers [13], bi-decomposition [14], the methods of cascades [15], Minato-Morreale [12], Reed-Muller [16] and DSD-decomposition [17]. The circuits were synthesized in bases of different logic gates without technology mapping processing. Power, delay and area of the circuits were estimated for the comparison. Boolean functions of size from 4 to 10 variables were considered in the experiment. The choice of a benchmark may have a significant impact on the results of experiments. In this study we decided to generate a set of tests based on information about the frequency of occurrence of NPN-equivalence classes of Boolean functions of four variables.

This paper is organized as follows: Section 2 provides a brief overview of considered methods. Section 3 describes the methodology of the tests carried out. Section 4 presents the results of this study and Section 5 is a conclusion.

2. Method overview

Akers method [13] is used to create majority-based circuits. It is the iterative algorithm that implies the manipulations of the table created from the given truth table. The columns of the converting table correspond to gates arguments. At each step of the algorithm, all sets of triples of the columns are iterated and a truth table of a majority gate is evaluated. Received truth tables are compared with each other, after which the best one is selected according to some heuristics. It is inserted into the table as a column. The algorithm terminates after obtaining the truth table that is equal to the given one.

Bi-decomposition [14] is based on extracting the superposition of two Boolean functions from a single source function:

$$f = \varphi(g_1(z_1), g_2(z_2)), \quad (1)$$

where f is a source function, g_1 and g_2 are Boolean functions with sets of arguments z_1 and z_2 respectively, φ is a given Boolean function of two arguments.

Function φ is typically represented by logic operations such as OR (NOR), AND (NAND), and XOR. Different constraints can be imposed on the arguments of new functions, such as requiring subfunctions to have disjoint support. Alternatively, these arguments can be given before bi-decomposition. In this article the method of heuristic decomposition was considered. This means that the arguments of g_1 and g_2 are not provided, and there is only one constraint for their arguments: the number of arguments must be less than the number of arguments of the source function. Nevertheless, the probability of existence of decomposition of this kind is law, especially for completely specified functions. If decomposition is impossible, when the source function is broken down into two functions, one or both of them have the same number of variables. The circuit synthesis algorithm using bi-decomposition is recursive. The two Boolean functions obtained at each step of recursion are decomposed in the same manner.

The method of cascades [15] is based on the Shannon [18] decomposition, which represents the source function as the sum of two sub-functions:

$$f(x_1, x_2, \dots, x_n) = x_n \cdot f(x_1, x_2, \dots, x_{n-1}, 1) \cup \bar{x}_n \cdot f(x_1, x_2, \dots, x_{n-1}, 0), \quad (2)$$

where f is a source Boolean function, x_i is a i argument of this function. This method is also recursive, and two sub-function are decomposed in the same way.

The Minato-Morreale [12] method is a technique for obtaining an irredundant sum-of-product (ISOP) of Boolean function. An ISOP is a sum-of-products (SOP) of Boolean function that cannot have any literal or product removed without losing equivalence to the source function. The study considered two variations of the method: with and without algebraic factoring. Algebraic factoring is an algorithm that transforms SOP, reducing the number of literals and products. This method does not guarantee obtaining the ISOP of a source Boolean function with the minimum number of products [11].

The Reed-Muller method [16] aims to obtain modulo-2 sums of products for a source Boolean function (polynomial). There are only positive or negative literals for each variable in this polynomial. In this article we considered expressions where all literals are positive (Zhegalkin polynomial [19]).

DSD-decomposition (Disjoint Support Decomposition) [17] is based on decomposition of a source Boolean function into several subfunctions with a disjoint support:

$$f = k(a_1, a_2, \dots, a_n), \quad (3)$$

where f is a source Boolean function, $a_1, a_2, \dots, a_n$ are subfunctions in decomposition, k is a function that links these subfunctions.

In this method the number of subfunctions is equal to or greater than 2 (depends on the source function). AND, OR, XOR or the Prime function (Boolean function of 3 variables or greater that cannot be further decomposed) can be used as k. The method is recursive, new functions are decomposed in the same way.

The bases of synthesized circuits for each considered method are presented in Table 1.

Table 1. Basis of synthesized circuits.

Method	Logic gates
Akers	MAJ, NOT, 0
Bi-decomposition	AND, NOT
Cascades	AND, NOT, 1, 0
Minato-Morreale	AND, NOT
Minato-Morreale with factoring	AND, NOT
DSD-decomposition	AND, XOR, NOT, 1, 0
Reed-Muller	AND, XOR, 1

3. Methodology

The synthesis methods were implemented in C++. The kitty [20] and STACCATO [21] libraries were used for Minato-Morreale and DSD-decomposition methods, respectively. The comparison of circuits synthesized by the methods was carried out without technology mapping processing.

To compare the methods with each other, a set of test cases was written in which a certain number of truth tables of Boolean functions over four to ten arguments were generated with a probability that was calculated based on the information about the frequency of occurrence of NPN-equivalence classes of Boolean functions of four variables. The circuits from the OpenABC-D [22] test set were used for obtaining this statistic. The collection process was organized as follows: all cuts of the size four [1] were iterated at each circuit, then the algorithm identified a NPN-equivalence class of the cut function. As a result, the frequency of occurrence of each class was achieved. To receive a function over more than four arguments a concatenation of the truth tables was used. The truth table

over four variables was obtained after generating a function of NPN-equivalence class according to the received probabilities, then four variables swapping, four input flipping and output flipping with a 50 percent probability were made.

The number of generated truth tables of Boolean functions over a particular number of arguments was 1000. These truth tables were supplied as input arguments to the synthesis methods, which constructed circuits consisting of logic gates.

In the first comparison, the arity of the gates in synthesized circuits was limited to two, therefore the Akers algorithm was not participated. Afterwards, using three-input gates was allowed and all the algorithms took part in the second comparison.

The following characteristics were used to compare circuits: the number of function arguments, area, delay, and switching activity. Additionally, the runtime of the methods was also taken into account. The switching activity of a circuit was calculated as the sum of the switching probabilities of all its gates:

$$Z = \sum_{i=0}^{n} \frac{s_i}{t}, \quad (4)$$

where Z is the switching activity of the logic circuit, s_i is the number of cell switches (from 1 to 0 and from 0 to 1), i is the cell index of the logic circuit, t is the number of simulations. In the experiment, the number of simulations of each logic circuit was 1024.

The implementations of Akers and bi-decomposition methods were tested on truth tables of Boolean functions over four to seven and four to eight arguments, respectively. The reason is the high asymptotic complexity of these methods [23, 14].

4. Results

The achieved results of the methods comparison are described in Tables 2 and 3. Table 2 illustrates the information about the runtime of the methods and characteristics of the synthesized circuits composed of two-input gates only, whereas Table 3 shows the same but for circuits that may include three-input gates. The cells of the tables contain the averages of the results obtained.

The part of Table 2, which corresponds to the statistics about switching activity of synthesized circuits, reveals that the leader in optimizing this parameter for functions over four variables is DSD-decomposition method, whereas for functions over five and six arguments is Minato-Morreale method with factoring and for functions over seven to ten arguments is the original Minato-Morreale method. Bi-decomposition method comes fourth in terms of power optimization for functions over four and five arguments, comes third for functions over six arguments, for the other functions this method reduces its effectiveness and exhibits the worst results. The method of cascades is the second from the end for optimization of functions over four to eight arguments and the worst one for others. Reed-Muller method takes the last place in optimization of functions over four to six variables but improves its position with an increase in the number of variables and becomes the third by functions over eight arguments.

According to Table 2, Minato-Morreale method optimizes the delay of circuits containing from five to ten inputs better than the other methods. This method is followed by Reed-Muller algorithm and then Minato-Morreale algorithm with factoring. DSD-decomposition method is the best for optimizing functions over four arguments but comes fourth in terms of delay optimization for the other number of arguments. Bi-decomposition and cascades methods are the least preferred for optimizing circuit delay.

The part of Table 2 Area illustrates that the leader for reducing the number of logic gates for circuits from five to nine inputs is Minato-Morreale method with factoring. DSD-decomposition is the first in terms of this optimization for functions over four and ten arguments, the method only the second and the third for functions over eight and nine arguments and over five and six arguments respectively. Minato-Morreale method without factoring comes second for functions over five and six arguments. Bi-decomposition and Reed-Muller methods are the worst in terms of optimizing circuit area.

The fastest algorithm is the method of cascades, while Minato-Morreale methods require a little more time for execution, these algorithms have a comparable runtime. DSD-decomposition and Reed-Muller methods are moderately slower than the leader. The most time consuming method is bi-decomposition.

Table 3 demonstrates the similar ranking to Table 2 in terms of switching activity optimization with the exception of Minato-Morreale method without factoring, which works more efficiently using three-input logic gates than a similar one with factoring. Akers method becomes consistently second, shifting the previous rating.

The Delay of Table 3 reveals that Akers method is not preferred for delay optimization, despite the fact that it becomes second for functions over five and six arguments. Minato-Morreale method without factoring is the absolute leader for optimization of circuit delay with allowing using tree-input gates.

A slightly different result in contrast to Table 2 is achieved in Area optimization. Akers algorithm comes first for functions over five to seven arguments and the original Minato-Morreale becomes second for functions over eight to ten arguments. Otherwise, the results are similar to those obtained in Table 2.

The runtime of the algorithms in Table 3 does not reveal any new data except that Akers algorithm is the longest-running method.

Table 2. Experiment results for circuits containing two-input gates only.

Arguments number	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Switching activity							
Bi-decomposition	4.54	8.71	17.66	35.36	70.31		
Method of cascades	5.11	9.78	18.57	34.06	61.29	109.68	197.85
Minato-Morreale	4.35	7.52	12.88	**20.89**	**33.11**	**49.41**	**72.61**
Minato-Morreale with factoring	4	**7.16**	**12.73**	22.63	41.74	75.22	134.7
DSD-decomposition	**3.69**	9.18	18.25	33.57	59.8	106.14	189.89
Reed-Muller	6.62	12.13	20.69	33.99	54.71	86.72	136.4
Delay							
Bi-decomposition	3.7	7.61	15.47	31.45	65.9		
Method of cascades	4.89	7.7	9.96	12	14	16	18
Minato-Morreale	3.11	**4.87**	**6.51**	**7.76**	**8.89**	**10.16**	**11.92**
Minato-Morreale with factoring	3.11	5.64	7.97	10.4	12.84	15.27	17.73
DSD-decomposition	**2.82**	6.84	9.71	11.93	14	16	18
Reed-Muller	4.38	6.11	7.77	9.03	10.06	11.26	12.96
Area							
Bi-decomposition	5.92	16.5	42.81	102.69	237.23		
Method of cascades	7.26	17.79	38.44	75.42	140.7	256.73	468.3
Minato-Morreale	5.97	15.52	35.91	75.78	155.87	301.82	574.24
Minato-Morreale with factoring	4.23	**11.26**	**25.37**	**53.92**	**114.1**	**231.55**	463.79
DSD-decomposition	**3.43**	15.88	36.77	77.45	134.61	244.54	**444.29**
Reed-Muller	12.8	30.26	64.77	134.16	272.82	548.99	1096.01
Runtime (ms)							
Bi-decomposition	4.07	7.41	16.39	44.22	167.53		
Method of cascades	2.65	**2.91**	**3.56**	**5.22**	**9.44**	**21.75**	**62.09**
Minato-Morreale	2.69	3.02	3.73	5.52	9.85	22.71	63.07
Minato-Morreale with factoring	2.65	3.01	3.68	5.47	9.94	22.85	63.55
DSD-decomposition	7.2	8.06	9.04	11.26	16.39	30.69	74.56
Reed-Muller	**2.64**	3.05	4.16	7.42	16.54	44.13	127.76

Table 3. Medical activities of multimodal human-computer interactions.

Arguments number	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Switching activity							
Akers	3.81	6.2	10.41	21.07			
Bi-decomposition	4.42	8.32	15.95	28.87	58.87		
Method of cascades	4.81	9.37	18.2	33.65	60.85	109.1	197.04
Minato-Morreale	3.57	**5.23**	**7.83**	**11.98**	**19.3**	**29.96**	**45.14**
Minato-Morreale with factoring	3.81	6.64	11.65	20.04	37.53	68.05	123.81
DSD-decomposition	**3.46**	9.17	18.25	33.57	59.8	106.14	189.89
Reed-Muller	5.43	9.39	15.56	25.5	40.87	64.69	101.25
Delay							
Akers	3.32	5.06	7.57	13.61			
Bi-decomposition	3.6	7.44	15.36	31.35	65.79		
Method of cascades	4.58	7.53	9.93	12	14	16	18
Minato-Morreale	**2.29**	**3.7**	**4.65**	**5.22**	**6.32**	**7.49**	**8**
Minato-Morreale with factoring	2.89	5.34	7.72	10.1	12.54	14.98	17.44
DSD-decomposition	2.56	6.83	9.71	11.93	14	16	18
Reed-Muller	3.26	4.57	5.57	6.28	7	8.18	9
Area							
Akers	4.67	**9.18**	**17.29**	**38.93**			
Bi-decomposition	5.66	15.55	38.62	90.02	213.62		
Method of cascades	6.57	16.77	37.56	74.41	139.63	255.25	466.14
Minato-Morreale	4.09	9.74	22.39	48.57	**102.94**	**206.14**	**409.41**
Minato-Morreale with factoring	3.78	10.02	22.8	48.74	104.58	215.87	441.36
DSD-decomposition	**2.92**	15.86	36.77	72.45	134.61	244.54	444.29
Reed-Muller	9.38	21.7	46.96	98.86	201.3	405.41	808.42
Runtime (ms)							
Akers	3.35	6.25	14.75	82.48			
Bi-decomposition	4	7.28	16.14	43.62	165.79		
Method of cascades	2.57	**2.83**	**3.46**	**5.06**	**9.2**	**21.1**	60.8
Minato-Morreale	2.6	2.9	3.55	5.2	9.28	21.49	**60.59**
Minato-Morreale with factoring	2.58	2.94	3.6	5.29	9.62	22.19	62.21
DSD-decomposition	7.3	7.8	8.7	10.75	15.63	29.31	71.96
Reed-Muller	**2.55**	2.92	3.96	7.05	15.92	43.48	127.38

5. Conclusion

The results of the work of seven methods for the synthesis of logic circuits implementing specified Boolean functions have been analyzed. The comparison of the obtained circuits has been carried out according to the three main criteria: the number of logic gates (area), depth (delay) and switching activity (power). The analysis of the results has shown that DSD-decomposition (only for functions over four arguments) and the both Minato-Morreale methods are the best choice for power optimization. The similar results have been demonstrated for delay optimization of logic circuits. Area optimization with a restriction on the arity of logic gates equal to two is better to carry out using DSD-decomposition (only for functions over four and ten arguments) and Minato-Morreale method with factoring. Akers algorithm is the leader in optimization of a number of three-input gates in circuits from five to seven inputs and for other circuits, the leader is Minato-Morreale algorithm without factoring. Minato-Morreale and cascades methods have demonstrated a compatible and minimal runtime, whereas Akers and bi-decomposition have shown the worst performance.

References

[1]. Riener H., Mishchenko A., Soeken M. Exact DAG-aware rewriting. In Proc. of Design, Automation and Test in Europe Conference and Exhibition, 2020, pp. 732–737.
[2]. Ammes G., Lau W., Ribas R. P. (2018) Comparative analysis of different Boolean function synthesis Methods. In Proc. of Microelectronics Students Forum. Available at: https://sbmicro.org.br/eventos/sforum/volume-18, accessed 01.07.2024.
[3]. McCluskey E. J. Minimization of Boolean functions. The Bell System Technical Journal, vol. 35, no. 6, 1956, pp. 1417–1444.
[4]. Choudhury M., Mohanram, K. Bi-decomposition of large Boolean functions using blocking edge graphs. In Proc. of the International Conference on Computer-Aided Design, 2010, pp. 586–591.
[5]. Wu D., Zhu J. FBDD: A folded logic synthesis system. In Proc. of International Conference on ASIC, 2005, pp. 746–751.
[6]. Singh E. M. S. K. J., Murgai L. L. C. M. R., Sangiovanni-Vincentelli R. K. B. A. SIS: A system for sequential circuit synthesis. University of California, Berkeley, vol. 94720, 1992, p. 4.
[7]. ABC. Available at: https://github.com/berkeley-abc/abc, accessed 01.07.2024.
[8]. Yang S. Logic synthesis and optimization benchmarks user guide: version 3.0. Research Triangle Park, NC, USA: Microelectronics Center of North Carolina (MCNC), 1991, pp. 502-508.
[9]. Brglez F., Bryan D., Kozminski K. Combinational Profiles of Sequential Benchmark Circuits. In Proc. of the International Symposium of Circuits and Systems, 1989, pp. 1929–1934.
[10]. Albrecht. C. IWLS 2005 Benchmarks. 2005. Available at: http://iwls.org/iwls2005/benchmarks.html, accessed 01.07.2024.
[11]. Sasao T., Butler J. T. Worst and best irredundant sum-of-products expressions. IEEE Transactions on Computers, vol. 50, no. 9, 2001, pp. 935–948.
[12]. Minato S. Fast generation of prime-irredundant covers from binary decision diagrams. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E76-A, no. 6, 1993, pp. 967-973.
[13]. Akers B. Synthesis of combinational logic using three-input majority gates. In Proc. of 3rd Annual Symposium on Switching Circuit Theory and Logical Design, 1962, pp. 149–157.
[14]. Pottosin Y. V. Synthesis of combinational circuits by means of bi-decomposition of Boolean functions. Informatics, vol. 19, no. 1, 2022, pp. 7–18.
[15]. Povarov G. N. A method for synthesis of computing and controlling contact circuits. Automatika i Telemekhanika, vol. 18, no. 2, 1957, pp. 145–162 (in Russian).
[16]. Harking B. Efficient algorithm for canonical Reed-Muller expansions of Boolean functions. IEE Proceedings E (Computers and Digital Techniques), vol. 137, no. 5, 1990, pp. 366–370.
[17]. Bertacco V. Scalable Hardware Verification with Symbolic Simulation. Springer Science & Business Media, 2006.
[18]. Shannon C. E. The synthesis of two-terminal switching circuits. The Bell System Technical Journal, vol. 28, no. 1, 1949, pp. 59–98.
[19]. Zhegalkin I. I. On the technique of calculating propositions in symbolic logic. Matematicheskii Sbornik, vol. 34, no. 1, 1927, pp. 9-28 (in Russian and French).
[20]. kitty. Available at: https://github.com/msoeken/kitty, accessed 01.07.2024.
[21]. STACCATO. Available at: https://web.eecs.umich.edu/staccato/, accessed 01.07.2024.
[22]. OpenABC-D. Available at: https://github.com/NYU-MLDA/OpenABC, accessed 01.07.2024.
[23]. Lee S.-Y., Riener H., De Micheli G. Logic resynthesis of majoritybased circuits by top-down decomposition. In Proc. of 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, 2021, pp. 105–110.

Информация об авторах / Information about authors

Максим Дмитриевич ВЕРШКОВ – студент 5-го курса Московского института электроники и математики им. А.Н. Тихонова Национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: оптимизация цифровых СБИС.

Maxim Dmitrievich VERSHKOV is a 5th year student of the A.N. Tikhonov Moscow Institute of Electronics and Mathematics of the National Research University Higher School of Economics. Research interests: optimization of digital VLSI.

Алексей Александрович ЯГЖОВ – студент 4-го курса Московского института электроники и математики им. А.Н. Тихонова Национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: оптимизация цифровых СБИС.

Alexey Aleksandrovich YAGZHOV is a 4th year student of the A.N. Tikhonov Moscow Institute of Electronics and Mathematics of the National Research University Higher School of Economics. Research interests: optimization of digital VLSI.

Никита Сергеевич РОМАНОВ – студент 5-го курса Московского института электроники и математики им. А.Н. Тихонова Национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: оптимизация цифровых СБИС.

Nikita Sergeevitch ROMANOV is a 5th year student of the A.N. Tikhonov Moscow Institute of Electronics and Mathematics of the National Research University Higher School of Economics. Research interests: optimization of digital VLSI.

Анна Алексеевна ФЕДОТОВА – студентка 3 курса Факультета компьютерных наук Национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: оптимизация цифровых СБИС.

Anna Alekseevna FEDOTOVA – is a 3th year student of the Faculty of Computer Science of the National Research University Higher School of Economics. Research interests: optimization of digital VLSI.

Егор Павлович ЗНАТНОВ – студент 3 курса Факультета компьютерных наук Национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: оптимизация цифровых СБИС.

Egor Pavlovich ZNATNOV – is a 3th year student of the Faculty of Computer Science of the National Research University Higher School of Economics. Research interests: optimization of digital VLSI.LSI.

DOI: 10.15514/ISPRAS-2024-36(4)-11

Generation of Spatial Time Series Data

[1] A.M. Kropacheva, ORCID: 0009-0002-7703-4762 <st086907@student.spbu.ru>
[1] D.V. Girdyuk, ORCID: 0000-0003-4679-7281 <d.girdyuk@spbu.ru>
[2] I.L. Iov, ORCID: 0000-0002-6140-9382 <illariov1809@gmail.com>
[1] A.Y. Pershin, ORCID: 0000-0003-2108-1906 <a.pershin@spbu.ru>

[1] St. Petersburg State University,
7/9, University Embankment, St. Petersburg, 199034, Russia.
[2] ITMO University,
49A, Kronverksky pr., St. Petersburg, 197101, Russia.

Abstract. In the era of deep learning, global-local deep neural networks are gradually replacing statistical approaches for time-series forecasting, especially for the spatiotemporal modeling field. However, the development of such methods is hindered by the lack of open benchmark datasets in this research domain. Generating synthetic data is an alternative solution to data collection, but prior works focus mainly on generating uncorrelated independent time series. In this work, we present a method for spatially correlated time-series generation. It uses a set of parametric autoregressive models for univariate time series generation in combination with the approach for sampling model parameters which allows one to simulate spatial relationships. We describe its implementation and conduct experiments showing the validity of the data for spatiotemporal modeling.

Keywords: time-series generation; spatiotemporal modeling; autoregressive model.

For citation: Kropacheva A.M., Girdyuk D.V., Iov I.L., Pershin A.Y. Generation of spatial time series data. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024. pp. 143-154. DOI: 10.15514/ISPRAS-2024-36(4)-11.

Acknowledgements. The research was carried out within the state assignment of Ministry of Science and Higher Education of the Russian Federation (project No. FSER-2024-0004).

Генерация временных рядов с пространственными взаимосвязями

[1] А.М. Кропачева, ORCID: 0009-0002-7703-4762 <st086907@student.spbu.ru>
[1] Д.В. Гирдюк, ORCID: 0000-0003-4679-7281 <d.girdyuk@spbu.ru>
[2] И.Л. Иов, ORCID: 0000-0002-6140-9382 <illariov1809@gmail.com>
[1] А.Ю. Першин, ORCID: 0000-0003-2108-1906 <a.pershin@spbu.ru>

[1] Санкт-Петербургский государственный университет,
Россия, 199034, г. Санкт-Петербург, Университетская наб., д. 7-9.
[2] Университет ИТМО,
Россия, 197101, г. Санкт-Петербург, д. 49, лит. А.

Аннотация. В эпоху глубокого обучения нейронные сети постепенно заменяют статистические подходы к прогнозированию временных рядов в различных областях, например, в сфере пространственно-временного моделирования. Однако недостаток открытых наборов данных в этой области препятствует развитию нейросетевых методов. Альтернативным решением для сбора данных

является генерация синтетических данных, но существующие методы фокусируются только на генерации некоррелированных независимых временных рядов. В этой работе мы представляем метод генерации временных рядов с пространственной корреляцией. Он использует набор параметризуемых авторегрессионных моделей для генерации одномерных временных рядов в сочетании с подходом к выбору параметров модели, что позволяет моделировать пространственные взаимоотношения. В работе приведена реализация метода и результаты экспериментов, которые показывают применимость данных для пространственно-временного моделирования.

Ключевые слова: генерация временных рядов; пространственно-временное моделирование; авторегрессионная модель.

Для цитирования: Кропачева А.М., Гирдюк Д.В., Иов И.Л., Першин А.Ю. Генерация временных рядов с пространственными взаимосвязями. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 143–154 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(4)–11.

Благодарности. Работа выполнена в рамках государственного задания Министерства науки и высшего образования Российской Федерации (проект № FSER-2024-0004).

1. Introduction

Time series are widely used in numerous domains, including forecasting financial data, examining road traffic patterns, predicting weather conditions, identifying anomalies in the network traffic, etc. [1]. In recent years, many time series datasets have been published in open source. Competitions focused on the time series forecasting became popular [2-3]. There is also a growing interest in tasks associated with time series, such as hierarchical forecasting [4] and spatiotemporal modeling [5].

However, there are several problems in the time series analysis field. Time series presented in many open datasets are quite homogeneous. Their application for evaluating the effectiveness of predictive models is limited [6]. Additionally, it is often difficult (or even impossible) to find open datasets for certain tasks [1]. Therefore, methods for generating time series can help create additional data not only to improve the training process of currently popular global neural network models for time series forecasting but also to create benchmark datasets suitable for their comprehensive testing. Examples of such models include both global models, like Informer, FEDformer, TFT, as well as foundation models like TimeGPT, TimesNet, and TimeLLM (see reviews [7-8]).

Methods for time series generation can be divided into two groups: statistical and neural network-based [9]. In the first group, the autoregressive approach is a popular choice. In the second group, methods are often based on the application of generative adversarial networks (GANs) or variational autoencoders (VAEs). Thus, the methods of the first group use parameterized autoregressive models to generate time series. Methods of the second group use neural network models that are trained to generate time series from random noise based on existing datasets. The first approach is implemented in such tools as GRATIS [6], timeseries-generator [10], and mockseries [11]. The second is widely used in fields with many real observations, e.g., economics and finance [12].

