# ВЕСТНИК

## ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2023
Т. 12, № 3

ISSN 2305-9052


## СЕРИЯ

# «ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

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


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


Тематика журнала:

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

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

2023
vol. 12, no. 3

ISSN 2305-9052

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

South Ural State University


The scope of the journal:

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

# Содержание

# Contents

DOI: 10.14529/cmse230301

# ON USING THE DECISION TREES TO IDENTIFY THE LOCAL EXTREMA IN PARALLEL GLOBAL OPTIMIZATION ALGORITHM*

© 2023 K.A. Barkalov, I.G. Lebedev, D.I. Silenko

*Lobachevsky State University of Nizhny Novgorod*
*(pr. Gagarina 23, Nizhny Novgorod, 603022 Russia)*
*E-mail: barkalov@vmk.unn.ru, ilya.lebedev@itmm.unn.ru, silenko@itmm.unn.ru*

Received: 31.07.2023

In the present work, the solving of the multidimensional global optimization problems using decision tree to reveal the attractor regions of the local minima is considered. The objective function of the problem is defined as a "black box", may be non-differentiable, multi-extremal and computational costly. We assume that the function satisfies the Lipschitz condition with a priory unknown constant. Global search algorithm is applied for the search of global minimum in the problems of such type. It is well known that the solution complexity essentially depends on the presence of multiple local extrema. Within the framework of the global search algorithm, we propose a method for selecting the vicinity of local extrema of the objective function based on analysis of accumulated search information. Conducting such an analysis using machine learning techniques allows making a decision to run a local method, which can speed up the convergence of the algorithm. This suggestion was confirmed by the results of numerical experiments demonstrating the speedup when solving a series of test problems.

*Keywords: global optimization, multiextremal functions, parallel computing, machine learning, decision tree.*

**FOR CITATION**

Barkalov K.A., Lebedev I.G., Silenko D.I. On Using the Decision Trees to Identify the Local Extrema in Parallel Global Optimization Algorithm. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 3. P. 5–18. DOI: 10.14529/cmse230301.

## Introduction

In the present work, we consider the parallel algorithms for solving the multidimensional global optimization problems. Such problems often arise in the cases, when it is necessary to select the values of parameters of the mathematical model being investigated, in which the simulation results best fit the experimental data. For solving the problems of this class, many algorithms are known: from the metaheuristic algorithms, based on the idea of random search [1–3], to the deterministic algorithms guarantying convergence to the global minimum [4–6].

Since in real global optimization problems each computing of the function value (hereafter called *a trial*) is a computation-costly operation, one has to reduce the number of such trials. It can be achieved by intentional choice of variants in the course of search for the optimal solution cutting off unpromising search subdomains and investigating only the ones, in which the solution of the problem can be found. The global search algorithm (GSA) is based on this idea [7]. In the present work, we tried to combine GSA and a local optimization method (Hooke–Jeeves pattern search method [8]) to reduce the number of trials executed. The decision on the run of the local method will be made using a decision tree.

*The paper is recommended for publication by the Program Committee of the International Scientific Conference "Parallel Computational Technologies (PCT) 2023".

The local optimization algorithms are intended for determining only one of local extrema within a set of feasible solutions, in which the objective function takes the extremal (maximum or minimum) value. Among the local methods, the zero-order methods and the gradient methods (of the 1st order and of the 2nd one) are traditionally distinguished [9]. Note that the application of the first-order methods (not to mention the second one) in the problems with the "black-box" type functions is difficult since the problem of numerical estimating the gradient arises here [10]. Therefore, further we will consider only the zero-order methods.

In order to determine at which moment it is best to run the local method within the framework of the global search, we will use an algorithm based on a decision tree. At the first search stage, the search information is accumulated in the form of the results of the trials executed. We used the trial results for training the decision tree that allows obtaining a piecewise constant approximation of the objective function and predicting the objective function behavior on its base. To do so, we will compare the function values at the points neighboring to the one, which we consider to be suspicious for a local minimum, and make a decision whether the next trial point falls into the attractor region of a local minimum or not.

The main part of the paper has the following structure. In Section 1, the global optimization problem statement is considered, the basic suggestions on the objective function are made. In Section 2, the description of the methods and approaches used is given. In particular, the main idea of global search algorithm is discussed and its computational rules are given. The section also contains the description of the local search method used and the method of constructing the approximation of the function using the decision trees. The novel algorithm combining the global search and the local one on the base of using the decision trees is presented in Section 3. In the next section, the features of the asynchronous parallelization scheme of the new algorithm are discussed. Section 5 contains the results of experiments, carried out using a parallel computer system with distributed memory, and a brief conclusion.

## 1. Problem statement

Let us consider a problem of searching the global minimum of a function $\varphi(y)$ in a hyperinterval $D = \{y \in R^N : a_i \leq y_i \leq b_i,\ 1 \leq i \leq n\}$.

$$\varphi(y^*) = min\{\varphi(y) : y \in D\}, D = \{y \in R^N : a_i \leq y_i \leq b_i, 1 \leq i \leq N\}, \quad (1)$$

where $a, b \in R$ are given vectors.

Also, assume the function to satisfy the Lipschitz condition with *a priori* unknown constant $L$ that corresponds to limited variation of the function values at limited variation of the argument.

$$|\varphi(y_1) - \varphi(y_2)| \leq L \parallel y_1 - y_2 \parallel, y_1, y_2 \in D, 0 < L < \infty.$$

This suggestion can be considered (with respect to the real-world problems) as the reflection of limited power generating the changes in the system being simulated.

The numerical solving of the problem (1) is reduced to constructing an estimate $y_k^* \in D$ matching some concept of nearness to the point $y^*$ (for example, $||y^* - y_k^*|| \leq \varepsilon$, where $\varepsilon \geq 0$ is a predefined accuracy) based on a finite number $k$ of the objective function values computed. With respect to the class of the considered problems, it is suggested that the objective function $\varphi(y)$ can be defined algorithmically, as a result of executing some subroutine or library.

The solving of the global optimization problems is much more computationally expensive as compared to the local optimization problems since finding the global optimum requires ex-

ploration of the whole search domain. As a result, the search of the global optimum is reduced to constructing some coverage in the search domain and choosing the best function value on this grid. When using the uniform grids, the computation costs of solving the problem grow exponentially with increasing dimensionality.

## 2. Algorithms

The main idea of the approach to constructing more efficient nonuniform grid is that its points are computed sequentially while the objective function is considered as a realization of some random process. The decision rules of the grid building algorithm are constructed in such a way that the next grid point corresponds to the global minimum point of the mathematical expectation of the function values. This point is stored in the list of known values, and the iterations are repeated until the stop criteria is satisfied: either the distance between the trial points becomes less than a predefined value or a preset maximum number of iterations is achieved.

When solving the multidimensional problems, the dimensionality reduction (i.e. the reduction of the multidimensional problem to an equivalent one-dimensional one) using Peano curves is applied. These ones allow reducing a multidimensional optimization problem in the domain $D$ to a one-dimensional minimization problem within the interval $[0,1]$

$$\varphi(y(x^*)) \ = \min\{\varphi(y(x)) : x \in [0,1]\},$$

where the function $\varphi(y(x^*))$ satisfies more general Hölder condition

$$|\varphi(y\,(x_1)) - \varphi(y\,(x_2))\,| \leq \ H\,|x_1 - x_2|^{\frac{1}{N}}\,,\ x_1,\ x_2 \epsilon [0,1].$$

Therefore, instead of the initial problem of minimizing the function $\varphi(y)$ in the domain $D$, we can consider the minimization of the one-dimensional function $f(x) = \varphi(y(x))$ satisfying the Hölder condition for $x \in [0,1]$.

### 2.1. Multidimensional parallel global search algorithm

The main steps of the parallel global search algorithm are as follows.

At the preliminary step, $p$ trials are executed in parallel in arbitrary internal points $x^1, ..., x^p$ of the interval $[0,1]$ that corresponds to the first iteration of the algorithm.

If $n \geq 1$ iterations are completed, which correspond to $k = k(n)$ trials executed at the points $x^i$, $1 \leq i \leq k$, then the points $x^{k+1}, \ldots, x^{k+p}$ of the search trials at the next $(n+1)^{\text{th}}$ iteration will be computed as the result of performing the following operations.

1. Renumber (by the lower indices) the points $x^i$, $1 \leq i \leq k$ as well as the boundary points of the interval $[0,1]$ in increasing order of the coordinate

$$0 = x_0 < \ x_1 < \ \ldots \ < x_{k+1} = 1. \tag{2}$$

   and juxtapose them with the values $z_i = f(x_i)$.
2. Compute current lower estimate $M$ of the unknown Hölder constant $H$:

$$\mu = max\left\{\frac{|z_i - z_{i-1}|}{(x_i - x_{i-1})^{1/N}},\ i = 1, \ldots, k\right\},\ M = \begin{cases} r\mu,\ \mu > 0, \\ 1,\ \mu = 0, \end{cases} \tag{3}$$

where $r > 1$ is a parameter of algorithm. This parameter controls the reliability of the algorithm: the higher values of $r$ ensure guaranteed finding of the global minimum, the choice of lower value speeds up the convergence of the algorithm.

3. For each interval $(x_{i-1}, x_i)$, $1 \le i \le k+1$, compute the value $R(i)$ called *a characteristic* of the interval according to the formulae

$$R(1) = 2\Delta_1 - 4\frac{z_1}{M},\ R(k+1) = 2\Delta_{k+1} - 4\frac{z_k}{M}, \tag{4}$$

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{M^2\Delta_i} - 2\frac{z_i + z_{i-1}}{M}, 1 < i < k+1, \tag{5}$$

where $\Delta_i = (x_i - x_{i-1})^{\frac{1}{N}}$.

4. Arrange the characteristics $R(i)$, $1 \le i \le k+1$, in the non-increasing order

$$R(t_1) \ge\ R(t_2) \ge ... \ge\ R(t_k) \ge\ R(t_{k+1}), \tag{6}$$

and select $p$ intervals with the indices $t_j$, $1 \le\ j \le\ p$ with the largest values of characteristics.

5. Compute the points $x^{k+j},\ 1 \le j \le p$ in the selected intervals according to the formulae

$$x^{k+j} = \frac{x_{t_j} + x_{t_j-1}}{2},\ t_j = 1,\ t_j = k+1, \tag{7}$$

$$x^{k+1} = \frac{x_{t_j} + x_{t_j-1}}{2} - sign\left(z_{t_j} - z_{t_j-1}\right)\frac{1}{2r}\left[\frac{\left|z_{t_j} - z_{t_j-1}\right|}{\mu}\right]^N,\ 1 < t_j < k+1. \tag{8}$$

The next $p$ trials are executed in parallel at the points $x^{k+j},\ 1 \le j \le p$, computed according to the formulae (7), (8). Upon completing the trials, the results of these ones are stored in the information database, and the algorithm goes to computing new trial points. Note that as a rule the process of executing a trial in the applied optimization problems is much more computation-costly as compared to computing the trial point.

The algorithm stops in the case if the condition $\Delta_{t_j} < \varepsilon$ is satisfied for even a single value $t_j$, $1 \le\ j \le\ p$, from (6). This stop criterion (along with the criterion limiting the number of executed iterations usual for the iteration methods) is used in the applied optimization problems, in which the global minimum point $y^*$ is unknown a priori.

When solving the test problems, in which the global minimum point $y^*$ is known, we can use the stop criterion upon falling into the vicinity of the global minimum as well. In this case, the method stops if the condition $\left\|y(x_{t_j}) - \ y^*\right\| < \varepsilon$ is satisfied even for one value of $t_j,\ 1 \le\ j \le\ p$, from (6).

As the final estimate of the global optimizer of the considered problem, the values

$$f_k^* = \min_{1 \le i \le k} f(x_i),\ x_k^* = arg\min_{1 \le i \le k} f(x_i). \tag{9}$$

are taken.

The substantiation of this method of organization of computations see in [7, 11]. The modifications taking into account the presence of inequality constraints as well as the information on the derivative of the objective function are presented in [12, 13].

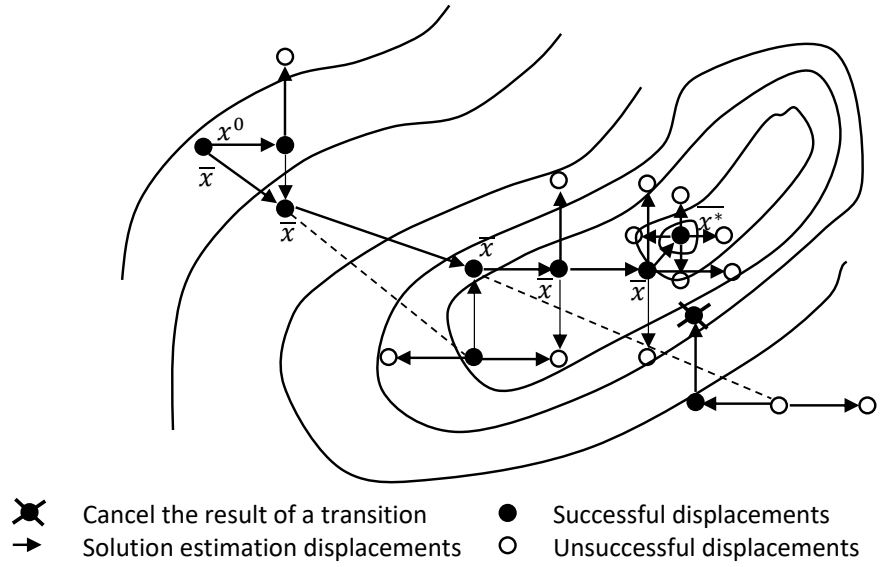



**Figure 1.** Example of iterations of the Hooke–Jeeves algorithm

## 2.2. Hooke–Jeeves method

For the Hooke–Jeeves pattern search algorithm belongs to the class of zero-order methods. Its computation rules are a combination of the investigating search (to select the direction) and search in the direction selected [14].

The investigating search is performed as follows.

1. The step value is determined (it is different for each coordinate and can be varied in the course of search).
2. The search step is considered to be successful if the value of the objective function at the check point does not exceed the value of the objective function at the initial point.
3. Otherwise, it is necessary to return to the previous item and make a step in the reverse direction.
4. After making the steps in all $N$ coordinates, the investigating search is completed. The point obtained is called the base point.

Figure 1 shows the example of the algorithm work. Function level lines are displayed, filled circles indicate successful steps, unfilled circles correspond to unsuccessful steps performed during the investigating search. With regard to the search in a selected direction, it consists in performing a step from the base point found during the investigating search along the line connecting this one with the previous base point. The magnitude of this step is defined by a parameter set in advance.

## 2.3. Decision tree

The decision tree is a tool used for an automated analysis of big data arrays, which is applied in the machine learning. Decision tree is a binary tree (a tree, in which each non-leaf node has two child nodes). It can be used for solving the classification problems as well as the regression ones. When solving the classification problems, each leaf of the tree is marked by a class label, and several leafs can have the same labels. In the case of constructing a regression, a constant is assigned to each leaf of the tree. Therefore, the approximating function obtained is a piecewise-

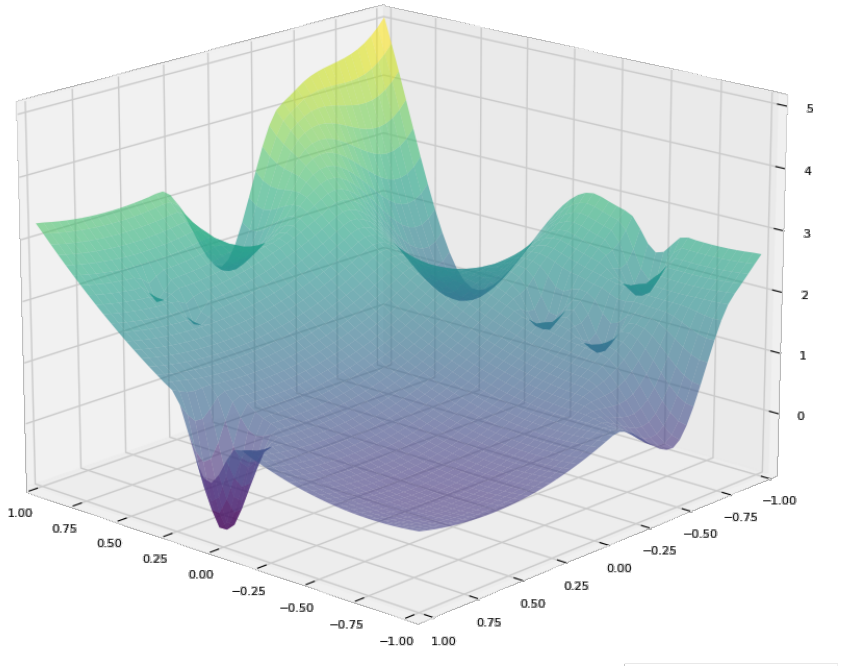



**Figure 2.** Objective function $\varphi(y)$ from the test class

constant one. In our case, the decision tree constructs the function $\psi(y)$ as a piecewise-constant approximation of $\varphi(y)$ in the search domain. Let us denote the value computed by the decision tree at the point $y$ as $z' = \psi(y)$. Figure 2 presents a graph of the objective function $\varphi(y)$; Fig. 3 presents corresponding approximation $\psi(y)$ built using decision tree.

When implementing the algorithm for finding the attraction areas of the local minima, we used the algorithms from OpenCV library to construct decision tree. OpenCV is an open source library of the algorithms of computer vision, image processing, and general-purpose numerical algorithms. More details on the decision tree can be found in [15].

## 3. Combination of local and global search algorithms for solving multidimensional problems

In the current section, let us present a detailed description of how we used the decision tree for finding the attraction areas of the local extrema. In the course of running GSA, it is necessary to determine whether it is worth to use current point as a start one for the local method or not. To do so, one can check the points of adjacent trials. If among these ones there are no points at which the function values are lower than in current one, one can suggest that we are in the attractor of a local minimum. In this case, we can run the local method from current point, which will rapidly converge to a local minimum of the function. We can do it in the multidimensional space only, not in one-dimensional interval after the dimensionality reduction. First, the reduced function $\varphi(y(x))$ changes its properties: one local minimum in the multidimensional space may divide into a set of minima after the dimensionality reduction. Second, after the use of the mapping $y(x)$, we can lose the information on the mutual arrangement of the points in the original space. The points located close to each other in the multidimensional space may appear to be essentially separated in the one-dimensional interval.

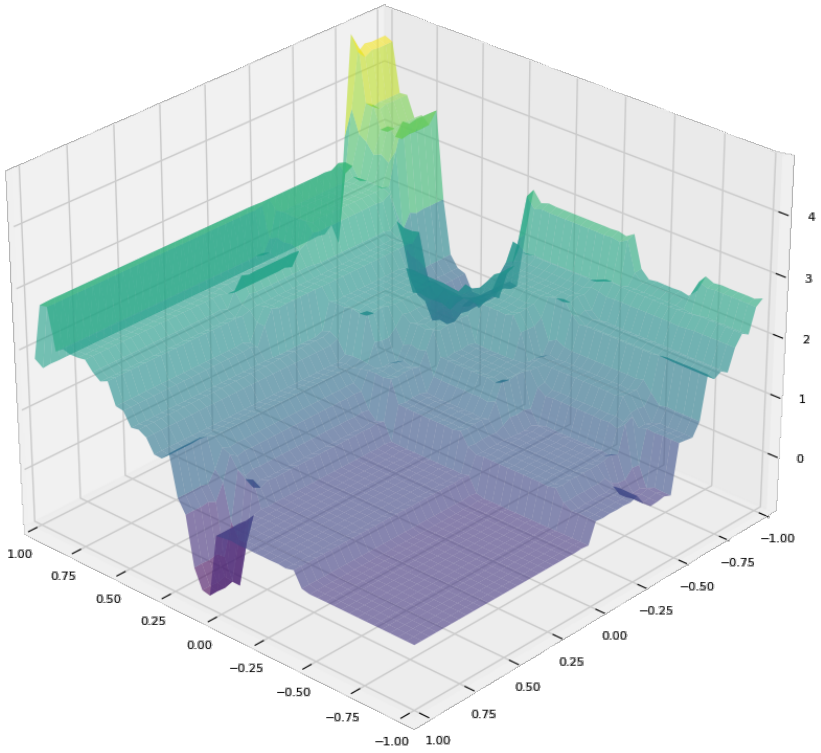



**Figure 3.** Piecewise-constant approximation $\psi(y)$ of the function $\varphi(y)$

Therefore, we will determine the attraction areas of the local extrema in the original multidimensional space. However, the determining of the adjacent points in the multidimensional space is quite expensive in terms of computing resources. To do so, we will use the decision tree.

After completing a certain number of trials (for example, $100 \cdot N$), let us use all accumulated points to initialize a suitable data structure and training the decision tree on its base. To make the determining of the adjacent points easier, we construct a uniform grid with certain step in each coordinate. Next, we compute the values of the approximation of the objective function in these points using the decision tree. Now, having a uniform grid of points and knowing the function values in each point, let us find the closest point to the initial one (from the viewpoint of the Euclidian distance). This closest point is a projection of the initial point onto the uniform grid, on which its neighbors can be determined easily. Since the decision tree constructs the approximation of the initial function (even piecewise constant), we can estimate the function values in the areas, in which the trials have not been executed earlier.

When considering the neighbors, it is necessary to take into account the following. If even a single neighbor has a lower function value than current point, there is no need to run the local method. If at even a single neighboring point the function value is the same as at the current one, this point must be also checked (i.e., to find all its neighbors and to check these ones in the same way). Only in the case when all function values at the neighboring points are greater than at the current point, we can run the local method.

## 4. Asynchronous algorithm parallelization scheme

The method of parallel computing described in previous section implies synchronous performing of $p$ search trials at the points computed at current iteration. The next iteration will be started only upon completing all trials. In the case of different computation costs of executing the trials at different points of the search domain, this approach may lead to a disbalance of the computing jobs. This drawback can be corrected by introduction of asynchrony. The idea of the asynchronous scheme is not to wait for completing all $p$ trials but to initiate the execution of a new trial immediately upon completing one of the trials.

Let us assume that the master process computes one point of the next trial at every iteration and sends it to the slave process to execute the trial. Note that executing a trial by the slave process in the applied optimization problems is much more computation costly than the choice of a new trial point by the master process. Therefore, the idle time of the processes will be negligible. In this case (unlike the synchronous parallel algorithms) total number of trials executed by each slave process will depend on the computation costs for executing particular trials and cannot be estimated in advance. When describing the parallel algorithm, let us assume that we have one master and $p$ slave processes at our disposal.

At the initial phase the master process (assume it to have the identifier 0) starts $p$ trials in parallel at $p$ different points $\{y\left(x^1\right), y\left(x^2\right), \ldots, y\left(x^p\right)\}$. The inverse images of two points are the boundary ones $x^1 = 0$, $x^p = 1$; the rest are the internal points $x^i \in (0,1)$, $i = 2, \ldots, p-1$.

Now assume $k \geq 0$ trials to be completed, and the slave processes are performing the trials at the points $y\left(x^{k+1}\right)$, $y\left(x^{k+2}\right), \ldots,$ $y\left(x^{k+p}\right)$.

Each slave process having completed the computing of the function at some time moment sends the computed value to the master process. Next, the master process selects a new point $x^{k+p+1}$ for the slave process according to the rules provided below. Note that in this case it will be a set of inverse images of the trial points $I_k = \left\{x^{k+1}, x^{k+2}, \ldots, x^{k+p}\right\}$, in which the trials have been already initiated but have not been completed yet.

So far, the parallel asynchronous global optimization algorithm using decision tree to find the local extrema attractors consists of the following steps.

1. Arrange in the increasing order (by the lower indices) the set of inverse images of the trial points

$$X_k = \left\{x^1, x^2, \ldots, x^{k+p}\right\},$$

containing all points, in which the trials either have been completed or are being executed, i.e. get an ordered set

$$0 = x_1 < x_2 < \ldots < x_{k+p} = 1.$$

2. Compute the values

$$M_1 = \max\left\{\frac{|z_i - z_{i-1}|}{(x_i - x_{i-1})^{1/N}} : x_{i-1} \notin I_k, x_i \notin I_k, 2 \leq i \leq k+p\right\},$$

$$M_2 = \max\left\{\frac{|z_{i+1} - z_{i-1}|}{(x_{i+1} - x_{i-1})^{1/N}} : x_i \in I_k, 2 \leq i < k+p\right\},$$

$$M = \max\{M_1, M_2\},$$

where $z_i = \varphi(y(x_i))$ if $x_i \notin I_k,\ 1 \le i \le k+p$. The values $z_i$ at the points $x_i \in I_k$ are not defined as far as the trials at these points have not been completed yet. If $M = 0$, then set $M = 1$.

3. Juxtapose each interval $(x_{i-1}, x_i),\ x_{i-1} \notin I_k, x_i \notin I_k,\ 2 \le i \le k+p$, with its characteristic $R(i)$ computed according to the formula

$$R(i) = rM\Delta_i + \frac{(z_i - z_{i-1})^2}{rM\Delta_i} - 2(z_i + z_{i-1}),$$

where $\Delta_i = (x_i - x_{i-1})^{1/N}$ and $r > 1$ is the reliability parameter of the method.

4. Find the interval $[x_{t-1}, x_t]$, which corresponds to the highest characteristic, i.e.

$$R(t) = \max\{R(i) :\ x_{i-1} \notin I_k, x_i \notin I_k,\ 2 \le i \le k+p\}.$$

5. Compute a new trial point $y^{k+p+1} = y\left(x^{k+p+1}\right)$, the inverse image of which is $x^{k+p+1} \in (x_{t-1}, x_t)$, according to the formula

$$x^{k+p+1} = \frac{x_t + x_{t-1}}{2} - \operatorname{sign}(z_t - z_{t-1})\frac{1}{2r}\left[\frac{|z_t - z_{t-1}|}{M}\right]^N.$$

6. Get the computed function value from the process $j$, add new trial $z_j = f(y(x_j))$ to the set $V$.
7. If $k < 100N$, return to Step 1.
8. Create the decision tree using the set $I_k$, get the approximating function $\psi(y)$.
9. If the decision tree is used for the first time, construct a uniform grid

$$Y' = \{y' \in R^N :\ a_i \le y'_i \le b_i,\ 1 \le i \le N\},$$

where number of the grid nodes is a parameter of the method.

10. Compute the approximation values: $Z' = \{z' = \psi(y'), y' \in Y'\}$
11. For all points $y' \in V$:
    Find the points $y'_q$ closest to $y'$,
    Make a detour of the neighbors $y'_q$ according to the principle described above.
    If not one neighbor point has a lower value than in $y'_q$, run the local method.
    Clear the set $V$.

After the next trial point is computed, the master process includes it to the set $I_k$ and sends to the slave process, which starts the new trial at this point. The master process stops the algorithm, if at least one of the two conditions is satisfied: $\Delta_t <$ or $k + p > K_{max}$. The real value $\epsilon > 0$ and the integer value $K_{max} > 0$ are the algorithm parameters. They correspond to the accuracy of the search and the maximum number of trials, respectively.

## 5. Numerical experiments

The numerical experiments were carried out on Lobachevsky supercomputer. Each supercomputer node had two processors Intel Sandy Bridge E5-2660 2.2 GHz, 64 Gb RAM.

In the experiments, we used GKLS generator of test problems, which can generate multiextremal optimization problems with known properties: the global minimum point, the number of local minima, etc.

To demonstrate the efficiency of investigated global search algorithm, here we present the results of comparing this one with well known methods DIRECT and DIRECTl. Table 1 presents the averaged numbers of iterations executed by respective methods when solving a series of GKLS problems. The sign " $>$ " points to the situation when not all problems are solved. In this case, the number of non-solved problems is given in brackets. As one can see, GSA overcomes DIRECT and DIRECTl methods in average number of iterations necessary for solving the problems with the same precision.

**Table 1.** Averaged number of iterations

| $N$ | Problem class | DIRECT | DIRECTl | GSA |
|---|---|---|---|---|
| 4 | Simple | $>$ 47 282 (4) | 18 983 | 11 953 |
| | Hard | $>$ 95 708 (7) | 68 754 | 25 263 |
| 5 | Simple | $>$ 16 057 (1) | 16 758 | 15 920 |
| | Hard | $>$ 217 215 (16) | $>$ 269 064 (4) | $>$ 148 342 (4) |

Below, the results of comparing two parallel algorithms — asynchronous global search algorithm (AGSA) and its modification using the decision tree to find the attractor areas of the local minima (Decision Tree) are presented. The numerical comparison was performed on two classes of the GKLS functions (Simple and Hard from [16]) of dimensionalities 2, 3, 4, and 5. These two classes differ in the size of the attraction region of the global minimum point. For the simple class of problems, the radius of the attraction region is three times larger than for a complex one.

The stop criterion for the algorithms was the falling of the next trial points into the $\varepsilon$-nearness of true global minimum. In Tab. 2, the averaged numbers of iterations executed by the algorithms when solving the problems and the values of time speedup relative to the sequential run are compared. The parallelization was performed using MPI technology, the run was performed on 8 processes.

**Table 2.** Averaged numbers of iterations and averaged numbers of trials executed by different algorithms

| $N$ | Problem class | Averaged number of iterations | | Speedup | |
|---|---|---|---|---|---|
| | | AGSA | Decision tree | AGSA | Decision tree |
| 2 | Simple | 3 823.1 | 485.6 | 7.4 | 12.7 |
| | Hard | 787.6 | 304.2 | 13.8 | 14.9 |
| 3 | Simple | 5 700.5 | 720.1 | 5.6 | 14.2 |
| | Hard | 2 411.9 | 634.3 | 5 | 5.9 |
| 4 | Simple | 31 101.5 | 1 037.4 | 5.1 | 7.1 |
| | Hard | 10 418.1 | 816.2 | 6.9 | 5 |
| 5 | Simple | 163 313.5 | 1 204.2 | 4.8 | 4.1 |
| | Hard | 27 524.5 | 999.4 | 4.6 | 6.3 |

As one can see from Tab. 2, the iteration speedup is large enough for the problems of any dimensionality.

## Conclusion

So far, as a result of the work done, we succeeded to combine the global search algorithm with a local optimization method. Unlike the known multistart schemes, the decision to run the local method is made using the decision tree. The use of such a combination of methods allows considerably accelerating the algorithm.

The parallel version of the algorithm preserves the properties of its sequential prototype that was confirmed by he numerical experiments on solving a series of several hundred problems of various dimensionalities. The proposed scheme allows us utilizing the advantages of both parallelization as well as fast search of local extrema.

In addition to using decision trees to identify attraction regions for local extrema of multiextremal functions, we also plan to use machine learning methods to separate the variables of the problem being solved. In many applied optimization problems the dependence of the objective function on some parameters is either linear or unimodal. Separating such variables into a special group and solving the problem using a parallel recursive optimization scheme [17] can reduce the time for solving the problem by orders of magnitude compared to using a global search for all variables at once.

*This study was supported by the Russian Science Foundation, project No. 21-11-00204.*

## References

1. Ferreiro A., Garcia J., Lopez-Salas J., Vazquez C. An efficient implementation of parallel simulated annealing algorithm in GPUs. J. Glob. Optim. 2013. Vol. 57, no. 3. P. 863–890. DOI: 10.1007/s10898-012-9979-z.
2. Garcia-Martinez J., Garzon E., Ortigosa P. A GPU implementation of a hybrid evolutionary algorithm: GPuEGO. J. Supercomput. 2014. Vol. 70, no. 2. P. 684–695. DOI: 10.1007/s11227-014-1136-7.
3. Langdon W. Graphics processing units and genetic programming: an overview. Soft Computing. 2011. Vol. 15, no. 8. P. 1657–1669. DOI: 10.1007/s00500-011-0695-2.
4. Evtushenko Y., Malkova V., Stanevichyus A.A. Parallel global optimization of functions of several variables. Comput. Math. Math. Phys. 2009. Vol. 49, no. 2. P. 246–260. DOI: 10.1134/S0965542509020055.
5. He J., Verstak A., Watson L., Sosonkina M. Design and implementation of a massively parallel version of DIRECT. Comput. Optim. Appl. 2008. Vol. 40, no. 2. P. 217–245. DOI: 10.1007/s10589-007-9092-2.
6. Paulavičius R., Žilinskas J., Grothey A. Parallel branch and bound for global optimization with combination of Lipschitz bounds. Optim. Method. Softw. 2011. Vol. 26, no. 3. P. 487–498. DOI: 10.1080/10556788.2010.551537.
7. Strongin R.G., Sergeyev Y.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Dordrecht: Kluwer Academic Publishers, 2000. DOI: 10.1007/978-1-4615-4677-1.
8. Hooke R., Jeeves T. "Direct Search" Solution of Numerical and Statistical Problems. J. ACM. 1961. Vol. 8, no. 2. P. 212–229. DOI: 10.1145/321062.321069.

9. Nocedal J., Wright S. Numerical Optimization. New York: Springer, 2006. DOI: 10.1007/b98874.

10. Kelley C.T. Iterative Methods for Optimization. Philadelphia: SIAM, 1999. DOI: 10.1137/1.9781611970920.

11. Barkalov K., Lebedev I. Solving multidimensional global optimization problems using graphics accelerators. Communications in Computer and Information Science. 2016. Vol. 687. P. 224–235. DOI: 10.1007/978-3-319-55669-7_18.

12. Barkalov K., Strongin R. A global optimization technique with an adaptive order of checking for constraints. Computational Mathematics and Mathematical Physics. 2002. Vol. 42, no. 9. P. 1289–1300.

13. Gergel V.P. A global optimization algorithm for multivariate functions with Lipschitzian first derivatives. Journal of Global Optimization. 1997. Vol. 10, no. 3. P. 257–281. DOI: 10.1023/A:1008290629896.

14. Himmelblau D. Applied Nonlinear Programming. New York: McGraw-Hill, 1972. 498 p.

15. Brahmbhatt S. Practical OpenCV (Technology in Action). New York: Apress, 2013. DOI: 10.1007/978-1-4302-6080-6_1.

16. Sergeyev Y., Kvasov D. Global Search Based on Efficient Diagonal Partitions and a Set of Lipschitz Constants. SIAM J. Optim. 2006. Vol. 16, no. 3. P. 910–937. DOI: 10.1137/040621132.

17. Barkalov K., Lebedev I., Kocheganova M., Gergel V. Combining local and global search in a parallel nested optimization scheme. Communications in Computer and Information Science. 2020. Vol. 1263. P. 100–112. DOI: 10.1007/978-3-030-55326-5_8.

УДК 519.853.4 DOI: 10.14529/cmse230301

# ОБ ИСПОЛЬЗОВАНИИ ДЕРЕВЬЕВ РЕШЕНИЙ ДЛЯ ВЫЯВЛЕНИЯ ОБЛАСТЕЙ ПРИТЯЖЕНИЯ ЛОКАЛЬНЫХ МИНИМУМОВ В ПАРАЛЛЕЛЬНОМ АЛГОРИТМЕ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ

**© 2023 К.А. Баркалов, И.Г. Лебедев, Д.И. Силенко**

*Нижегородский государственный университет им. Н.И. Лобачевского*
*(603022 Нижний Новгород, пр. Гагарина, д. 23)*
*E-mail: barkalov@vmk.unn.ru, ilya.lebedev@itmm.unn.ru, silenko@itmm.unn.ru*

Поступила в редакцию: 31.07.2023

В работе рассматривается решение многомерных задач многоэкстремальной оптимизации с использованием деревьев решений для выявления областей притяжения локальных минимумов. Целевая функция представлена как «черный ящик», она может быть недифференцируемой, многоэкстремальной и вычислительно трудоемкой. Для функции предполагается, что она удовлетворяет условию Липшица с априори неизвестной константой. Для решения поставленной задачи многоэкстремальной оптимизации применятся алгоритм глобального поиска. Хорошо известно, что сложность решения существенно зависит от наличия нескольких локальных экстремумов. В данной работе предложена модификация алгоритма, в которой определяются окрестности локальных минимумов целевой функции на основе анализа накопленной поисковой информации. Проведение такого анализа с использованием методов машинного обучения позволяет принять решение о запуске локального метода, что может ускорить сходимость алгоритма. Данный подход был подтвержден результатами численных экспериментов, демонстрирующих ускорение при решении набора тестовых задач.

*Ключевые слова: глобальная оптимизация, многоэкстремальные функции, параллельные вычисления, машинное обучение, дерево решений.*

## ОБРАЗЕЦ ЦИТИРОВАНИЯ

Barkalov K.A., Lebedev I.G., Silenko D.I. On Using the Decision Trees to Identify the Local Extrema in Parallel Global Optimization Algorithm // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 3. С. 5–18. DOI: 10.14529/cmse230301.



## Литература

1. Ferreiro A., Garcia J., Lopez-Salas J., Vazquez C. An efficient implementation of parallel simulated annealing algorithm in GPUs // J. Glob. Optim. 2013. Vol. 57, no. 3. P. 863–890. DOI: 10.1007/s10898-012-9979-z.
2. Garcia-Martinez J., Garzon E., Ortigosa P. A GPU implementation of a hybrid evolutionary algorithm: GPuEGO // J. Supercomput. 2014. Vol. 70, no. 2. P. 684–695. DOI: 10.1007/s11227-014-1136-7.
3. Langdon W. Graphics processing units and genetic programming: an overview // Soft Computing. 2011. Vol. 15, no. 8. P. 1657–1669. DOI: 10.1007/s00500-011-0695-2.
4. Евтушенко Ю.Г., Малкова В.У., Станевичюс А.А. Параллельный поиск глобального экстремума функций многих переменных // Вычислительная математика и математическая физики. 2009. Т. 49, № 2. С. 246–260. DOI: 10.1134/S0965542509020055.

5. He J., Verstak A., Watson L., Sosonkina M. Design and implementation of a massively parallel version of DIRECT // Comput. Optim. Appl. 2008. Vol. 40, no. 2. P. 217–245. DOI: 10.1007/s10589-007-9092-2.

6. Paulavičius R., Žilinskas J., Grothey A. Parallel branch and bound for global optimization with combination of Lipschitz bounds // Optim. Method. Softw. 2011. Vol. 26, no. 3. P. 487–498. DOI: 10.1080/10556788.2010.551537.

7. Strongin R.G., Sergeyev Y.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Dordrecht: Kluwer Academic Publishers, 2000. DOI: 10.1007/978-1-4615-4677-1.

8. Hooke R., Jeeves T. "Direct Search" Solution of Numerical and Statistical Problems // J. ACM. 1961. Vol. 8, no. 2. P. 212–229. DOI: 10.1145/321062.321069.

9. Nocedal J., Wright S. Numerical Optimization. New York: Springer, 2006. DOI: 10.1007/b98874.

10. Kelley C.T. Iterative Methods for Optimization. Philadelphia: SIAM, 1999. DOI: 10.1137/1.9781611970920.

11. Barkalov K., Lebedev I. Solving multidimensional global optimization problems using graphics accelerators // Communications in Computer and Information Science. 2016. Vol. 687. P. 224–235. DOI: 10.1007/978-3-319-55669-7_18.