Nevertheless, the overwhelming majority of existing solutions lack support for generating correlated time series. These time series are often observed in tasks involving spatiotemporal dependencies between time series and the objects that generate them. For example, in road traffic intensity forecasting, sensors installed on roads measure the number of passing vehicles and/or their average speed. The corresponding time series between adjacent crossroads often exhibit strong correlation, which can be used to improve the forecasting quality, for example, using spatiotemporal graph neural networks (STGNN) [13]. The emergence of open traffic datasets, such as PEMS-BAY and METR-LA [5], has allowed researchers to standardize the way they compare their models. This has significantly accelerated progress in the development of modeling spatiotemporal relationships. However, similar types of relationships are encountered in many other areas, such as telecommunications, neurobiology, epidemiology, meteorology, and others. Yet, at the current moment, in many fields, either open datasets are completely absent, or they are insignificant in size. This fact once again confirms the necessity of developing methods for time series generation.

Moreover, time series analysis is not limited to prediction alone. Different tasks such as anomaly detection in observations, searching for causal relationships, and adaptive prediction of time series that allow changes in value distributions currently attract significant interest.

This paper presents a method for generating time series with spatial dependencies, generated by a set of objects (e.g., sensors on roads or cellular base stations). Time series are generated using a set of classical parameterized autoregressive models, such as ETS [14]. Spatiotemporal relationships are modeled by sampling process parameters. The proximity of objects, determined e.g. by Euclidean or geodesic distances on some manifold in parameter space corresponds to the similarity of autoregressive models. The closer the process parameters are, the more similar the time series are to each other.

The paper is structured as follows. Section 2 describes existing approaches to time series generation. In Section 3, the method for generating time series with spatial dependencies is presented. The implementation of the method is described in Section 4. Section 5 contains an evaluation of the generator's results. In the conclusion, the results of the work and further research plans are outlined.

2. Related Works

Methods for time series generation can be divided into the following groups: statistical (autoregressive) and neural network-based. An overview of modern neural network approaches to time series generation is presented in the work [9]. The method described in this paper belongs to autoregressive approaches. In this section, we review several existing methods and open-source projects that fall into this group.

The method for generating time series with diverse controlled characteristics GRATIS [6] utilizes Gaussian mixture autoregressive (MAR) models. Tuning the parameters of these models allows for the generation of a time series with desired features. GRATIS generates realistic datasets necessary for research tasks such as forecasting comparison, model training on generated data, and others. Thus, the method provides an efficient benchmarking tool but does not consider the generation of a set of spatially correlated time series.

In [15], a method is proposed to generate synthetic time series for simulation, control, and optimization tasks for hybrid energy systems. Fourier series and ARMA models were trained on real measurements, and then employed to generate time series. In addition to generating independent time series, the method offers a way to obtain synthetic correlated time series based on correlated input data. However, this method only partially meets the goals of the work. The correlation of time series is fully determined by the training data and applied to the entire resulting system. However, the concept of spatial correlation only implies correlation between those time series generated by similar objects.

In addition to the methods described above, there are several open-source tools for time series generation. A lot of them are Python libraries for time series generation based on various stochastic processes. A process here is defined as some function that describes the behaviour of series values over time. However, these solutions do not support specifying spatial dependencies. Thus, such methods are only of interest because of the implemented processes, generation methods, and solution design. This paper considers two such packages: timeseries-generator [10] and mockseries [11].

The Python package timeseries-generator is widely known, providing the ability to generate synthetic time series. The authors propose the following implementation. A time series is represented as $ts = v_0 * f_1 * f_2 * ... * f_N + u$, where v_0 is the initial value, $f_1, ..., f_N$ are integer factors reflecting the characteristics of the time series (trend, seasonality, and others), and u is random noise. This tool supports specifying time series features such as trend and seasonality and allows users to simulate changes in observations related to real events (weekends, holidays, and others). Nevertheless, this package does not support the usage of an external dataset to parameterize stochastic processes and uses only one user-defined model for time series generation. Thus, the

timeseries-generator package contains effective methods for generating time series but does not meet all the requirements of our study.

Another Python package providing flexible capabilities for creating time series is mockseries. It has more components than timeseries-generator and represents time series as an additive or multiplicative combination of various signals (trend, seasonality, noise, and several others). For example, $y_t = \big(S_1(t) * T_l(t) + S_2(t)\big) * u$, where $S_1(t)$, $S_2(t)$ are different seasonal components, $T_l(t)$ is a linear trend, and u is random noise [16]. Moreover, the package implements methods for changing time series values at specified time points or over a specified time interval. Such a method allows users to simulate anomalies and change time series values over a specified interval according to a certain function. Thus, using the mockseries tool, a set of time series with characteristics changing over time can be generated.

In conclusion, there appear to be no methods that can generate a set of time series with spatial dependencies. There might be no suitable software tools to address the issue.

3. Generation Method

The proposed method for generating time series consists of the following components and ideas:

- a set of stochastic processes supporting parameterization, such as ETS (error, trend, seasonal) [14]) family;
- generation of a schedule, i.e., a mechanism for creating complex time series models;
- generation of points with a clustered division on the sphere as a time series source;
- process parameterization method.

The generation scheme is shown in Fig. 1.

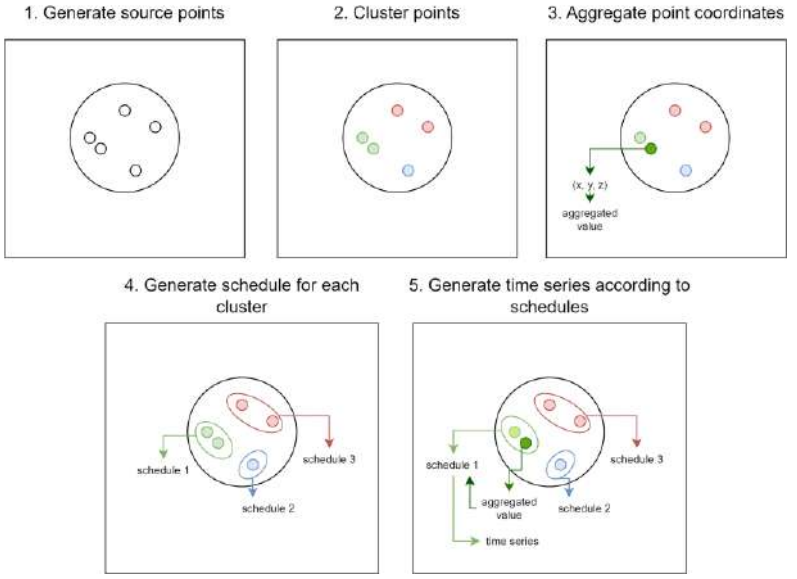


Fig. 1. Time series generation scheme.

3.1 Stochastic Processes

Stochastic processes are the functions that determine the behavior of a time series and include a component of random error. They describe the growth, decline, stationarity, seasonality, dispersion, and other characteristics of the time series. Processes allow for the computation of a new value in

the series based on several previous ones. Time series created by the same sequence of processes are expected to be similar.

In our method, the process is represented as follows: $p = f(X_t, X_{t-1}, \dots, X_{t-l}, \xi; \theta)$, where $X_t, X_{t-1}, \dots, X_{t-l}$ are the set of previous values in the series, θ is the parameter vector, and ξ is the random error component [17]. Thus, the time series becomes a sequence of values from a process list. At the start of each time series generation, the empty sequence is filled with a small set of random initial data. This set size is equal to the number of values that the first process requires. Processes then use the required number of previous values and add their generated data to the sequence.

The method uses ETS models (Error, Trend and Seasonality) to ensure the generated time series resembles real data. In the implementation of the time series generator, all ETS models are additive, represented as a sum of several components:

$$ts = a \cdot L(t) + b \cdot T(t) + c \cdot S_l(t) + d \cdot U(t),$$

where $L(t)$ is the long-term component, $T(t)$ is the trend, $S_l(t)$ is the seasonal component with frequency l, and $U(t)$ is the random error [14]. Parameters $a, b, c, d \in N$ denote the number of components of the corresponding type, which the user specifies.

3.2 Time Series Schedule Generator

A schedule is an order of processes, each associated with a sequence of parameters. The order of processes is a sequence of pairs $(process_i, steps_i)$, where $process_i$ is a randomly selected process from the list of available process types, and $steps_i \in N$ is the number of time steps allocated for the process. Let k specify the number of pairs in the sequence, and n be the total number of observations in the time series. Then we have $\sum_{i=1}^{k} steps_i = n$.

For each pair $(process_i, steps_i)$, we generate another sequence of pairs $(steps_{ij}, parameters_{ij})$, where $parameters_{ij}$ is the set of parameters of the process, and $steps_{ij}$ is the number of steps allocated to the process to work with this set of parameters. If q determines the number of pairs in the parameter sequence, then we have $\sum_{j=1}^{q} steps_{ij} = steps_i$.

3.3 Source Data Generation

In previous sections, we described the generation of uncorrelated time series. To update the method with spatial correlations, we first need to obtain a set of initial parameter sets for the generator, each representing an object. Sensors installed on roads may be treated as objects, the physical location of each acting as a parameter, as described in Section 1. The closer the sensors are placed, the more similar their time series will be. Using the spatial relationships between objects, one can unite the objects into a graph based on Euclidean distances. In the current implementation, points from the space R^3 are used as objects, and their coordinates naturally serve as parameters of the points.

Points are sampled and clustered with the k-means method on the unit sphere centered at zero. The cluster structure of objects allows for the parameterization of a complex time series model. For each cluster, a schedule is generated, and all objects generate time series using the schedule of the parent cluster. In our method, the parent cluster fully defines the schedule parameters for each object.

3.4 Process Parameterization

We propose a parameterization method to allow the time series to depend on the characteristics of the parent object. If the characteristics of objects are similar, a time series will be generated from sets of processes with close parameters. One challenge of parameterization is the transformation of characteristics into process parameters: the dimensionality may vary, specific limitations exist for many parameters, etc. This issue can be addressed with the following approaches. One is to treat characteristics as parameters in order and adjust them to the required range. If the number of

parameters is not enough, use additional aggregated values. Another way is to aggregate all the characteristics and use the resulting value to generate all process parameters. This approach is used in the current implementation.

Time series generated from the parameters chosen randomly may be unstable. The user needs to specify the following hyperparameters: the range $[a, b]$ and the number of intervals m. Additionally, the method can use generated source data to calculate these parameters. Suppose the input data is the set of points in space R^3. In that case, the parameter a is considered a minimum value over all coordinates, and the parameter b is a maximum. Process parameters and initial values of the time series are generated concerning these constraints.

The main aggregation function used is the weighted arithmetic mean with a sum of weights equal to 1. The aggregation function considers the order of coordinates and allows for smoothing of the values. Other aggregation functions are the sum, maximum and minimum of all values, allowing one to parameterize the processes with up to 4 parameters. The set of aggregation functions may be extended with the skewness, kurtosis, quantiles and other parameters. However, high-dimensional parameter spaces are either unreachable, or all the parameters will be similar, as the results or aggregation functions are correlated.

Let $A = (a_1, a_2, a_3) \in R^3$ be a point, a_{mw} is the weighted arithmetic mean of A, a_m is the maximum value of A. Consider the following examples of parameter calculations:

- standard deviation,
- coefficient of the time series level,
- trend coefficient,
- seasonality coefficient.

Consider the *standard deviation* being a parameter for all processes. Let's denote it as σ. Let $s = \frac{|b-a|}{m}$, where a, b, m are hyperparameters described above. In the current implementation, the formula for calculating the standard deviation looks as follows: $\sigma = s' + k$, where $k \sim N(0, \frac{s}{2})$. If no source data is passed to the process, then $s' = s$, otherwise $s' = s * (1 + \frac{a_{mw}}{a_m})$.

The *time series level* is a separate time series component, i.e., it does not depend on the trend and seasonality. It is present in the simple exponential smoothing and Holt/Holt-Winters models. The smoothing parameter $\alpha \in [0,1]$ of the level determines the weight of the last points in calculating the new value of the time series. In the current implementation, α belongs to the interval $(0,0.3)$ so generated time series using the simple exponential smoothing model are stationary [17].

The *trend* component $(T(t))$ is present in the Holt and Holt-Winters models. The trend coefficient $\beta \in [0,1]$ is the smoothing parameter responsible for the extent to which the time series exhibits growth or decline. If no source data is passed to the process, then $\beta \sim U[0,0.05]$, otherwise $\beta = \frac{a_{mw}}{20a_m}$.

The *seasonal* component $(S(t))$ is used in the Holt-Winters model. Similarly, to the trend, it has a coefficient $\gamma \in [0,1]$, a smoothing parameter determining the impact of seasonal patterns on the generation [18]. An empirical rule for γ is $\gamma \in [0.5, 1]$, as with $\gamma < 0.5$, the seasonal component of the time series becomes barely discernible. If no source data is passed to the process, then $\gamma \sim U[0.5, 1]$, otherwise $\gamma = 1 - \frac{a_{mw}}{2a_m}$.

4. Implementation

The implementation of the application was divided into two parts: creating a time series generator that supports complex user-defined or random models, and incorporating parameterization of stochastic processes that generate time series based on input data. The scenario of time series generation looks as follows.

1) Setting hyperparameters by the user, initializing the generator.
2) Creating a schedule for all-time series, either collectively or for each individually (depending on the configured parameters).
3) Generating time series according to the schedule.
4) Adding time series to the resulting list.

To implement the time series simulator, the following technologies were used:

- Python 3.10 [19];
- NumPy [20];
- matplotlib [21];
- scikit-learn [22].

The class diagram is shown in Appendix A. The implementation code is available in the repository on GitHub [23].

5. Evaluation

To assess the quality of the generator's performance, experiments were conducted on sampling points on the sphere and generating corresponding time series. They were conveyed in the following environment:

- CPU: AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx 2.30 GHz
- RAM: 16GB
- OS: Windows 10 (64-bit)

The experiments consisted of generating 5 points on a sphere and 5 corresponding time series with 100 observations. The average time of a single generation is about 1-2 seconds. Peak CPU load is about 40-50%, and memory consumption is about 100MB.

5.1 Metrics

The quality criterion for the time series generator is the similarity of the time series generated by close objects. There is an algorithm to find optimal matches between time sequences — Dynamic Time Warping (DTW). It is effective when comparing time series, one of which is shifted, compressed, or stretched along the time axis relative to the other. However, such an algorithm is not suitable for this implementation of the generator. The resulting time series may differ in dynamics and trend direction, even if they are based on the same model. Thus, quality assessment is carried out using visual comparison.

5.2 Results

In Fig. 2, a plot of the time series returned by the generator is presented on the left, and the location of the points on the sphere that generated the time series is shown on the right. Points belonging to the same cluster, along with their corresponding time series, are marked with the same color.

The time series generated by the same schedule (cluster) are distinguishable, i.e., they have similar dispersion and common segments where the characteristics of the processes and the stochastic processes themselves change. The closer the points within the cluster, the closer their initial values and the generated time series: this is reflected in the plots where three time series from the same cluster are presented – two series are similar, and the third differs from the pair but still preserves a common behavioral model.

Different objects generate time series from their own clusters, so their behavior is weakly correlated. Nevertheless, the similarity of time series considering the coordinates of points varies with each

generation. In some datasets, similar time series appear regardless of the proximity of their parent objects. Thus, the generation method has shown promise, but further testing is required.

One of the proposed applications of the generation method is to mock the real data and use artificial data to pretrain models for time series forecasting, i.e. data augmentation. The technique is currently being used for experiments with existing datasets with spatiotemporal correlations in data, such as PEMS-BAY [24] and others (see section 1). We create custom complex processes to mock the origin data better, then train the time-series forecast models on the synthetic data. We expect that after fine-tuning on the origin data models will outperform their basic versions which were not trained on the synthetic data. An example of a custom process is shown in Appendix B.

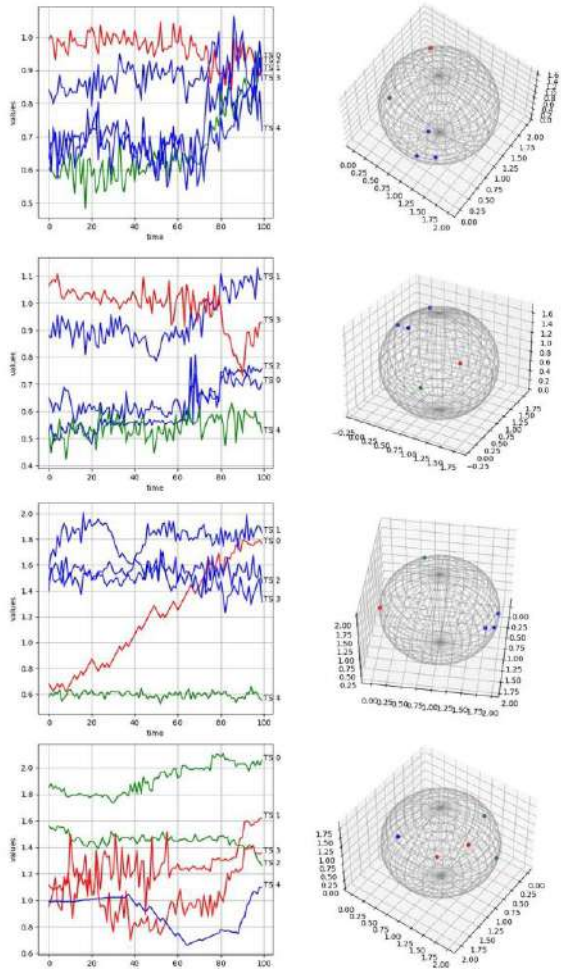


Fig. 2. Results of the time series generator.

Conclusion and Future work

Within the scope of this work, an approach to generating time series with spatial dependencies was presented. The following results have been achieved.

1) A review of existing methods and software implementations of time series generators based on autoregressive processes was conducted: GRATIS, Correlated synthetic time series generation, timeseries-generator, mockseries.
2) A method for generating time series was developed, including:

a) a set of stochastic processes generating time series;
b) a method for constructing complex models for time series generation;
c) a method for generating and clustering points on a sphere;
d) a method for generating time series parameters depending on input data.

3) The method was implemented in software.
4) The generator's performance has been tested.

In the further development of the method, the following tasks are set: to implement various methods for sampling points on arbitrary surfaces, to implement functions for constructing a graph of relationships between objects and approximating geodesic distances on surfaces, to describe abstractions for changing process parameters, and to conduct testing of new functions of the time series generator.

References

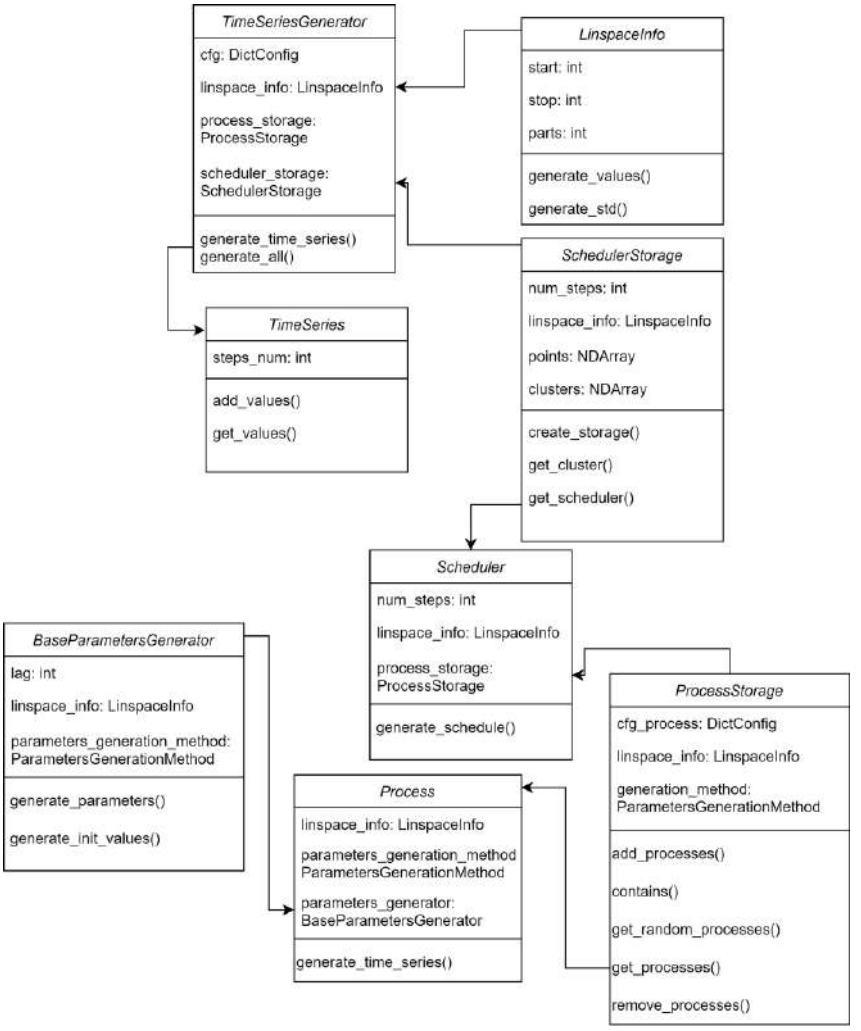
[1]. Y. Hahn, T. Langer, R. Meyes, and T. Meisen, "Time series dataset survey for forecasting with deep learning," Forecasting, vol. 5, no. 1, pp. 315–335, 2023, Available at: https://www.mdpi.com/25719394/5/1/17, accessed 02.04.2024.
[2]. S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," International Journal of Forecasting, vol. 36, no. 1, pp. 54–74, 2020, m4 Competition.
[3]. S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m5 competition: Background, organization, and implementation," International Journal of Forecasting, vol. 38, no. 4, pp. 1325–1336, 2022, special Issue: M5 competition.
[4]. M. Abolghasemi, G. Tarr, and C. Bergmeir, "Machine learning applications in hierarchical time series forecasting: Investigating the impact of promotions," International Journal of Forecasting, vol. 40, no. 2, pp. 597–615, 2024.
[5]. Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Graph convolutional recurrent neural network: Data-driven traffic forecasting," ArXiv, vol. abs/1707.01926, 2017, Available at: https://api.semanticscholar.org/CorpusID:195346050, , accessed 02.04.2024.
[6]. Y. Kang, R. J. Hyndman, and F. Li, "Gratis: Generating time series with diverse and controllable characteristics," 2019.
[7]. Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," arXiv preprint arXiv:2202.07125, 2022.
[8]. J. A. Miller, M. Aldosari, F. Saeed, N. H. Barna, S. Rana, I. B. Arpinar, and N. Liu, "A survey of deep learning and foundation models for time series forecasting," arXiv preprint arXiv:2401.13912, 2024.
[9]. S. Wang, Y. Du, X. Guo, B. Pan, Z. Qin, and L. Zhao, "Controllable data generation by deep learning: A review," ACM Comput. Surv., mar 2024, Available at: https://doi.org/10.1145/3648609, accessed 02.04.2024.
[10]. "timeseries-generator", Available at: https://github.com/NikeInc/timeseries-generator, accessed 18.05.2024.
[11]. "mockseries", Available at: https://github.com/cyrilou242/mockseries, accessed 18.05.2024.
[12]. M. Dogariu, L.-D. S¸tefan, B. A. Boteanu, C. Lamba, B. Kim, and B. Ionescu, "Generation of realistic synthetic financial time-series," ACM Transactions on Multimedia Computing, Communications, and Applications, vol. 18, no. 4, p. 1–27, 2022.
[13]. Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," arXiv preprint arXiv:1707.01926, 2017.
[14]. P. J. Brockwell and R. A. Davis, Introduction to time series and forecasting. Springer, 2016.
[15]. P. W. Talbot, C. Rabiti, A. Alfonsi, C. Krome, M. R. Kunz, A. Epiney, C. Wang, and D. Mandelli, "Correlated synthetic time series generation for energy system simulations using fourier and arma signal processing," International Journal of Energy Research, vol. 44, no. 10, p. 8144–8155, 2020.
[16]. "mockseries documentation.", Available at: https://mockseries.catheu.tech/docs/tutorials/interaction-scalaroperations, accessed 19.05.2024.
[17]. К.В. Воронцов, "Машинное обучение. Прогнозирование временных рядов. К.В. Воронцов, Школа анализа данных, Яндекс", 2020, Available at:

https://youtu.be/Rmh6b96u6UU?si=o3I20WlIP5EKW2kw (in Russian), accessed 08.10.2023.
[18]. R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and practice, 3rd ed. OTexts, 2021.
[19]. "Python", Available at: https://docs.python.org/3/tutorial/index.html, accessed 11.04.2024.
[20]. "Numpy", Available at: https://numpy.org/doc/stable/, accessed 19.05.2024.
[21]. "matplotlib", Available at: https://matplotlib.org/stable/, accessed 17.05.2024.
[22]. "scikit-learn", Available at: https://scikitlearn.org/stable/index.html, accessed 19.05.2024.
[23]. "time-series-generator", Available at: https://github.com/stilmmo/time-series-generator, accessed 01.06.2024.
[24]. Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in International Conference on Learning Representations (ICLR '18), 2018.