12. Баркалов К.А., Стронгин Р.Г. Метод глобальной оптимизации с адаптивным порядком проверки ограничений // Вычислительная математика и математическая физики. 2002. Т. 42, № 9. С. 1289–1300.

13. Gergel V.P. A global optimization algorithm for multivariate functions with Lipschitzian first derivatives // Journal of Global Optimization. 1997. Vol. 10, no. 3. P. 257–281. DOI: 10.1023/A:1008290629896.

14. Himmelblau D. Applied Nonlinear Programming. New York: McGraw-Hill, 1972. 498 p.

15. Brahmbhatt S. Practical OpenCV (Technology in Action). New York: Apress, 2013. DOI: 10.1007/978-1-4302-6080-6_1.

16. Sergeyev Y., Kvasov D. Global Search Based on Efficient Diagonal Partitions and a Set of Lipschitz Constants // SIAM J. Optim. 2006. Vol. 16, no. 3. P. 910–937. DOI: 10.1137/040621132.

17. Barkalov K., Lebedev I., Kocheganova M., Gergel V. Combining local and global search in a parallel nested optimization scheme // Communications in Computer and Information Science. 2020. Vol. 1263. P. 100–112. DOI: 10.1007/978-3-030-55326-5_8.

Баркалов Константин Александрович, д.т.н., доцент, кафедра математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (Нижний Новгород, Российская Федерация)

Лебедев Илья Генадьевич, заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений, Нижегородский государственный университет им. Н.И. Лобачевского (Нижний Новгород, Российская Федерация)

Силенко Дмитрий Игоревич, магистр, Нижегородский государственный университет им. Н.И. Лобачевского (Нижний Новгород, Российская Федерация)

DOI: 10.14529/cmse230302

# INTERMEDIATE FUSION APPROACH FOR PNEUMONIA CLASSIFICATION ON IMBALANCED MULTIMODAL DATA*

© 2023 O.N. Ivanova, A.V. Melekhin, E.V. Ivanova, S. Kumar, M.L. Zymbler

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

*E-mail: onivanova@susu.ru, temamel540@gmail.com elena.ivanova@susu.ru, kumars@susu.ru, mzym@susu.ru*

Received: 03.08.2023

In medical practice, the primary diagnosis of diseases should be carried out quickly and, if possible, automatically. The processing of multimodal data in medicine has become a ubiquitous technique in the classification, prediction and detection of diseases. Pneumonia is one of the most common lung diseases. In our study, we used chest X-ray images as the first modality and the results of laboratory studies on a patient as the second modality to detect pneumonia. The architecture of the multimodal deep learning model was based on intermediate fusion. The model was trained on balanced and imbalanced data when the presence of pneumonia was determined in 50% and 9% of the total number of cases, respectively. For a more objective evaluation of the results, we compared our model performance with several other open-source models on our data. The experiments demonstrate the high performance of the proposed model for pneumonia detection based on two modalities even in cases of imbalanced classes (up to 96.6%) compared to single-modality models' results (up to 93.5%). We made several integral estimates of the performance of the proposed model to cover and investigate all aspects of multimodal data and architecture features. There were accuracy, ROC AUC, PR AUC, F1 score, and the Matthews correlation coefficient metrics. Using various metrics, we proved the possibility and meaningfulness of the usage of the proposed model, aiming to properly classify the disease. Experiments showed that the performance of the model trained on imbalanced data was even slightly higher than other models considered.

*Keywords: multimodal model, intermediate fusion, pneumonia, deep learning, imbalanced data.*

**FOR CITATION**

Ivanova O.N., Melekhin A.V., Ivanova E.V., Kumar S., Zymbler M.L. Intermediate Fusion Approach for Pneumonia Classification on Imbalanced Multimodal Data. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 3. P. 19–30. DOI: 10.14529/cmse230302.

## Introduction

Pneumonia is the most common diagnosis in the world among all diagnosed lung diseases. During the pandemic, this disease took the first place among all diagnosed human diseases [1]. Timely, fast, reliable detection of pneumonia can allow doctors to start using treatment as early as possible, achieve positive dynamics, improve the prognosis of the course of the disease, and, ultimately, improve the health of the population due to fewer resources. Modern deep learning technologies allow the processing of data from several modalities at once, which is very practical in the field of medicine. Indeed, in the clinic, the patient passes a lot of laboratory tests and many different types of studies. The aggregation of the results of various types of medical research was previously performed only by an experienced doctor. Now, this task can be taken over by

*The paper is recommended for publication by the Program Committee of the International Scientific Conference "Global Smart Industry Conference (GloSIC) 2023".

multimodal deep learning models, at least with a recommendation and informative purpose, acting in automatic mode.

The imbalance of data in the classification is a fairly typical situation in various fields of science and practice. In medicine, the presence of a sufficient and approximately equal number of training examples for each class of multiclass classification can be considered a great success for a researcher designing a classifier model. Improbability methods in machine learning involve the desire to align data in different classes. This is usually done by reducing a larger class (for example, by random deletions) or increasing a smaller class (most often by combining an encoder/decoder or using GAN). In turn, probabilistic machine learning models for solving binary classification problems are weakly dependent on the balance of classes.

Logistic regression and decision trees certainly respond to class imbalance: a significant change in the value of the free term and the impurity of leaves measure relative. However, neither one nor the other change has a significant impact on the result of the prediction. In regression, the determining factor is the slope coefficient, not the intercept. Trees share samples with impurities approximately proportionally. Therefore, in the case when you do not need to use SVM or other improbability models, you can work with imbalanced classes using the unequal class weights for random forests, gradient boost, and its variations.

For multimodal data, probabilistic deep learning models are usually used. We solved the problem of binary classification by attributing multimodal information about a patient to a class of healthy people or a class of people with diagnosed pneumonia. We used [2] as the dataset. The number of patients diagnosed with pneumonia was 50. The number of healthy patients was 500. The imbalance of classes, when the smaller class is from 1% to 20% of the total capacity of the dataset, refers to a moderate degree. The imbalance of classes in our multimodal model is moderate, since the minority class accounted for 9.1% of the total data set.

When building multimodal models, an important issue is the fusion of modalities. Late or early fusion of modalities is used only for certain types of tasks under certain data constraints. The detection of pneumonia from X-ray images and electronic medical records of the patient's laboratory tests implies a careful choice of the fusion model due to semantic heterogeneity and the remoteness of the modals from each other. The technical issues of implementing gradient boosting in multimodal models are also non-trivial. If many single-modal models can often be significantly improved by tuning hyperparameters, then the design of the architecture of a multi-modal model is a more complex process. Here, it is necessary to take into account the side effects of multimodality in the coordination and harmonization of learning outcomes, including intermediate ones. Normalization methods that are used for single-modal learning may not be sufficient. Metrics for evaluating the accuracy of multi-modality processing models are also the subject of close study by many researchers [3–5].

In this paper, we have proposed approaches to solving the listed problems in addition to the problem of detecting pneumonia.

The scientific novelty of the proposed solution is determined by the following:

1. A model of multimodal deep learning with intermediate fusion for detecting pneumonia from X-ray images and the results of clinical studies of patients is proposed.
2. It is shown that balancing imbalanced data in a multimodal dataset using probabilistic models is acceptable and does not lead to deterioration of classification results.

The practical significance of the study is as follows:

1. The proposed multimodal model of deep neural network training on moderately imbalanced classes allowed for a significant improvement in the quality of the binary classifier compared to the results of training on a single modality with medical images, and maintained comparable accuracy with the model trained on balanced multimodal data.
2. The necessity of studying and calculating various metrics of the quality of the multimodal model for the formation of an adequate multiparametric assessment of various aspects of the model is shown.

The article is organized as follows. In Section 1, we provide brief review of related works. Section 2 introduces our multimodal model of intermediate fusion. In Section 3, we discuss the results of the evaluation of the proposed model. Conclusions summarize the results of the research.

## 1. Literature review

Imbalanced data in multiclass classification is a typical situation in real business and production processes. In the works [6–8] various methods of working with imbalanced data in the field of medicine are proposed. All these works were devoted to solving problems of multiclass classification according to the data of one modality.

At the same time, when creating multimodal models, there are often their own peculiarities of working with data. Multimodal deep learning models are subject to such problems as data alignment between modalities, problems of mapping, translation, fusion and co-learning of modalities [9–11].

It would be logical to assume that the complexity of multimodal models increases complementarily when learning probabilistic models on imbalanced classes. However, in this study, we found out that balancing classes in a multimodal model of pneumonia detection is an optional step that can be abandoned when solving this problem.

When reviewing the modern literature, we could not find examples of the random removal of samples from multimodal models for class alignment. The simplicity of implementing this method is outweighed by a major disadvantage — when using it, valuable samples with useful information may be lost. However, other methods of majority class undersampling are quite often used by researchers to restore balance. So, in the work [12], the Tomek links search method is used. This method works well on sets with a small number of features.

The authors of [13] suggest using the Condensed Nearest Neighbor Rule. This method is often used in a situation of imbalance of classes of a strong degree ($< 1\%$).

In [14], data preprocessing was performed using one-side sampling of the majority class (or one-sided selection). Computationally, it is quite demanding. It makes sense to use this method when the dataset contains a small total number of samples.

Another type of algorithms — neighborhood cleaning rule — is suggested by [15]. The main result of this method is the removal of noise that interferes with the training of the model.

In [16], the authors propose an imbalanced multimodal model for estimating and forecasting the value of real estate objects. The proposed model is based on oversampling, that is, increasing the number of examples of a minority class. At the same time, a simple algorithm for duplicating randomly selected samples was used.

In [17], the SMOTE algorithm (Synthetic Minority Oversampling Technique) [18] was used to solve the oversampling problem. The basis of this algorithm is the generation of artificial

samples. Using the nearest neighbor method, a certain number of neighbors are selected for the sample. Then the feature vectors of each pair of neighboring features are multiplied by a random value from the interval $(0, 1)$. Thus, the feature vector of an artificially created sample in the area of a minority class is calculated. Unfortunately, this approach works well only for single-modal models with well-defined class domains. In the case when the vector space of features of different classes intersects or is mixed, as is often the case when analyzing medical observation data, the use of this method leads to a deterioration in the accuracy of classification.

The work [19] uses the ACM (Adaptive Synthetic Minority Oversampling) algorithm, which is a modification of SMOTE. The authors propose to perform the generation of artificial neighbor feature vectors only within the cluster. This modification of the oversampling algorithm is applicable to scattered classes, but greatly slows down the system, since it actually solves both the clustering problem and the sampling problem. There is also a requirement for the presence of clusters in the source data.

The authors of the paper [20] have demonstrated that the use of the oversampling algorithms described above is advisable only for binary classification, or if, in a multiclass classification, all minority classes differ from the majority by the same amount. In conditions where all classes (more than two) are imbalanced to varying degrees, the authors suggest using the ADASYN algorithm. This algorithm assumes calculating the coefficient of the distribution of sample weights within a minority class in order to generate artificial samples similar to the most important samples for learning. The coefficient is calculated based on the analysis of distances to samples of the majority class, which is a measure of complexity for training the model.

Thus, many researchers suggest balancing classes before training multimodal models. Such preprocessing takes a lot of time and resources. At the same time, the training of modalities is still carried out by probabilistic models that are insensitive to imbalance. Our idea is to eliminate the class balancing stage when building the architecture of a multimodal model of binary classification of medical data on lung diseases.

## 2. Methods

### 2.1. Data preprocessing

To test our hypothesis, we had to prepare two versions of the dataset — the imbalanced and balanced ones. The imbalanced dataset contained information about 50 patients with diagnosed pneumonia and 500 healthy people. We would like to make a reservation that by healthy people we meant patients who were represented in the same medical multimodal dataset, with diagnosed brain diseases, primarily with sleep disorders of various nature.

The dataset consisted of two modalities — X-ray images and the results of laboratory tests of patients' blood and urine. Images in the frontal position of the earliest registration were selected for each patient. As it is correct, for patients with pulmonary diseases, X-ray examinations were carried out repeatedly, with the preservation of all images in the medical history. With the course of treatment and the development of the disease, changes in the condition of the lungs were displayed on later images. Therefore, we took only primary images to train the model.

With regard to clinical data, the dataset in question had some redundancy for solving a particular problem of detecting pneumonia. In general, there are 1630 parameters of clinical analyses in the dataset. The reduction of the number of parameters occurred in several stages. First, all parameters that were not found in all patients were removed. After this operation, 523 parameters of clinical analyses remained. At the second stage, we made maps of the frequency

of parameters and removed those that mainly occurred only in patients of a certain class. At the same time, we were guided by a threshold of 80%: if the parameter was observed in 80% of healthy and 80% of sick people, then such a parameter fell into our sample.

In medical practice, the patient receives an appointment for the delivery of parameters repeatedly. Usually, when a patient is hospitalized, the main indicators of blood and urine are studied. Further, with a primary and clarified diagnosis, the doctor prescribes additional studies. Therefore, at the last, third stage, we selected for consideration only those tests that were taken from patients at the beginning of hospitalization, thus eliminating disease-specific tests from consideration. In the end, we left 49 parameters of medical tests.

We obtain a balanced dataset through the sampling procedure, which is described below.

## 2.2. Data sampling

Imbalanced classes were the source data for sampling for us. To obtain balanced classes, we used the method of generating artificial samples of the greatest importance — ADASYN [21]. First, we used the SVM method to visualize class boundaries in two-dimensional space and found out that a number of samples of the minority class are quite close to the samples of the majority class, sometimes even mixed. That is why, of all the sampling methods, we chose and applied the ADASYN method in order to set the samples at the borders with greater weight and generate most of the artificial samples of the minority class in such "mixed" zones. This allowed us to make the boundaries between classes clearer. Also, we compared the predictive ability of the proposed model with the single-modal models described in [6–8]. In all the works, references to the source codes were given and the authors' permission to use their code to conduct experiments on other datasets was posted. All the work was processed only X-ray images and determined the presence of pneumonia.

## 2.3. Multimodal model of intermediate fusion

The general architecture of the proposed multimodal model is shown in figure. To process the modality of chest X-ray images, we chose the MobileNetV2 model. To process clinical data, we chose the Attention model presented in the Keras framework. The fusion of modalities was carried out according to an intermediate type. In the final metamodel, four embeddings were submitted for input — two from each modality. The first embedding of the clinical data modality came from the last learning layer of the transformer model. The second embedding moved from the last layer after pooling and compaction. The embeddings of the convolutional neural network used to train the classification of X-ray images were received, respectively, after the first and last block of convolutional layers.

The deep concatenation method was used for fusion, after which the metamodel created a fully connected level using the ReLU method and collapsed it using the SoftMax method.

We applied the classical loss function to self-tune the model:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\left(p(P_i)\right) + (1 - P_i)\log\left(1 - p(P_i)\right). \quad (1)$$

## 2.4. Model validation

To test the accuracy of the model and the hypothesis as a whole, we compared the results not only between the two proposed models with balanced and imbalanced classes, but also with models that are publicly available for binary classification of healthy people and people with

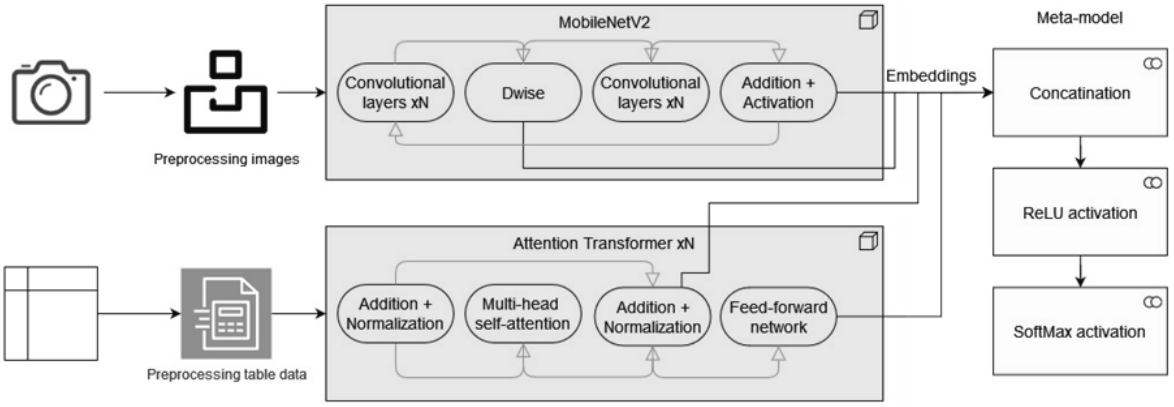



**Fig.** Architecture of the proposed multimodal classification model

diagnosed pneumonia [6–8]. The presented models were single-modal and could only process lung X-rays.

We chose several metrics for training because classical single metrics cannot specify all the aspects of the multimodal data. The evaluation of the quality of the multimodal model is carried out both individually for each class (using the precision and recall metrics), and integrally, for all classes (metrics F1, PR AUC, and Matthews correlation coefficient).

The classical Accuracy metric for evaluating the accuracy of the model is useless in a situation with imbalanced classes. This metric will tend to inflate values for incorrectly defined negative classes, which will lead to low predictive ability.

The Precision metric can be interpreted as the proportion of correctly classified objects of a positive class, while the Recall metric shows what proportion of objects of a positive class out of all objects of a positive class the algorithm has found. An important feature of these metrics is that they are calculated not on the basis of predicted estimates, but on the basis of predicted classes.

There are several metrics called $F_\beta$ which harmonize two metrics above — Precision and Recall. When evaluating the effectiveness of multimodal models, the F1 metric is traditionally used, which equally takes into account the importance of Precision and Recall.

For our task, it is more important to correctly identify the present disease than its absence. Thus, the F1 metric is suitable for our case, since we are very interested in the correct definition of a positive class.

The ROC AUC metric should not be used in our case. This metric is good when we have well-balanced classes and care about both true positive and true negative prognosis. In the case of 1:10 balance between classes, the false positive rate for highly imbalanced datasets is pulled down due to a large number of true negatives.

Unlike ROC AUC, PRAYS metric, like F1, focuses mainly on the positive class (Precision and True Positive Rate), pays less attention to the frequent negative class in imbalanced data. Therefore, this metric is also an adequate choice for our case.

Another popular metric for evaluating a model with imbalanced classes is the Matthews correlation coefficient. This coefficient is more complete compared to the F1 metric. The fact is that when using the F1 and PR AUC metrics, the key is the statement about which class is interpreted as positive — minority or majority. When exchanging labels, the value of the F1 metric will change, it will need to be recalculated. At the same time, the F1 metric completely

ignores the true negative rate, which gives a certain limitation in the interpretation of the results of the model. The Matthews metric does not have these disadvantages, so it is advisable to use it in our case.

## 3. Results and discussion

We have run the full training cycle of the multimodal model five times. Here we present the averaged results of the model (Tab. 1). To generate balanced classes, we used the ADASYN method.

**Table 1.** Comparison of accuracy and losses in different models

| Model | Test accuracy, % | Test loss, % |
|---|---|---|
| Single modality CXR [1] | 93.0 | 4.81 |
| Single modality CXR [2] | 92.8 | 3.93 |
| Single modality CXR NSGANETV2 [3] | 93.5 | 4.11 |
| Proposed model on balanced classes | 95.9 | 2.14 |
| Proposed model on imbalanced classes | 96.0 | 2.11 |

As can be seen from the above data, the accuracy of the model using artificial balancing of classes and in the original representation of classes is almost the same, even with a slight advantage in favor of imbalanced classes. As we expected, the use of probabilistic deep learning models embedded in modern transformers and convolutional networks for binary classification allows us to work without pre-calibration of imbalanced classes in the presence of a sufficient number of samples. This result could be due to the nature and power of the dataset used, as well as the successful architecture of the multimodal model. We will not dare to assert the possibility of extrapolating this statement in relation to other tasks, models and datasets. However, the potential possibility of such elimination of one of the steps of data preprocessing may become a decisive factor when choosing models in conditions of limited time and computational capabilities of researchers. Table 2 presents the results of calculating metrics for evaluating the accuracy of the proposed multimodal model on imbalanced classes.

**Table 2.** Comparison of metrics on the proposed model

| Metric / Dataset | ROC AUC | PR AUC | F1 score | Matthews coefficient |
|---|---|---|---|---|
| Train | 0.9827 | 0.9627 | 0.9577 | 0.9583 |
| Test | 0.9780 | 0.9480 | 0.9501 | 0.9455 |

The ROC AUC metric, as expected, has lower values, as it has a lower sensitivity to the minority class. This metric gives a false sense of the high accuracy of the model, but does not describe the real predictive ability of the model. The remaining metrics, PR AUC, F1 score, Matthews coefficient, give a more adequate assessment of the accuracy of the model on imbalanced data.

## Conclusions

In this study, we tested the hypothesis that imbalanced data in a multimodal deep learning model for binary classification of the definition of pneumonia is not always necessary to undergo a balancing and alignment procedure.

For the dataset under consideration with a moderate degree of class imbalance, we found out that the generation of artificial samples with subsequent affixing of their correspondence to analogues in other modalities is optional. The training of a multimodal deep learning model by probabilistic algorithms for processing individual modalities is even slightly higher in accuracy than the results of the same model on artificially balanced data.

Excluding the data balancing step from data preprocessing before training the model can significantly increase the learning rate, and at the same time solves the problem of matching artificially generated samples in various modalities.

We conducted a study of the predictive ability of the model using several metrics. For imbalanced classes, the F1 and Matthews coefficient metrics showed a more adequate assessment of the accuracy of the proposed multimodal model.

*This study is supported by the Russian Science Foundation regional grant No. 23-21-10009.*

## References

1. COVID-19 and vascular disease. EBioMedicine. 2020. Aug. Vol. 58. P. 102966. DOI: 10.1016/j.ebiom.2020.102966.
2. Soenksen L.R., Ma Y., Zeng C., *et al.* Code for generating the HAIM multimodal dataset of MIMIC-IV clinical data and x-rays. 2022. DOI: 10.13026/3F8D-QE93.
3. Qiu S., Chang G.H., Panagia M., *et al.* Fusion of deep learning models of MRI scans, Mini–Mental State Examination, and logical memory test enhances diagnosis of mild cognitive impairment. Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring. 2018. Jan. Vol. 10, no. 1. P. 737–749. DOI: 10.1016/j.dadm.2018.08.013.
4. Parcalabescu L., Frank A. MM-SHAP: A Performance-agnostic Metric for Measuring Multimodal Contributions in Vision and Language Models Tasks. 2022. DOI: 10.48550/ARXIV.2212.08158.
5. Bakalos N., Voulodimos A., Doulamis N., *et al.* Fusing RGB and Thermal Imagery with Channel State Information for Abnormal Activity Detection Using Multimodal Bidirectional LSTM. Cyber-Physical Security for Critical Infrastructures Protection. Springer International Publishing, 2021. P. 77–86. DOI: 10.1007/978-3-030-69781-5_6.
6. Sarada N., Rao K.T. A Neural Network Architecture Using Separable Neural Networks for the Identification of "Pneumonia" in Digital Chest Radiographs. International Journal of e-Collaboration. 2021. Jan. Vol. 17, no. 1. P. 89–100. DOI: 10.4018/ijec.2021010106.
7. Vashisht S., Sharma B., Lamba S. Using Support Vector Machine and Generative Adversarial Network for Multi-Classification of Pneumonia Disease. 2023 4th International Conference for Emerging Technology (INCET). IEEE, May 2023. DOI: 10.1109/incet57972.2023.10170180.
8. Yadav P., Menon N., Ravi V., Vishvanathan S. Lung-GANs: Unsupervised Representation Learning for Lung Disease Classification Using Chest CT and X-Ray Images. IEEE

Transactions on Engineering Management. 2023. Aug. Vol. 70, no. 8. P. 2774–2786. DOI: 10.1109/tem.2021.3103334.

9. Fang M., Peng S., Liang Y., *et al.* A Multimodal Fusion Model with Multi-Level Attention Mechanism for Depression Detection. SSRN Electronic Journal. 2022. DOI: 10.2139/ssrn.4102839.
10. Cai S., Wakaki R., Nobuhara S., Nishino K. RGB Road Scene Material Segmentation. Computer Vision – ACCV 2022. Springer Nature Switzerland, 2023. P. 256–272. DOI: 10.1007/978-3-031-26284-5_16.
11. Msuya H., Maiseli B.J. Deep Learning Model Compression Techniques: Advances, Opportunities, and Perspective. Tanzania Journal of Engineering and Technology. 2023. June. Vol. 42, no. 2. P. 65–83. DOI: 10.52339/tjet.v42i2.853.
12. Pereira R.M., Costa Y.M., Jr. C.N.S. MLTL: A multi-label approach for the Tomek Link undersampling algorithm. Neurocomputing. 2020. Mar. Vol. 383. P. 95–105. DOI: 10.1016/j.neucom.2019.11.076.
13. Tang B., He H., Zhang S. MCENN: A variant of extended nearest neighbor method for pattern recognition. Pattern Recognition Letters. 2020. May. Vol. 133. P. 116–122. DOI: 10.1016/j.patrec.2020.01.015.
14. Xin L., Mou T. Research on the Application of Multimodal-Based Machine Learning Algorithms to Water Quality Classification. Wireless Communications and Mobile Computing / ed. by C.-H. Wu. 2022. July. Vol. 2022. P. 1–13. DOI: 10.1155/2022/9555790.
15. Aridas C.K., Karlos S., Kanas V.G., *et al.* Uncertainty Based Under-Sampling for Learning Naive Bayes Classifiers Under Imbalanced Data Sets. IEEE Access. 2020. Vol. 8. P. 2122–2133. DOI: 10.1109/access.2019.2961784.
16. Li Y., Branco P., Zhang H. Imbalanced Multimodal Attention-Based System for Multiclass House Price Prediction. Mathematics. 2022. Dec. Vol. 11, no. 1. P. 113. DOI: 10.3390/math11010113.
17. Mathew R.M., Gunasundari R. An Oversampling Mechanism for Multimajority Datasets using SMOTE and Darwinian Particle Swarm Optimisation. International Journal on Recent and Innovation Trends in Computing and Communication. 2023. Mar. Vol. 11, no. 2. P. 143–153. DOI: 10.17762/ijritcc.v11i2.6139.
18. Chawla N.V., Bowyer K.W., Hall L.O., Kegelmeyer W.P. SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. 2002. Vol. 16. P. 321–357. DOI: 10.1613/jair.953.
19. Siriseriwan W., Sinapiromsaran K. Adaptive neighbor synthetic minority oversampling technique under 1NN outcast handling. Songklanakarin Journal of Science and Technology (SJST). 2017. Vol. 39. P. 5. DOI: 10.14456/SJST-PSU.2017.70.
20. Alhudhaif A. A novel multi-class imbalanced EEG signals classification based on the adaptive synthetic sampling (ADASYN) approach. PeerJ Computer Science. 2021. May. Vol. 7. P. 523. DOI: 10.7717/peerj-cs.523.
21. He H., Bai Y., Garcia E.A., Li S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008, part of the IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, China, June 1-6, 2008. IEEE, 2008. P. 1322–1328. DOI: 10.1109/IJCNN.2008.4633969.

УДК 004.891.3 DOI: 10.14529/cmse230302

# ПОДХОД К КЛАССИФИКАЦИИ МНОГОМОДАЛЬНЫХ ДАННЫХ О ЗАБОЛЕВАНИЯХ ПНЕВМОНИЕЙ НА ОСНОВЕ ПРОМЕЖУТОЧНОГО СЛИЯНИЯ

**© 2023 О.Н. Иванова, А.В. Мелехин, Е.В. Иванова, С. Кумар, М.Л. Цымблер**

*Южно-Уральский государственный университет*
*(454080 Челябинск, пр. им. В.И. Ленина, д. 76)*
*E-mail: onivanova@susu.ru, temamel540@gmail.com elena.ivanova@susu.ru, kumars@susu.ru, mzym@susu.ru*

Поступила в редакцию: 03.08.2023

В медицинской практике первичную диагностику заболеваний следует проводить быстро и по возможности автоматически. Обработка многомодальных данных в медицине стала повсеместно распространенным методом классификации, прогнозирования и обнаружения заболеваний. Пневмония – одно из наиболее распространенных заболеваний легких. В нашем исследовании для выявления пневмонии мы использовали рентгенограммы органов грудной клетки в качестве первой модальности и результаты лабораторных исследований пациента в качестве второй модальности. Архитектура многомодальной модели глубокого обучения была основана на промежуточном слиянии. Модель обучалась на сбалансированных и несбалансированных данных, когда наличие пневмонии определялось в 50% и 9% от общего числа случаев соответственно. Для более объективной оценки результатов мы сравнили производительность нашей модели с несколькими другими моделями с открытым исходным кодом на наших данных. Эксперименты демонстрируют высокую эффективность предложенной модели выявления пневмонии по двум модальностям даже в случаях несбалансированных классов (до 96.6%) по сравнению с результатами одномодальных моделей (до 93.5%). Мы сделали несколько интегральных оценок производительности предлагаемой модели, чтобы охватить и исследовать все аспекты многомодальных данных и особенностей архитектуры. Были показатели точности, ROC AUC, PR AUC, показателя F1 и коэффициента корреляции Мэтьюса. Используя различные метрики, мы доказали возможность и целесообразность использования предложенной модели с целью правильной классификации заболевания. Эксперименты показали, что производительность модели, обученной на несбалансированных данных, даже немного выше, чем у других рассмотренных моделей.

*Ключевые слова: многомодальная модель, промежуточное слияние, пневмония, глубокое обучение, несбалансированные данные.*

## ОБРАЗЕЦ ЦИТИРОВАНИЯ

Ivanova O.N., Melekhin A.V., Ivanova E.V., Kumar S., Zymbler M.L. Intermediate Fusion Approach for Pneumonia Classification on Imbalanced Multimodal Data // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 3. С. 19–30. DOI: 10.14529/cmse230302.



## Литература

1. COVID-19 and vascular disease // EBioMedicine. 2020. Aug. Vol. 58. P. 102966. DOI: 10.1016/j.ebiom.2020.102966.
2. Soenksen L.R., Ma Y., Zeng C., *et al.* Code for generating the HAIM multimodal dataset of MIMIC-IV clinical data and x-rays. 2022. DOI: 10.13026/3F8D-QE93.

3. Qiu S., Chang G.H., Panagia M., *et al.* Fusion of deep learning models of MRI scans, Mini–Mental State Examination, and logical memory test enhances diagnosis of mild cognitive impairment // Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring. 2018. Jan. Vol. 10, no. 1. P. 737–749. DOI: 10.1016/j.dadm.2018.08.013.

4. Parcalabescu L., Frank A. MM-SHAP: A Performance-agnostic Metric for Measuring Multimodal Contributions in Vision and Language Models Tasks. 2022. DOI: 10.48550/ARXIV.2212.08158.

5. Bakalos N., Voulodimos A., Doulamis N., *et al.* Fusing RGB and Thermal Imagery with Channel State Information for Abnormal Activity Detection Using Multimodal Bidirectional LSTM // Cyber-Physical Security for Critical Infrastructures Protection. Springer International Publishing, 2021. P. 77–86. DOI: 10.1007/978-3-030-69781-5_6.

6. Sarada N., Rao K.T. A Neural Network Architecture Using Separable Neural Networks for the Identification of "Pneumonia" in Digital Chest Radiographs // International Journal of e-Collaboration. 2021. Jan. Vol. 17, no. 1. P. 89–100. DOI: 10.4018/ijec.2021010106.

7. Vashisht S., Sharma B., Lamba S. Using Support Vector Machine and Generative Adversarial Network for Multi-Classification of Pneumonia Disease // 2023 4th International Conference for Emerging Technology (INCET). IEEE, May 2023. DOI: 10.1109/incet57972.2023.10170180.

8. Yadav P., Menon N., Ravi V., Vishvanathan S. Lung-GANs: Unsupervised Representation Learning for Lung Disease Classification Using Chest CT and X-Ray Images // IEEE Transactions on Engineering Management. 2023. Aug. Vol. 70, no. 8. P. 2774–2786. DOI: 10.1109/tem.2021.3103334.

9. Fang M., Peng S., Liang Y., *et al.* A Multimodal Fusion Model with Multi-Level Attention Mechanism for Depression Detection // SSRN Electronic Journal. 2022. DOI: 10.2139/ssrn.4102839.

10. Cai S., Wakaki R., Nobuhara S., Nishino K. RGB Road Scene Material Segmentation // Computer Vision – ACCV 2022. Springer Nature Switzerland, 2023. P. 256–272. DOI: 10.1007/978-3-031-26284-5_16.

11. Msuya H., Maiseli B.J. Deep Learning Model Compression Techniques: Advances, Opportunities, and Perspective // Tanzania Journal of Engineering and Technology. 2023. June. Vol. 42, no. 2. P. 65–83. DOI: 10.52339/tjet.v42i2.853.

12. Pereira R.M., Costa Y.M., Jr. C.N.S. MLTL: A multi-label approach for the Tomek Link undersampling algorithm // Neurocomputing. 2020. Mar. Vol. 383. P. 95–105. DOI: 10.1016/j.neucom.2019.11.076.

13. Tang B., He H., Zhang S. MCENN: A variant of extended nearest neighbor method for pattern recognition // Pattern Recognition Letters. 2020. May. Vol. 133. P. 116–122. DOI: 10.1016/j.patrec.2020.01.015.

14. Xin L., Mou T. Research on the Application of Multimodal-Based Machine Learning Algorithms to Water Quality Classification // Wireless Communications and Mobile Computing / ed. by C.-H. Wu. 2022. July. Vol. 2022. P. 1–13. DOI: 10.1155/2022/9555790.

15. Aridas C.K., Karlos S., Kanas V.G., *et al.* Uncertainty Based Under-Sampling for Learning Naive Bayes Classifiers Under Imbalanced Data Sets // IEEE Access. 2020. Vol. 8. P. 2122–2133. DOI: 10.1109/access.2019.2961784.

16. Li Y., Branco P., Zhang H. Imbalanced Multimodal Attention-Based System for Multiclass House Price Prediction // Mathematics. 2022. Dec. Vol. 11, no. 1. P. 113. DOI: 10.3390/math11010113.

17. Mathew R.M., Gunasundari R. An Oversampling Mechanism for Multimajority Datasets using SMOTE and Darwinian Particle Swarm Optimisation // International Journal on Recent and Innovation Trends in Computing and Communication. 2023. Mar. Vol. 11, no. 2. P. 143–153. DOI: 10.17762/ijritcc.v11i2.6139.

18. Chawla N.V., Bowyer K.W., Hall L.O., Kegelmeyer W.P. SMOTE: Synthetic Minority Over-sampling Technique // J. Artif. Intell. Res. 2002. Vol. 16. P. 321–357. DOI: 10.1613/jair.953.

19. Siriseriwan W., Sinapiromsaran K. Adaptive neighbor synthetic minority oversampling technique under 1NN outcast handling // Songklanakarin Journal of Science and Technology (SJST). 2017. Vol. 39. P. 5. DOI: 10.14456/SJST-PSU.2017.70.

20. Alhudhaif A. A novel multi-class imbalanced EEG signals classification based on the adaptive synthetic sampling (ADASYN) approach // PeerJ Computer Science. 2021. May. Vol. 7. P. 523. DOI: 10.7717/peerj-cs.523.

21. He H., Bai Y., Garcia E.A., Li S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning // Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008, part of the IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, China, June 1-6, 2008. IEEE, 2008. P. 1322–1328. DOI: 10.1109/IJCNN.2008.4633969.

Иванова Ольга Николаевна, к.п.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Мелехин Артем Викторович, студент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Иванова Елена Владимировна, к.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Кумар Сэчин, PhD, с.н.с., лаборатория больших данных и машинного обучения, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Цымблер Михаил Леонидович, д.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

УДК 004.021, 004.032.24, 004.051, 004.272 DOI: 10.14529/cmse230303

# АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ И ИСПОЛНЕНИЕ ЭФФЕКТИВНЫХ ПРОГРАММ ДЛЯ ЧИСЛЕННЫХ АЛГОРИТМОВ

© 2023 В.Н. Алеева

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

Поступила в редакцию: 29.06.2023

Проектировать эффективные параллельные программы для многопроцессорных архитектур сложно, так как нет четких формальных правил, которых необходимо придерживаться. Для решения этой проблемы при реализации численных алгоритмов может применяться концепция $Q$-детерминанта. Данная теория позволяет проводить автоматизированный анализ ресурса параллелизма алгоритма, автоматизированное сравнение ресурсов параллелизма алгоритмов, решающих одну и ту же алгоритмическую проблему, проектировать эффективные программы для реализации алгоритмов с помощью специально разработанного метода проектирования, повысить эффективность реализации численных методов и алгоритмических проблем. Результаты, полученные на основе концепции $Q$-детерминанта, представляют собой один из вариантов решения проблемы эффективной реализации численных алгоритмов, методов и алгоритмических проблем на параллельных вычислительных системах. Однако пока остается не решенной фундаментальная проблема автоматизированного проектирования и исполнения для любого численного алгоритма программы, реализующей алгоритм эффективно. В статье описана разработка единой для численных алгоритмов программной системы проектирования и исполнения $Q$-эффективных программ — эффективных программ, спроектированных с помощью концепции $Q$-детерминанта. Система предназначена для использования на параллельных вычислительных системах с общей памятью. Она состоит из компилятора и виртуальной машины. Компилятор преобразует представление алгоритма в форме $Q$-детерминанта в исполняемую программу, использующую ресурс параллелизма алгоритма полностью. Виртуальная машина исполняет программу, полученную с помощью компилятора. В статье также приведено экспериментальное исследование созданной программной системы с применением суперкомпьютера «Торнадо ЮУрГУ».

*Ключевые слова: $Q$-детерминант алгоритма, представление алгоритма в форме $Q$-детерминанта, $Q$-эффективная реализация алгоритма, ресурс параллелизма алгоритма, программная $Q$-система, параллельная вычислительная система, параллельная программа, $Q$-эффективная программа.*

## ОБРАЗЕЦ ЦИТИРОВАНИЯ

Алеева В.Н. Автоматизированное проектирование и исполнение эффективных программ для численных алгоритмов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 3. С. 31–49. DOI: 10.14529/cmse230303.

## Введение

Концепция $Q$-детерминанта является одним из подходов к распараллеливанию численных алгоритмов. Впервые она была изложена в работе [1], ее развитие и применение описаны в работах [2, 3, 13–18]. Результаты, полученные на основе концепции $Q$-детерминанта, представляют собой один из вариантов решения проблемы эффективной реализации численных алгоритмов, численных методов и алгоритмических проблем на параллельных вычислительных системах (ПВС). Кратко их можно сформулировать так.

1) Разработана программная $Q$-система для исследования ресурса параллелизма алгоритмов [2, 14–17].