Appendix

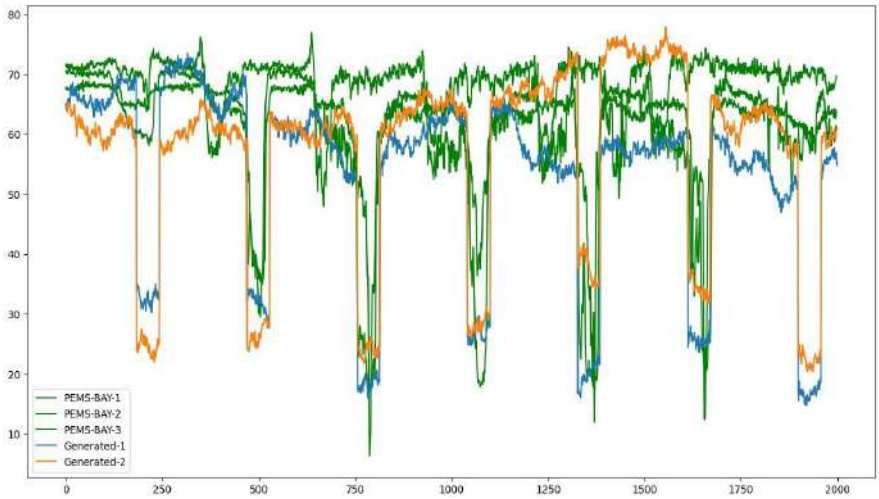
A. Overall Generation Scheme

This scheme demonstrates the class architecture of the time series generator implementation in Python.



B. Custom process

This graph demonstrates a custom process that mocks origin data from PEMS-BAY dataset. All of the origin time series are colored in green.



Информация об авторах / Information about authors

Алена Михайловна КРОПАЧЕВА является обучающейся бакалавриата факультета Математики и механики Санкт-Петербургского государственного университета. Область научных интересов: анализ временных рядов, классические алгоритмы компьютерного зрения, машинное обучение и глубокое обучение.

Alena Mikhailovna KROPACHEVA is an undergraduate student at the Faculty of Mathematics and Mechanics of Saint Petersburg State University. Her research interests include time series analysis, classical computer vision algorithms, machine learning, and deep learning.

Дмитрий Викторович ГИРДЮК является ассистентом кафедры диагностики функциональных систем Санкт-Петербургского государственного университета. Его научные интересы включают компьютерное зрение, анализ временных рядов, математическая иммунология.

Dmitry Viktorovich GIRDYUK is an assistant of the Department of Functional Systems Diagnostics of Saint Petersburg State University. His research interests include computer vision, time series analysis, and mathematical immunology.

Илларион Лаврентьевич ИОВ – аспирант лаборатории моделирования природных систем Университета ИТМО. Область научных интересов: автоматическое машинное обучение, применение больших языковых моделей для оптимизации, анализ временных рядов.

Illarion Lavrentievich IOV – post graduate student at Natural Systems Simulation Lab at ITMO University. Research interests: automated machine learning, large language models applications to optimization, time series analysis.

Антон Юрьевич ПЕРШИН – Ph.D., доцент кафедры фундаментальной информатики и распределенных систем Санкт-Петербургского государственного университета. Сфера

научных интересов: хаотические динамические системы, ламинарно-турбулентный переход, численные методы анализа устойчивости, анализ временных рядов.

Anton Yurievich PERSHIN – Ph.D., associate professor of the Department of Fundamental Informatics and Distributed Systems of Saint Petersburg State University. Research interests: chaotic dynamical systems, transition to turbulence, numerical methods for stability analysis, time series analysis.

DOI: 10.15514/ISPRAS-2024-36(4)-12

Merging Directly-Follows Graphs and Sankey Diagrams for Visualizing Acyclic Processes

I.D. Derezovskiy, ORCID: 0009-0006-0082-2102 <iderezovskiy@hse.ru>
N.D. Shaimov, ORCID: 0000-0003-3843-5379 <nshaimov@hse.ru>
I.A. Lomazova, ORCID: 0000-0002-9420-3751 <ilomazova@hse.ru>
A.A. Mitsyuk, ORCID: 0000-0003-2352-3384 <amitsyuk@hse.ru>

HSE University, Faculty of Computer Science, PAIS Lab
11 Pokrovsky boulevard, Moscow, 101000, Russia

Abstract. This paper proposes a method to visualize models of acyclic processes based on merging Directly-Follows Graphs (DFG) and Sankey diagrams. DFG is a popular graphical model to visualize discrete process models, while Sankey diagrams are used to represent flows of any kind. Our approach, based on flow diagrams, allows us to highlight individual cases or groups of cases in the overall model. The approach is implemented as a web-based tool that allows us, given an event log of an acyclic process, to construct and analyze the process behavior. We illustrate and evaluate the applicability of the proposed approach using learning processes as examples.

Keywords: data visualization; directly-follows graphs; Sankey diagram; process mining; web application; data analysis.

For citation: Derezovskiy I.D., Shaimov N.D., Lomazova I.A., Mitsyuk A.A. Merging Directly-Follows Graphs and Sankey Diagrams for Visualizing Acyclic Processes. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 4, 2024. pp. 155-168. DOI: 10.15514/ISPRAS-2024-36(4)-12.

Acknowledgements: This work is supported by the Basic Research Program at the HSE University, Russia. Merging Directly-Follows Graphs and Sankey Diagrams for Visualizing Acyclic Processes**.**

Объединение графов непосредственного следования и диаграмм Санкей для визуализации ациклических процессов

И.Д. Дерезовский, ORCID: 0009-0006-0082-2102 <iderezovskiy@hse.ru>
Н.Д. Шаимов, ORCID: 0000-0003-3843-5379 <nshaimov@hse.ru>
И.А. Ломазова, ORCID: 0000-0002-9420-3751 <ilomazova@hse.ru>
А.А. Мицюк, ORCID: 0000-0003-2352-3384 <amitsyuk@hse.ru>

Национальный исследовательский университет «Высшая школа экономики», Россия, 101000, Москва, Покровский бульвар, 11

Аннотация. В данной статье предлагается метод визуализации моделей ациклических процессов, основанный на объединении графов непосредственного следования и диаграмм Санкей. Графы непосредственного следования – популярная графическая модель для визуализации моделей дискретных процессов, в то время как диаграммы Санкей используются для представления потоковых данных. Наш метод, основанный на потоковых диаграммах, позволяет выделять на общей модели отдельные экземпляры или группы экземпляров процесса. Для подхода, предложенного в работе, представлена реализация в виде веб-приложения, которое позволяет, на основе журналов событий ациклических процессов, строить и анализировать поведение процессов. Применимость предложенного подхода иллюстрируется и оценивается на примерах образовательных процессов.

Ключевые слова: визуализация данных; process mining; графы непосредственного следования; диаграмма Санкей; анализ процессов; веб-приложение.

Для цитирования: Дерезовский И.Д., Шаимов Н.Д., Ломазова И.А., Мицюк А.А. Объединение графов непосредственного следования и диаграмм Санкей для визуализации ациклических процессов. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 155–168 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(4)–12.

Благодарности: Работа поддержана Программой фундаментальных исследований Национального исследовательского университета «Высшая школа экономики» (НИУ ВШЭ), Россия.

1. Introduction

The visual representation of information makes it possible to analyze data quickly and in an accessible form. Visual analytics is widely used in the analysis of various processes [1], [2], [3], [4]. In the field of process analysis, process mining techniques are widely used to automatically synthesize and evaluate process models based on event logs [5], [6]. The resulting models are used to detect and eliminate flaws in existing processes.

Currently, process mining is particularly relevant and in demand. Recent research shows interest in the application of process mining in many fields such as business, healthcare and education [7], [8], [9]. Process mining techniques allow representing the behavior of a process in a visual form suitable for visual analytics [10].

Today, experts and researchers in the field of process analysis use various approaches to visualize process models. Specifically, processes can be represented as Directly-Follows Graphs (DFG), BPMN-models and Petri Nets. DFG is a simple and popular model that is easy to discover. However, due to its simplicity, it may represent the process too imprecisely. In particular, a DFG model can allow impossible behavior, which in turn can lead to incorrect conclusions about the features of the process [11]. An acyclic process has no cycles and can be considered as a flow of events. The Sankey diagram is often used to visualize various flows. This paper takes into account the advantages of both DFG and Sankey methods of visualizing processes and expands them to get a better visual representation of the acyclic processes.

The main goal of this paper is threefold: (1) to present a method to visualize an acyclic process model based on merging Directly-Follows Graphs (DFG) and Sankey diagrams that makes the analysis of such processes more convenient, (2) to illustrate and evaluate the proposed approach

using real examples, (3) and to implement the proposed visualization method as a web application, describe its functionality and development technology.

The paper is organized as follows. Section 2 presents the problem we solve. Section 3 provides an overview of other existing solutions, highlighting their strengths and weaknesses. We discuss an illustrative motivating example in Section 4. Then, Section 5 presents the idea of our solution, whereas Section 6 discusses its technical aspect. Section 7 contains some findings and evaluations made when analyzing an example of visualization of a model of a real educational process, built from an event log. Finally, Section 8 concludes the paper.

2. Problem statement

Digitized processes in various fields generate significant amounts of data, including event data reflecting user behavior. Through investigation of the process models, it is possible to obtain various insights about existing processes and come up with necessary changes in processes. The data does not take into account human relationships or other individual aspects. That means that decisions cannot be performed in an automatic manner. Automated tools for process analysis should only help experts by providing the necessary information for decision-making.

It is a very important task to visualize generated models in an accessible and understandable way. There are many tools for visualizing process models, some of the popular ones are considered in the next section. However, existing tools do not provide an approach for the analysis of acyclic processes. Such processes occur when events in a process are arranged in a chain-like manner with no cyclic behavior. For example, an educational process where students attend lectures and seminars, perform homeworks and tests [7].

In this paper, we propose a way to visualize acyclic processes and present the tool that allows users to generate and visualize models of acyclic processes, highlight specified cases on the existing model and construct intersections for specified groups of cases.

We propose a way to visualize an acyclic process based on merging Directly-Follows Graphs (DFG) and Sankey diagrams. This method we implement as a client-server application that enables building models of acyclic processes from event logs, and visualizing them in the form of very specific oriented graphs. Moreover, the acyclic nature of the process makes it possible to compare different groups of cases within the process through the intersection of flows. The proposed visualization will simplify the analysis, search for relationships, deviations and anomalies in acyclic processes.

3. Overview of existing solutions

Many tools for visualizing process models based on event logs are available. We compare our tool with the following process mining tools and instruments for visualizing data in the form of various graphs and diagrams. Results of the comparison are summarized in Table 1 with the following numbering:

1) PMTK (novel web-based Process Mining ToolKit) [12].
2) Celonis (commercial online platform for analysis business processes) [13].
3) SankeyMATIC (website for building Sankey diagrams) [14].
4) Fluxicon Disco (application for analysis business processes) [15].
5) ProM (plugin-based framework for process mining) [16].
6) Proceset (analytical system for collecting and uploading data, conducting research in process mining) [17].
7) Solution proposed in this paper.

Table 1. Analysis of existing solutions.

Feature	1	2	3	4	5	6	7
Function of automatic building models based on the event log	+	+	-	+	+	+	+
Function of visualizing models in the form of an oriented graph (diagram)	+	+	+	+	+	+	+
Interactive selection of vertices and transitions in the graph for filtering	+	+	-	+	+	+	+
Visualization of a model with intersection of subsets	-	-	-	-	-	-	+
Filtering by multiple values	+	+	-	+	+	+	+
Filtering by events and transitions	+	+	-	+	+	+	+
Creating and saving subsets of cases	-	-	-	-	+	-	+
Saving all information about the user's projects	+	+	-	+	-	+	+
Free of charge for non-commercial and academic use	+	-	+	-	+	-	+

3.1 The Process Mining ToolKit (PMTK)

The Process Mining ToolKit, i.e., PMTK presents process mining algorithms and techniques in an easy-to-use solution. PMTK is built on top of the PM4Py library and allows non-technical users to use the advanced process mining technology implemented in PM4Py [18].

The important advantage is that PMTK provides a workspace in which the user is able to organize various event logs and projects. Subsequently, various objects, e.g., event logs and filters can be stored in the corresponding project's folder. Also, PMTK implements a process map with various filtering options (i.e., filtering of edges and activities). As such, PMTK, can be seen as a front-end solution for the advanced open-source process mining library PM4Py. But a user cannot really interact with the graph: for example, one cannot select vertices and transitions by clicking on the corresponding elements.

3.2 Celonis

Celonis is one of the leading products in the field of Process Mining. This commercial comprehensive software is designed to analyze, visualize and optimize processes within an organization based on data obtained from various IT systems. Including the analysis of educational processes.

The main advantage of Celonis is the ability to build different stages of processing applications on a straight line. By revealing different parts of the process, users can find out how many people participated in the processing of this incident, see the duration of each stage, and track statistics.

But it is difficult to integrate this online platform into the work of a non-commercial organization, since most of the functions are paid. There are also serious limitations in working with data: in particular, there are few data filtering options.

3.3 SankeyMATIC

SankeyMATIC is a free online tool that provides a wide range of controls that allow the user to customize the design of the constructed diagrams. User can export final design as either a PNG or SVG file. The advantage of this web tool is that it specializes in building Sankey diagrams.

However, SankeyMATIC does not provide the ability to create and manage projects, organize long-term work with data. There are serious limitations in working with data, as with previous analogues.

3.4 Fluxicon Disco

Disco's process development technology helps to build visual diagrams based on uploaded data. The program allows to perform process data analysis in order to optimize the efficiency of processes, control deviations, or explore various options for process tracks.

The advantages of the application are detailed statistics in a convenient visual form, convenient tools for filtering data, convenient tools for creating and working with user projects.

It is worth noting that a significant part of the functions is provided on a paid basis.

3.5 ProM

ProM is used to work with business process-related data and offers a variety of tools for analyzing, modeling, and improving business processes. ProM Tools provides a wide range of algorithms for process analysis, such as algorithms for detecting process models, algorithms for analyzing process performance, and many others. ProM also supports various process modeling standards, such as BPMN (Business Process Model and Notation), Petri Nets and others.

ProM is an extensible framework that supports a wide variety of process mining techniques in the form of plugins. It is platform independent as it is implemented in Java, and can be downloaded free of charge.

3.6 Proceset

Proceset is an active business intelligence system designed to manage and analyze business processes in real time, which allows you to quickly respond to changes and events within business processes.

The system integrates with various data sources, such as databases, process management systems (BPM), customer relationship management systems (CRM) and others, to obtain up-to-date information about processes.

Proceset provides visualization of business process data in a convenient and informative form, using only simple line graphs, process maps and tables.

An important disadvantage of the system, as with all the considered analogues, is also the inability to select subsets of cases using the specified filters from the general event log. It is also impossible to visualize models with multiple subsets with intersecting elements.

These analogues have a number of important common disadvantages:

- The inability to filter the event log (graph) by trajectory options through interactive selection of events and transitions (by clicking on the vertices and edges of the graph);
- The lack of visualization options for models with intersecting subsets;
- The inability to create and save subsets of cases.

4. Motivating example

Let us consider a process where in each trace events do not repeat themselves, and the resulting model is acyclic. It can be represented as a flow, where cases flow from one event to another.

Generally, such cases are represented with DFG, where events and transitions from one event to another are shown as nodes and arcs whose thickness is based on the number of involved cases.

The problem with DFG is that in some cases it does not allow us to identify dependencies between events. As an example, consider the event log in Table 2, which contains data on students' academic performance, and its DFG model in Fig. 1. For each trace in this event log, events «LR17 high» and «Accum low» do not occur in the same trace. However, the model in Fig. 1 allows a trace passing through these two events. Also, when we need to analyze certain groups of cases, the most common approach is to construct a separate model for this subgroup. Then an existing event log is filtered, leaving only the relevant cases, and a new model, based on the filtered event log, is synthesized and visualized. This leads to a quite cumbersome analysis and requires a visual comparison of two models.

Table 2. Example of an eventlog.

Case_id	Activity	Timestamp	Grade	Lector_id	Group
#19273	LR1 high	08.11.2021	8	#13761	BPI203
#19273	LR2 med	09.11.2021	6	#13761	BPI203
#19273	LR3 med	16.11.2021	5	#13761	BPI203
...					
#45951	LR17 med	11.12.2021	6	#13976	BPI206
#45951	LR18 low	17.12.2021	3	#13976	BPI206

To better reflect the flow of cases and dependencies between certain events, we can merge DFG with the Sankey diagram that is used to display flows. Such visualization allows to track changes in case flow at each moment of the process. As for analyzing groups of cases, a selected case or group of cases can be highlighted on the model as a separate flow. In this way, analysis can be performed on the same model without the need to create a separate model. Additionally, a selected group can be visually compared with the rest of the cases or with other groups.

Let us take a look at an example with educational flow where students pass control elements of a course. For instance, consider the event log in Table 2. This log contains real data about students, grades received by students in a certain educational discipline, teachers and study groups. The grades for the test work are divided into pairs in accordance with the order in which they are carried out. For example, we want to take a look at students with low accumulated grade to find potential causes. Using the PM4PY python library, we can generate the DFG model of the process (see Fig. 1). Now we can apply filter to the event log and leave only traces containing «Accum low» events. Building a model with filtered log provides us with the following model (see Fig. 2). To search for potential causes both of the models should be analyzed side by side. The situation with the generation of Petri Net models is similar (see Fig. 3). Using DFG merged with the Sankey diagram, we can highlight traces with the «Accum low» event (see Fig. 4). Both models allow detecting differences between the specified group of students and the rest of the students. But DFG merged with the Sankey diagram eliminates the need to compare two models' side by side and more clearly shows the flow of these cases in proportion to the others.

5. Proposed solution

In this work, we propose the development of a tool in the form of a web application focused on the generation and visualization of acyclic process models. This tool will allow us to build interactive acyclic process models from event logs. The application provides the ability to specify subsets of cases from loaded event logs. The user can select subsets to be highlighted in the visualization, in order to display the trajectories of the selected subset among the other cases. In addition, it is possible

to simultaneously visualize two selected subsets of cases and highlight the cases included in both subsets.

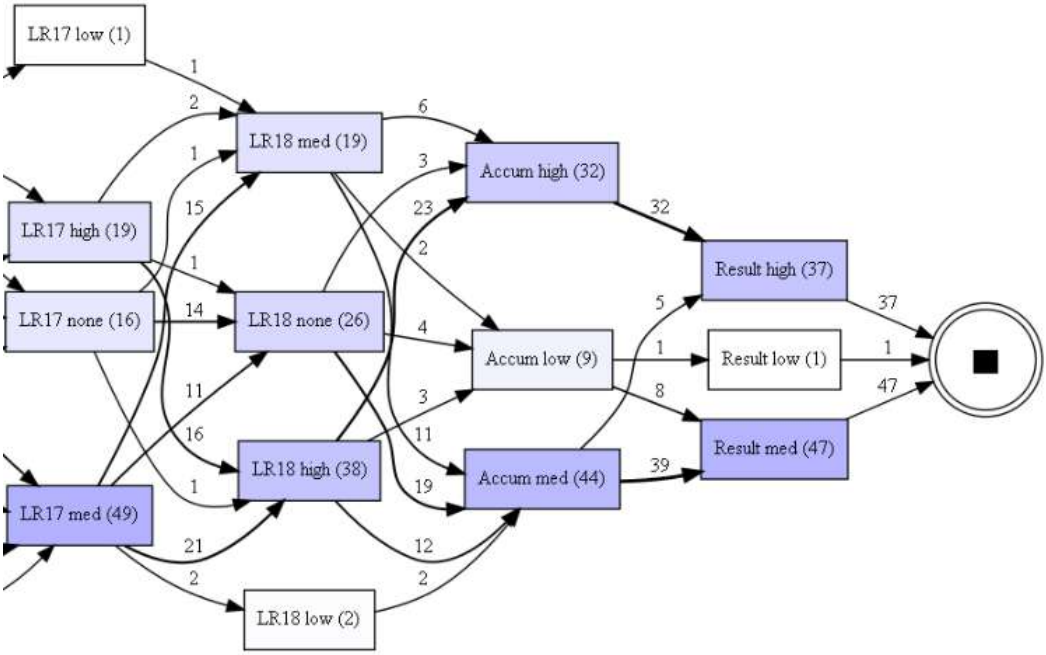


Fig. 1. An example of a DFG model of the educational process.

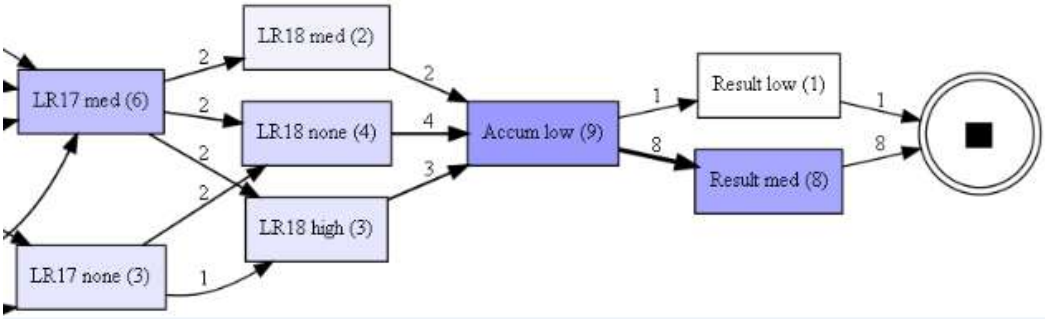


Fig. 2. The DFG model for traces containing the «Accum low» event.

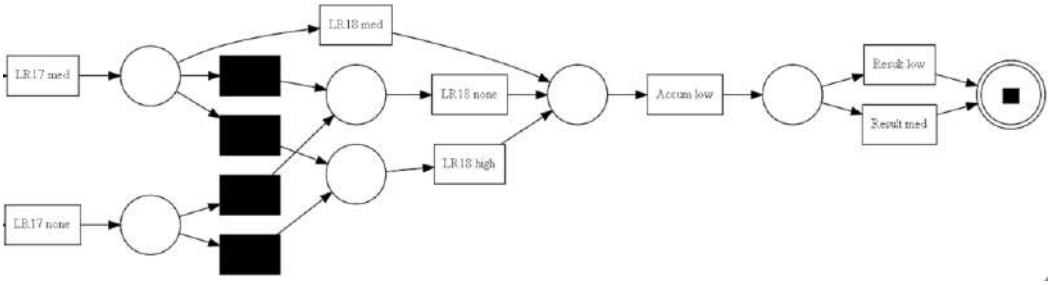


Fig. 3. The Petri Net model for traces containing the «Accum low» event.

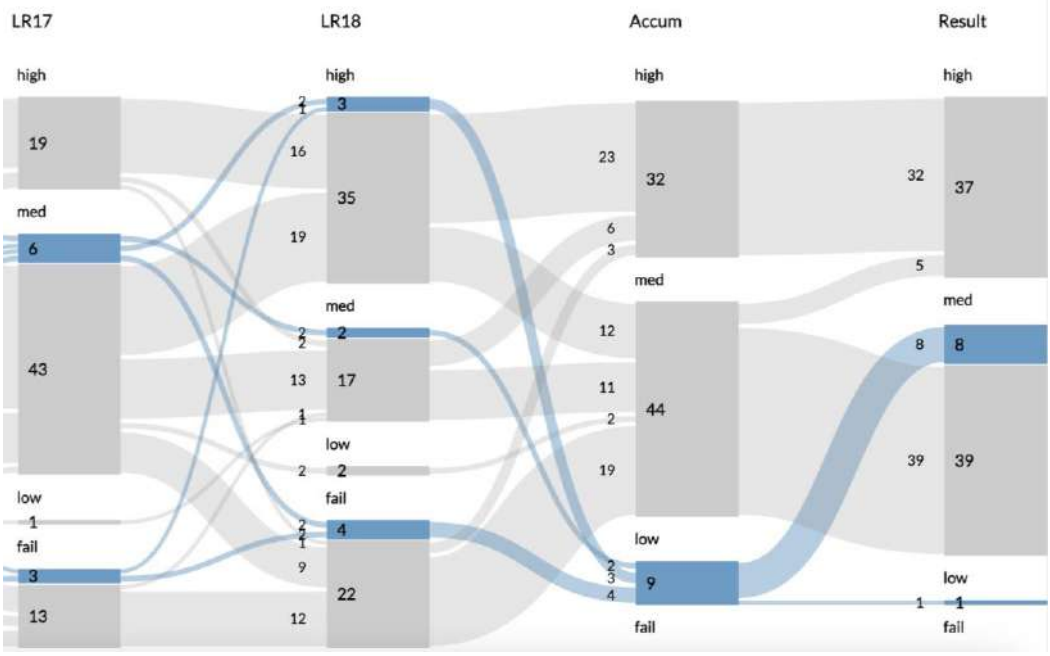


Fig. 4. The DFG model merged with the Sankey diagram with traces containing the «Accum low» event highlighted.

As a base for our data flow graphical representation, we chose a Sankey diagram. The classic version of the Sankey diagram, which is shown in Fig. 5, specializes on visualization of flows, but such diagrams do not contain vertices as such. This makes it difficult to view and analyze activities of a process, and, compared with the DFG, cuts off a significant part of the important information about the number of elements involved in the event.