2) Предложен метод проектирования $Q$-эффективных программ для выполнения $Q$-эффективных реализаций алгоритмов, использующих ресурс параллелизма алгоритмов полностью [3, 13–16].
3) Описана технология $Q$-эффективного программирования для повышения эффективности реализации численных методов решения алгоритмических проблем и самих алгоритмических проблем [14, 15].

Пока остается не решенной проблема автоматизированного проектирования и исполнения для любого численного алгоритма программы, реализующей алгоритм эффективно.

Цель данного исследования заключается в том, чтобы показать возможность создания программной системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов. Для достижения цели решаются следующие задачи.

1) Проектирование и реализация компилятора, преобразующего представление алгоритма в форме $Q$-детерминанта в исполняемую программу, использующую ресурс параллелизма алгоритма полностью.
2) Проектирование и реализация виртуальной машины, исполняющей программу, полученную с помощью компилятора.
3) Проведение экспериментального исследования функционирования программной системы, состоящей из компилятора и виртуальной машины, с применением суперкомпьютера «Торнадо ЮУрГУ».

В решение данных задач внесли вклад студенты кафедры системного программирования ЮУрГУ Юферов А.В. (задачи 1 и 2) и Иванов А.Д. (задача 3).

Статья организована следующим образом. Раздел 1 содержит обзор работ по теме исследования. В разделе 2 приведены используемые в статье понятия концепции $Q$-детерминанта. Раздел 3 содержит описание проектирования, а раздел 4 описание реализации программной системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов. В разделе 5 представлены результаты функционального тестирования на персональном компьютере созданной программной системы и приведено ее экспериментальное исследование с применением суперкомпьютера «Торнадо ЮУрГУ». Заключение содержит краткое изложение полученных результатов, выводы об их значении и применении, описание дальнейших исследований.

## 1. Обзор работ по теме исследования

В сфере высокопроизводительных вычислений одним из ведущих ученых мира является Джек Донгарра (Jack Dongarra). На протяжении десятков лет производительность вычислительной техники росла в соответствии с законом Мура [25]. И если большинство программных средств не поспевало за аппаратными достижениями, то программное обеспечение высокопроизводительных вычислений справлялось с этим успешно. Во многом это объясняется использованием алгоритмов Дж. Донгарры. В университете Теннесси для поддержки библиотеки матрично-векторных операций DPLASMA [22] научным коллективом с участием Дж. Донгарры развивается подход к конструированию и исполнению параллельных программ на основе машинно-независимого представления прикладного алгоритма в виде бесконтурного ориентированного графа. Это позволяет обеспечивать высокопроизводительную реализацию прикладных алгоритмов благодаря планированию вычислений и динамической поддержке исполнительной системы PaRSEC [23]. Дж. Донгарра занимается

также вопросами автоматического распараллеливания алгоритмов [20]. Ассоциация вычислительной техники (ACM) присудила Джеку Донгарре премию Тьюринга за 2021 год.

В Российской Федерации одним из наиболее развитых направлений исследований в области параллельных вычислений является направление, созданное В.В. Воеводиным. Именно оно наиболее близко к направлению, использующему концепцию $Q$-детерминанта. Из работ, связанных с направлением исследований В.В. Воеводина, отметим монографию [4], где проводится очень важное и развитое исследование параллельной структуры алгоритмов и программ для их реализации на ПВС. Исследование параллельной структуры алгоритмов основано на описании и изучении информационных графов алгоритмов. Оно применяется в открытой энциклопедии AlgoWiki [9, 19]. Этот проект сформировался под руководством Вл. В. Воеводина и Дж. Донгарры. В энциклопедии описываются и изучаются графы конкретных алгоритмов. Это позволяет реализовывать алгоритмы эффективно. Однако в работах данного направления исследований программное обеспечение для автоматизированного исследования и использования ресурса параллелизма алгоритмов не рассматривается.

Еще один подход к эффективной реализации алгоритмов на ПВС разрабатывается в Стэндфордском университете под руководством А. Айкена (A. Aiken). Подход заключается в том, что отдельно описывается прикладной алгоритм и способ его отображения на вычислительные ресурсы ПВС. Научным коллективом, разрабатывающим подход, создана система Legion [21], реализующая этот подход. Недостатком системы является отсутствие автоматизации в конструировании способа отображения прикладного алгоритма на вычислительные ресурсы.

Существуют направления исследований, в которых предлагаются универсальные подходы к созданию параллельных программ. Одним из результатов таких исследований является Т-система [26]. Она обеспечивает среду программирования с поддержкой автоматического динамического распараллеливания программ. Вместе с тем нельзя утверждать, что параллельные программы, созданные с помощью Т-системы, полностью используют ресурс параллелизма реализуемого ими алгоритма, поскольку распараллеливаемые программы могут не содержать всех реализаций алгоритма, в частности, при их создании могла быть не учтена самая параллельная реализация. Метод синтеза параллельных программ — еще один подход к созданию параллельных программ. Он состоит в том, чтобы с помощью базы знаний параллельных алгоритмов конструировать параллельные алгоритмы для решения более сложных задач. На основе метода разработаны технология фрагментарного программирования, язык его реализации и система программирования LuNA [12]. Такой подход к созданию параллельных программ является универсальным, но не решает проблему автоматизированного исследования и использования ресурса параллелизма алгоритмов. Отметим еще одно направление исследований. Для преодоления ресурсных ограничений ПВС предлагаются методы построения параллельных архитектурно-независимых программ с использованием функционального языка программирования [24]. Однако не показано, что созданные программы используют весь ресурс параллелизма алгоритмов.

Многочисленными являются исследования, заключающиеся в разработке параллельных программ, учитывающих специфику алгоритмов, а также архитектуру ПВС. Их примерами являются [8, 11]. Такие исследования повышают эффективность реализации конкретных алгоритмов или реализации алгоритмов на ПВС определенной архитектуры, однако они не обеспечивают универсального подхода к разработке эффективных параллельных программ.

Приведенный обзор, а также факт, что ресурс параллелизма алгоритмов при реализации на ПВС часто используется не полностью, по-видимому, доказывают, что в настоящее время нет признанных в научном сообществе и широко применяемых на практике решений задач исследования и использования ресурса параллелизма численных алгоритмов, тем более в автоматизированном режиме. Другими словами, проблема автоматизированного распараллеливания алгоритмов, проектирования и исполнения реализующих их эффективных программ не решена. По мнению автора перспективным для ее решения является подход, основанный на унифицированном представлении алгоритма, показывающем ресурс параллелизма в полной мере. Например, подход, основанный на концепции $Q$-детерминанта.

## 2. Теоретические основы разработки программной системы

Приведем понятия концепции $Q$-детерминанта, используемые в данном исследовании.

Рассмотрим алгоритмическую проблему $\bar{y} = F(N, B)$, где $N = \{n_1, \ldots, n_k\}$ — множество параметров размерности проблемы или $N$ — пустое множество, $B$ — множество входных данных, $\bar{y} = \{y_1, \ldots, y_m\}$ — множество выходных данных, при этом целое число $m$ является либо константой, либо значением вычислимой функции параметров $N$ при условии, что $N \neq \varnothing$. Здесь $n_i$ $(i \in \{1, \ldots, k\})$ равно любому положительному целому числу. Если $N = \{n_1, \ldots, n_k\}$, то через $\bar{N} = \{\bar{n}_1, \ldots, \bar{n}_k\}$ обозначим набор из $k$ положительных целых чисел, где $\bar{n}_i$ — некоторое заданное значение параметра $n_i$ для каждого $i \in \{1, \ldots, k\}$. Через $\{\bar{N}\}$ обозначим множество всех возможных $k$-наборов $\bar{N}$. Пусть $\mathfrak{A}$ — алгоритм для решения алгоритмической проблемы, $Q$ — набор операций, используемых алгоритмом $\mathfrak{A}$.

**Определение 1.** Определим выражение над $B$ и $Q$, как терм в стандартном смысле математической логики [5]. Каждое выражение $w$ имеет уровень вложенности, обозначим его через $T^w$.

**Пример 1.** Выражения и их уровни вложенности:
1) $w_1 = b_1 \times (b_2 + b_3)/b_4$, $T^{w_1} = 3$:
2) $w_2 = (b_1 \geq b_2) \vee ((b_3 \times b_4) \leq b_5 \times (b_6 + b_7))$, $T^{w_2} = 4$.

**Определение 2.** Мы называем выражение цепочкой длины $n$, если оно является результатом применения некоторой ассоциативной операции из $Q$ к $n$ выражениям.

**Определение 3.** Если $N = \varnothing$, то любое выражение $w$ над $B$ и $Q$ мы называем безусловным $Q$-термом. Пусть $N \neq \varnothing$ и $V$ — множество всех выражений над $B$ и $Q$. Тогда любое отображение $w : \{\bar{N}\} \to V \cup \varnothing$ также называется безусловным $Q$-термом.

Пусть $N = \varnothing$ и $w$ — безусловный $Q$-терм. Предположим, что выражение $w$ над $B$ и $Q$ имеет значение логического типа при любой интерпретации переменных $B$. Тогда безусловный $Q$-терм $w$ называется безусловным логическим $Q$-термом. Пусть $N \neq \varnothing$ и $w$ — безусловный $Q$-терм. Если выражение $w(\bar{N})$ для каждого $\bar{N} \in \{\bar{N}\}$ имеет значение логического типа при любой интерпретации переменных $B$, то безусловный $Q$-терм $w$ называется безусловным логическим $Q$-термом.

Пусть $u_1, \ldots, u_l$ — безусловные логические $Q$-термы, $w_1, \ldots, w_l$ — безусловные $Q$-термы. Тогда множество $l$ пар $(\widehat{u}, \widehat{w}) = \{(u_i, w_i)\}_{i \in \{1,\ldots,l\}}$ называется условным $Q$-термом длины $l$.

Пусть $(\widehat{u}, \widehat{w}) = \{(u_i, w_i)\}_{i=1,2,\ldots}$ — счетное множество пар безусловных $Q$-термов. Предположим, что $\{(u_i, w_i)\}_{i \in \{1,\ldots,l\}}$ — условный $Q$-терм для любого $l < \infty$. Тогда мы называем $(\widehat{u}, \widehat{w})$ условным бесконечным $Q$-термом.

$Q$-термы можно вычислять. Процесс нахождения значений $Q$-термов описан в [2, 15].

**Определение 4.** Пусть $M = \{1, \dots, m\}$. Предположим, что алгоритм $\mathfrak{A}$ состоит в нахождении для каждого $i \in M$ значения $y_i$ путем вычисления значения $Q$-терма $f_i$. Тогда набор $Q$-термов $\{f_i \mid i \in M\}$ называется $Q$-детерминантом алгоритма $\mathfrak{A}$. Система уравнений $\{y_i = f_i \mid i \in M\}$ называется представлением алгоритма $\mathfrak{A}$ в форме $Q$-детерминанта.

**Определение 5.** Процесс вычисления $Q$-термов $\{f_i \mid i \in M\}$ алгоритма $\mathfrak{A}$ называется реализацией алгоритма $\mathfrak{A}$. Реализация алгоритма $\mathfrak{A}$ называется параллельной, если существуют операции, которые выполняются одновременно.

Предположим, что $U$, $C$ и $I$ образуют разбиение множества $M = \{1, \dots, m\}$ с пустыми членами, то есть: $U \cup C \cup I = M$; $U \cap C = U \cap I = C \cap I = \varnothing$; кроме того, одно или два подмножества $U$, $C$ и $I$ могут быть пустыми. Предположим также, что подмножества $U$, $C$ и $I$ можно связать с подмножествами множества $Q$-термов $\{f_i \mid i \in M\}$ таким образом:

1) для каждого $i \in U$ существует $Q$-терм $f_i$, который является безусловным, и $f_i = w^i$;
2) для каждого $i \in C$ существует $Q$-терм $f_i$, который является условным, и $f_i = \{(u^i_j, w^i_j)\}_{j \in \{1,\dots,l(i)\}}$, где $l(i)$ — либо константа, либо значение вычислимой функции от $N$, если $N \neq \varnothing$;
3) для каждого $i \in I$ существует $Q$-терм $f_i$, который является условным бесконечным, и $f_i = \{(u^i_j, w^i_j)\}_{j \in \{1,2,\dots\}}$.

**Определение 6.** Опишем реализацию алгоритма $\mathfrak{A}$, называемую $Q$-эффективной.

Пусть $N = \varnothing$. Зададим интерпретацию переменных $B$. Будем вычислять выражения

$$W = \left\{w^i(i \in U); u^i_j, w^i_j(i \in C, j \in \{1, \dots, l(i)\}); u^i_j, w^i_j(i \in I, j \in \{1, 2, \dots\})\right\} \tag{1}$$

одновременно, параллельно. Мы говорим, что операция готова к выполнению, если вычислены значения всех ее операндов. При вычислении каждого из выражений $W$ (см. (1)) мы выполняем операции, как только они готовы к выполнению. Если несколько операций цепочки готовы к выполнению, то они выполняются по схеме сдваивания. Например, вычисление цепочки $a_1 + a_2 + a_3 + a_4$ по схеме сдваивания выполняется так: сначала вычисляем $b_1 = a_1 + a_2$ и $b_2 = a_3 + a_4$ одновременно, затем $c = b_1 + b_2$. Если для $i \in C \cup I$ и $j \in \{1, 2, \dots\}$ $u^i_j$ имеет значение **false**, то вычисление соответствующего $w^i_j$ прекращается. Если для $i \in C \cup I$ и $j \in \{1, 2, \dots\}$ вычисление пары $(u^i_j, w^i_j)$ приводит к тому, что значение одного выражения не определено, то вычисление второго выражения прекращается. Если для $i \in C \cup I$ вычисление некоторой пары $(u^i_{j_0}, w^i_{j_0})$ приводит к определению их значений и значение $u^i_{j_0}$ **true**, то вычисление выражений $u^i_j, w^i_j$ прекращается для любого $j \neq j_0$.

Теперь пусть $N \neq \varnothing$. Зададим интерпретацию переменных $B$ и $\bar{N} \in \{\bar{N}\}$. Получаем множество выражений

$$W(\bar{N}) = \left\{w^i(\bar{N})(i \in U); u^i_j(\bar{N}), w^i_j(\bar{N})(i \in C, j \in \{1, \dots, l(i)\}); \right. \\ \left. u^i_j(\bar{N}), w^i_j(\bar{N})(i \in I, j \in \{1, 2, \dots\})\right\}. \tag{2}$$

Выражения $W(\bar{N})$ (см. (2)) вычисляются по аналогии с выражениями $W$ (см. (1)).

Цепочки должны вычисляться по схеме сдваивания, так как она обеспечивает минимальное время вычисления цепочки. Обоснуем это утверждение. Высота каждого из алгоритмов, вычисляющих цепочку длины $n$, составляет не менее $\lceil \log n \rceil$ и не более $n - 1$. При

этом высота алгоритма, использующего схему сдваивания, равна $\lceil \log n \rceil$, т.е. является минимальной. Предположим, что программы, реализующие алгоритмы, имеют одинаковую вычислительную инфраструктуру, т.е. условия разработки и выполнения [3, 15]. Тогда меньше всего времени для выполнения последовательных инструкций понадобится программе для реализации алгоритма, использующего схему сдваивания. Следовательно, в соответствии с законом Амдала программа, реализующая алгоритм с использованием схемы сдваивания, будет иметь время выполнения не больше, чем каждая из остальных программ.

**Определение 7.** Реализация алгоритма $\mathfrak{A}$ называется выполнимой, если одновременно должно выполняться конечное (непустое) множество операций.

$Q$-эффективная реализация полностью использует ресурс параллелизма алгоритма, который опишем далее. Для этого будем использовать следующие условия и обозначения.

1. Алгоритм $\mathfrak{A}$ представлен в форме $Q$-детерминанта $y_i = f_i$ для всех $i \in M$.
2. Значения $Q$-термов $f_i$ для всех $i \in M$ определяются при любой интерпретации переменных $B$ и для любого $\bar{N} \in \{\bar{N}\}$, если $N \neq \varnothing$.
3. Пусть $N = \varnothing$ и $I \neq \varnothing$. Тогда при заданной интерпретации переменных $B$ для любого $i \in I$ существует пара выражений $u^i_{j_i}, w^i_{j_i}$ такая, что значение $u^i_{j_i}$ равно **true** и значение $w^i_{j_i}$ определено. Введем обозначение

$$\widetilde{W} = \left\{w^i (i \in U); u^i_j, w^i_j (i \in C, j \in \{1, \ldots, l(i)\}); u^i_{j_i}, w^i_{j_i} (i \in I)\right\}.$$

4. Пусть $N \neq \varnothing$ и $I \neq \varnothing$. Тогда при заданной интерпретации переменных $B$ для любого $\bar{N} \in \{\bar{N}\}$ и $i \in I$ существует пара выражений $u^i_{j_i}(\bar{N}), w^i_{j_i}(\bar{N})$ такая, что значение $u^i_{j_i}(\bar{N})$ равно **true**, а значение $w^i_{j_i}(\bar{N})$ определено. Введем обозначение

$$\widetilde{W}(\bar{N}) = \left\{w^i(\bar{N}) (i \in U); u^i_j(\bar{N}), w^i_j(\bar{N}) (i \in C, j \in \{1, \ldots, l(i)\}); u^i_{j_i}(\bar{N}), w^i_{j_i}(\bar{N}) (i \in I)\right\}.$$

5. $Q$-эффективная реализация алгоритма $\mathfrak{A}$ выполнима.

**Определение 8.** Обозначим через $D_{\mathfrak{A}}$ высоту и через $P_{\mathfrak{A}}$ ширину алгоритма $\mathfrak{A}$, характеризующие его ресурс параллелизма, и определим их следующим образом.

Если $N = \varnothing$, то

$$D_{\mathfrak{A}} = \begin{cases} \max\limits_{w \in W} T^w, & \text{если } I = \varnothing, \\ \max\limits_{w \in \widetilde{W}} T^w, & \text{если } I \neq \varnothing; \end{cases} \qquad P_{\mathfrak{A}} = \max_{1 \leq r \leq D_{\mathfrak{A}}} \sum_{w \in W} O^w_r,$$

где $O^w_r$ — количество операций уровня вложенности $r$ выражения $w$.

Если $N \neq \varnothing$, то

$$D_{\mathfrak{A}}(\bar{N}) = \begin{cases} \max\limits_{w(\bar{N}) \in W(\bar{N})} T^{w(\bar{N})}, & \text{если } I = \varnothing, \\ \max\limits_{w(\bar{N}) \in \widetilde{W}(\bar{N})} T^{w(\bar{N})}, & \text{если } I \neq \varnothing; \end{cases} \qquad P_{\mathfrak{A}}(\bar{N}) = \max_{1 \leq r \leq D_{\mathfrak{A}}(\bar{N})} \sum_{w(\bar{N}) \in W(\bar{N})} O^{w(\bar{N})}_r,$$

где $O^{w(\bar{N})}_r$ — количество операций уровня вложенности $r$ выражения $w(\bar{N})$.

$D_{\mathfrak{A}}$ характеризует время выполнения $Q$-эффективной реализации алгоритма, а $P_{\mathfrak{A}}$ — количество вычислителей вычислительной системы (вычислительных ядер, процессоров), необходимое для выполнения $Q$-эффективной реализации алгоритма, а также масштабируемость алгоритма.

**Определение 9.** Программа называется $Q$-эффективной, если она выполняет $Q$-эффективную реализацию алгоритма.

В исследованиях на основе концепции $Q$-детерминанта для оценки эффективности параллельных программ используются их динамические характеристики: время выполнения, ускорение и эффективность. Ускорение программы вычисляется по формуле $S = T_1/T_p$, а эффективность по формуле $E = S/p$, где $T_1$ — время выполнения программы на одном вычислительном ядре процессора ПВС, $T_p$ — время выполнения программы на $p$ вычислительных ядрах одного или нескольких процессоров ПВС, $p$ — количество используемых программой вычислительных ядер. Мы считаем, что программа эффективнее другой, если она имеет динамические характеристики лучше, чем другая программа.

$Q$-эффективная программа использует весь ресурс параллелизма алгоритма. Но когда она выполняется на ПВС, то возможно, что из-за нехватки вычислительных ресурсов ПВС не все операции будут выполняться по мере их готовности к выполнению. В результате будет выполняться реализация алгоритма, не использующая весь ресурс параллелизма алгоритма. Вместе с тем в статье [15, раздел 5.4] доказана следующая теорема. Предположим, что для численного алгоритма имеются $Q$-эффективная и не $Q$-эффективная программы с одинаковой вычислительной инфраструктурой. Тогда динамические характеристики $Q$-эффективной программы не уступают динамическим характеристикам не $Q$-эффективной программы. Следовательно, каждая из $Q$-эффективных программ эффективна для своей вычислительной инфраструктуры. $Q$-эффективная реализация получила свое название из-за этого факта. Этот факт подтвержден также экспериментально [3, 15]. Он убеждает в том, что разрабатывать и использовать $Q$-эффективные программы целесообразно.

## 3. Проектирование программной системы

Разработанный ранее метод проектирования параллельной программы для выполнения $Q$-эффективной реализации численного алгоритма [3, 13–16] может применять любой разработчик при проектировании для любого численного алгоритма программы, использующей ресурс параллелизма алгоритма полностью. Однако для этого нужно знать, что такое $Q$-детерминант алгоритма, и уметь его строить для конкретных алгоритмов. Но представим, что разработчику достаточно уметь описывать классическим способом численные алгоритмы, которые он применяет, например, с помощью блок-схемы, и иметь доступ к программному обеспечению, которое по описанию алгоритма получает $Q$-эффективную реализацию, а затем ее выполняет. По-видимому, такая возможность важна. Но можно ли ее реализовать? Подсистема программной $Q$-системы [2, 14–17] преобразует блок-схему алгоритма в его представление в форме $Q$-детерминанта, которое является унифицированным представлением численных алгоритмов. Для лучшего понимания приведем простой пример представления алгоритма в форме $Q$-детерминанта, полученного с помощью $Q$-системы.

**Пример 2.** Рассмотрим алгоритм скалярного произведения векторов

$$\vec{A} = (A(1), A(2), \ldots, A(8)) \text{ и } \vec{B} = (B(1), B(2), \ldots, B(8)),$$

использующий схему сдваивания. Его представление в форме $Q$-детерминанта имеет вид

$$S = ((A(1)\cdot B(1) + A(2)\cdot B(2)) + (A(3)\cdot B(3) + A(4)\cdot B(4)))+ \\ + ((A(5)\cdot B(5) + A(6)\cdot B(6)) + (A(7)\cdot B(7) + A(8)\cdot B(8))). \quad (3)$$

На рис. 1 показано представление в форме $Q$-детерминанта алгоритма скалярного произведения векторов $\vec{A}$ и $\vec{B}$, полученное с помощью $Q$-системы. Представление использует обозначения: «op» — операция, «f0» — первый операнд операции, «s0» — второй операнд операции. Уравнение (3) и рис. 1 описывают одно и то же вычисление.


```
S= ;{
     "op":"+","f0":{
                    "op":"+","f0":{
                                   "op":"+","f0":{
                                                  "op":"+","f0":{
                                                                 "op":"*","f0":"A(1)","s0":"B(1)"
                                                                 },
                                                  "s0":{
                                                                 "op":"*","f0":"A(2)","s0":"B(2)"
                                                                 }
                                                  },
                                   "s0":{
                                                  "op":"+","f0":{
                                                                 "op":"*","f0":"A(3)","s0":"B(3)"
                                                                 },
                                                  "s0":{
                                                                 "op":"*","f0":"A(4)","s0":"B(4)"
                                                                 }
                                                  }
                                   },
                    "s0":{
                                   "op":"+","f0":{
                                                  "op":"+","f0":{
                                                                 "op":"*","f0":"A(5)","s0":"B(5)"
                                                                 },
                                                  "s0":{
                                                                 "op":"*","f0":"A(6)","s0":"B(6)"
                                                                 }
                                                  },
                                   "s0":{
                                                  "op":"+","f0":{
                                                                 "op":"*","f0":"A(7)","s0":"B(7)"
                                                                 },
                                                  "s0":{
                                                                 "op":"*","f0":"A(8)","s0":"B(8)"
                                                                 }
                                                  }
                                   }
                    }
}
```


**Рис. 1.** Представление в форме $Q$-детерминанта алгоритма скалярного произведения векторов длины 8

Используя представление алгоритмов в форме $Q$-детерминантов, можно разрабатывать единое для всех численных алгоритмов программное обеспечение. Примером является программное обеспечение $Q$-системы для получения $Q$-эффективной реализации любого численного алгоритма и вычисления на ее основе характеристик ресурса параллелизма алгоритма. Аналогично возможно разработать единое для всех численных алгоритмов программное обеспечение для получения $Q$-эффективной реализации алгоритма и проектирования на ее основе исполняемой программы для выполнения $Q$-эффективной реализации. Эта идея лежит в основе создания единой программной системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов.

Для создания программной системы мы использовали следующую постановку задачи. В состав программной системы должны входить два программных продукта:

1) компилятор — преобразователь представления алгоритма в форме $Q$-детерминанта в код программы;
2) виртуальная машина — исполнитель создаваемых компилятором программ.

На вход компилятор должен принимать два файла:

1) файл с представлением алгоритма в форме $Q$-детерминанта, полученный $Q$-системой;
2) текстовый файл с необходимыми для построения программы метаданными алгоритма, создаваемый пользователем системы.

В качестве выходных данных компилятор должен создавать бинарный файл с исполняемой виртуальной машиной программой. В заголовочной секции файл должен содержать информацию о входных и выходных данных алгоритма, получаемых компилятором из файла метаданных алгоритма, а также дополнительных параметрах, позволяющих виртуальной машине исполнять программу, например, объем необходимой алгоритму памяти.

Виртуальная машина в качестве входных параметров должна принимать:

1) исполняемую программу, созданную с помощью компилятора;
2) пути файлов входных и выходных данных исполняемого алгоритма.

Файл входных данных алгоритма готовит пользователь. Результатом работы виртуальной машины является файл со значениями выходных данных. Пользователь на своем рабочем компьютере может с помощью компилятора готовить программу для исполнения, а исполнение может осуществляться на персональном компьютере или ПВС. Виртуальная машина в данном исследовании должна быть ориентирована на ПВС с общей памятью.

При проектировании программной системы была разработана диаграмма размещения ее компонентов, показанная на рис. 2. Компонент «$Q$-система» представляет собой подси-

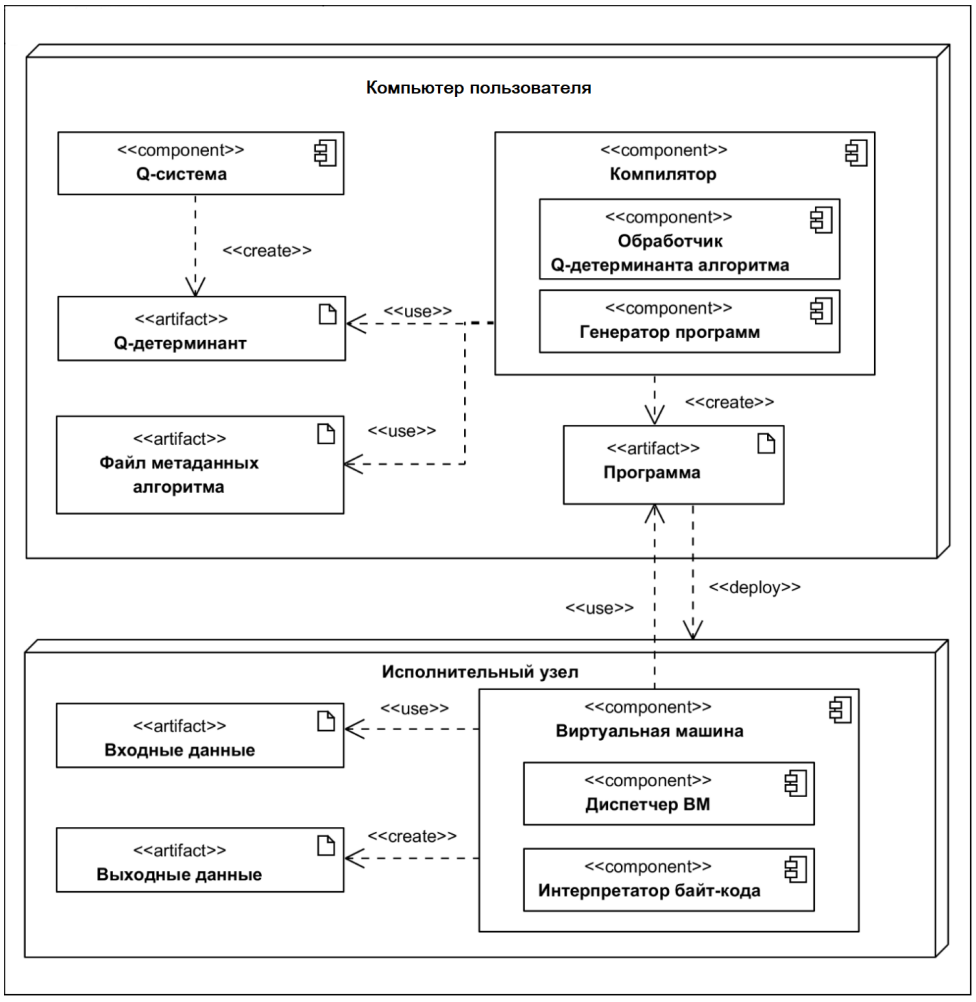



**Рис. 2.** Диаграмма размещения компонентов программной системы

стему $Q$-системы для формирования по блок-схеме алгоритма его представления в форме $Q$-детерминанта, которое обозначено на диаграмме артефактом «$Q$-детерминант». Компонент «Компилятор» преобразует представление алгоритма в форме $Q$-детерминанта в код программы. Артефакт «Файл метаданных алгоритма» — текстовый файл, создаваемый пользователем. Он используется компилятором для информации о входных данных алгоритма, их описания, а также текстового описания самого алгоритма. Подкомпонент «Обработчик $Q$-детерминанта алгоритма», используя $Q$-детерминант, строит в памяти структуру, содержащую операции $Q$-детерминанта и связи между ними. Назовем эту структуру деревом алгоритма. Подкомпонент «Генератор программ» записывает в бинарном виде метаданные алгоритма и сформированную с помощью дерева алгоритма очередь команд для выполнения $Q$-эффективной реализации алгоритма в файл программы, обозначенный артефактом «Программа». Компонент «Виртуальная машина» исполняет программу, созданную компонентом «Компилятор». Подкомпонент «Диспетчер ВМ» обеспечивает распределение задач по потокам исполнения, в которых работают объекты компонента «Интерпретатор байт-кода». Виртуальная машина получает поток команд из файла программы, а также читает входные данные алгоритма из файла, обозначенного артефактом «Входные данные». После окончания работы виртуальной машины полученные выходные данные записываются в файл, обозначенный артефактом «Выходные данные».

## 4. Реализация программной системы

Основной упор при выборе инструментов для реализации был сделан на кроссплатформенность, открытость исходного кода и производительность конечного решения, чтобы не потерять преимущество, которое предоставляет $Q$-эффективная реализация алгоритма. В качестве основного языка разработки был выбран язык $C{+}{+}$ из-за скорости его работы, гибкости и кроссплатформенности.

Рассмотрим некоторые основные решения по реализации программной системы.

Файл метаданных алгоритма, который использует компилятор, имеет формат YAML. Он содержит три корневых атрибута: description — описывающая строка, которая будет показана пользователю при выводе информации о программе виртуальной машиной; input parameters — список входных данных алгоритма; output parameters — список выходных данных алгоритма. Во время компиляции компилятор проверяет, что все индексы в пределах заданных в этом файле значений и при обнаружении выхода за пределы прерывает процесс с ошибкой.

В качестве формата файла программы используется разработанный бинарный формат, описанный далее. Таким образом, программы, создаваемые компилятором, имеют специфику. Поэтому, чтобы исполнять их, потребовалось разработать программное обеспечение, которое было названо виртуальной машиной. Файл программы включает заголовок, секции с дополнительными данными и секцию с командами. Заголовок схематично показан на рис. 3. В первой строке размещаются сигнатура данных и версия формата файла. Поле «Сигнатура данных» содержит константу, позволяющую однозначно идентифицировать тип данных. Для того чтобы убедиться, что файл является программой, которую виртуальная машина способна выполнить, реализация класса, читающего данные программы, проверяет значение сигнатуры данных. Пользователь, открыв программу в текстовом редакторе, по значению сигнатуры данных сможет понять, является ли файл программой. Поле «Версия формата файла» позволяет устанавливать версии файлов программ. Это

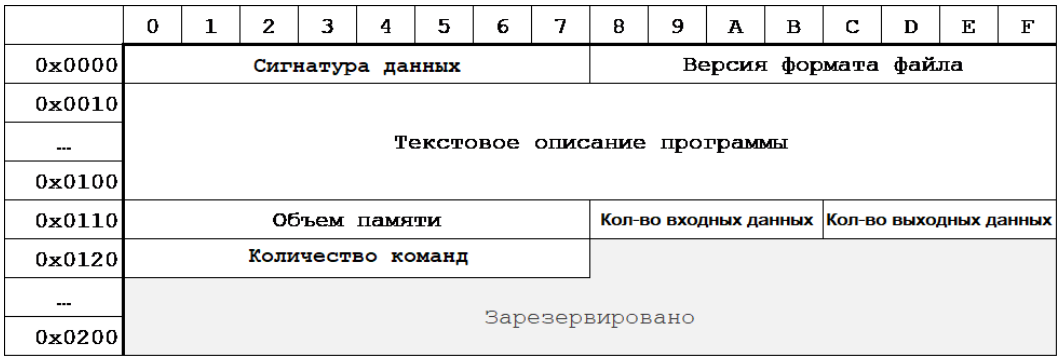



**Рис. 3.** Схема заголовка файла программы

необходимо для того, чтобы добавлять новые возможности в связку компилятор — виртуальная машина. Поле «Текстовое описание программы» содержит описание, показываемое пользователю в справке о программе. Компилятор получает значение этого поля из файла метаданных алгоритма из корневого параметра description. Поле «Объем памяти» содержит число, характеризующее необходимый для выделения объем памяти, измеряемый в ячейках памяти виртуальной машины. Значения полей «Кол-во входных данных» и «Кол-во выходных данных» компилятор рассчитывает автоматически на основе списков входных и выходных данных алгоритма из файла метаданных алгоритма. Поле «Количество команд» содержит общее число команд в программе.

После заголовка начинается раздел с метаинформацией о входных и выходных данных алгоритма. Каждое данное описывается с помощью структуры, схематически представленной на рис. 4. В файле сначала размещаются все метаданные о входных данных алгоритма, а затем все метаданные о выходных данных. Их количество известно из параметров «Кол-во входных данных» и «Кол-во выходных данных» заголовка. Поле «Обозначение (наимено-

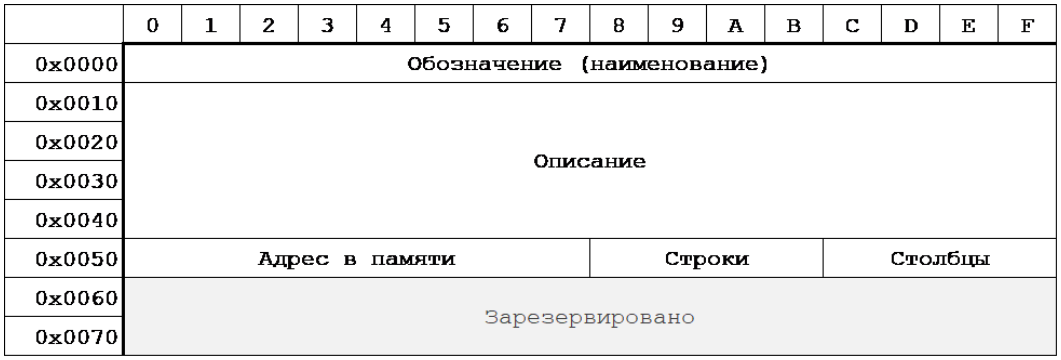



**Рис. 4.** Схема структуры метаданных о данных алгоритма в файле программы

вание)» содержит имя данного, использованное в представлении алгоритма в форме $Q$-детерминанта в виде строки. Содержимое поля «Описание» представляет текстовое описание данного, показываемое пользователю при выводе справочной информации о программе. Поле «Адрес в памяти» содержит адрес размещения данного. Если данное является матрицей, то это адрес первого элемента первой строки матрицы. Содержимое полей «Строки» и «Столбцы» является количеством строк и столбцов матрицы соответственно. Параметр «Адрес в памяти» компилятор рассчитывает автоматически, а значения остальных параметров получает из структур, описывающих эти данные в файле метаданных адгоритма.

Последним в файле программы идет раздел с очередью команд. Схема структуры команды показана на рис. 5. Число команд определяется параметром «Количество команд» из заголовка. Интерпретатору байт-кода отдаются две первые строки схемы.

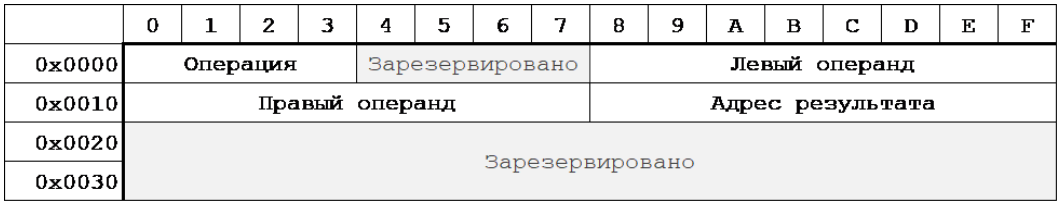



**Рис. 5.** Схема структуры команды в файле программы

В качестве формата файлов для входных и выходных данных алгоритма был выбран формат CSV с разделителем точка с запятой. Распределение памяти для данных происходит на этапе компиляции. Скалярные данные рассматриваются как квадратные матрицы порядка 1. Объем памяти, необходимый входному/выходному данному, рассчитывается как произведение количества строк на количество столбцов этого данного. В представлении алгоритма в форме $Q$-детерминанта, получаемом от $Q$-системы, индексация массивов и матриц имеет начальный индекс 1. Получение значения по индексу рассчитывается по формуле $address + columns \cdot (i-1) + j - 1$, где $address$ — адрес ячейки с индексом (1,1), $columns$ – количество столбцов матрицы, $i$ и $j$ — индексы, по которым происходит обращение. Если использован один индекс, то $j$ будет иметь значение 1. Если оба индекса упущены, то $i$ и $j$ будут иметь значение 1. Вначале распределяется память для входных данных, начиная с ячейки 1. Все матрицы «укладываются» в память построчно по порядку появления в файле метаданных программы. Затем резервируется память для выходных данных точно также, как и для входных. Далее распределяется память под все операции, кроме корневых. Корневыми считаются операции, значение которых будет являться значением $Q$-термов. Значения $Q$-термов — выходные данные программы, а память для них уже была выделена.