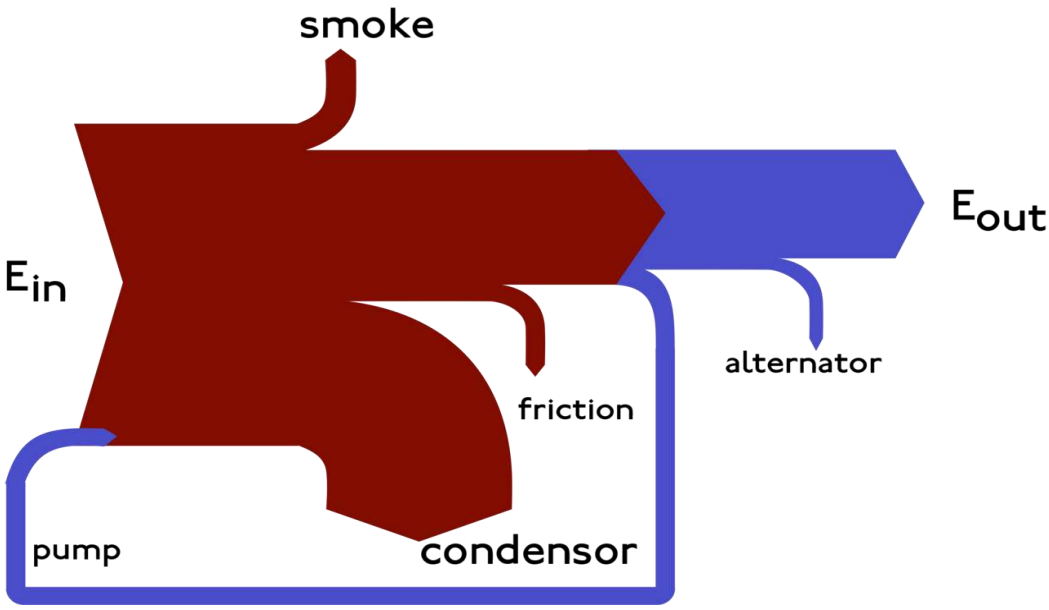


Fig. 5. An example of a classic Sankey diagram of a thermodynamic steam cycle [19].

The proposed solution to the problem is to merge the Sankey diagram with the DFG (see Fig. 6). By modifying the DFG, replacing the edges with wide transitions of the Sankey diagram and changing the size of the vertices in accordance with the number of cases involved, we obtain a visualization that clearly shows the proportional distribution and movement of cases throughout the process.

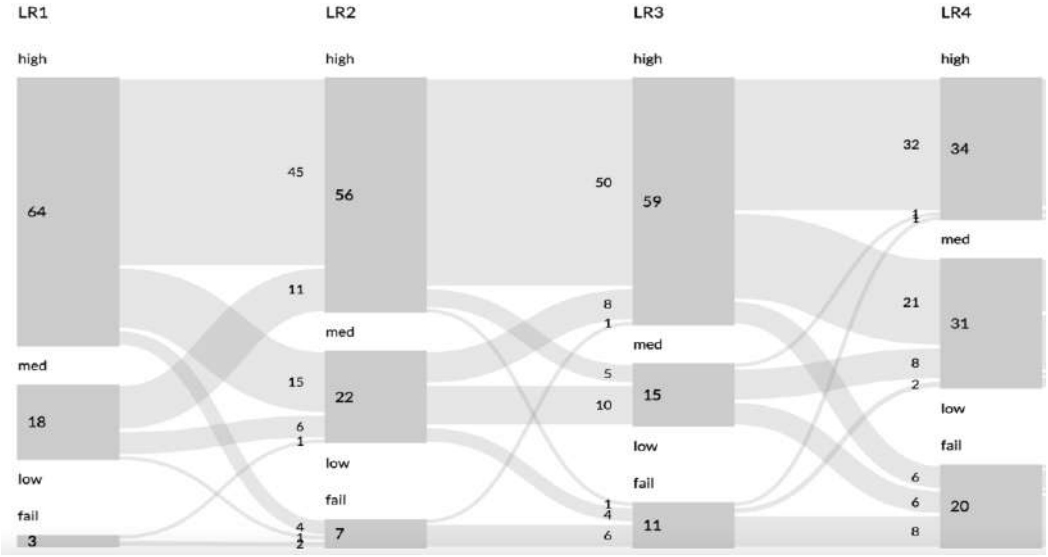


Fig. 6. An example of visualization for acyclic processes based on merging DFG and Sankey diagram.

Let us review the obtained visualization for process models. The model columns represent process actions. For example, in the case of educational processes, they may correspond to academic disciplines or forms of control (seminar, test, exam, etc.). Columns may contain several nodes. Each node may represent a different action category in the process. For example, they can correspond to student grades (excellent, good, satisfactory, unsuccessful or «none» if the student missed a form of control). The arcs of the model show the transitions between events. Thickness of nodes and arcs correlate to the number of cases passing through them.

The resulting visualization is envisioned to be part of the web application. The application will allow us to bring interactivity into the visualization. For example, users can filter out a specific group of cases based on events and transitions through interaction with process visualization. This could be done by clicking on the corresponding elements (events or transitions).

Next, we describe the main features of the application in more detail.

The application provides the ability to apply filters. Each applied filter is added to the list of currently applied filters. The selected set of cases is defined by the list of current filters. Cases can be filtered by the following categories:

1) *by events* – cases should or should not contain a specified event or events;
2) *by transitions* – cases should or should not contain a specified transition or transitions;
3) *by attributes* – cases should or should not contain certain attribute values.

For example, for an educational process, an attribute can be specific student or teacher IDs, grades, etc.

In addition, we would like to emphasize the ability to create subsets of cases based on a list of filters. Cases that fit the current filters can be saved as a subset. The resulting subsets of cases can be highlighted in the model or used to create a visualization of the intersection of two selected subsets. Visualization of the intersection of two selected subsets shows one subset in one color and the other one in another color. Cases that present in both subsets will be shown as a gradient of two colors.

The rest of the cases are shown in neutral color. As a result, visualization will display 4 different categories of cases that:

1) present only in the first subset;
2) present only in the second subset;
3) present in both subsets;
4) not present in any of the subsets.

It is important to mention the limitations of our visualization. The process must be acyclic, since cycles can disrupt the visual coherence of the model. Therefore, each action for each case in the event log should occur no more than one time.

The application provides an account system to implement the saving of the user's work progress. After registering or logging in, the user can create a project and upload the necessary event logs. Projects store uploaded event logs and created subsets of cases.

And of course, the most important function of the application is model generation using uploaded event logs and subsets. Generation takes place at the time of request for visualization of the process model and mostly follows standard DFG generation algorithm. Unique event names in the event log create the set of graph vertices and edges reflect direct transitions from one event to another.

6. Tool architecture and implementation

Let us describe the architecture of the developed web-based application. It follows the client-server application model and shown in Fig. 7. The main three components of our client-server application are as follows:

- **Web client** handles and displays the visualization of the generated model. It communicates with the server via HTTP requests, providing information about the selected event log data, applied filters, and authorization requests;
- **Web server** constructs educational process models and filters event logs. Upon receiving requests from the web client, it processes the data and returns an HTTP response containing the educational process model with applied filters, along with user data;
- **Database** stores event logs and responds to data requests from the server. It provides information about users, projects, and event logs back to the server when it receives a request.

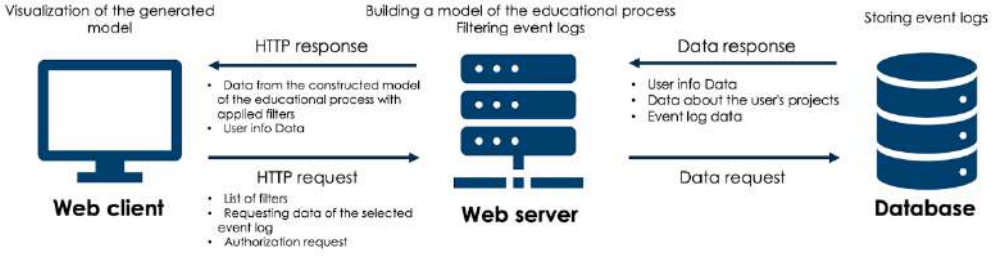


Fig. 7. The software architecture.

The user interface (the client-side part of the application) is implemented in JavaScript using HTML and CSS. This combination consists of the typical technologies used to build web-based applications. JavaScript provides dynamic behavior and interactivity, while HTML defines the structure of web pages, and CSS handles presentation and styling. It is supposed to use a JavaScript library React JS for creating user interfaces. This library supports component-based architecture, which makes it easier to manage and reuse UI elements. React's document object model helps update

the user interface efficiently, leading to faster rendering and an improved user experience. This architecture encourages cleaner code organization via components, promoting improved maintainability and scalability of web-based applications. The part of the program responsible for plotting and visualizing graphs is implemented using the open-source JavaScript library D3.js (Data Driven Documents 7.6). This library provides extensive capabilities for creating custom visualizations, allowing developers to design bespoke charts and graphs tailored to specific needs. D3.js uses SVG, HTML and CSS to render visualizations. This provides flexibility and control over the appearance and behaviour of the elements, giving more power over data representations.

The developed tool for visualizing acyclic processes is implemented as a client-server application. Python was chosen to implement the backend of the application. It should be noted that a lot of Python frameworks are available to implement the backend of the web application, including Django, Flask and Pyramid. These frameworks provide opportunities for developing server-side components of applications, such as handling HTTP requests and communicating with the backend of the application with databases. An important advantage of using Python for programming is that it can be easily integrated with other programming languages. It is especially important that it can be used in combination with JavaScript. Additionally, Python is widely used for data analysis, including in the field of process mining. Python provides many libraries and frameworks for analyzing and visualizing process data: numpy, pandas, pm4py, and others.

In our application, PostgreSQL has been chosen as the database management system for storing information about users, created projects, and uploaded event logs. In general, PostgreSQL supports the management of databases of unlimited size.

Visual Studio Code is a popular web application development tool which supports programming languages such as JavaScript, HTML, and CSS, as well as Python, React JS, which we used for the development of our visualization tool.

7. Findings and evaluations

Let us now provide some observations that were made in the course of analyzing the visualization of the model built on the basis of event logs of the real learning process. The event logs contain information about students' grades for the algebra practicum course.

Fig. 8 shows the visualization with the intersection of two subsets of cases. The first subset contains students who received an excellent grade for the last work (node «LR18»), and the second subset contains students who received a low accumulated grade (node «Accum»). The resulting grade for the discipline is formed as follows: the resulting grade consists of the accumulated grade and the final exam grade; the resulting grade is a sum of those grades where each grade has a certain coefficient. The accumulated grade is a sum of all the grades that the student received throughout his studies in the discipline, each with their own coefficient. The final exam takes place at the end of the academic discipline. The visualization shows the trajectories of students from the two subsets, and students who belong to the intersection of these subsets (those who had received an excellent grade for the last test work and a low accumulated grade), are highlighted by a gradient.

From the model, it is clear that most of the students who received a low accumulated grade received a medium final grade. This may indicate that these students received a medium or high mark for the final exam and successfully completed the course. To look at the trajectory for such students throughout the course, we can click on the transition between the «Accum low» and «Result med» vertices to form a subset of students who contain this transition in their trajectory. The subset, highlighted in blue color, is displayed on the model (see Fig. 9).

Upon further analysis of the diagram, it becomes clear that these students missed a large number of assessments, resulting in a low accumulated grade.

Detailed analysis of the educational process models combined with visualization for subsets of students will help to identify areas and events that may be a cause for concern and require special attention from educational program administrators.

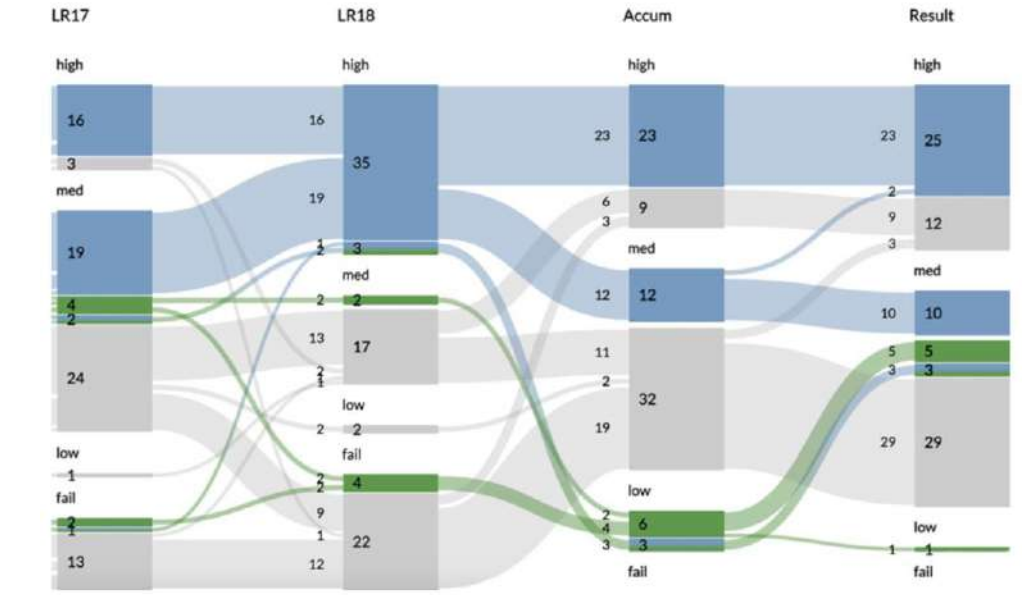


Fig. 8. Visualization of the intersection of two subsets of cases.

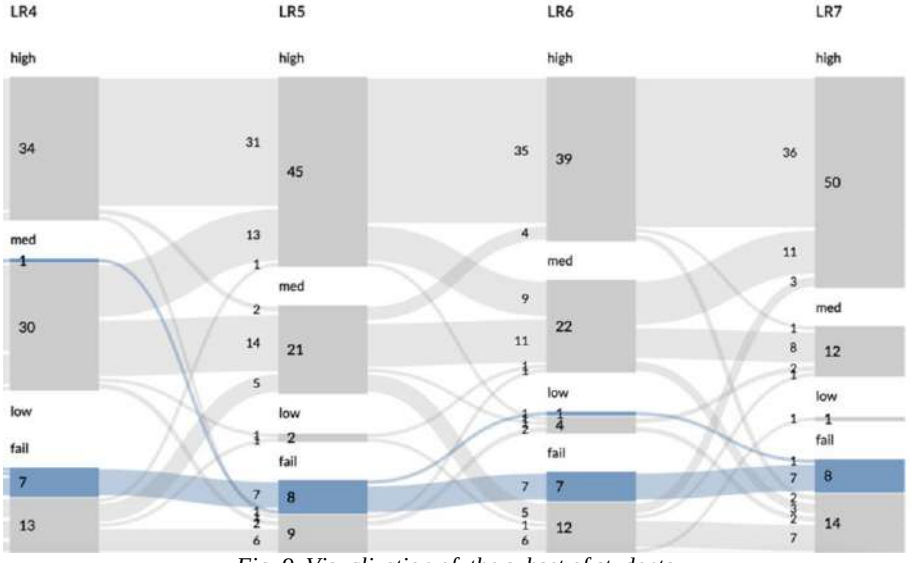


Fig. 9. Visualization of the subset of students.

8. Conclusion

In this paper, we presented a new method for visualizing acyclic processes. The visualization is based on merging of DFG and Sankey diagrams, taking advantage of the benefits of each. Our approach, based on the flow diagrams, allows us to highlight individual cases or groups of cases in an overall model. The approach is implemented as a web-based tool that allows us, given an event log of an acyclic process, to construct and analyze the process behavior. It is assumed that in combination with other data analysis methods, the causes of particular events and trajectories can be easily identified.

The application provides the ability to specify subsets of cases from uploaded event logs to highlight them in the visualization. In addition, one can simultaneously visualize two selected subsets of cases and highlight overlapping cases that are included in both subsets.

As an example of the practical usage of the developed application, we show how it allows analysts and managers of educational programs to analyze data about the learning process and students' trajectories by visualizing educational trajectories in order to identify deviations, relationships and anomalies that require special attention from administrators of educational programs.

An important constraint of our solution is that it deals with acyclic models only. For future work we plan to generalize our visualization method for models with cycles.

References

[1]. T. Gschwandtner, «Visual Analytics Meets Process Mining: Challenges and Opportunities», in Data-Driven Process Discovery and Analysis. SIMPDA 2015. Lecture Notes in Business Information Processing, vol 244. Springer, Cham, 2017, pp. 142–151. doi: 10.1007/978-3-319-53435-0_7.

[2]. Y. V. Kotylev and A. A. Mitsyuk, «Software System Behavior Can Be Analyzed with Visual Analytics», in Proceedings of the Conference on Modeling and Analysis of Complex Systems and Processes 2020 (MACSPro 2020) / Ed. by Alexander Shapoval, V. Popov, I. Makarov. Vol. 2795. CEUR Workshop Proceedings, 2020. pp. 46-56.

[3]. S. Miksch, «Visual Analytics Meets Process Mining: Challenges and Opportunities», 3rd International Conference on Process Mining (ICPM 2021), Eindhoven, Netherlands, 2021, pp. xiv-xiv, doi: 10.1109/ICPM53251.2021.9576854.

[4]. J. Rehse, L. Pufahl, M. Grohs and L. Klein, «Process mining meets visual analytics: the case of conformance checking». arXiv preprint arXiv:2209.09712 (2022). doi: 10.48550/arXiv.2209.09712.

[5]. W. M. P. van der Aalst, Process Mining - Data Science in Action, Second Edition. Springer, 2016, isbn: 978-3-662-49850-7. doi: 10.1007/978-3-662-49851-4. [Online]. Available: https://doi.org/10.1007/978-3--49851-4.

[6]. J. Carmona, W. M. P. van der Aalst. Process Mining Handbook: Lecture Notes. Springer Nature, Switzerland, 2022. 504 p. doi: 10.1007/978-3-031-08848-3.

[7]. N. D. Shaimov, I. A. Lomazova, A. A. Mitsyuk, and I. Y. Samonenko, «Analysis of Students' Academic Performance using LMS Event Logs», Modeling and analysis of information systems, vol. 29, no. 4, pp. 286-314, 2022.

[8]. J. Munoz-Gama, N. Martin, C. Fernandez-Llatas, O. Johnson, M. Sepúlveda, E. Helm, et al. - Process mining for healthcare: Characteristics and challenges. J. Biomed, 2022. doi: https://doi.org/10.1016/j.jbi.2022.103994.

[9]. P. Badakhshan, B. Wurm, T. Grisold, J. Geyer-Klingeberg, J. Mendling, J. Vom Brocke - Creating business value with process mining. J Strateg Inf Syst., vol. 31(4), 2022. doi: https://doi.org/10.1016/j.jsis.2022.101745.

[10]. I. Sitova and J. Pecerska, «Process Data Analysis Using Visual Analytics and Process Mining Techniques», 61st International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), 2020. doi: https://doi.org/10.1109/ITMS51158.2020.9259296.

[11]. Wil M.P. van der Aalst, «A practitioner's guide to process mining: Limitations of the directly-follows graph», Procedia Comput. Sci., vol. 164, pp. 321-328, 2019. doi: https://doi.org/10.1016/j.procs.2019.12.189.

[12]. The Process Mining ToolKit (PMTK) Documentation. [Online] Available: https://pmtk.fit.fraunhofer.de/ [Accessed Mar. 05, 2024].

[13]. The Celonis Documentation. [Online] Available: http://celonis.com/ [Accessed Mar. 05, 2024].

[14]. The SankeyMATIC Documentation. [Online] Available: https://sankeymatic.com/ [Accessed Mar. 05, 2024].

[15]. The Fluxicon Disco Documentation. [Online] Available: https://fluxicon.com/disco/ [Accessed Mar. 05, 2024].

[16]. The ProM Documentation. [Online] Available: https://promtools.org/prom-documentation/ [Accessed Mar. 05, 2024].

[17]. The Proceset Documentation. [Online] Available: https://infomaximum.ru/docs?v=02.2024 [Accessed Mar. 05, 2024].

[18]. Alessandro Berti, Sebastiaan van Zelst, Daniel Schuster, «PM4Py: A process mining library for Python», Software Impacts, vol. 17, p. 100556, 2023. doi: 10.1016/j.simpa.2023.100556. [Online]. Available: https://doi.org/10.1016/j.simpa.2023.100556 [Accessed Mar. 25, 2024].

[19]. Wikimedia Commons, the free media repository. [Online]. Available: https://www.google.com/url?q=https://commons.wikimedia.org/wiki/File:Sankeysteam.png&sa=D&source=docs&ust=1712231868106450&usg=AOvVaw389upF5qS7BipkMnvrzqon [Accessed Mar. 30, 2024].

Информация об авторах / Information about authors

Илья Денисович ДЕРЕЗОВСКИЙ – студент бакалавриата факультета компьютерных наук НИУ Высшая Школа Экономики (ВШЭ). Сфера научных интересов: моделирование и формальный анализ поведения процессов в информационных системах, объектно-ориентированное программирование и проектирование пользовательских интерфейсов.

Ilya Denisovich DEREZOVSKIY is a bachelor student at the faculty of computer science, HSE University. Research interests mainly include modeling and formal analysis of the behavior of processes in information systems, object-oriented programming and user interface design.

Никита Денисович ШАИМОВ – аспирант факультета компьютерных наук НИУ Высшая Школа Экономики (ВШЭ), стажер-исследователь научно-учебной лаборатории процессно-ориентированных информационных систем (ПОИС) НИУ ВШЭ. Сфера научных интересов: анализ процессов (process mining), обработка и анализ данных, программная симуляция.

Nikita Denisovich SHAIMOV is a postgraduate student at the faculty of computer science in HSE University and a research assistant at the Laboratory for Process-Aware Information Systems (PAIS Lab), HSE University. Research interests mainly include computer modeling and simulation, data analysis, process mining, educational process mining.

Ирина Александровна ЛОМАЗОВА – доктор физико-математических наук, профессор факультета компьютерных наук НИУ ВШЭ, заведующий лабораторией научно-учебной лаборатории процессно-ориентированных информационных систем (ПОИС) НИУ ВШЭ. Область научных интересов: анализ и моделирование бизнес-процессов, сети Петри, вложенные сети Петри, процессно-ориентированные информационные системы, формальные модели распределённых систем.

Irina Alexandrovna LOMAZOVA – Dr. Sci. (Phys.-Math.), professor of the faculty of computer science in HSE University, and laboratory head of the Laboratory for Process-Aware Information Systems (PAIS Lab), HSE University. Doctor of Sciences in Theoretical Foundations of Computer Science Russian Academy of Sciences Dorodnitsyn Computation Center since 2002. Research interests mainly include analysis and modeling of business processes, Petri nets, process-oriented information systems, formal models of distributed systems.

Алексей Александрович МИЦЮК – доцент факультета компьютерных наук НИУ ВШЭ, старший научный сотрудник научно-учебной лаборатории процессно-ориентированных информационных систем (ПОИС) НИУ ВШЭ. Имеет степень кандидата компьютерных наук (физико-математические науки) НИУ ВШЭ (2019 г.). Область научных интересов: алгоритмы, архитектура информационных систем, сети Петри, программная архитектура, извлечение и анализ процессов, анализ данных, визуализация данных, проектирование пользовательских интерфейсов.

Alexey Alexandrovich MITSYUK is an associate professor of the faculty of computer science in HSE University and senior researcher at the Laboratory for Process-Aware Information Systems (PAIS Lab), HSE University. Cand. Sci. (Phys.-Math.) of HSE University since 2019. Research interests mainly include process mining, algorithms, information systems architecture, Petri nets, software architecture, data analysis, data visualization, user interface design.

DOI: 10.15514/ISPRAS-2024-36(4)-13

Проблема неопределенности в анализе трасс на основе высокоуровневых моделей в контексте динамической верификации

А.А. Карнов, ORCID: 0000-0002-2066-9946 <karnov@ispras.ru>

Институт системного программирования РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. В данной статье обсуждается проблема применения метода динамической верификации к большим и сложным системам, в частности, операционным системам общего назначения. Современные практики и стандарты разработки критических систем требуют наличия формальной модели политики безопасности. Полнота и непротиворечивость формальных требований, указанных в модели политики безопасности, должна быть верифицирована с использованием формальных методов. Позже, когда будет разработана реализация системы, необходимо установить, что реализованные механизмы безопасности соответствуют указанным в модели требованиям. При использовании такого подхода удобно иметь единую модель, подходящую как для формальной верификации, так и для тестирования реализации. Для достижения этой цели необходимо, с одной стороны, выделить подмножество языковых конструкций модели, подходящих для обоих методов, а с другой, разработать специальные методики анализа трасс выполнения, позволяющие эффективно выполнять тысячи тестов. В статье приводится анализ языковых конструкций, позволяющих эффективное использование модели в рамках динамической верификации. Также в статье представлены методы оптимизации процесса динамической верификации систем. Предложенные методы были реализованы в прототипе инструмента анализа трасс и протестированы на модели системы контроля доступа для операционных систем на основе Linux.

Ключевые слова: динамическая верификация; анализ трасс; язык описания моделей Event-B.

Для цитирования: Карнов А. А. Проблема неопределенности в анализе трасс на основе высокоуровневых моделей в контексте динамической верификации. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 169–182. DOI: 10.15514/ISPRAS–2024–36(4)–13.

Uncertainty Problem in High-Level Model-Based Trace Analysis as Part of Runtime Verification

A.A. Karnov, ORCID: 0000-0002-2066-9946 <karnov@ispras.ru>

Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. The article discusses the problem of applying runtime verification to large and complex systems such as general-purpose operating systems. When verifying the security mechanisms of operating systems, modern practices and standards require a formal security policy model (SPM). The SPM must be verified using formal model methods, and it must also be used to verify the completeness and consistency of the operating system's security mechanisms by confirming compliance with the formal requirements of the SPM. In this case, it is convenient to have a single model suitable for both formal verification and implementation testing. For practical application, it is necessary, on the one hand, to select a subset of model language constructs suitable for both acts, and on the other hand, to develop special techniques for analyzing execution traces that allow to effectively perform thousands of test cases. The article addresses both of these issues. We present an analysis of language constructs that allow us to use the model for both verification and execution trace analysis. We also offer techniques that have been developed to optimize the runtime verification of Linux-based systems. We also implemented the proposed methods in the trace analysis tool prototype.

Keywords: runtime verification; trace analysis; Event-B modelling language.

For citation: Karnov A.A. Uncertainty problem in high-level model-based trace analysis as part of runtime verification. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024. pp. 169-182 (in Russian). DOI: 10.15514/ISPRAS-2024-36(4)-13

1. Введение

Динамическая верификация [1] – это подход к анализу и исполнению вычислительных систем, основанный на извлечении из системы информации во время ее работы и последующем анализе полученной информации. Информация извлекается в виде последовательности событий в порядке их возникновения в системе, при этом события могут дополняться данными, например, параметрами и неким результатом. Такая последовательность событий называется трассой выполнения [2] и представляет собой наблюдаемое поведение системы. Анализ трассы показывает, является ли наблюдаемое поведение корректным, иначе говоря, соответствует ли оно определенным свойствам или же нарушает их.

Динамическую верификацию также можно определить как набор формальных методов для установления соответствия между трассой выполнения и спецификацией системы. В статье рассматривается частный случай, когда в роли такой спецификации выступает формальная модель. Как правило, модель системы содержит меньшее количество деталей и имеет меньший объем по сравнению с реализацией. Кроме того, одной модели может соответствовать сразу несколько различных реализаций. По этим причинам формальная верификация модели системы является более доступной и выгодной по сравнению с формальной верификацией самой системы. Использование верифицированной модели, как источника определения корректного поведения системы, значительно повышает надежность динамической верификации.

Область приведенного в этой статье исследования ограничена анализом трасс выполнения и формальной моделью. Мы абстрагируемся от других важных компонентов динамической верификации, оставляя за рамками исследования методики инструментирования и тестирования системы. Реализация является лишь источником трасс, которые могут быть собраны как во время запусков неких тестовых сценариев, так и во время штатной работы системы. Этапы сбора и извлечения информации также не рассматриваются.

Исследование направлено на эффективное проведение анализа трасс на основе формальной модели системы и на преодоление проблем, с которыми придется неизбежно столкнуться. Раздел 2 посвящен мотивации выбора используемых инструментов и подходов, в нем также перечисляются цели исследования. Раздел 3 содержит краткое описание проблемы на простом примере. В разделе 4 представлен краткий обзор существующих подходов к анализу трасс и исполнению формальных моделей. Текущие результаты исследования представлены в разделе 5. В разделе 6 перечислены будущие направления работы.

2. Мотивация

Для работы над критически важными системами, такими как средства защиты информации, современные стандарты и практики требуют [3] наличия формальной модели политики безопасности. Модель политики безопасности должна быть верифицирована [4] при помощи формальных методов, например, дедуктивной верификации. Возможность использовать одну и ту же формальную модель как для формальной спецификации критической системы, так и для ее тестирования является одним из важнейших преимуществ динамической верификации. Используя такой подход, мы получаем возможность построить замкнутую систему верификации, в которой корректность наблюдаемого поведения системы будет сопоставляться с верифицированной моделью политики безопасности.

Именно по этой причине нам необходимо представить формальную модель тестируемой системы в форме, удобной как для методов формальной верификации, так и для динамической верификации. Оказывается, предложить такое представление на практике достаточно сложно. Одно из возможных решений предложено в данной статье.

Основное практическое применение предложенного подхода – верификация средств защиты информации операционных систем. В этой области было проделано уже достаточное количество работ [5], [6], [7]. В качестве основного средства описания формальных моделей был выбран язык Event-B. Event-B имеет набор преимуществ: простые и понятные языковые конструкции, Rodin [8] – удобную среду разработки моделей, включающую инструменты дедуктивной верификации и множество других полезных инструментов. Среди инструментов есть инструменты для выполнения (анимации) моделей, что демонстрирует принципиальную возможность использования языка для анализа трасс.

Нет никаких серьезных причин отказываться от Event-B, но существуют нетривиальные проблемы, которые усложняют, а иногда и не позволяют использовать модели для динамической верификации. Цели данного исследования:

1. Провести анализ и классификацию языковых конструкций и приемов написания спецификаций на Event-B, усложняющих динамическую верификацию;
2. Выделить подмножество языка и шаблоны написания фрагментов спецификаций, которые либо устраняют проблемы анализа трасс, либо позволяют привести модели к виду, удобному для динамической верификации;
3. Разработать набор техник трансформации моделей;
4. Разработать методы анализа моделей.

3. Проблема неопределенности

Рассмотрим простую модель светофоров на пешеходном переходе. Состояние модели состоит из двух логических переменных:

$$cars\_go \in BOOL \quad (1)$$

$$peds\_go \in BOOL \quad (2)$$

Для обеспечения безопасности движения модель содержит инвариант, запрещающий одновременное движение автомобилей и пешеходов:

$$\neg(cars\_go = TRUE \wedge peds\_go = TRUE) \quad (3)$$

В качестве начального состояния мы можем взять любое состояние, не нарушающее инвариант 3. События $cars$ и $peds$ могут изменять значение переменных на значение параметра go. Чтобы гарантировать, что инвариант 3 не будет нарушен, события имеют условия безопасности, которые называются охранными условиями. Охранное условие 4 относится к событию $cars$, а охранное условие 5 относится к $peds$.

$$go = TRUE \Rightarrow peds\_go = FALSE \quad (4)$$

$$go = TRUE \Rightarrow cars\_go = FALSE \quad (5)$$

Пространство состояний модели показано на рис. 1.

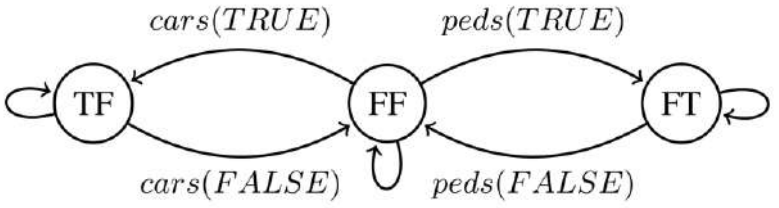


Рис. 1. Пространство состояний модели светофоров.
Fig. 1. Traffic lights state space.

Такая модель не подходит для тестирования, ведь мы не можем напрямую получить информацию о том, двигаются ли пешеходы и автомобили. Мы можем отслеживать только сигналы светофора, поэтому нам необходимо добавить контекст модели. Контекст будет содержать константы red и $green$, обозначающие разные цвета сигналов:

$$red \in COLORS \quad (6)$$

$$green \in COLORS \quad (7)$$

$$red \neq green \quad (8)$$

Чтобы получить возможность реагировать на сигналы светофора, в уже существующие события нужно добавить новый параметр $colors$. Этот параметр соответствует множеству загоревшихся на светофоре сигналов, что обозначено в охранном условии 9. При этом параметр go в трассе будет отсутствовать, и вывод о его значении приходится делать из охранных условий 10–11. Охранные условия 9–11 нужно добавить в оба события.

$$colors \subseteq COLORS \quad (9)$$

$$green \in colors \Rightarrow go = TRUE \quad (10)$$

$$green \notin colors \Rightarrow go = FALSE \quad (11)$$

В данной модели наличие зеленого света разрешает движение независимо от того, какие еще сигналы горят в этот момент на светофоре. Несмотря на это неудобное для пешеходов и водителей допущение, модель является полной и непротиворечивой. Но при попытке ее исполнить мы неизбежно столкнемся со следующими сложностями.

Рассмотрим трассу $\{cars(\{green\}),\ cars(\{red\}),\ peds(\{green\})\}$. Трасса соответствует модели только в том случае, если в начальном состоянии переменная $peds\_go$ имеет значение $FALSE$: состояния FF и TF на рис. 1. Но мы также допустили, что начальное состояние может быть FT. С этой ситуацией мы столкнемся в рамках Event-B, если будем

использовать действия с неопределенными присваиваниями. Всего в языке существует три вида присваиваний: простое присваивание, произвольный выбор значения из множества возможных значений и произвольный выбор значения, удовлетворяющего предикату. Так как произвольный выбор значения по предикату – единственный способ компактно записать конструкцию if-then-else, нежелательно ограничиваться только простым присваиванием и не рассматривать неопределенные присваивания вообще.

Если мы посмотрим на данные с реального светофора, мы сможем наблюдать события $cars(\{yellow\})$, $cars(\{red,\ yellow\})$ и $cars(\emptyset)$. Значение множества $COLORS$ следует из набора предикатов логики первого порядка 6–8. Эти предикаты вполне допускают наличие дополнительных цветов на светофоре, так что значение множества, как совокупности всех его элементов, остается неопределенным. Это важная проблема, так как предикаты – единственный способ определить значение констант в Event-B, кроме того, существуют и другие обстоятельства, когда значение нужно вывести из предиката. Иногда существует только одно решение, удовлетворяющее предикату, но для множества $COLORS$ из примера существует бесконечное количество решений. Кроме того, модель не содержит упомянутый в трассе идентификатор $yellow$. С точки зрения модели желтый вполне может оказаться не дополнительным цветом, а тем же самым красным (тогда участникам движения надо стоять), зеленым (тогда нужно двигаться) или вообще не быть цветом (тогда в трассе кроется ошибка). Эта проблема важна, так как на практике мы не можем описать каждую сущность, например, каждый файл или пользователя в модели в виде отдельной константы.

Как уже упоминалось, параметр go в трассе будет отсутствовать. Эта проблема имеет корни в выразительных возможностях языка Event-B. Единственный способ вычислить некоторое промежуточное значение – добавить еще один параметр к событию и ограничить его значение через охранные условия. Такой параметр мы называем вычислимым. Очевидное решение проблемы, не использовать вычислимых параметров вовсе, ведет к ухудшению читаемости модели. Есть и еще одна сторона данной проблемы. С точки зрения синтаксиса модели случай, когда трасса не является полной из-за ошибки на этапе сбора информации, не отличим от случая, когда значение параметра нужно вычислить из охранных условий.

Возможность абстрагироваться от лишних деталей является основным преимуществом моделирования. Например, в модели светофоров мы абстрагировались от наличия дополнительных цветов и оставили множество $COLORS$ неопределенным полностью. Подобный прием в теории может понадобиться и для других типов данных, например, для чисел. Тогда во время анализа трассы важное свойство должно быть проверено без использования точного значения. Но это также может оказаться спорной ситуацией: абстракция неотличима от случайной неточности в спецификации.

4. Другие работы

Набор инструментов ProB [9] включает в себя инструмент, называемый аниматором. Аниматор позволяет выполнять события Event-B модели, значения параметров при этом ограничиваются или точно определяются при помощи дополнительных предикатов. Ядро ProB написано на языке SICStus Prolog, поэтому ProB активно использует возможности логического программирования для того, чтобы подобрать подходящее значение для каждого объекта. На вид логических формул при этом не накладывается никаких формальных ограничений. Аниматор может дать пользователю возможность самому выбрать определенное значение, если от этого зависит следующее состояние модели. Трассы аниматора нелинейны и могут содержать ветви в разные состояния. К сожалению, использовать аниматор для больших тестовых случаев невозможно, так как он не может работать с большими наборами значений, длинными трассами и большим количеством переменных. Это является следствием выбранного метода и приоритетов разработчиков ProB: поддержка всех конструкций языка важнее, чем эффективность.

Существуют работы по комбинированию SMT-солверов [10-11] с Event-B моделями для их исполнения. SMT-солверы активно используются на отдельных частях модели в процессе дедуктивной верификации, но при исполнении всей модели целиком они сталкиваются со значительными трудностями. Не все солверы поддерживают теорию множеств. Также солверы не могут поддерживать предикаты над таким объектом, как множество всех подмножеств конечного множества. В этом случае формула, задающая объект, выходит за рамки логики первого порядка [12]. Таким образом, SMT-солверы могут быть использованы не на всей модели и, как и логические программы, имеют проблемы с временной эффективностью.

Наиболее эффективным с точки зрения времени выполнения является подход генерации кода [13]. Event-B модели транслируются [14-16] в исполняемую программу на императивном языке, которая может быть запущена. Несложно передать такой программе на вход трассу и обработать ее. Но существующие инструменты генерации кода не рассматривают проблему неопределенности: точные значения объектов, в том числе констант, должны передаваться в конфигурационных файлах.

Таким образом, не существует эффективного решения, покрывающего все потребности тестирования на основе формальной Event-B модели.

5. Текущие результаты

5.1 Пользовательские типы

В Event-B объекты могут иметь логический тип (как элементы встроенного булевого множества), целочисленный тип (как элементы встроенного целочисленного множества), пользовательский тип (как элементы определяемого пользователем множества, называемого в Event-B несущим множеством), являться упорядоченной парой (элемент декартова произведения двух множеств) или множеством (как элемент множества всех подмножеств базового типа). Event-B нотации всех типов приведены в табл. 1. Примером пользовательского типа является $COLORS$ в разделе 3. Объекты пользовательских типов не имеют никакого явного значения и могут быть связаны с другими объектами того же типа только через отношения равенства и неравенства. Примером такого отношения может быть предикат 8. Неопределенность значения следует уже из природы этих объектов.

В качестве значения такого объекта можно взять пару из имени его несущего множества (типа) и некоторого натурального числа. В табл. 2 показан пример интерпретации несложного набора предикатов. Метод выбора целого числа рассматривается в следующем подразделе.

Табл.1. Типы объектов Event-B.
Table 1. Event-B object types.

Тип	**Нотация Event-B**
Логический	$x \in BOOL$
Целочисленный	$x \in \mathbb{Z}$
Пользовательский	$x \in A$
Упорядоченная пара	$x \in A \times B$
Множество	$x \in \mathbb{P}(A)$

Этот подход довольно похож на тот, что используется в ProB. Множества (в том числе несущие) представлены в ProB в виде списков без повторяющихся элементов. Значением

объекта пользовательского типа является конкатенация из имени несущего множества и индекса элемента в соответствующем списке. В примере из таб. 2 символьные значения для x и y будут $A1$ и $A2$, соответственно. Данная нотация удобна для подстановки в трассу новых объектов, не объявленных в модели в качестве констант.

Табл.2. Интерпретация пользовательских типов.
Table 2. Interpretation of user-given type.

Event-B	**Интерпретация**	**Решение**
$x \in A$ $y \in A$ $z \in A$ $x \neq y$ $x = z$	$GIVEN(A,i) \in A$ $GIVEN(A,j) \in A$ $GIVEN(A,k) \in A$ $GIVEN(A,i) \neq GIVEN(A,j)$ $GIVEN(A,i) = GIVEN(A,k)$ $x = GIVEN(A,i)$ $y = GIVEN(A,j)$ $z = GIVEN(A,k)$	$i = 1$ $j = 2$ $k = 1$

Предложение A.1. *Для всех объектов пользовательских типов, значение которых должно явно отличаться от значений заданных в модели констант, в трассе используется идентификатор, состоящий из имени типа и целого числа.*

Альтернативный подход широко используется при генерации кода, когда вместо несущего множества можно использовать тип данных. Это значительно упрощает работу с большими несущими множествами, так как не требует хранения в памяти всех возможных значений. Но это может усложнять обработку логических формул: проверка принадлежности несущему множеству превращается в проверку типа данных, а выражения над несущими множествами (например, «все цвета, кроме красного») необходимо переписывать в другой форме. Разумеется, возможен и описанный выше подход с обычными множествами.

В SMT-солверах также можно использовать оба подхода. Можно считать, что все объекты пользовательского типа являются строкой и использовать в качестве значения таких объектов реальные данные (например, для объекта типа «пользователь» значением будет имя пользователя в системе). Это упрощает процесс тестирования, не требуя отображения имен. Несущим множеством будет множество из всех строк. Также можно создать тип данных (так называемый сорт) и значение объекта будет комбинацией из типа и целого числа.

5.2 Вычисление значения из предиката

В Event-B объекты могут обозначаться идентификаторами, например, константы и переменные. В случае переменных значение объекта определяется через действие присваивания. Но константы и вычисляемые параметры событий определяются иначе: их значение необходимо вывести из набора предикатов.

Значение объекта, на которое указывает идентификатор, может быть точно определено, если оно задается простым предикатом равенства. Из предикатов 12–14 легко извлечь точные значения объектов.

$$NUMBER = 20 \tag{12}$$

$$2 \mapsto TRUE = PAIR \tag{13}$$

$$SET = \{A, B\} \tag{14}$$

Но в некоторых случаях предикаты ведут к неопределенным значениям. Из предиката 15 следует, что $NUMBER$ – любое число, меньшее либо равное 20. Предикат 16 указывает, что $PAIR$ – любая упорядоченная пара из множества REL, представляющего из себя некое

отображение. В предикатах 17–18 множество SET содержит элементы A, B и может содержать любые другие элементы того же типа.

$$NUMBER \leq 20 \tag{15}$$

$$PAIR \in REL \tag{16}$$

$$A \in SET \tag{17}$$

$$B \in SET \tag{18}$$

Далее мы называем предикаты, из которых делается вывод о значении объекта, определяющими. Если определяющий предикат является равенством, в котором один из операндов – идентификатор объекта, то второй операнд мы называем определяющим выражением. Для спецификаций систем, подходящих для тестирования, мы должны ограничить виды определяющих предикатов, чтобы избежать неопределенности.

Предложение B.1. *Для всех логических и целочисленных объектов, а также для упорядоченных пар, значение которых следует из предикатов, определяющий предикат должен иметь форму равенства между идентификатором объекта и определяющим выражением.* Так, для $NUMBER$ и $PAIR$ мы допускаем определяющие предикаты 12–13 и запрещаем определяющие предикаты 15–16. Определяющие выражения могут содержать другие идентификаторы, пока все идентификаторы точно определены и в их определениях нет циклов:

$$NUMBER \mapsto TRUE = PAIR \tag{19}$$

$$NUMBER = 2 \tag{20}$$

Предложение B.2. *Для каждого объекта пользовательского типа должно быть задано отношение равенства или неравенство со всеми остальными объектами того же типа.* Так как у объектов пользовательского типа в рамках модели отсутствует какое-либо значение, необходимо знать, во-первых, какие объекты равны друг другу, во-вторых, для всех остальных объектов необходимо явное указание на их различие. В примере из табл. 2 это требование соблюдено. В таком случае мы можем пронумеровать все объекты, учитывая отношения равенства. Именно так конструируется решение, указанное в последнем столбце табл. 2. В дополнение к равенствам и неравенствам, в современных версиях Event-B существует предикат раздела (partition). Он позволяет выразить неравенство объектов в более компактной форме:

$$partition(S, S1, S2) \Leftrightarrow (S1 \cup S2 = S) \wedge (S1 \cap S2 = \emptyset) \tag{21}$$

Если несущее множество S содержит четыре константы A, B, C и D, гораздо удобнее записать в формате раздела, чем прописывать шесть отдельных неравенств для каждой пары из A, B, C и D:

$$partition(S, \{A\}, \{B\}, \{C\}, \{D\}) \tag{22}$$

Множества также могут быть определены через предикат равенства, но в некоторых случаях это невозможно. Для такого определения требуется перечислить все элементы множества, при этом для каждого элемента необходимо ввести значение, а в случае пользовательского типа данных – идентификатор соответствующей константы. Если мощность множества велика, это неудобно, если вообще возможно. Поэтому необходимо добавить альтернативный способ.

Предложение B.3. *Определяющие предикаты для множеств должны быть предикатами равенства из Предложения B.1, либо предикатами из табл. 3.* Набор определяющих предикатов может быть рассмотрен, как задача теории множеств. Для этой задачи будет существовать общее решение, но при этом точное значение множества останется неопределенным. Предполагая изначально, что множество SET является пустым, мы можем обработать предикаты и построить минимальное частное решение. Все остальные предикаты,

в которых фигурирует множество SET, могут быть использованы, чтобы убедится, что минимальное решения является корректным в рамках модели. Эту же логику можно применить к отображениям и функциям, которые могут быть рассмотрены, как множества упорядоченных пар.

Табл.3. Определяющие предикаты для множеств.
Table 3. Defining predicates for sets.

Предикат	**Решение**
$x \in SET$	$\{x\} \subseteq SET \subseteq \Omega^*$
$s \subseteq SET$	$s \subseteq SET \subseteq \Omega$
$partition(SET, s1, s2)$	$SET = s1 \cup s2$
$partition(s1, SET, s2)$	$SET = s1 \setminus s2$

* Ω обозначает множество всех объектов того же типа, что и элементы SET.

Минимальное решение может оказаться некорректным из-за предикатов, не входящих в множество определяющих. Такой случай проиллюстрирован предикатами 23–24. Необходимо дополнить множества S и REM новыми элементами, чтобы выполнить условие из предиката 24.

$$partition(S, \{A\}, \{B\}, REM) \tag{23}$$

$$REM \neq \emptyset \tag{24}$$

Также необходимо полностью заполнить множества, которые фигурируют в присваиваниях: отсутствие даже одного элемента в таких множествах приведет к некорректному состоянию модели. Для того чтобы дополнить множество, необходимо знать его точную мощность и факт, что оно конечно.

Предложение B.4. *Все множества, появляющиеся в присваиваниях или требующих дополнения для выполнения аксиом, должны быть конечными и их мощность должна быть точно определена.* Примером необходимого определения является предикат 25.

$$finite(SET) \wedge card(SET) = 20 \tag{25}$$

Во всех остальных случаях в качестве значения можно оперировать минимальным решением, динамически расширяя его по мере того, как в трассе появляются новые элементы.

В настоящее время мы выбираем, дополнить ли множество новыми элементами заранее или расширять его динамически уже в ходе анализа трассы, по тому, определена ли его точная мощность. Но этот подход не учитывает распространенный случай, когда мощность указана точно, но при этом велика, а в предварительном дополнении множества нет никакой необходимости. В таком случае динамическое расширение этого множества полезно для повышения эффективности. Одна из будущих целей этого исследования – реализация алгоритма, который может обнаруживать такие множества и оптимизировать вычисления над ними.

5.3 Неопределенные присваивания

В Event-B существует три вида действий присваивания: простое присваивание, произвольный выбор из множества возможных значений, произвольный выбор значения, удовлетворяющего предикату. Из всех видов только простое присваивание лишено неопределенности. В качестве примеров можно рассмотреть присваивания 26–31, которые полностью аналогичны определяющим предикатам 12–18.

$$NUMBER \coloneqq 20 \tag{26}$$

$$PAIR \coloneqq 2 \mapsto TRUE \tag{27}$$

$$SET \coloneqq \{A, B\} \tag{28}$$

$$NUMBER :| NUMBER' \leq 20 \tag{29}$$

$$PAIR :\in REL \tag{30}$$

$$SET :| A \in SET' \wedge B \in SET' \tag{31}$$

Произвольный выбор из множества значений будет лишен неопределенности только в случае, когда множество выбора содержит одно и только одно значение. В таком виде данный вид присваивания полностью лишен смысла. В случае неопределенности алгоритм анализа, тем не менее, понятен: мы должны проанализировать остаток трассы многократно, каждый раз выбирая следующее из возможных значений. Если произошла некая ошибка, это означает, что возможно, выбор был некорректным и в тестируемой системе он был сделан иначе. Необходимо вернуться в точку выбора и изменить его.

Несмотря на прозрачность процесса, долгие возвраты и экспоненциальный рост количества комбинаций с каждым новым выбором делают такой анализ очень неэффективным. Кроме того, если мы рассматриваем тестирование, то подобные неточности в спецификации неизбежно вызовут вопросы. По этим причинам мы не допускаем произвольный выбор из множества в любой его форме.

Предложение C.1. *Произвольный выбор из множества значений недопустим.*

Присваивание значения, удовлетворяющего предикату, в большинстве случаев также является неопределенным. При этом такое присваивание – единственный способ реализации конструкции if-then-else в рамках Event-B. Поэтому вместо запрета необходимо сформулировать ограничения. Если сформулирован предикат p, соответствующий условию, и две альтернативы A и B, соответствующие итоговым значениям переменной в зависимости от выполнения условия p, то присваивание можно записать следующим образом:

$$var \coloneqq p \wedge var' = A \vee \neg p \wedge var' = B \tag{32}$$

Эту форму можно расширить для любого числа условий, пока выполняются следующие условия. Во-первых, дизъюнкция всех условий должна быть тождественна истине (условие полноты). Полнота гарантирует, что всегда найдется хотя бы одно итоговое значение для переменной. Во-вторых, конъюнкции всех возможных пар условий должны быть невыполнимы. Это условие определенности, гарантирующее, что возможно только одно итоговое значение. В примере выше условием полноты будет формула $p \vee \neg p \Leftrightarrow \top$, а условием определенности $p \wedge \neg p \Leftrightarrow \bot$. Оба условия будут выполнены.

При этом в событии, которое содержит присваивание, могут быть охранные условия. Предположим, что охранное условие g – единственное. В таком случае мы можем ослабить условие полноты и не рассматривать те случаи, которые заведомо противоречат охранному условию. Тогда присваивание и условие полноты примут следующий вид:

$$var \coloneqq p \wedge var' = A \vee q \wedge var' = B \tag{33}$$

$$(g \Rightarrow p \vee q) \Leftrightarrow \top \tag{34}$$

Продолжая эти рассуждения, мы можем, наконец, сформулировать общий случай для n альтернативных значений и k охранных условий.