Опишем работу виртуальной машины. Ядром виртуальной машины являются три ее подсистемы: диспетчер задач, интерпретатор байт-кода программы, подсистема для реализация памяти виртуальной машины. Перед началом работы производятся подготовительные действия: в интерпретаторе байт-кода программы регистрируются все обработчики команд, выделяется память, создается и конфигурируется диспетчер задач. Обработка $Q$-эффективной программы начинается с создания рабочих потоков. Этим процессом занимается диспетчер задач. В каждом потоке создается объект интерпретатора байт-кода (далее интерпретатор). Количество потоков определяется, как количество вычислителей, доступных на ПВС, минус один или задается пользователем вручную. Один поток (основной) выделяется на работу самого диспетчера. После создания потоков начинается процесс создания для них заданий. Этим процессом также занимается диспетчер. Диспетчер получает команды с помощью передаваемого в него объекта, который представляет из себя реализацию шаблона проектирования итератор (Iterator). За счет этой абстракции команды могут быть получены из любого источника: из файла, из заранее заданного массива (актуально для тестов) или по сети. Максимальный размер задания для потока определяется как константа, заданная в коде виртуальной машины, или заданная пользователем. Как только задание создано, оно ставится в очередь на выполнение. Первый свободный поток берет задание из очереди и начинает его выполнение.

Задание — это очередь команд. Для обработки очередной команды интерпретатор выбирает необходимый обработчик команды и делегирует ему ответственность по вычислению. Если команда готова к выполнению, то она исполняется, а результат записывается в память по адресу, указанному в самой команде. В противном случае команда помещается в конец задания. Процесс обработки задания интерпретатором продолжается пока исходное задание не опустеет.

Репозиторий компилятора доступен по ссылке [6], а виртуальной машины по ссылке [7].

## 5. Экспериментальное исследование программной системы

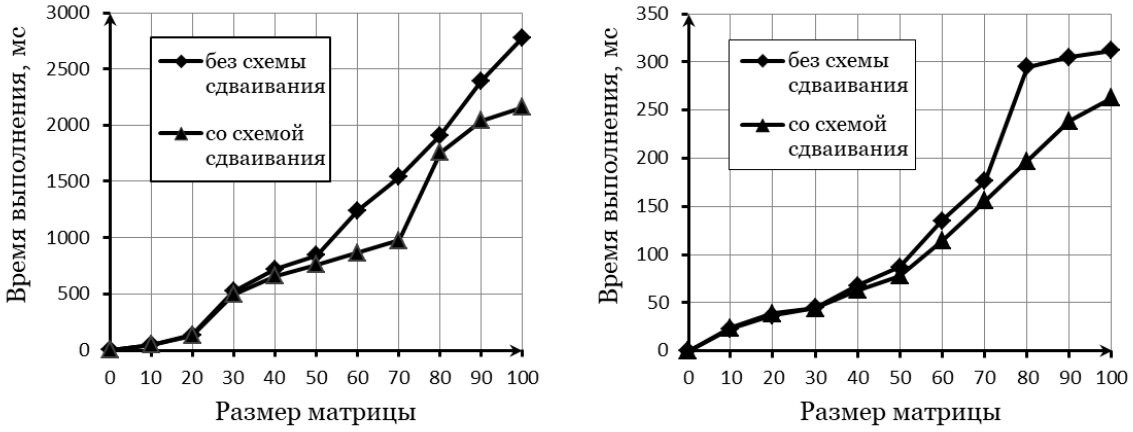
При тестировании программной системы на персональном компьютере было проведено функциональное тестирование с использованием алгоритмов с различными структурами $Q$-детерминантов, для которых с помощью $Q$-системы были получены представления в форме $Q$-детерминантов. Для функционального тестирования применялись следующие алгоритмы: скалярное произведение векторов без использования схемы сдваивания, скалярное произведение векторов с использованием схемы сдваивания, умножение матриц без использования схемы сдваивания, умножение матриц с использованием схемы сдваивания, поиск максимального элемента в массиве чисел, алгоритмы, реализующие методы Гаусса—Жордана, Гаусса—Зейделя и Якоби для решения СЛАУ.

При тестировании входные данные алгоритмов генерировались случайным образом, а выходные сравнивались с эталонными данными, полученными с помощью реализаций в среде MatLAB. Для всех алгоритмов все тесты успешно прошли этапы компиляции и исполнения виртуальной машиной, полученные значения выходных данных совпали с эталонными, при этом для итерационных алгоритмов они совпали с заданной точностью.

Экспериментальное исследование разработанной программной системы проводилось на суперкомпьютере «Торнадо ЮУрГУ». В экспериментах были задействованы два центральных процессора Intel Xeon X5680 с частотой 3.33 GHz одного вычислительного узла, каждый из которых имеет 6 ядер и поддерживает 12 потоков, оперативная память узла 24 Гб ECC DDR3 Full buffered [10].

Для экспериментального исследования использовались два алгоритма умножения двух квадратных матриц: алгоритм $\mathfrak{B}$ без использования схемы сдваивания и алгоритм $\mathfrak{C}$ с использованием схемы сдваивания. Результаты экспериментального исследования приведены на рис. 6. Графики слева показывают время выполнения $Q$-эффективных программ, выполняющих $Q$-эффективные реализации алгритмов $\mathfrak{B}$ и $\mathfrak{C}$, а графики справа время выполнения операций этих $Q$-эффективных реализаций. Таким образом, программная система выполняет $Q$-эффективную программу алгоритма $\mathfrak{C}$ быстрее, чем алгоритма $\mathfrak{B}$. Также для алгоритма $\mathfrak{C}$ на выполнение операций $Q$-эффективной реализации требуется меньше времени, чем для алгоритма $\mathfrak{B}$. Полученные результаты объясняются тем, что высота алгоритма $\mathfrak{C}$ меньше, чем алгоритма $\mathfrak{B}$. Исследование времени выполнения операций $Q$-эффективных реализаций проводилось с целью оценки его вклада в общее время выполнения $Q$-эффективных программ.

Нам не известны аналоги разработанной в данном исследовании программной системы, поэтому сравнение с аналогами не проводилось. Приведенные результаты экспериментов дают возможность читателю оценить быстродействие $Q$-эффективных программ. Следует иметь ввиду, что оно зависит от ресурса параллелизма алгоритмов и от условий разработки и выполнения программ, т.е. вычислительной инфраструктуры программ [3]. В статье [15]




а) $Q$-эффективная реализация

б) Операции $Q$-эффективной реализации

**Рис. 6.** Результаты экспериментального исследования

доказано, что при одной и той же вычислительной инфраструктуре $Q$-эффективная программа имеет динамические характеристики не хуже, чем любая программа, выполняющая другую реализацию того же алгоритма, т.е. Q-эффективная программа эффективна для своей вычислительной инфраструктуры. По нашему мнению, хотя быстродействие важно, ценность программных систем, подобных разработанной в данном исследовании, состоит в том, что они дают возможность не тратить время разработчика на программирование.

## Заключение

В статье представлено исследование по созданию на основе концепции $Q$-детерминанта программной системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов. Оно включает решение следующих задач.

1) Проектирование и реализация компилятора, преобразующего представление алгоритма в форме $Q$-детерминанта в исполняемую программу, использующую ресурс параллелизма алгоритма полностью.
2) Проектирование и реализация виртуальной машины, исполняющей программу, полученную с помощью компилятора.
3) Проведение экспериментального исследования функционирования программной системы, состоящей из компилятора и виртуальной машины, с применением суперкомпьютера «Торнадо ЮУрГУ».

Результаты тестирования разработанной программной системы на персональном компьютере и ее экспериментального исследования на суперкомпьютере «Торнадо ЮУрГУ» подтвердили адекватность и эффективность использованных решений.

Главным результатом исследования является то, что показана возможность создания единой для численных алгоритмов программной системы, которая по блок-схеме алгоритма проектирует программу, использующую ресурс параллелизма алгоритма полностью, и исполняет ее. Разработанная программная система предназначена для использования на ПВС с общей памятью. Целью дальнейших исследований является возможность использования распределенной памяти ПВС. Кроме того, планируется использование вариантов решений на основе концепции $Q$-детерминанта, отличных от описанного в данном исследовании.

## Литература


1. Алеева В.Н. Анализ параллельных численных алгоритмов. Препринт № 590. Новосибирск: ВЦ СО АН СССР, 1985. 23 с.
2. Алеева В.Н., Зотова П.С., Склезнев Д.С. Расширение возможностей исследования ресурса параллелизма численных алгоритмов с помощью программной $Q$-системы // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 66–81. DOI: 10.14529/cmse210205.
3. Алеева В.Н., Шатов М.Б. Применение концепции $Q$-детерминанта для эффективной реализации численных алгоритмов на примере метода сопряженных градиентов для решения систем линейных уравнений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 56–71. DOI: 10.14529/cmse210304.
4. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
5. Ершов Ю.Л., Палютин Е.А. Математическая логика. М.: Наука, 1987. 336 с.
6. Репозиторий компилятора программной системы. URL: `https://github.com/yuferovalex/qc` (дата обращения: 29.06.2023).
7. Репозиторий виртуальной машины программной системы. URL: `https://github.com/yuferovalex/qvm` (дата обращения: 29.06.2023).
8. Недожогин Н.С., Копысов С.П., Новиков А.К. Параллельное решение систем линейных уравнений на гибридной архитектуре CPU+GPU // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 2. С. 40–54. DOI: 10.14529/cmse200203.
9. Открытая энциклопедия свойств алгоритмов. URL: `https://algowiki-project.org/ru` (дата обращения: 30.05.2023).
10. Суперкомпьютер «Торнадо ЮУрГУ». URL: `http://supercomputer.susu.ru/computers/tornado/` (дата обращения: 30.05.2023).
11. Afanasyev I.V., Voevodin V.V., Komatsu K., *et al.* Distributed Graph Algorithms for Multiple Vector Engines of NEC SX-Aurora TSUBASA Systems // Supercomputing Frontiers and Innovations. 2021. Vol. 8, no. 2. P. 95–113. DOI: 10.14529/jsfi210206.
12. Akhmed-Zaki D., Lebedev D., Malyshkin V., *et al.* Automated Construction of High Performance Distributed Programs in LuNA System // Parallel Computing Technologies (PaCT 2019). Vol. 11657. Springer, 2019. P. 3–9. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-25636-4_1.
13. Aleeva V. Designing a Parallel Programs on the Base of the Conception of $Q$-Determinant // Supercomputing. RuSCDays 2018. Vol. 965. Springer, 2019. P. 565–577. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-05807-4_48.
14. Aleeva V.N. Improving Parallel Computing Efficiency // Proceedings – 2020 Global Smart Industry Conference, GloSIC 2020. IEEE, 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.
15. Aleeva V., Aleev R. Investigation and Implementation of Parallelism Resources of Numerical Algorithms // ACM Transactions on Parallel Computing. 2023. Vol. 10. no. 2. Article number 8. P. 1–64. DOI: 10.1145/3583755.

16. Aleeva V.N., Aleev R.Zh. High-Performance Computing Using Application of $Q$-determinant of Numerical Algorithms // Proceedings – 2018 Global Smart Industry Conference, GloSIC 2018. IEEE, 2018. 8 p. Article number 8570160. DOI: 10.1109/GloSIC.2018.8570160.

17. Aleeva V., Bogatyreva E., Skleznev A., *et al.* Software $Q$-system for the Research of the Resource of Numerical Algorithms Parallelism // Supercomputing. RuSCDays 2019. Vol. 1129. Springer, 2019. P. 641–652. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-36592-9_52.

18. Aleeva V.N., Sharabura I.S., Suleymanov D.E. Software System for Maximal Parallelization of Algorithms on the Base of the Conception of $Q$-determinant // Parallel Computing Technologies (PaCT 2015). Vol. 9251. Springer, 2015. P. 3–9. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-21909-7_1.

19. Antonov A.S., Dongarra J., Voevodin V.V. AlgoWiki Project as an Extension of the Top500 Methodology // Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 1. P. 4–10. DOI: 10.14529/jsfi180101.

20. Balaprakash P., Dongarra J., Gamblin T., *et al.* Autotuning in High-Performance Computing Applications // Proceedings of the IEEE. 2018. Vol. 106, no. 11. P. 2068–2083. DOI: 10.1109/JPROC.2018.2841200.

21. Bauer M., Treichler S., Slaughter E., Aiken A. Legion: Expressing locality and independence with logical regions // SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 2012. P. 1–11. DOI: 10.1109/SC.2012.71.

22. Bosilca G., Bouteiller A., Danalis A., *et al.* Flexible Development of Dense Linear Algebra Algorithms on Massively Parallel Architectures with DPLASMA // 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum. 2011. P. 1432–1441. DOI: 10.1109/IPDPS.2011.299.

23. Bosilca G., Bouteiller A., Danalis A., *et al.* PaRSEC: Exploiting Heterogeneity to Enhance Scalability // Computing in Science & Engineering. 2013. Vol. 15, no. 6. P. 36–45. DOI: 10.1109/MCSE.2013.98.

24. Legalov A.I., Vasilyev V.S., Matkovskii I.V., *et al.* A Toolkit for the Development of Data-Driven Functional Parallel Programmes // Parallel Computational Technologies (PCT'2018). Vol. 910. Springer, 2018. P. 16–30. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-99673-8_2.

25. Moore G. Cramming More Components onto Integrated Circuits // Electronics Magazine. 1965. Vol. 38, no. 8. P. 114–117.

26. Moskovsky A., Roganov V., Abramov S. Parallelism Granules Aggregation with the T-System // Parallel Computing Technologies (PaCT 2007). Vol. 4671. Springer, 2007. P. 293–302. Lecture Notes in Computer Science. DOI: 10.1007/978-3-540-73940-1_30.

Алеева Валентина Николаевна, к.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse230303

# COMPUTER-AIDED DESIGN AND EXECUTION OF EFFECTIVE PROGRAMS FOR NUMERICAL ALGORITHMS

© 2023 V.N. Aleeva

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

*E-mail: aleevavn@susu.ru*

Received: 29.06.2023

Designing effective parallel programs for multiprocessor architectures is difficult because there are no clear formal rules to follow. The concept of the $Q$-determinant can be applied to solve this problem when implementing numerical algorithms. This theory allows for automated analysis of the algorithm parallelism resource, automated comparison of the parallelism resources of algorithms solving the same algorithmic problem. In addition, it makes it possible to design effective programs for the implementation of numerical algorithms using a specially developed design method, improve the efficiency of the implementation of numerical methods and algorithmic problems. The results obtained on the basis of the $Q$-determinant concept are one of the options for solving the problem of effective implementation of numerical algorithms, methods and algorithmic problems on parallel computing systems. However, the fundamental problem of computer-aided design and execution for any numerical algorithm of a program that implements the algorithm effectively remains unresolved. The paper describes the development of a software system for designing and executing $Q$-effective programs that is unified for numerical algorithms. A $Q$-effective program is an effective program designed using the concept of a $Q$-determinant. The system is intended for use on parallel computing systems with shared memory. It consists of a compiler and a virtual machine. The compiler converts the representation of the algorithm in the form of a $Q$-determinant into an executable program that uses the algorithm's parallelism resource completely. The virtual machine executes the program generated by the compiler. The paper also provides an experimental study of the created software system using the SUSU Tornado supercomputer.

*Keywords: Q-determinant of algorithm, representation of algorithm in form of Q-determinant, Q-effective implementation of algorithm, parallelism resource of algorithm, software Q-system, parallel computing system, parallel program, Q-effective program.*

**FOR CITATION**

Aleeva V.N. Computer-aided Design and Execution of Effective Programs for Numerical Algorithms. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 3. P. 31–49. (in Russian) DOI: 10.14529/cmse230303.



## References

1. Aleeva V.N. Analysis of Parallel Numerical Algorithms. Preprint no. 590. Novosibirsk, Computing Center of the Siberian Branch of the Academy of Sciences of the USSR, 1985. 23 p. (in Russian)
2. Aleeva V.N., Zotova P.S., Skleznev D.S. Advancement of Research for the Parallelism Resource of Numerical Algorithms with the Help of Software $Q$-system. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 2. P. 66–81. (in Russian) DOI: 10.14529/cmse210205.
3. Aleeva V.N., Shatov M.B. Application of the $Q$-determinant Concept for Efficient

Implementation of Numerical Algorithms by the Example of the Conjugate Gradient Method for Solving Systems of Linear Equations. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 3. P. 56–71. (in Russian) DOI: 10.14529/cmse210304.

4. Voevodin V.V., Voevodin Vl.V. Parallel Computing. St.Petersburg, BHV-Petersburg, 2002. 608 p. (in Russian)
5. Ershov Yu.L., Palyutin E.A. Mathematical Logic. Moscow, Mir, 1984. 303 p.
6. Software system compiler repository. URL: `https://github.com/yuferovalex/qc` (accessed: 29.06.2023).
7. Software system virtual machine repository. URL: `https://github.com/yuferovalex/qvm` (accessed: 29.06.2023).
8. Nedozhogin N.S., Kopysov S.P., Novikov A.K. Parallel Solving of Linear Equations Systems on Hybrid Architecture CPU+GPU. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2020. Vol. 9, no. 2. P. 40–54. (in Russian) DOI: 10.14529/cmse200203.
9. Open Encyclopedia of Parallel Algorithmic Features. URL: `https://algowiki-project.org/en` (accessed: 30.05.2023).
10. "Tornado SUSU" Supercomputer. URL: `http://supercomputer.susu.ru/en/computers/tornado/` (accessed: 30.05.2023).
11. Afanasyev I.V., Voevodin V.V., Komatsu K., *et al.* Distributed Graph Algorithms for Multiple Vector Engines of NEC SX-Aurora TSUBASA Systems. Supercomputing Frontiers and Innovations. 2021. Vol. 8, no. 2. P. 95–113. DOI: 10.14529/jsfi210206.
12. Akhmed-Zaki D., Lebedev D., Malyshkin V., *et al.* Automated Construction of High Performance Distributed Programs in LuNA System. Parallel Computing Technologies (PaCT 2019). Vol. 11657. Springer, 2019. P. 3–9. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-25636-4_1.
13. Aleeva V. Designing a Parallel Programs on the Base of the Conception of $Q$-Determinant. Supercomputing. RuSCDays 2018. Vol. 965. Springer, 2019. P. 565–577. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-05807-4_48.
14. Aleeva V.N. Improving Parallel Computing Efficiency. Proceedings – 2020 Global Smart Industry Conference, GloSIC 2020. IEEE, 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.
15. Aleeva V., Aleev R. Investigation and Implementation of Parallelism Resources of Numerical Algorithms. ACM Transactions on Parallel Computing. 2023. Vol. 10. no. 2. Article number 8. P. 1–64. DOI: 10.1145/3583755.
16. Aleeva V.N., Aleev R.Zh. High-Performance Computing Using Application of $Q$-determinant of Numerical Algorithms. Proceedings – 2018 Global Smart Industry Conference, GloSIC 2018. IEEE, 2018. 8 p. Article number 8570160. DOI: 10.1109/GloSIC.2018.8570160.
17. Aleeva V., Bogatyreva E., Skleznev A., *et al.* Software $Q$-system for the Research of the Resource of Numerical Algorithms Parallelism. Supercomputing. RuSCDays 2019. Vol. 1129. Springer, 2019. P. 641–652. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-36592-9_52.

18. Aleeva V.N., Sharabura I.S., Suleymanov D.E. Software System for Maximal Parallelization of Algorithms on the Base of the Conception of $Q$-determinant. Parallel Computing Technologies (PaCT 2015). Vol. 9251. Springer, 2015. P. 3–9. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-21909-7_1.

19. Antonov A.S., Dongarra J., Voevodin V.V. AlgoWiki Project as an Extension of the Top500 Methodology. Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 1. P. 4–10. DOI: 10.14529/jsfi180101.