Предложение C.2. *Присваивание значения, удовлетворяющего предикату, допустимо, если оно имеет вид 35, удовлетворяет условию полноты 36 и условию определенности 37. Любые другие формы данного вида присваивания запрещены.*

$$var : | \bigvee_{i=1}^{n} (p_i \wedge var' = Val_i) \tag{35}$$

$$((\bigwedge_{j=1}^{k} g_j) \Rightarrow (\bigvee_{i=1}^{n} p_i)) \Leftrightarrow \top \tag{36}$$

$$\forall i, j \cdot 0 < i < j \leq n: p_i \wedge p_j \Leftrightarrow \bot \tag{37}$$

5.4 Отсутствующие параметры

Как упоминалось в разделе 3, значения некоторых параметров, осмысленных в рамках модели, могут отсутствовать в тестируемой системе и должны быть вычислены из охранных условий. С точки зрения анализа трассы эти параметры попросту отсутствуют, что усложняет автоматическое вынесение вердикта в этих случаях.

Предложение D.1. *Если значение параметра может отсутствовать в трассе, оно должно быть однозначно вычислимо из охранных условий.*

В этом случае, как и прежде, требуем точных значений этих параметров, но не требуем формул специального вида. В настоящее время мы решаем эту проблему при помощи медиатора – компонента, который переводит трассу из набора собранных в тестируемой системе данных в термины модельных сущностей. Этот компонент специфичен для каждой пары из модели и реализации. Медиатор позволяет провести анализ трассы, абстрагируясь от конкретной реализации системы. На этот же компонент можно переложить задачу по вычислению и добавлению в трассу отсутствующих ранее параметров.

Предложение D.2. *Значения всех вычислимых параметров должны быть вычислены до анализа корректности трассы.*

Снова рассмотрим пример со светофорами из раздела 3. Трасса выполнения подается на вход медиатору. Медиатор должен подставить в трассу характерные для модели идентификаторы red и $green$, если на светофоре зажглись соответствующие сигналы. Если зажегся желтый сигнал, то в соответствии с Предложением A.1 в трассу будет подставлен идентификатор $COLORS3$. Также, если зажегся зеленый сигнал, то значение параметра go станет $TRUE$ согласно охранному условию 10, в противном случае в трассе окажется значение $FALSE$ согласно охранному условию 11.

Необходимо учитывать, что таким образом мы добавляем в трассу свойства, которые отсутствуют в наблюдаемом поведении. При неточной реализации медиатора это может сказаться на корректности вынесенного тестового вердикта. Этот подход усложняет разработку медиатора и может породить новые ошибки в процессе тестирования.

С другой стороны, такой подход увеличивает эффективность анализа и учитывает природу вычислимых параметров. При таком подходе вычислимый параметр явно отличается от просто отсутствующего, что помогает обнаруживать ошибки этапа сбора трассы.

Предложение D.3. *Если при анализе трассы охранное условие не может быть проверено из-за отсутствующего параметра, тест не пройден.* Данное решение сужает класс корректных трасс, поэтому для некоторых целей анализа оно может быть спорным.

Автоматическое вычисление отсутствующих параметров является целью дальнейших исследований.

5.5 Оптимизации

Для тестирования данного подхода был разработан прототип инструмента анализа трасс. В качестве языка программирования был выбран Java, так как для этого языка уже реализован набор библиотек для работы с Event-B моделями, что удобно для прототипа. Реализация,

использующая предложенный подход и делающая непосредственные вычисления всех объектов показала лишь небольшое ускорение по сравнению с аниматором ProB. Тем не менее, выяснилось, что прототип, не смотря на простейшую реализацию, все еще работал быстрее. Для достижения требуемых показателей эффективности анализа было необходимо реализовать некоторые оптимизации.

Предложение E.1. *Множества всех множеств, декартово произведение, множества отображений и функций должны иметь представление, отличное от обычных множеств.* В целях оптимизации стоит отказаться от хранения таких множеств в памяти и ввести для них дополнительное представление. Для этого также необходимо интерпретировать некоторые формулы иначе. Например, предикат 38 может быть переписан в более простой форме 39. Трансформация формул происходит автоматически.

$$SUBSET \in \mathbb{P}(SET) \tag{38}$$

$$SUBSET \subseteq SET \tag{39}$$

Предложение E.2. *При обработке кванторов необходим автоматический анализ формулы, нацеленный на ограничение области значений подкванторных переменных.* Измерения времени работы инструмента показали, что решение кванторных формул является наиболее затратной по времени задачей. Если добавить в анализатор дополнительные правила обработки таких формул, ограничивая множество перебираемых значений, то обработка формулы ускоряется на порядок.

Предложение E.3. *Проверка инвариантов должна быть опциональной.* Инварианты являются большим набором предикатов, который нужно проверять после выполнения каждого события в трассе. Инварианты часто содержат кванторы, так что даже с оптимизациями их проверка занимает значительную часть времени анализа.

При этом если анализ трассы проводится на основе ранее верифицированной модели, в проверке инвариантов нет никакой необходимости. Для верифицированной модели выполнение инвариантов логически следует из выполнения охранных условий. Все, что действительно требуется для проверки корректности состояния – проверить выполнение охранных условий. Разумеется, при работе с произвольными моделями этого недостаточно, но для повышения эффективности необходима опция отключения проверки инвариантов. В ProB такая возможность отсутствует.

Если модель не верифицирована, возможный способ оптимизации – проверять только те инварианты, которые содержат измененные последним событием переменные.

6. Будущие исследования

В настоящий момент все трансформации модели выполняются автоматически на уровне индивидуальных формул. Не было предложено техник трансформации моделей вручную в целях повышения эффективности тестирования. Необходимо провести более обширный анализ существующих практик создания формальных моделей. В настоящий момент основным ориентиром является модель контроля доступа и потока информации в ОС Linux [5, 17], но требуются эксперименты и с другими моделями.

Ограничения, предложенные в подразделе 5.2, являются строгими, но разумными, если речь идет о моделях, используемых в процессе тестирования. В случае, когда эти ограничения все же невозможно соблюдать, возможным решением может быть использование SMT-солверов на определенных частях модели. Например, это может оказаться полезно для подбора значений констант, так как в рамках одной модели это достаточно сделать один раз и в это никак не повлияет на скорость последующего анализа трасс. Необходимо провести эксперименты и оценить эффективность такого подхода.

Текущая версия инструмента не поддерживает отложенных вычислений, если не считать динамически расширяемых множеств. В примере из раздела 3 вполне возможно определить

неопределенное начальное состояние модели по первым событиям. Этот подход может быть использован, чтобы достичь в моделях большего уровня абстракции.

Желаемый результат данного исследования – достичь скорости выполнения тестов, максимально близкой к генерации кода. Тестирование прототипа инструмента анализа трасс на модели системы безопасности ОС Linux и 39000 тестовых трассах показало, что все трассы проверяются за 8 минут по сравнению с десятичасовым анализом с использованием ProB. Созданный вручную на языке Python исполняемый аналог модели позволил выполнить тот же набор тестов за 1 минуту.

7. Заключение

Данная статья посвящена исследованию проблем, которые возникают в процессе тестирования формальных систем, таких как системы безопасности операционных систем общего назначения, в частности, Linux. При этом необходимо построить формальную модель политики безопасности для того, чтобы определить требования к системе и иметь возможность доказать их непротиворечивость и полноту. Верифицированная модель политики безопасности используется при тестировании реализации системы для анализа корректности наблюдаемого поведения. Но этот анализ требует большого количества ресурсов и в некоторых случаях встречается с непреодолимыми препятствиями.

Это исследование выявило конкретные проблемы, связанные с неопределенностью значений. Мы обозначили подмножество языка Event-B, позволяющее избежать неопределенности и предложили подход для вычисления определенных значений и эффективного анализа трасс. Результаты исследования могут быть применены к другим формальным языкам моделирования, использующим логику первого порядка.

Также мы реализовали предложенный метод в прототипе инструмента анализа трасс. Прототип был протестирован на большом наборе тестовых трасс и реальной модели системы безопасности и показал значительно ускорение по сравнению с аниматором ProB. Этот анализ может быть выполнен непрерывно в рамках одного пространства состояний модели, что демонстрирует поддержку инструментом длинных тестовых трасс.

Список литературы / References

[1]. Bartocci E., Falcone Y., Fracalanza A., Reger G. Introduction to runtime verification. In Bartocci E., Falcone Y. (editors) Lectures on Runtime Verification. Lecture Notes in Computer Science, vol. 10457, Springer, 2018. pp. 1–33. DOI: 10.1007/978-3-319-75632-5_1.

[2]. Reger G., Havelund K. What Is a Trace? A Runtime Verification Perspective. In Margaria T., Steffen B. (editors) Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications. Lecture Notes in Computer Science, vol. 9953, Springer, 2016. DOI: 10.1007/978-3-319-47169-3_25.

[3]. ГОСТ Р 59453.1. Защита информации. Формальная модель управления доступом. Часть 1. Общие положения: описание стандарта и тендеры. – Введен 2021-06-01 – Москва: Стандартинформ, 2021. / State Std. GOST R 59453.1. Information protection. Formal access control model. Part 1. General principles. Moscow, 2021 (in Russian).

[4]. ГОСТ Р 59453.2. Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификации формальной модели управления доступом. – Введен 2021-06-01 – Москва: Стандартинформ, 2021. / State Std. GOST R 59453.1. Information protection. Recommendations on verification of formal access control model. Part 2. General principles. Moscow, 2021 (in Russian).

[5]. Девянин П. Н., Ефремов Д. В., Кулямин В. В., Петренко А. К., Хорошилов А. В., Щепетков И. В. Моделирование и верификация политик безопасности управления доступом в операционных системах. Горячая линия – Телеком, Москва, Россия, 2019. / Devyanin P.N., Efremov D. V., Kulyamin V. V., Petrenko A. K., Khoroshilov A. V., Shchepetkov I. V. Modeling and verification of access control security policies in operating systems. Moscow, Russia: Hotline-Telecom, 2019 (in Russian).

[6]. Ефремов Д. В., Копач В. В., Корныхин Е. В., Кулямин В. В. , Петренко А. К., Хорошилов А. В., Щепетков И. В. Мониторинг и тестирование модулей операционных систем на основе абстрактных моделей поведения системы. Труды ИСП РАН, 2021, том 33, вып. 6, стр. 15–26. DOI: 10.15514/ISPRAS-2021-33(6)-2. / Efremov D. V., Kopach V. V., Kornykhin E. V., Kulyamin V. V., Petrenko A.K., Khoroshilov A. V., Shchepetkov I. V. Runtime verification of operating systems based on abstract models. Proceedings of ISP RAS, 2021, vol. 33, no. 6, pp. 15–26 (in Russian). DOI: 10.15514/ISPRAS-2021-33(6)-2.

[7]. Девянин П. Н., Леонова М. А. Приёмы описания модели управления доступом ОССН Astra Linux Special Edition на формализованном языке метода Event-B для обеспечения её верификации инструментами Rodin и ProB. ПДМ, 2021, № 52, стр. 83–96. DOI: 10.17223/20710410/52/5. / Devyanin P. N., Leonova M. A. The techniques of formalization of OS Astra Linux Special Edition access control model using Event-B formal method for verification using Rodin and ProB. Prikladnaya Diskretnaya Matematica, 2021, no. 52, pp. 83–96, (in Russian). DOI: 10.17223/20710410/52/5.

[8]. Abrial J.-R., Butler M., Hallerstede S., Hoang T., Mehta F., Voisin L. Rodin: an open toolset for modelling and reasoning in Event-B. The International Journal on Software Tools for Technology Transfer, vol. 12. Springer, 2010, pp. 447–466. DOI: 10.1007/s10009-010-0145-y.

[9]. Leuschel M., Butler M. ProB: an automated analysis toolset for the B method. The International Journal on Software Tools for Technology Transfer, vol. 10, Springer, 2008, pp. 185–203. DOI: 10.1007/s10009-007-0063-9.

[10]. Déharbe D. Integration of SMT-solvers in B and Event-B development environments. Science of Computer Programming, vol. 78, 2013, pp. 310– 326. DOI: 10.1016/j.scico.2011.03.007.

[11]. Schmidt J., Leuschel M. SMT solving for the validation of B and Event-B models. The International Journal on Software Tools for Technology Transfer, vol. 24, Springer, 2022, pp. 1043–1077. DOI: 10.1007/s10009-022-00682-y.

[12]. Brauer E. Second-order logic and the power set. Journal of Philosophical Logic, vol. 47, 2018, pp. 123–142. DOI: 10.1007/s10992-016-9422-x.

[13]. Fürst A., Hoang T., Basin D., Desai K., Sato N., Miyazaki K. Code generation for Event-B. Presented at the Integrated Formal Methods 2014, Bertinoro, Italy, 09 2014.

[14]. Wright S., Automatic generation of C from Event-B. Presented at the Workshop on Integration of Model-based Formal Methods and Tools, Bangkok, Thailand, 02 2009.

[15]. Yang F., Jacquot J.-P., Souquières J. JeB: Safe simulation of Event-B models. Presented at the The 20th Asia-Pacific Software Engineering Conference, Bangkok, Thailand, 12 2013. DOI: 10.1109/APSEC.2013.83

[16]. Cataño N., Rivera V., EventB2Java: A code generator for Event-B. In Rayadurgam, S., Tkachuk, O. (editors) NASA Formal Methods. Lecture Notes in Computer Science, vol. 9690, Springer, 2016. DOI: 10.1007/978-3-319-40648-0_13.

[17]. Девянин П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Горячая линия – Телеком, Москва, Россия, 2013. / Devyanin P. N. Security models of computer systems. Control for access and information flows. Moscow, Russia: Hotline-Telecom, 2013, (in Russian).

Информация об авторах / Information about authors

Алексей Александрович КАРНОВ — аспирант. Научные интересы: формальные спецификации, верификация и тестирование, статический и динамический анализ.

Aleksei Aleksandrovich KARNOV — postgraduate student. Research interests: formal specifications, verification and testing, static and dynamic analysis.

DOI: 10.15514/ISPRAS-2024-36(4)-14

Идентификация термокарстовых объектов по спутниковым графическим данным с помощью нейронной сети

В.В. Жебсаин, ORCID: 0000-0002-2976-8721 <zhebs@mail.ru>
А.Ф. Посельский, ORCID: 0009-0002-2602-8044 <al.poselsky@gmail.com>

Северо-Восточный федеральный университет им. М.К. Аммосова,
Россия, 677027, Якутск, ул. Белинского, 58.

Аннотация. В работе представлены результаты численного эксперимента по идентификации термокарстовых объектов, образовавшихся в результате климатических изменений в регионах распространения вечной мерзлоты, на основе спутниковых графических данных. Разработана прикладная компьютерная программа, предназначенная для идентификации спутниковых графических данных реализующая трехслойную нейронную сеть. Идентификация термокарстовых объектов прикладной программой производилась на основе метода обучения нейронной сети с использованием тренировочных данных. В качестве функции активации в нейронной сети применялась сигмоидальная функция, корректировка весовых коэффициентов сети основывалась на методе обратного распространения ошибок. Изучена зависимость эффективности идентификации объектов от различных начальных параметров нейронной сети, таких как, скорость обучения, количество нейронов в скрытом слое и количество эпох обучения. Выявлены оптимальные значения вышеуказанных параметров, обеспечивающих наибольшие показатели эффективности нейронной сети. Проведено сравнение полученных результатов с данными других исследователей. В целом, результаты проведенного в работе исследования показали перспективность рассмотренного метода для развития автоматизированных средств дистанционного мониторинга термокарстовых процессов.

Ключевые слова: нейронные сети; термокарстовые образования; программирование; численные эксперименты.

Для цитирования: Жебсаин В.В., Посельский А.Ф. Идентификация термокарстовых объектов по спутниковым графическим данным с помощью нейронной сети. Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 183–190. DOI: 10.15514/ISPRAS–2024–36(4)–14.

Благодарности. Исследование выполнено при поддержке Российского научного фонда (проект № 24-21-20043).

Identification of Thermokarst Objects from Satellite Graphical Data Using a Neural Network

V.V. Zhebsain, ORCID: 0000-0002-2976-8721 <zhebs@mail.ru>
A.F. Poselsky, ORCID: 0009-0002-2602-8044 <al.poselsky@gmail.com>

M. K. Ammosov North-Eastern Federal University,
58, Belinsky st., Yakutsk, Republic of Sakha (Yakutia), 677027, Russia.

Abstract. The paper presents the results of a numerical experiment on the identification of thermokarst objects formed as a result of climatic changes in the regions of the cryolithozone, based on satellite graphical data. An applied computer program has been developed designed to identify satellite graphical data implementing a three-layer neural network. The dependence of the object identification efficiency on various initial parameters of the neural network, such as the learning rate, the number of neurons in the hidden layer and the number of learning epochs, has been studied. The optimal values of the above parameters have been identified, providing the highest efficiency indicators of the neural network. The results obtained were compared with the data of other researchers.

Keywords: neural networks; thermokarst formations; programming; numerical experiments.

For citation: Zhebsain V.V., Poselsky A.F. Identification of Thermokarst Objects from Satellite Graphical Data Using a Neural Network. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 4, 2024. pp. 183-190 (in Russian). DOI: 10.15514/ISPRAS-2024-36(4)-14.

Acknowledgements. The investigation was conducted under the help of the Russian Science Foundation (project No 24-21-20043).

1. Введение

Как известно, область практического применения программных средств, основанных на технологии нейронных сетей постоянно расширяется. Развиваются новые подходы и методы, усовершенствуются существующие решения. В данной работе представлены итоги работы по разработке прикладной компьютерной программы, реализующей нейронную сеть, а также её первоначальные результаты применения для решения практических задач по идентификации термокарстовых полигональных образований, находящихся на ранней стадии развития, на основе космических графических данных. Причина образования полигональных термокарстовых объектов заключается в повсеместном повышении температуры мерзлых пород в северных регионах [1-3], как следствие глобального потепления климата. Обзор научно-практической литературы, показал, что несмотря на значительное количество работ, посвященных дистанционному зондированию термокарстовых полигональных образований, вопросы автоматизированной идентификации, основанные на технологии нейронных сетей, не рассматриваются.

Актуальность решения задач идентификации и дешифровки аэрокосмических (спутниковых) графических образов термокарстовых полигональных образований, характерных для северных регионов России, в частности Якутии, обусловлена транспортной труднодоступностью указанных регионов и растущей интенсификацией глобальных климатических изменений.

Последствия деградации ландшафтов в указанных регионах могут быть катастрофическими для инфраструктуры, в частности, для водоводов (рис. 1), трубопроводов, железных дорог, газопроводов и других крупных инженерных сооружений, а также жилых и производственных зданий.

2. Компьютерная программа

Для реализации нейронной сети, при помощи сред разработки программного обеспечения PyCharm (Python) и графического интерфейса Qt Designer разработана прикладная

компьютерная программа. В программе реализована трехслойная нейронная сеть с переменным количеством узлов (нейронов), предназначенная для решения задачи классификации (идентификации) на основе обучения тренировочному набору данных.

Рис. 1. Термокарстовые полигональные образования (объекты) на трассе водовода Лена – Туора Кюель – Татта» (п. Майя, вдхр. Мундуллах). Фото А. М. Сальва, 2016 г [1].
Fig. 1. Thermokarst polygonal formations (objects) on the route of the Lena–Tuora Kyuel–Tatta water pipeline" (Maya village, Mundullah reservoir). Photo by A. M. Salva, 2016[1].

На входной слой сети подается матрица I_i цифровых кодов графического образа. Количество элементов данной матрицы определяется размером графического образа, например для изображения 70х70 пикселей матрица состоит из 4900 элементов (нейронов). Данная матрица умножается на матрицу весовых коэффициентов W_{i+1}, генерируемых случайным образом.

$$X_{i+1} = W_{i+1} \cdot I_i \tag{1}$$

где I_i – матрица входного сигнала;

W_{i+1} – матрица весовых коэффициентов;

X_{i+1} – матрица входного сигнала скрытого слоя.

Таким образом, на первый скрытый слой подается матрица Xi+1. В качестве функции активации применяется сигмоида. В результате получаем выходной сигнал скрытого слоя в виде:

$$O_j = \frac{1}{1+e^{-x_{i+1}}}; \tag{2}$$

В случае трехслойной сети, выходной сигнал скрытого слоя подается на выходной слой сети. Таким образом, выходной сигнал сети определяется как:

$$O_k = W_{jk} \cdot O_j \tag{3}$$

где O_k – матрица выходных сигналов k слоя (выходного);

O_j – матрица выходных сигналов предыдущего j слоя (скрытого);

W_{jk} – матрица весовых коэффициентов между слоями j и k;

В случае сети со множеством скрытых слоев, матрица выходного слоя выполняет роль следующего скрытого слоя. Алгоритм обучения и расчета нейронной сети основан на методе обратного распространении ошибок и корректировке весовых коэффициентов. В процессе

обучения сети, производится минимизация функции ошибки E_k при помощи коррекции весовых коэффициентов [4, 5]:

$$\Delta W_{jk} = \alpha \cdot \frac{\partial E_k}{\partial W_{jk}} \tag{4}$$

где, α – коэффициент обучения;

$\frac{\partial E_k}{\partial W_{jk}}$ – градиент функции ошибки.

Здесь, ΔW_{jk} – матрица коррекции весовых коэффициентов, которая определяется при помощи выражения:

$$\Delta W_{jk} = \alpha \cdot E_k \cdot O_k \cdot (1 - O_k) \cdot O_j^T; \tag{5}$$

где, $E_k = (t_k - O_k)^2$ - матрица функции ошибки, t_k - целевые (правильные) значения узлов выходного слоя сети;

O_j^T – транспонированная матрица сигналов скрытого слоя.

Таким образом, в программе реализован стандартный метод решения классификационной задачи нейронной сетью. Программа разработана в виде прикладного средства и оснащена для удобства пользователя графическим интерфейсом (рис. 2).

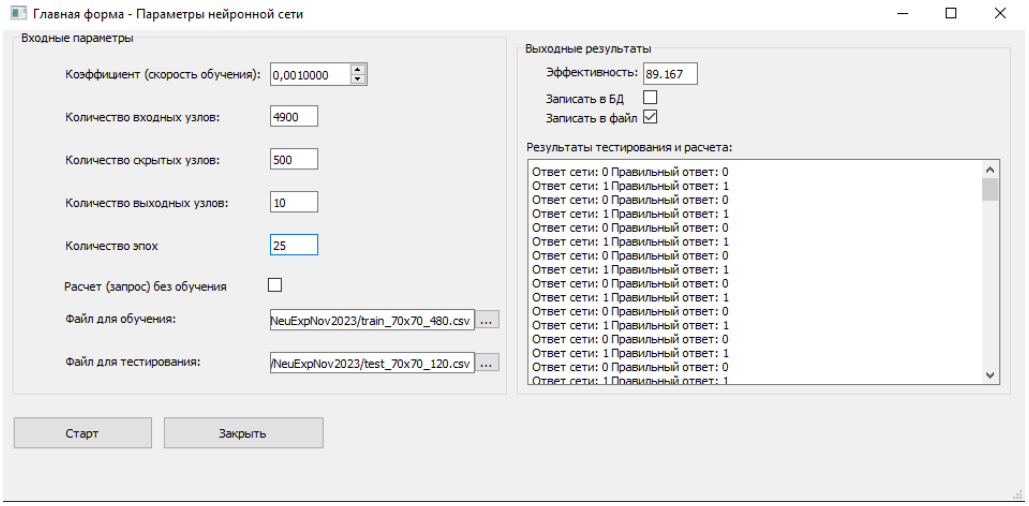


Рис. 2. Главное окно программы.
Fig. 2. The main window of the program.

В программе существует возможность записи результатов расчетов на внешний текстовый файл, а также сохранения значений весовых коэффициентов отдельных нейронов в базу данных SQLite.

3. Обучение сети

Для обучения сети были подготовлены тренировочные наборы данных, состоящие из 480 оцифрованных графических образов, некоторые образцы которых приведены на рис. 3. На данном рисунке образцы 1, 3 и 4 имеют характерные признаки термокарстовых полигональных образований, представленных также на рис.1. Остальные образцы на рис. 3 являются примерами ландшафтов из той же местности, не имеющих признаков деградации.

Графические образы для тренировочных и тестовых наборов сформированы на основе спутниковых данных Google и географически относятся к Чурапчинскому району Республики Саха (Якутия), где в настоящее время наблюдается активная деградация ландшафтов из-за интенсивных климатических изменений последних десятилетий. Обучающий набор состоял из предварительно обработанных и оцифрованных фотоснимков в цветовой палитре оттенков серого.

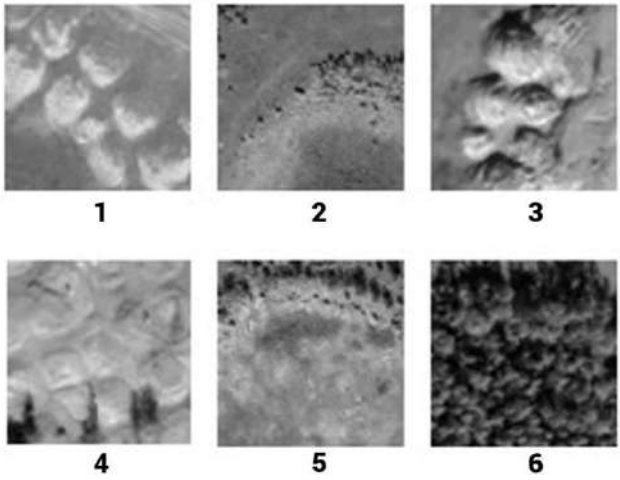


Рис. 3. Примеры графических образов из тренировочного набора данных.
Fig. 3. Examples of graphical images from the training dataset.

4. Результаты численного эксперимента

Суть численного эксперимента состояла, в том, чтобы обученную нейронную сеть испытывать на тестовом наборе данных, с целью выявления зависимости эффективности от различных начальных параметров. Тестовый набор данных, состоял из 120 случайных образов, содержащих термокарстовые объекты, так и не содержащих их. Критерием качества обучения сети служила эффективность, определенная как процент количества правильных ответов, относительно общего количества образов тестового набора. Изучение зависимости эффективности сети при идентификации данных тестового набора от скорости обучения показало, что максимальная эффективность наблюдается при скорости обучения в интервале значений 0.001-0.003, независимо от количества нейронов в скрытом слое. Зависимость эффективности сети от скорости обучения при различных количествах нейронов в скрытом слое приведена на рис. 4.

Получено, что количество эпох обучения безусловно оказывает влияние на эффективность сети, максимальные значения которой наблюдаются при количестве эпох в диапазоне 19-20, достигая значений 90-92 %. Дальнейшее увеличение количества эпох не приводит к повышению эффективности и далее происходит её постепенное снижение (рис. 5).

Исследование зависимости эффективности сети от количества нейронов в скрытом слое показало, что при скорости обучения 0.001 и количестве эпох 10 наибольшая эффективность наблюдается при количестве нейронов в диапазоне 3500-3750, достигая значения в 92%. Указанная зависимость представлена на рис. 6.

5. Заключение

Как показал численный эксперимент, эффективность нейронной сети при идентификации тестовых термокарстовых объектов варьирует в пределах 88-92 %. Изучение зависимости эффективности идентификации от количества эпох показало, что наибольшая эффективность

достигается при количестве эпох обучения сети 19-25, далее происходит ухудшение результатов (рис. 5). Исследование зависимости эффективности от скорости обучения показало, что максимальная результативность сети достигается при значениях скорости обучения в пределах 0,002-0,003.

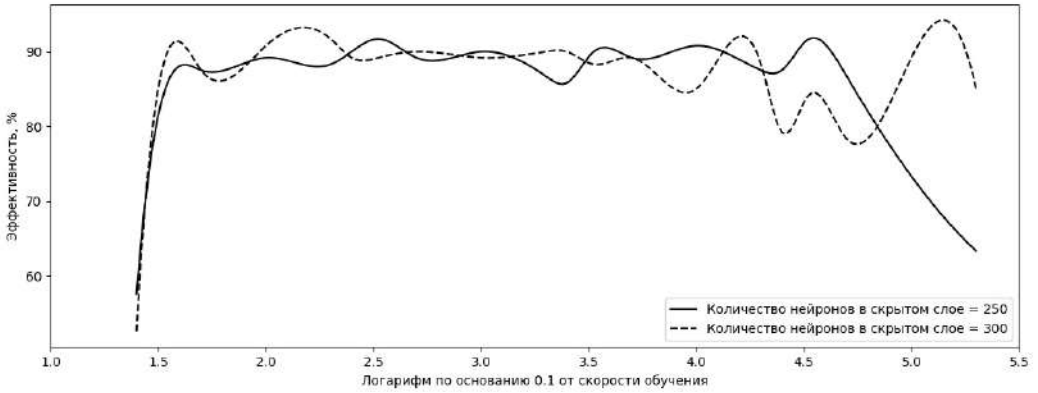


Рис. 4. Зависимость эффективности от скорости обучения.
Fig. 4. Dependence of efficiency on learning speed.

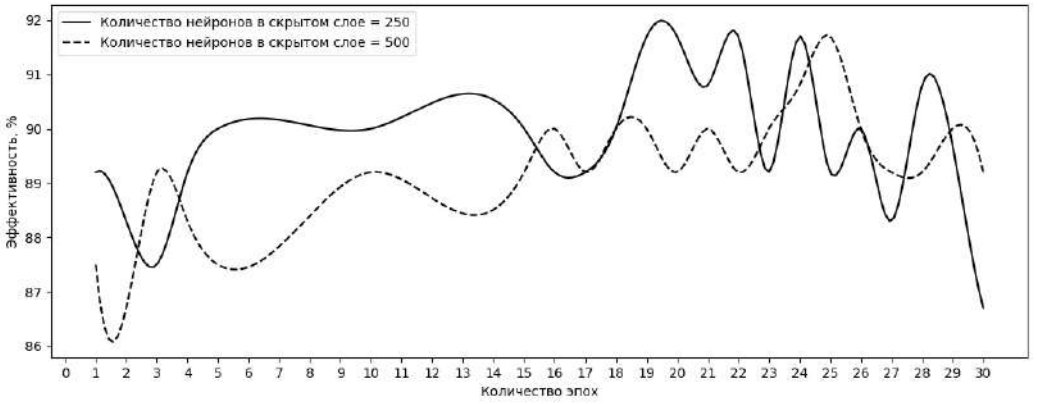


Рис. 5. Зависимость эффективности от количества эпох.
Fig. 5. Dependence of efficiency on the number of epochs.

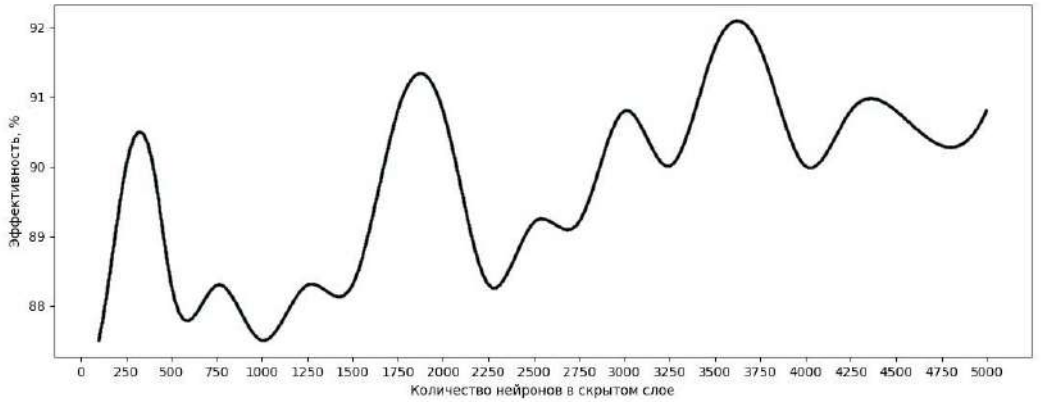


Рис. 6. Зависимость эффективности от количества нейронов в скрытом слое.
Fig. 6. Dependence of efficiency on the number of neurons in the hidden layer.

По вопросу подбора оптимального количества нейронов в скрытом слое существует ряд рекомендаций. В частности, в работе [5], приводится следующая рекомендация по оптимальному количеству нейронов скрытого слоя:

$$N_h = \frac{2 \cdot N_i}{3} + N_o \tag{6}$$

где N_i – количество нейронов входного слоя;

N_o – количество нейронов выходного слоя.

В нашем случае, согласно соотношению (6), наибольшая эффективность должна была наблюдаться при N_h = 3277, но по результатам проведенных нами исследований максимальная эффективность сети приходится на количество нейронов в скрытом слое в диапазоне значений 3500-3750.

В целом, результаты исследования показали перспективность рассмотренного метода идентификации графических образов термокарстовых объектов, основанного на технологии нейронных сетей для развития автоматизированных средств дистанционного мониторинга термокарстовых процессов.

Список литературы / References

[1]. Сальва А.М. Отслеживание участков термокарстовых проявлений по космическим снимкам (на примере трассы магистрального водовода в центральной Якутии)//Арктика и Антарктика. 2020. №2. с. 126-137.

[2]. Konstantinov P., Zhelezniak M., Basharin N., Misailov I., Andreeva V.//Land (MDPI, Basel, Switzerland, 2020). 2020. №9 (12). p. 1-10.

[3]. Н.И.Башарин, Л.С. Егорова, Н.Ф. Васильев, Н.А. Федоров, А.Н. Федоров // Вестник Северо-Восточного федерального университета. 2020. №3 (19), с. 36-44.

[4]. Тарик Рашид. Создаем нейронную сеть. СПб. ООО «Альфа книга». 2017. 272 с.

[5]. Heaton Jeff. Heaton Research. The Number of Hidden Layers (online). https://www.heatonresearch.com/2017/06/01/hidden-layers.html. 12/06/2024.

Информация об авторах / Information about authors

Василий Васильевич ЖЕБСАИН – кандидат физико-математических наук, заведующий кафедрой «Радиофизика и электронные системы» Физико-технического института Северо-Восточного федерального университета им. М.К. Аммосова с 2015 года. Сфера научных интересов: технологии нейронных сетей, информационные технологии, разработка программного обеспечения, технологии баз данных, проблемы и вопросы дистанционного зондирования Земли.

Vasiliy Vasilievich ZHEBSAIN – Cand. Sci. (Phys.-Math,), Head of the Department of Radiophysics and Electronic Systems of the Institute of Physics ad Technology of the North-Eastern Federal University since 2015. Research interests: neural network technologies, information technologies, software development, database technologies, problems and issues of remote sensing of the Earth.

Айаал Федорович ПОСЕЛЬСКИЙ является аспирантом кафедры «Радиофизика и электронные системы» Физико-технического института Северо-Восточного федерального университета им. М.К. Аммосова. Его научные интересы включают технологии нейронных сетей, информационные технологии, разработка программного обеспечения, анализ данных.

Ayal Fedorovich POSELSKY is a postgraduate student of the Department of Radiophysics and Electronic Systems of the Institute of Physics ad Technology of the North-Eastern Federal University. His research interests include neural network technologies, information technology, software development, data analysis.

DOI: 10.15514/ISPRAS-2024-36(4)-15

К моделированию задачи осаждения твердой частицы в вязкой несжимаемой жидкости методом гидродинамики сглаженных частиц (SPH)

И.И. Потапов, ORCID: 0000-0002-3323-2727 <potapov2i@gmail.com>
О.В. Решетникова, ORCID: 0000-0002-0099-3589 <ov13r@yandex.ru>

Вычислительный центр Дальневосточного отделения Российской академии наук, Россия, 680000, г. Хабаровск, ул. Ким-Ю-Чена, д. 65

Аннотация. В работе сформулирована и решена задача определения гидравлической крупности для единичной частицы. Для решения задачи предложена оригинальная разновидность метода сглаженных частиц (SPH), в котором осаждаемая частица оказывает влияние на движение окружающих ее частиц воды, но расчет сил, действующих на частицу (кроме силы, учитывающих эффекты присоединенной массы), выполняется по формулам аналитической механики для материальной точки. Верификация предложенной математической постановки и реализующего ее алгоритма расчета выполнена с использованием открытого авторского кода «SPH_Lab2d». Получены зависимости для гидравлической крупности частицы от ее диаметра для различных случаев дискретизации объема жидкости. Численные результаты демонстрируют хорошую согласованность с экспериментальными данными и известными феноменологическими зависимостями для кварцевого песка.

Ключевые слова: гидравлическая крупность, метод SPH, составное ядро, радиус сглаживания, открытое программное обеспечение**.**

Для цитирования: Потапов И.И., Решетникова О.В. К моделированию задачи осаждения твердой частицы в вязкой несжимаемой жидкости методом гидродинамики сглаженных частиц (SPH). Труды ИСП РАН, том 36, вып. 4, 2024 г., стр. 191–202. DOI: 10.15514/ISPRAS–2024–36(4)–15.

On Modeling the Grain Settling through Viscous Incompressible Fluid Problem using Smoothed Particle Hydrodynamics Method

I.I. Potapov, ORCID: 0000-0002-3323-2727 <potapov2i@gmail.com>
O.V. Reshetnikova, ORCID: 0000-0002-0099-3589 <ov13r@yandex.ru>

Computer center of Far East Branch of the Russian Academy of Sciences
65 Kim U Chena Street, Khabarovsk City, 680000 Russian Federation

Abstract. In this paper was formulates the settling velocity determining problem for a single particle. To solve it, an original version of the smoothed particle method (SPH) is proposed where settling particle affects on surrounding fluid particles movement. Herewith the calculation of forces acting on particle (except for inertial forces which takes attached mass effect) is performed according by analytical mechanics formulas for a material point. This mathematical formulation and calculation algorithm was verificated by using open source code «SPH_Lab2d». Dependences for the particle settling velocity on its diameter are obtained for various cases of fluid volume discretization. The results demonstrate a good conformance this property to values, that determined by the experimental data and known phenomenological dependences for quartz sand.

Keywords: settling velocity, mesh-free SPH method, composite kernel, smoothing radius, open source software.

For citation: Potapov I. I., Reshetnikova O. V. On modeling the grain settling through viscous incompressible fluid problem using smoothed particle hydrodynamics method**.** Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 4, 2024. pp. 191-202 (in Russian). DOI: 10.15514/ISPRAS-2024-36(4)-15.

1. Введение

Моделирование русловых процессов находит множество практических применений – от проектирования конструкций и судоходных трасс до прогнозирования чрезвычайных ситуаций и их последствий. Важным механизмом изменения формы русла является процесс движения донных частиц, математическое описание которого одним из параметров включает среднюю скорость падения (осаждения) твердой частицы в стоячей воде.

Движение донных частиц (взвешивание и осаждение) характеризуется наличием сильнодеформируемых границ – донной поверхности, а также свободно перемещаемых разрозненных (в том числе и единичных) гранул твердой фракции в расчетном объеме.

В настоящее время при моделировании процессов взвешивания и осаждения донных частиц доминируют сеточные методы [1] разработанные для решения задач механики сплошных сред. Существуют многочисленные примеры таких исследований с более чем вековой историей [2]. Однако классические сеточные методы сталкиваются с известными трудностями при описании воздушно-песчаных или водно-песчаных смесей высокой концентрации [3], которые обусловлены необходимостью построения уравнений состояния многофазной среды и сложностью верификации таких уравнений.

Следует отметить, что разностные или проекционные сеточные методы разработаны для решения задач механики сплошных сред. Но сыпучий материал (песок) как дискретная среда имеет характерные пространственные и временные масштабы, ограничивающие возможности его описания уравнениями механики сплошных сред, поскольку не всегда оказывается выполнимым предельный переход, когда сыпучесть среды приводит к хаотизации процессов развития донной неустойчивости [4, 5, 6] (хорошо известные донные волны). По этой причине использование проекционно-сеточных методов для численной реализации моделей движения высоконаполненных водно-песчаных смесей требует формулировки сложных реологических моделей, для валидации которых часто отсутствуют экспериментальные данные.

Авторам на данный момент не известны математические модели и соответственно сеточные методы, позволяющие корректно моделировать данные явления. Определенное

преимущество здесь получают бессеточные методы моделирования водно-песчаных смесей, поскольку позволяют не выполнять предельного перехода для гранулометрической фазы, т.е. не рассматривать ее как сплошную среду. В настоящее время ведутся активные исследования в данном направлении [7], например возможностями платформы «OpenFOAM 2.0.0: Particle Tracking» исследуются комбинации сеточных методов для описания гидродинамики и DEM методов для описания твердой фазы.

Также известны работы по моделированию водогрунтовых потоков методом SPH. Однако здесь для случаев движения единичных гранул твердой фракции в жидкости, или просто незначительных ее объемов (в сравнении с расчетным) имеет место нефизичность получаемого решения [8] в связи с ошибками определения плотности фазы при использовании классического SPH-подхода. Поэтому зачастую предлагается не учитывать в решении (относить к погрешности) случаи обособленного расположения гранул в объеме жидкости, а рассматривать движение только макрообъемов твердой фазы. В работе [9], посвященной моделированию всплывания пузырьков газа в жидкости (задача, обратная осаждению) означенную проблему предлагается компенсировать присвоением фракции газа повышенной на несколько порядков плотности, чтобы она составляла не менее одной десятой доли от плотности жидкости. Понятно, что модели с подобными параметрами также неадекватны физической природе процессов.

Особенностью данной работы является попытка описания жидкой среды и движения в ней твердых гранул SPH методом, оставаясь в рамках механики сплошных сред. Для гранулы твердого материала здесь используется комбинация походов классической механики материальной точки и SPH подхода для учета обратного ее влияния на жидкую среду. Авторы на данный момент ведут исследовательский поиск возможностей такого подхода, решая различные модельные верификационные задачи.

2. Постановка задачи

В работе исследуется процесс осаждения сферической твердой частицы в вязкой ламинарной тяжелой несжимаемой жидкой среде. Расчетная область задачи Ω и ее границы Γ_i, приведены на рис. 1.

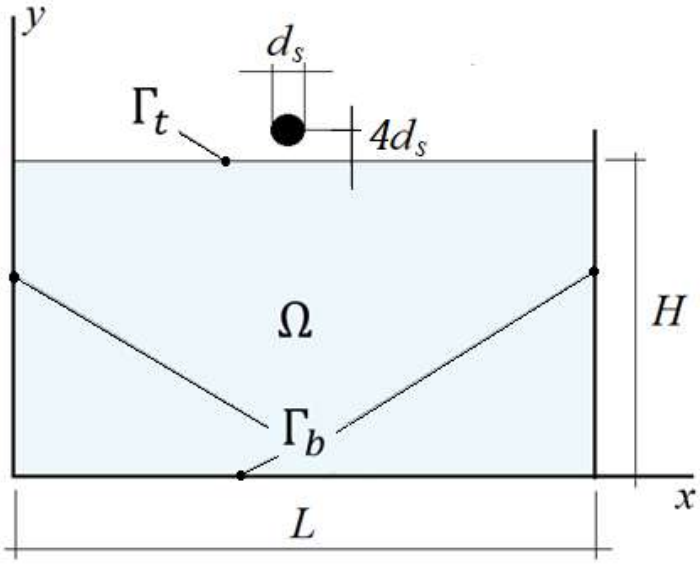


Рис. 1. Представление расчетной области задачи.
Fig. 1. Task computational domain presentation.

Математическая модель задачи включает уравнения Навье-Стокса, описывающие жидкую среду

$$\rho_f \frac{d\boldsymbol{u}_f}{dt} + \nabla P = \mu\, \Delta \boldsymbol{u}_f + \rho_f\, \boldsymbol{g}, \tag{1}$$

$$\nabla \cdot \mathrm{u_f} = 0, \tag{2}$$

а также уравнения для оседающей частицы [2]

$$m_s = const. \tag{3}$$

$$m_s \frac{d\boldsymbol{u}_s}{dt} = \boldsymbol{F}^{st} + \boldsymbol{F}^{m} + \boldsymbol{F}^{a}. \tag{4}$$

Здесь ρ_f и ρ_s – плотности жидкости (индекс «f») и оседающей частицы (индекс «s») соответственно; $\boldsymbol{u}_f$ и $\boldsymbol{u}_s$ – векторы скоростей жидкости и оседающей частицы; P – давление в жидкости; ∇ – оператор Гамильтона; m_s – масса оседающей частицы; $\boldsymbol{F}^{st}$ – сила Стокса для частицы; $\boldsymbol{F}^{m}$ – сила от присоединенной массы; $\boldsymbol{F}^{a}$ – архимедова сила (суммарная сила от действия тяготения и выталкивания); μ – динамическая вязкость жидкости; $\boldsymbol{g} = (0, -g)$ – вектор ускорения свободного падения.

Выражения для сил, действующих на оседающую частицу, имеют вид

$$\boldsymbol{F}^{a} = m_s \left(1 - \frac{\rho_f}{\rho_s}\right) \boldsymbol{g}, \tag{5}$$

$$\boldsymbol{F}^{st} = 18 \frac{V_s}{d_s^2} \mu \left(\boldsymbol{u}_f - \boldsymbol{u}_s\right) = 18 \frac{m_s}{d_s^2 \rho_s} \mu \left(\boldsymbol{u}_f - \boldsymbol{u}_s\right), \tag{6}$$

$$\boldsymbol{F}^{m} = -\nabla E_f, \qquad E_f = \frac{(u_s)^2}{2} \rho_f \int \left(\frac{u_f}{u_s}\right)^2 dV_f\,. \tag{7}$$

Здесь V_s – объем частицы; d_s –диаметр частицы, $u_S = |\boldsymbol{u}_S|,\ u_f = |\boldsymbol{u}_f|$.

Сила от присоединенной массы $\boldsymbol{F}^{m}$ выражается через кинетическую энергии жидкости E_f, увлекаемой оседающей частицей при ее ускоренном движении. Поскольку масса является величиной скалярной, фактический знак $\boldsymbol{F}^{m}$ определяется знаком ускорения частицы.

Уравнения (1)-(4) замыкаются начальными (здесь $\boldsymbol{x}$ – координаты точки в пространстве)

$$\boldsymbol{u}_f(\boldsymbol{x}, 0) = 0,\ \ \boldsymbol{x} \in \Omega; \tag{8}$$

$$\boldsymbol{x}_s(\boldsymbol{x}, 0) = \boldsymbol{x}_{0s}(\boldsymbol{x}, 0),\ \ \boldsymbol{u}_s(\boldsymbol{x}, 0) = 0,\ \ \boldsymbol{x} \in \Omega, \tag{9}$$

и граничными условиями

$$\boldsymbol{u}_f(\boldsymbol{x}, t) = 0,\ \ \boldsymbol{x} \in \Gamma_b\,. \tag{10}$$

$$\frac{d\boldsymbol{u}_f}{dy}(\boldsymbol{x}, t) = 0,\ \ \boldsymbol{x} \in \Gamma_t\,. \tag{11}$$

3. Дискретизация задачи методом SPH

Идея метода гидродинамики сглаженных частиц (SPH) основана на представлении расчетного объема конечным набором лагранжевых частиц, каждая из которых обладает объемом, массой и является носителем физических свойств (плотности, скорости, давления и т.д.). Величина некоторого свойства SPH-частицы в точке пространства определяется «сглаженным» вкладом от близлежащих частиц-соседей через весовую функцию – ядро сглаживания W (далее – ядро). Для численного выражения значения свойства в SPH-частице используются аппроксимации, получаемые из следующих базовых форм [10]:

$$f_i = \sum_j \frac{m_j}{\rho_j} f_j W(r_{ij}, H),$$

$$\nabla f_i = \sum_j \frac{m_j}{\rho_j} f_{ij} \nabla W(r_{ij}, H).$$

Здесь $f_i = f(x_i^\alpha)$ – значение функции свойства f, аппроксимируемое в частице i с координатами x_i^α ($\alpha = 1, \dots, D$; D – размерность пространства задачи); f_{ij} – интерполяция функции f на отрезке между точками x_i^α и x_j^α по двум соответственным значениям f_i и f_j; m_j и ρ_j – соответственно масса и плотность частицы, находящейся в области сглаживания; $W(r_{ij}, H)$ – ядро, удовлетворяющая условиям четности $W(r_{ij}, H) = W(-r_{ij}, H)$ и компактности $W(r_{ij}, H) = 0$, $r_{ij} \geq H$; $\nabla W(r_{ij}, H)$ – производная для ядра; $r_{ij} = \sqrt{\sum_\alpha (x_j^\alpha - x_i^\alpha)^2}$ – расстояние между частицами i и j; H – размер области сглаживания, определяемый произведением сглаживающей длины h и безразмерного радиуса сглаживания R.

Для построения дискретного аналога задачи (1)-(11) методом SPH производится исходное разбиение расчетной области Ω на конечное множество SPH-частиц размерами d_f и d_s. Дискретный аналог для уравнений (1)-(2) Навье-Стокса формулируется в рамках классического подхода [10, 11] и определяется уравнением движения для каждой из множества SPH-частиц, представляющих расчетный объем жидкости. Далее по тексту такая частица жидкости обозначена индексом «i», и тем же индексом снабжено аппроксимируемое по методу SPH переменное свойство в ней. Уравнение движения жидкости

$$\frac{d\boldsymbol{u}_i}{dt} = -m_f \sum_j \left(\frac{P_i + P_j}{\rho_i} + \Pi_{ij} \right) \frac{dW_f(r_{ij})}{dr} \frac{\mathbf{r}_{ij}}{|r_{ij}|} + \boldsymbol{g}, \tag{12}$$

где j – локальный номер любой SPH-частицы внутри области сглаживания; Π_{ij} – член, именуемый искусственной вязкостью [10, 11]

$$\Pi_{ij} = \begin{cases} -\delta c_0 h \frac{\mathbf{r}_{ij} \cdot (\boldsymbol{u}_i - \boldsymbol{u}_j)}{|r_{ij}|^2}, & \mathbf{r}_{ij} \cdot (\boldsymbol{u}_i - \boldsymbol{u}_j) < 0, \\ 0, & other. \end{cases} \tag{13}$$

Согласно [10, 11] искусственная вязкость (13) может заменять собой действие физической вязкости для задач с малыми числами Маха M. При этом имеет место соотношение $\delta c_0 h \sim 2(D+2)\,\mu/\rho_f$. Для задачи об осаждении частицы $M \ll 1$, поскольку движение общего объема жидкости по условию (10) задачи отсутствует. Следовательно, выражение (12) тождественно записи уравнения (1).

Помимо (12) требуется уравнение сохранения массы для жидкой частицы

$$\rho_i = m_f \sum_j W_f(r_{ij}), \tag{14}$$

а также уравнение состояния, связывающее плотность среды и давление в ней [12]

$$P_i = \begin{cases} \rho_{f_0} c_0^2 \left(\frac{\rho_i}{\rho_{f_0}} - 1 \right), \rho_i > \rho_{f_0}, \\ 0, \qquad \rho_i \leq \rho_{f_0}, \end{cases} \tag{15}$$

где ρ_{f_0} – плотность жидкости в естественном состоянии, c_0 – модельная скорость звука, m_f – масса жидких частиц.

Закон сохранения массы для твердой оседающей частицы (она и ее переменные свойства обозначаются далее индексом «k») определяется уравнением (3) и имеет вид

$$\rho_k = \sum_j m_j W_s(r_{kj}), \tag{16}$$

дополняемый также уравнением состояния для твердой частицы

$$P_k = \begin{cases} \rho_{s_0} c_0^2 \left(\frac{\rho_k}{\rho_{s_0}} - 1\right), \rho_k > \rho_{s_0}, \\ 0, \qquad\qquad \rho_k \le \rho_{s_0}. \end{cases} \tag{17}$$

Дискретный аналог уравнения движения (4) с учетом (5) для оседающей частицы имеет вид

$$\frac{d\boldsymbol{u}_k}{dt} = \boldsymbol{A}_k^{st} + \boldsymbol{A}_k^m + \left(1 - \frac{\rho_{f_0}}{\rho_{s_0}}\right)\boldsymbol{g}, \tag{18}$$

Для выражения $\boldsymbol{A}^{st}$ из (6) используем SPH-аппроксимацию Морриса [13] для вязкой силы, действующей на частицу и учитывающую локальные изменения плотности среды в пределах области сглаживания. Полагая, что коэффициент вязкости жидкости остается неизменным, а сопротивление движению определяется мгновенной плотностью фаз, получим

$$\boldsymbol{A}_k^{st} = 18\,\mu_0 \frac{m_s}{\rho_s} \sum_j \frac{\rho_k + \rho_j}{\rho_k\,\rho_j} \frac{\mathbf{r}_{kj} \cdot \nabla W_{kj}}{|r_{kj}|^2} \left(\boldsymbol{u}_k - \boldsymbol{u}_j\right), \qquad \nabla W_{kj} = \frac{dW_s(r_{kj})}{dr} \frac{\mathbf{r}_{kj}}{|r_{kj}|}, \tag{19}$$

Здесь μ_0 – модельное значение вязкости среды, которое определяется по формуле

$$\mu_0 = \rho_f \frac{c_0 d_f}{Re_d}, \tag{20}$$

где Re_d – число Рейнольдса для тела при заданных параметрах задачи.

Аппроксимация для $\boldsymbol{A}^m$ сформирована исходя из определения выражения кинетической энергии, увлекаемой твердой частицей жидкости. А именно, энергия в объеме жидкости вокруг твердой частицы определяется как сумма энергий в объемах жидких частиц

$$E_k = \frac{1}{2} m_f^2 \sum_j E_j = \frac{1}{2} m_f^2 \sum_j \frac{u_j^2}{\rho_j} W_s(r_{kj}). \tag{21}$$

Градиент энергии выразим в соответствии с базовой аппроксимацией SPH для производной функции

$$\nabla E_k = \frac{m_s}{\rho_s} \sum_j \left(E_s + E_j\right) \nabla W_{kj}. \tag{22}$$

Тогда подставив в (16) вместо E_j выражение (15) и учтя, что энергия твердой частицы есть $E_s = m_s \frac{u_s^2}{2}$, в соответствии с (7) получим

$$\boldsymbol{A}_k^m = -\frac{1}{2\rho_s} \sum_j \left(m_s u_s^2 + m_f^2 \frac{u_j^2}{\rho_j} W_s(r_{kj})\right) \nabla W_{kj}. \tag{23}$$

Для случая дискретизации расчетного объема частицами разного размера для фаз $d_f \neq d_s$, аппроксимации для свойств жидкой фазы (14) и (16) требуют модификации. Здесь необходимо учесть означенное неравенство размеров частиц, находящихся в области сглаживания, для чего вводится коэффициент приведения объема $o_j = \frac{V_s}{V_j}$. А именно

$$\rho_i = m_f \sum_j o_j W_f(r_{ij}), \tag{24}$$

$$\nabla P_i = m_f \sum_j \left(\frac{P_i + P_j}{\rho_i} + \Pi\right) o_j \frac{dW_f(r_{ij})}{dr} \frac{\mathbf{r}_{ij}}{|r_{ij}|}. \tag{25}$$

Важно отметить, что для восстановления мгновенных значений свойств в частицах твердой и жидкой фаз используются разные SPH-ядра. Причем сглаживающая длина h для каждой

фазы равна соответственному размеру SPH-частицы. При двумерном представлении расчетного объема функции W имеют следующий вид

- для жидкой фазы (ядро Монагана [11]) с безразмерным радиусом сглаживания $R = 2$

$$W_f(r_{ij}) = \frac{15}{7\pi d_f^2}\begin{cases}\frac{2}{3} - \left(\frac{r_{ij}}{d_f}\right)^2 + \frac{1}{2}\left(\frac{r_{ij}}{d_f}\right)^3, & 0 \le \frac{r_{ij}}{d_f} \le 2,\\ \frac{1}{6}\left(2 - \frac{r_{ij}}{d_f}\right)^3, & 1 \le \frac{r_{ij}}{d_f} \le 2,\\ 0, & \frac{r_{ij}}{d_f} > 2;\end{cases}$$

- для твердой фазы (составное ядро [15] с $R = 1$), в котором функции $W_s(r_{kj})$ и $\frac{dW_s(r_{kj})}{dr}$ определены независимо друг от друга

$$W_s(r_{kj}) = \frac{15}{13\ d_s^2}\begin{cases}\left(1 - \left(\frac{r_{kj}}{d_s}\right)^2\right)^2\left(1 - \left(4 - \frac{52}{5\pi}\right)\left(\frac{r_{kj}}{d_s}\right)^2\right), & 0 \le \frac{r_{kj}}{d_s} \le 1,\\ 0, & \frac{r_{kj}}{d_s} > 1,\end{cases}$$

$$\frac{dW_s(r_{kj})}{dr} = \frac{264}{665\ \pi d_s^3}\begin{cases}22\frac{r_{kj}}{d_s} - 25, & 0 \le \frac{r_{kj}}{d_s} \le 1,\\ 0, & \frac{r_{kj}}{d_s} > 1.\end{cases}$$

Использование для аппроксимации свойств жидкости и твердой частицы разных ядер обусловлено в первую очередь тем, что ядро с $R = 1$ исключает эффект потери массы даже при отсутствии соседей в области сглаживания. Эта особенность наилучшим образом подходит для моделирования движения гранулированных сред. Кроме того, потеря массы является одной из основных причин получения нефизичных эффектов в численном решении задачи. С другой стороны, ядра с $R \ge 2$ обеспечивают большую гладкость распределения аппроксимируемых свойств в локальном объеме области сглаживания, поэтому такие ядра лучшим образом подходят для построения моделей движения жидкостей.

4. Алгоритм решения задачи

Для решения задачи (1)-(11) использовался алгоритм Верле, рассмотренный в работах [3, 17]. Дискретизация расчетного объема жидкости осуществляется частицами одинакового диаметра d_f. При этом соблюдается плотность исходного заполнения: центры жидких частиц равномерно распределены в пространстве так, что расстояние между любой соседней парой составляет d_f. Осаждаемая частица диаметром d_s, в начальный момент времени размещалась над свободной поверхностью жидкой фазы на расстоянии $4d_s$.

Использование формул (14) и (16), реализующих межфазный обмен массой, требуют пояснений. В процессе движения оседающая частица попадает в области сглаживания окружающих ее жидких частиц. При этом на мгновенное значение плотности в жидкой частице ρ_i масса твердой частицы не оказывает, а в аппроксимации (14) учитывается только ее объем через значение $W_f(r_{ik})$, но не масса.

С другой стороны, при движении оседающей частицы возникает присоединение к ней массы за счет увеличения плотности ρ_k от воздействия окружающих ее частиц жидкой фазы (налипания). Это реализуется в выражении (16) за счет учета различия масс всех соседей в области сглаживания оседающей частицы.

5. Результаты численного моделирования

Верификация численного решения рассматриваемой задачи в постановке (1)-(11) производилась относительно задачи на определение гидравлической крупности частицы. Средняя скорость установившегося осаждения w сферической частицы в жидкости [1] в зависимости от ее диаметра есть

$$w = \frac{\mu}{\rho_s d_s} Re_d, \tag{26}$$

где $Re_d = f(d_b)$ – число Рейнольдса для придонных частиц; d_b – безразмерное выражение диметра частицы $d_b = \frac{\rho_f(\rho_s - \rho_f) g d^3}{\mu^2}$. В [1] приведена зависимость, позволяющая получить Re_d на большом диапазоне d_b. Далее в работе расчетная по (26) гидравлическая крупность обозначена как $w^{(26)}$.

Кроме того, известно множество феноменологических формул для песков конкретной местности в малом диапазоне размеров частиц. В частности, рассмотрена формула Руби [18], согласованная с результатами серии экспериментов на осаждение крупиц кварцевого песка плотностью 2650 $\frac{\text{кг}}{\text{м}^3}$ в воде при температуре 16 °C

$$w^{(27)} = \frac{\mu}{\rho_s d_s} \begin{cases} 6\left(\sqrt{1 + \frac{d_b}{54}} - 1\right), & d_b \le 38500; \\ 1.05\sqrt{d_b}, & d_b > 38500. \end{cases} \tag{27}$$

Также нами для сравнения использованы справочные данные о гидравлической крупности $w^{\text{спр}}$ частиц грунта с плотностью 2650 $\frac{\text{кг}}{\text{м}^3}$ при температуре 15 °C [19].

В работе произведено численное решение задачи на определение гидравлической крупности для четырех значений диаметра d_s круглой падающей частицы (табл. 1). Физические параметры и размеры расчетной области следующие: $\rho_{f_0} = 1000 \frac{\text{кг}}{\text{м}^3}$; $\rho_{s_0} = 2650 \frac{\text{кг}}{\text{м}^3}$; $\mu = 0.001 \frac{\text{кг}}{\text{с м}}$; толщина слоя жидкости $H = 0.103$ м; длина пути свободного падения частицы $4d_f$.

Табл. 1. Значения диаметров частиц и соответствующие им значения гидравлической крупности по результатам численного решения.
Table 1. The particle diameters values and corresponding its settling velocity values according by numerical solution results.

	1	**2**	**3**	**4**
d_s, (м)	0.00100	0.00065	0.00052	0.00040
d_b	25996	7139	3655	1664
Re_d	200	90	60	36
$w^{\text{числ}}$, (м/с)	0.137	0.108	0.094	0.078

Первая серия вычислений производилась при дискретизации обеих фаз частицами одинакового размера, $d_f = d_s$. Результаты численного решения такой постановки задачи на нахождение гидравлической крупности $w^{\text{числ}}$ представлены в табл. 1 и на рис. 2 – 4. Из графиков рис. 2 и 3 (цифра на графике обозначает порядковый номер значения d_s в соответствии с табл. 1) видно, что некоторое время твердая частица движется с ускорением, затем происходит удар о жидкость и резкое уменьшение скорости. Затем имеет место переходный период формирования и схлопывания каверны (снова небольшое ускорение) и период плавного гашения импульса от первоначальной скорости вхождения частицы в жидкость. Примерно с отсечки 0.2 с. частица приобретает в среднем постоянную скорость

осаждения $w^{\text{чсл}}$. Результаты показывают (см. табл. 1 и рис. 4), что значения $w^{\text{чсл}}$ получаются наиболее близкими к вычисляемым по закону Руби $w^{(27)}$.

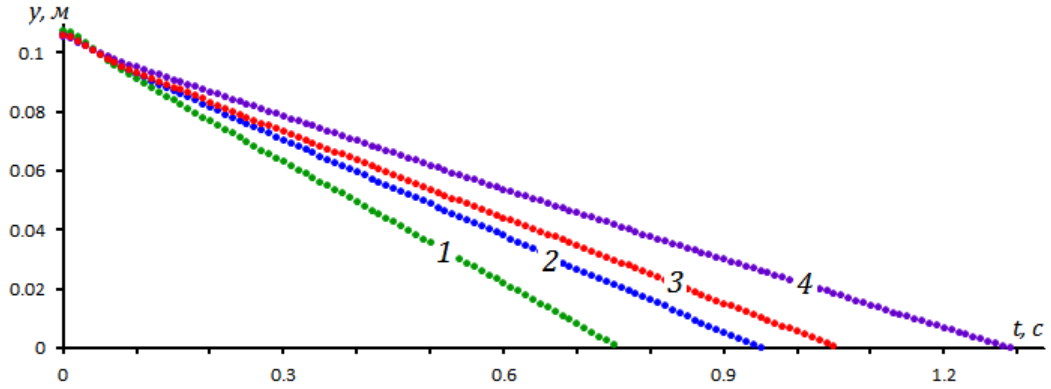


Рис. 2. Изменение вертикальной координаты твердой частицы по времени.
Fig. 2. The solid particle vertical coordinate changing on time.

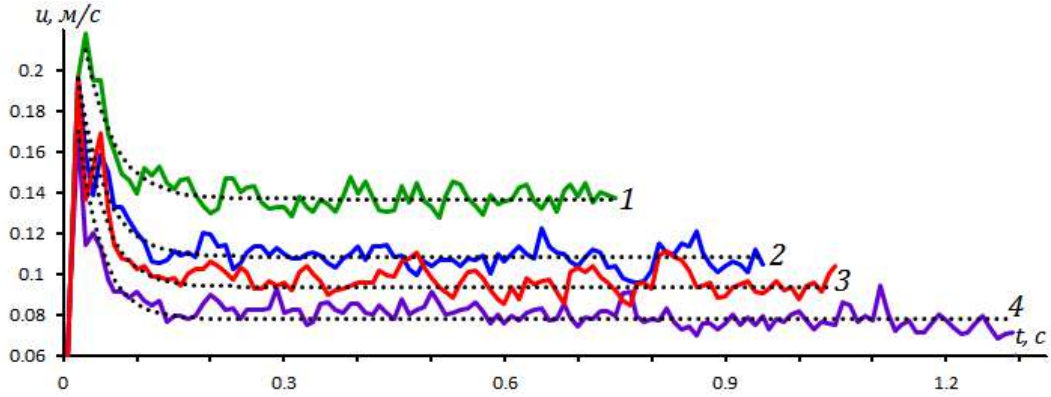


Рис.3. Изменение скорости падения частицы по времени.
Fig. 3. The solid particle falling speed changing on time.

Нахождение значений $w^{\text{чсл}}$ производилось путем анализа полученных временных рядов $u(t)$ на нахождение отклонения их от среднего – тренда $T(t)$. Здесь принято представление его экспоненциальной функцией

$$T(t) = k + (U - k)e^{-at},$$

где U – скорость падения частицы в жидкость; k и a – параметры. При этом параметр k – определяет значение $w^{\text{чсл}}$, a – интенсивность гашения первоначальной скорости U частицы. Выявленные при анализе рядов расчетных данных параметры трендов представлены в табл. 2. Соответственные графики $T(t)$ см. на рис. 2 (пунктирные линии).

Влияния значения диаметра частицы на амплитуду флуктуаций значений расчетных рядов $u(t)$ относительно трендов не прослеживается (рис. 1) – среднеквадратическое отклонение для всех вариантов составляет $\cong 0.006 \frac{\text{м}}{\text{с}}$. Видимое по линиям графиков существенное большее его значение для кривой 3 (см. рис. 2) объясняется случайностью данных ряда $u(t)$ для каждого запуска вычислений. Иногда получались решения с возникновением небольших горизонтальных «планирований» твердой частицы, в результате которых происходило

перемещение ее в позицию с наименьшим сопротивлением со стороны жидкости и последующий ускоренный «провал». Здесь приведен один такой «негладкий» случай с целью более полного представления общих результатов численного решения поставленной задачи.

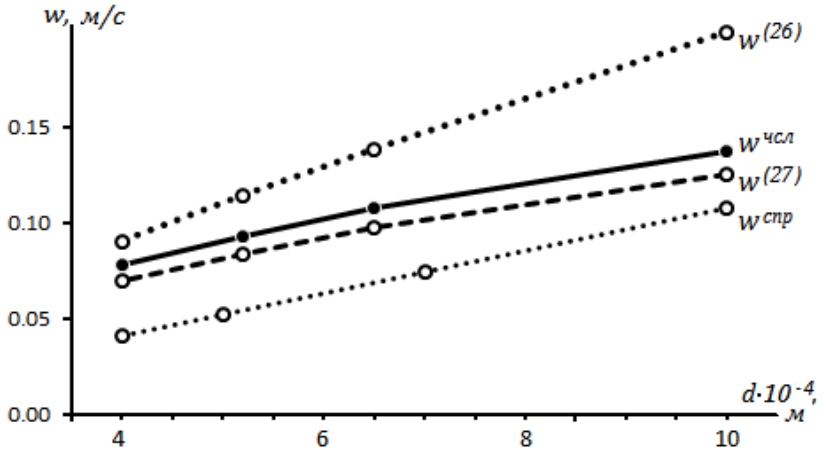


Рис. 4. Зависимости гидравлической крупности частицы от ее диаметра по разным источникам ($w^{(20)}$, $w^{(21)}$, $w^{\text{спр}}$) и для численного решения ($w^{\text{чсл}}$).
Fig. 4. The dependence of particle settling velocity on its diameter form by different sources ($w^{(20)}$, $w^{(21)}$, $w^{\text{спр}}$) and by numerical solution ($w^{\text{чсл}}$).

Табл. 2. Значения параметров функций $T(t)$ для исследуемых вариантов диаметров частиц.
Table 2. The functions T(t) parameters values for the studied particle diameters variants.

	1	**2**	**3**	**4**
k	0.137	0.108	0.094	0.078
a	0.250	0.270	0.310	0.320

Вторая серия вычисления производилась для случаев различных значений отношений $\frac{d_s}{d_f} = \theta$. При этом в первую очередь выявлено, что для получения устойчивого решения должно выполняться условие $1 \leq \theta < 2$. В противном случае не возникает адекватных величин сил гидравлического сопротивления. Такая согласованность размеров SPH частиц для жидкой и твердой фаз, а также ограничение снизу на размер жидкой частицы объясняется значениями радиусов сглаживания R соответственных ядер. Если $d_f < d_s$ – частица жидкости просто не будет попадать в область сглаживания твердой частицы. Если $2d_f < d_s$ – уже наоборот ядро сглаживания жидкой фазы слабо отвечает на вхождение твердой частицы в область сглаживания, то есть наблюдается эффект потери массы и не формируется адекватных величин уплотнения и дивергенции относительной скорости.

Влияние на получаемое решение значения θ представлено на рис. 5: увеличение этого отношения относительно единицы сопровождается уменьшением $w^{\text{чсл}}$. То есть увеличение размера твердой частицы относительно жидкой приводит к увеличению площади воздействия сил гидравлического сопротивления. Это можно рассматривать как добавление влияния шероховатости поверхности твердой частицы в постановку задачи.

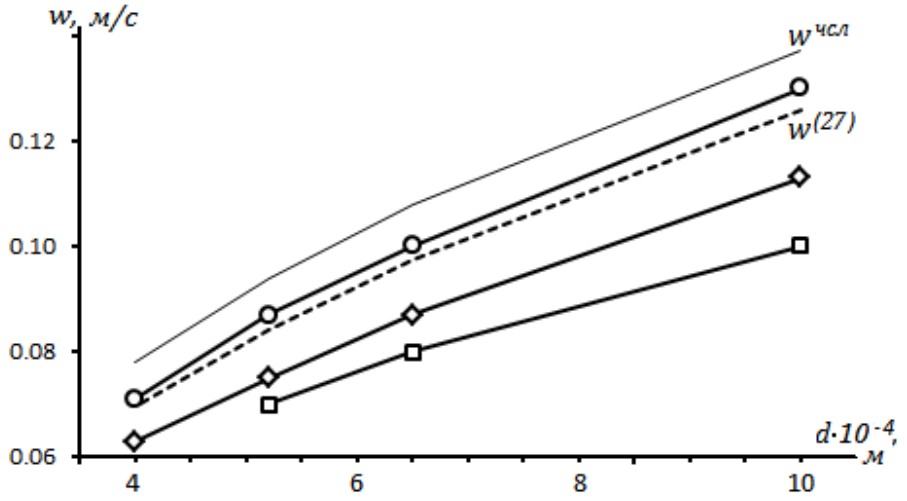


Рис. 5. Влияние отношения θ на гидравлическую крупность частицы в численном решении: $\theta = 1.1$ (линия —○—), $\theta = 1.3$ (линия —◊—) и $\theta = 1.6$ (линия —□—).

Fig. 5. The ratio θ impact on particle settling velocity in numerical solution: $\theta = 1.1$ (line —○—), $\theta = 1.3$ (line —◊—) and $\theta = 1.6$ (line —□—).

Список литературы / References

[1]. Петров А.Г., Потапов И.И. Избранные разделы русловой динамики. – М.: ЛЕНАНД, 2019. – 244 с. / Petrov A.G., Potapov I.I. Selected sections of riverbed dynamics. – Moscow: LENAND, 2019. – 244 P.

[2]. Караушев А.В. Теория и методы расчета речных наносов. – Л.: Гидрометеоиздат, 1977. – 270 с. / Karaushev A. V. Theory and methods of river sediments calculation. Leningrad : Gidrometeoizdat, 1977. 1977. – 270 P.

[3]. Потапов И.И., Решетникова О.В. Применение стационарной гипопластической модели для моделирования движения сыпучей среды // Вычислительные технологии. – 2019. – Т.24, № 6. – С. 90-98. / Potapov I. I., Reshetnikova O. V. The use of a stationary hypoplastic model for modeling the motion of a granular medium // Computing technologies. – 2019. – Vol.24, no 6. – P. 90-98.

[4]. Coleman S.E., Nikora V.I. Initiation and growth of fluvial dunes // Marine and River Dune Dynamics, 1 – 3 April 2008, Leeds, United Kingdom. – P. 43 – 49.

[5]. Sanne L.N. Modeling of sand dunes in steady and tidal flow. Ph.D.-thesis MEK-DTU, Technical University of Copenhagen, Denmark. 2003.– 185 P.

[6]. Tjerry S. Morphological calculations of dunes in alluvial rivers. Ph.D.-thesis ISVA, Technical University of Denmark. 1995. – 193 P.

[7]. Boac, J.M., Ambrose, R.P.K., Casada, M.E. et al. Applications of Discrete Element Method in Modeling of Grain Postharvest Operations // Food Eng Rev. 2014 – Vol. 6. – P. 128-149.

[8]. Szewc K., Pozorski J., Minier J.-P. Simulations of single bubbles rising through viscous liquids using Smoothed Particle Hydrodynamics // International Journal of Multiphase Flow. – 2013. – Vol. 50. P. 98-105.

[9]. Vahabi M., Sadengy K. On the use of SPH method for simulating gas bubbles rising in viscoelastic liquids // Nihon Reoroji Gakkaishi. 2014. – Vol. 42, №. 5. – P. 309-319.

[10]. Monaghan J.J. Shock Simulation by the Particle Method SPH // Journal of computational physics. – 1983. – Vol. 52. – P. 374-389.

[11]. Monaghan J.J. Smoothed particle hydrodynamics and its diverse applications // Annu. Rev. Fluid Mech. – 2012. – Vol. 44. – P. 323-346.

[12]. Molteni D., Colagrossi A. A simple procedure to improve the pressure evaluation in hydrodynamic context using the SPH // Computer Physics Communications. – 2009. Vol. – 180, №. 6. – P. 861-872.

[13]. Morris J.P., Fox P.J., Zhu Y. Modeling low Reynolds number incompressible flows using SPH // Comput. Phys. – 1997. – Vol. 136. – P. 214-226.

[14]. Monaghan J.J. Smoothed particles hydrodynamics // Reports on Progress in Physics. – 2005. – Vol. 68. – P. 1703-1759.

[15]. Потапов И. И., Решетникова О. В. Исследование влияния двух геометрических параметров на точность решения гидростатической задачи методом гидродинамики сглаженных частиц // Компьютерные исследования и моделирование. – 2021. – Т. 13, № 5. – С. 979-992. / Potapov I. I., Reshetnikova O. V. The two geometric parameters influence study on the hydrostatic problem solution accuracy by the SPH method // Computer research and modeling. – 2021. – Vol. 13, no 5. – P. 979-992.

[16]. Потапов И.И., Решетникова О.В. Реализация модели движения сыпучей среды методом сглаженных частиц // Информатика и системы управления. – 2019. – Т. 62, № 4. – С. 26-34. / Potapov I. I., Reshetnikova O. V. About the SPH- implementation of the bulk medium motion model // Information Science and Control Systems. – 2019. – Vol. 62, no 4. – P. 26–34.

[17]. Rubey W. W. Setting velocities of gravel, sand and silt particles // American journal of science. 1933. Vol. 25. P. 325-338.

[18]. Справочник по гидравлике / Под ред. В.А. Большакова. – 2-е изд., перераб. и доп. – К.: Вища шк. Головное изд-во, 1984. – 343 с. / Handbook of Hydraulics / Et al. V.A. Bolshakov. – 2nd et. – Kiev: Higher education. Main publishing, 1984. – 343 P.

Информация об авторах / Information about authors

Игорь Иванович ПОТАПОВ – доктор физико-математических наук, главный научный сотрудник Вычислительного центра Дальневосточного отделения Российской Академии наук. Область научных интересов: исследование русловых процессов для равнинных рек с несвязным основанием, речной гидродинамики, донной неустойчивости русла, математическое моделирование эволюции несвязного дна под действием речного потока.

Igor Ivanovich POTAPOV – Dr. Sci. (Phys.-Math.), chief researcher of the Computer center of Far East Branch of the Russian Academy of Sciences. The research interests: channel processes for flat rivers with an incoherent bed, river hydrodynamics, bed instability of the river channel, and mathematical modeling of the incoherent bed evolution under the influence of a river flow.

Ольга Владимировна РЕШЕТНИКОВА – кандидат технических наук, научный сотрудник Вычислительного центра Дальневосточного отделения Российской Академии наук. Область научных интересов: исследование и математическое моделирование механических процессов в сплошных средах.

Olga Vladimirovna RESHETNIKOVA – Cand. Sci. (Tech.), researcher of the Computer center of Far East Branch of the Russian Academy of Sciences. The research interests: analysis and mathematical modeling of mechanical processes in continuous media.