20. Balaprakash P., Dongarra J., Gamblin T., *et al.* Autotuning in High-Performance Computing Applications. Proceedings of the IEEE. 2018. Vol. 106, no. 11. P. 2068–2083. DOI: 10.1109/JPROC.2018.2841200.

21. Bauer M., Treichler S., Slaughter E., Aiken A. Legion: Expressing locality and independence with logical regions. SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 2012. P. 1–11. DOI: 10.1109/SC.2012.71.

22. Bosilca G., Bouteiller A., Danalis A., *et al.* Flexible Development of Dense Linear Algebra Algorithms on Massively Parallel Architectures with DPLASMA. 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum. 2011. P. 1432–1441. DOI: 10.1109/IPDPS.2011.299.

23. Bosilca G., Bouteiller A., Danalis A., *et al.* PaRSEC: Exploiting Heterogeneity to Enhance Scalability. Computing in Science & Engineering. 2013. Vol. 15, no. 6. P. 36–45. DOI: 10.1109/MCSE.2013.98.

24. Legalov A.I., Vasilyev V.S., Matkovskii I.V., *et al.* A Toolkit for the Development of Data-Driven Functional Parallel Programmes. Parallel Computational Technologies (PCT'2018). Vol. 910. Springer, 2018. P. 16–30. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-99673-8_2.

25. Moore G. Cramming More Components onto Integrated Circuits. Electronics Magazine. 1965. Vol. 38, no. 8. P. 114–117.

26. Moskovsky A., Roganov V., Abramov S. Parallelism Granules Aggregation with the T-System. Parallel Computing Technologies (PaCT 2007). Vol. 4671. Springer, 2007. P. 293–302. Lecture Notes in Computer Science. DOI: 10.1007/978-3-540-73940-1_30.

УДК 004.272.25, 004.421, 004.032.24 DOI: 10.14529/cmse230304

# ОБНАРУЖЕНИЕ АНОМАЛИЙ ВРЕМЕННОГО РЯДА НА ОСНОВЕ ТЕХНОЛОГИЙ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ И НЕЙРОННЫХ СЕТЕЙ

**© 2023 Я.А. Краева**

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

Поступила в редакцию: 20.05.2023

В статье рассмотрена задача поиска аномальных подпоследовательностей временного ряда, решение которой в настоящее время востребовано в широком спектре предметных областей. Предложен новый метод обнаружения аномальных подпоследовательностей временного ряда с частичным привлечением учителя. Метод базируется на концепциях диссонанса и сниппета, которые формализуют соответственно понятия аномальных и типичных подпоследовательностей временного ряда. Предложенный метод включает в себя нейросетевую модель, которая определяет степень аномальности входной подпоследовательности ряда, и алгоритм автоматизированного построения обучающей выборки для этой модели. Нейросетевая модель представляет собой сиамскую нейронную сеть, где в качестве подсети предложено использовать модификацию модели ResNet. Для обучения модели предложена модифицированная функция контрастных потерь. Формирование обучающей выборки выполняется на основе репрезентативного фрагмента ряда, из которого удаляются диссонансы, маломощные сниппеты со своими ближайшими соседями и выбросы в рамках каждого сниппета, трактуемые соответственно как аномальная, нетипичная деятельность субъекта и шумы. Вычислительные эксперименты на временных рядах из различных предметных областей показывают, что предложенная модель по сравнению с аналогами показывает в среднем наиболее высокую точность обнаружения аномалий по стандартной метрике VUS-PR. Обратной стороной высокой точности метода является большее по сравнению с аналогами время, которое затрачивается на обучение модели и распознавание аномалии. Тем не менее, в приложениях интеллектуального управления отоплением зданий метод обеспечивает быстродействие, достаточное для обнаружения аномальных подпоследовательностей в режиме реального времени.

*Ключевые слова: временной ряд, поиск аномалий, диссонанс, сниппет, сиамская нейронная сеть.*

## ОБРАЗЕЦ ЦИТИРОВАНИЯ

Краева Я.А. Обнаружение аномалий временного ряда на основе технологий интеллектуального анализа данных и нейронных сетей // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 3. С. 50–71. DOI: 10.14529/cmse230304.

## Введение

В настоящее время разработка эффективных моделей, методов и алгоритмов поиска аномалий временного ряда остается одной из наиболее актуальных исследовательских проблем [1]. Поиск аномалий требуется в широком спектре предметных областей, связанных с обработкой временных рядов: Интернет вещей [2], умное управление зданиями [3] и городом [4], персональная медицина [5] и др. При этом целью поиска может быть точечная аномалия (одиночный выброс) или аномальная подпоследовательность (непрерывный интервал элементов ряда, не все из которых являются выбросами). Случай аномальных подпоследовательностей является наиболее востребованным на практике и сложным для поиска, поскольку среди прочих параметров требует учета всевозможных длин аномалии [1].

Методы поиска аномальных подпоследовательностей временного ряда разделяют на три группы [6, 7]: поиск с учителем (supervised), поиск без учителя (unsupervised) и поиск с частичным привлечением учителя (semi-supervised).

Методы поиска аномалий *с учителем* моделируют нормальное и девиантное поведение во временном ряде и включают в себя этап обучения, после которого возможно их применение во временных рядах, не известных модели. Методы данной группы требуют для обучения предварительно размеченный экспертом или специализированной программой временной ряд (ряды), где подпоследовательности имеют одну из двух меток: «норма» или «аномалия». По своей природе методы поиска аномалий с учителем ограничены в своей способности обнаруживать аномалии, не задействованные на этапе обучения, и поэтому в настоящее время они редко применяются на практике [6].

Методы поиска аномалий *без учителя* не требуют предварительных знаний о временном ряде и не включают в себя этап обучения. Указанные методы основываются на предположениях о свойствах, которыми обладают аномальные подпоследовательности: они встречаются реже, имеют иную форму, происходят из другого распределения вероятностей и др.

Методы *с частичным привлечением учителя* включают в себя этап обучения, на котором пытаются изучить только нормальное поведение временного ряда, который фигурирует в качестве обучающей выборки модели. Модель, будучи примененной к тестовому временному ряду, помечает как аномальные подпоследовательности, которые не соответствуют нормальному поведению.

В данной статье предлагается новый метод обнаружения аномальных подпоследовательностей временного ряда с частичным привлечением учителя. Метод включает в себя нейросетевую модель, которая определяет степень аномальности входной подпоследовательности, и алгоритм автоматизированного построения обучающей выборки для этой модели. Нейросетевая модель представляет собой сиамскую нейронную сеть (Siamese Neural Network) [8], где в качестве подсети фигурирует модификация нейросетевой модели ResNet [9], и для обучения которой предлагается модифицированная функция потерь. Алгоритм построения обучающей выборки для нейросетевой модели предполагает очистку репрезентативного ряда от аномальных подпоследовательностей, отражающих аномальные и нетипичные активности субъекта, для поиска которых применяются понятия диссонанса [10] и сниппета [11] соответственно.

Остаток текста статьи организован следующим образом. Раздел 1 содержит краткий обзор работ по тематике исследования. В разделе 2 приводятся формальные определения базовых понятий. В разделе 3 представлен метод обнаружения аномалий временного ряда в реальном времени, основанный на совместном применении технологий нейронных сетей и интеллектуального анализа данных. Раздел 4 описывает результаты вычислительных экспериментов по исследованию эффективности предложенного метода. Заключение подводит итоги исследования.

## 1. Обзор работ

В недавно опубликованных обзорных статьях о методах поиска аномалий временного ряда [6, 12] суммарно рассматривается около ста различных методов, в том числе более 25 методов с частичным привлечением учителя. Поэтому в данном разделе кратко рассмотрено лишь малое число основных подходов к поиску аномалий, некоторые из которых далее были задействованы в вычислительных экспериментах данного исследования.

В группу методов поиска аномалий без учителя входят поиск диссонансов [13] на основе матричного профиля временного ряда [14], алгоритмы DRAG [10], MERLIN [15], DAMP [16], метод NormA [17, 18] и др.

Как указывалось выше, методы поиска аномалий с учителем редко применяются на практике [6], и поэтому можно привести лишь три относительно недавние разработки, входящие в данную группу методов: MultiHMM [19], HIF [20] и NF [21].

Типичными представителями группы методов поиска аномалий с частичным привлечением учителя являются следующие разработки. Метод LSTM-AD [22] выполняет обнаружение аномалий в многомерных временных рядах на основе применения нейронных сетей долгой краткосрочной памяти (LSTM, Long Short-Term Memory). Метод предполагает, что размеченные данные распределяются на следующие группы. Нормальные подпоследовательности делятся на четыре группы: обучающая выборка ($s_N$), две валидационные выборки ($v_{N1}$ и $v_{N2}$) и тестовая выборка ($t_N$). Аномальные подпоследовательности делятся на две группы: валидационная ($v_A$) и тестовая выборки ($t_A$). Метод использует двухступенчатую схему «предсказание-детекция»: сперва модель на основе многослойной сети LSTM предсказывает значения временного ряда, а затем вычисляется распределение ошибок предсказания, с помощью которого обнаруживаются аномалии. Многослойная сеть LSTM организуется следующим образом. Во входном слое для каждой из $m$ размерностей ряда имеется один нейрон, $d \times \ell$ нейронов в выходном слое таких, что на каждое из $\ell$ предсказанных значений для каждой из $d$ размерностей имеется один нейрон (где $d$, $\ell$ — параметры и $1 \leqslant d \leqslant m$). Нейроны скрытого слоя LSTM являются полносвязными, что реализовано с помощью рекуррентных связей. Несколько LSTM слоев (обычно два) объединяются в стек таким образом, чтобы каждый нейрон в скрытом слое LSTM снизу был полностью соединен с каждым нейроном в скрытом слое LSTM над ним посредством прямых соединений. Обучение описанной модели выполняется на выборке $s_N$, выборка $v_{N1}$ используется для раннего останова обучения при подборе весов нейросети. Фаза детекции выполняется следующим образом. Для каждой из выбранных $d$ размерностей $\ell$ раз выполняется предсказание $\ell$ значений. Далее применяется вектор ошибок, элемент которого представляет собой разность между реальным и предсказанным значениями. Модель, обученная на выборке $s_N$, используется для вычисления векторов ошибок для последовательностей валидационной и тестовой выборок. Векторы ошибок моделируются таким образом. Векторы ошибок для элементов из выборки $v_{N1}$ используются для оценки параметров распределения с использованием оценки максимального правдоподобия. Подпоследовательность классифицируется как аномалия, если функция оценка максимального правдоподобия меньше наперед заданного параметра $\tau$, иначе она помечается как «норма». При этом выборки $v_{N2}$ и $v_A$ применяются для определения $\tau$ посредством максимизации значения $F$-меры, когда аномальные подпоследовательности считаются принадлежащими положительному классу, а нормальные — напротив, отрицательному.

Метод DeepAnT [23] позволяет обнаруживать одиночные выбросы и аномальные подпоследовательности временного ряда как в онлайн, так и в офлайн режиме. DeepAnT использует не содержащие разметку временные ряды для изучения распределения данных, которое затем используется для прогнозирования нормального поведения временного ряда. DeepAnT состоит из двух модулей: предсказателя и детектора аномалий временного ряда. Модуль предсказания использует глубокую сверточную нейронную сеть для прогнозирования будущего значения ряда на определенном горизонте, используя в качестве контекста

окно предыдущих значений ряда. Нейронная сеть включает в себя два сверточных слоя с размером ядра 32 и Линейным выпрямителем (ReLU, Rectified linear unit) в качестве функции активации. К выходу каждого из слоев применяется операция подвыборки по максимальному значению (MaxPooling). Последним слоем нейросети является полносвязный. В качестве функции потерь пр обучении нейросети применяется средняя абсолютная ошибка (MAE, Mean Absolute Error). Полученное прогнозное значение затем передается детектору аномалий, который отвечает за разметку значения как нормального или аномального. DeepAnT допускает обучение на временных рядах, из которых не удаляются выбросы и аномальные подпоследовательности.

## 2. Теоретический базис

### 2.1. Временной ряд и подпоследовательность

*Временной ряд* $T$ представляет собой последовательность вещественных значений, взятых в хронологическом порядке:

$$T = \{t_i\}_{i=1}^{n}, \quad t_i \in \mathbb{R}. \tag{1}$$

Число $n$ называется длиной ряда и обозначается $|T|$.

*Подпоследовательность* $T_{i,\,m}$ временного ряда $T$ представляет собой непрерывный промежуток из $m$ элементов ряда, начиная с позиции $i$:

$$T_{i,m} = \{t_k\}_{k=i}^{i+m-1}, \quad 3 \leqslant m \ll n, \quad 1 \leqslant i \leqslant n - m + 1. \tag{2}$$

Множество всех подпоследовательностей ряда $T$, имеющих длину $m$, обозначим как $S_T^m$.

### 2.2. Диссонансы

Подпоследовательности $T_{i,\,m}$ и $T_{j,\,m}$ ряда $T$ считаются *не пересекающимися*, если $|i - j| \geqslant m$. Некая подпоследовательность ряда, не пересекающаяся с данной подпоследовательностью $C$, обозначается как $M_C$.

Подпоследовательность $D$ ряда $T$ является *диссонансом* [10], если

$$\min_{M_D \in T} \big(\mathrm{Dist}(D, M_D)\big) \geqslant r, \tag{3}$$

где $\mathrm{Dist}(\cdot\,,\cdot\,)$ — неотрицательная симметричная функция расстояния, $r$ — порог расстояния (параметр). Иными словами, некая подпоследовательность ряда является диссонансом, если ее ближайший сосед (ближайшая и не пересекающаяся с ней подпоследовательность) находится на расстоянии не менее чем $r$. Для поиска диссонансов в качестве функции расстояния могут быть выбраны евклидова метрика [10], квадрат z-нормализованного евклидова расстояния [24] и др.

### 2.3. Сниппеты

*Сниппеты* [11] временного ряда представляют собой подпоследовательности, выражающие типичные активности некоего субъекта, деятельность которого описывает данный ряд. Формальное определение сниппетов выглядит следующим образом.

Пусть имеется временной ряд $T$ и задана длина подпоследовательности $m$ ($m \ll n$). Разобьем ряд на не пересекающиеся *сегменты* длины $m$, без ограничения общности считая,

что $n$ кратно $m$. Рассмотрим множество сегментов $Seg_T^m$:

$$Seg_T^m = \{Seg_i\}_{i=1}^{n/m}, \quad Seg_i = T_{m\cdot(i-1)+1,\,m}. \tag{4}$$

Сниппеты представляют собой непустое подмножество $Seg_T^m$ из $K$ сегментов, где $K$ $(1 \leqslant K \leqslant n/m)$ — параметр, отражающий количество активностей субъекта, интересующее исследователя. Обозначим множество сниппетов ряда $T$, имеющих длину $m$, как $C_T^m$:

$$C_T^m = \{C_i\}_{i=1}^{K}, \quad C_i \in Seg_T^m. \tag{5}$$

С каждым сниппетом ассоциированы следующие атрибуты: индекс, профиль, ближайшие соседи и мощность (значимость) данного сниппета. *Индекс сниппета* $C_i \in C_T^m$ обозначается как $C_i.index$ и представляет собой номер $j$ сегмента $Seg_j$, которому соответствует подпоследовательность ряда $T_{m\cdot(j-1)+1,\,m}$.

*Профиль сниппета*, обозначаемый как $C_i.profile$, представляет собой вектор MPdist-расстояний [25] между данным сниппетом и подпоследовательностями ряда:

$$C_i.profile = \{d_k\}_{k=1}^{n-m+1}, \quad d_k = \mathrm{MPdist}(C_i,\, T_{i,\,m}). \tag{6}$$

*Множество ближайших соседей сниппета* $C_i \in C_T^m$ обозначается как $C_i.NN$ и содержит подпоследовательности ряда, которые более близки данному сниппету, чем другим сегментам ряда, в смысле расстояния MPdist:

$$C_i.NN = \{T_{j,\,m} \mid Seg_{C_i.index} = \arg\min_{1\leqslant q\leqslant n/m} \mathrm{MPdist}(T_{j,\,m},\, Seg_q),\ 1 \leqslant j \leqslant n-m+1\}. \tag{7}$$

*Мощность сниппета* $C_i \in C_T^m$ обозначается как $C_i.frac$ и вычисляется как доля мощности множества ближайших соседей сниппета от общего количества подпоследовательностей ряда, имеющих длину $m$, при этом сниппеты упорядочиваются по убыванию их мощности:

$$C_i.frac = \frac{|\,C_i.NN|}{n-m+1}. \tag{8}$$

$$\forall C_i, C_j \in C_T^m :\ i < j \iff C_i.frac \geqslant C_j.frac. \tag{9}$$

Расстояние $\mathrm{MPdist}(\cdot\,,\cdot\,)$ между подпоследовательностями $A$ и $B$ $(|A| = |B| = m)$ определяется следующим образом [25]. Фиксируем параметр $\ell$ $(\lceil 0.3m\rceil \leqslant \ell \leqslant \lceil 0.8m\rceil)$, который отражает длину семантически значимого непрерывного промежутка точек подпоследовательности. Вычисление MPdist предполагает последовательное выполнение следующих операций: 1) вычисление матричных профилей $A$ и $B$, взятых в указанном и обратном порядке; 2) конкатенация вычисленных профилей; 3) упорядочение элементов полученного временного ряда по возрастанию; 4) взятие в качестве ответа $k$-го элемента результирующего ряда. Формальная запись выглядит следующим образом:

$$\mathrm{MPdist}_\ell(A,B) = \mathrm{AscSort}\big(P_{ABBA}\big)(k), \quad P_{ABBA} = P_{AB} \bullet P_{BA}, \tag{10}$$

где $\mathrm{AscSort}(\cdot\,)$ — операция упорядочивания элементов последовательности по возрастанию, символ $\bullet$ обозначает операцию конкатенации, $k$ $(0 < k < m)$ — задаваемый аналитиком параметр (типичное значение $k = \lceil 0.1m\rceil$).

*Матричным профилем* [26] рядов $A$ и $B$ для длины называется ряд $P_{AB}$, $i$-м элементом которого является расстояние между $i$-й подпоследовательностью ряда $A$, имеющей длину $\ell$, и ее ближайшим соседом в ряде $B$:

$$P_{AB} = \{\mathrm{ED}^2_{\mathrm{norm}}(A_{i,\ell}, B_{j,\ell})\}_{i=1}^{m-\ell+1}, \quad B_{j,\ell} = \arg \min_{1 \leqslant q \leqslant m-\ell+1} \mathrm{ED}^2_{\mathrm{norm}}(A_{i,\ell},\, B_{q,\ell}), \tag{11}$$

где функция $\mathrm{ED}^2_{\mathrm{norm}}(\cdot,\cdot)$ означает квадрат евклидова расстояния между z-нормализованными подпоследовательностями. Аналогичным образом определяется матричный профиль рассматриваемых рядов, взятых в порядке $B$ и $A$, и обозначается как $P_{BA}$.

# 3. Метод обнаружения аномалий во временных рядах

В данном разделе представлен новый метод обнаружения аномалий во временном ряде в режиме реального времени, получивший название DiSSiD (Discord, Snippet, and Siamese Neural Networket-based Detector of anomalies). Метод включает в себя следующие компоненты: нейросетевая модель, построенная на основе сиамской нейронной сети, и алгоритм подготовки обучающей выборки для указанной модели, — описанные ниже в разделах 3.1 и 3.2 соответственно.

## 3.1. Нейросетевая модель

### *3.1.1. Архитектура модели*

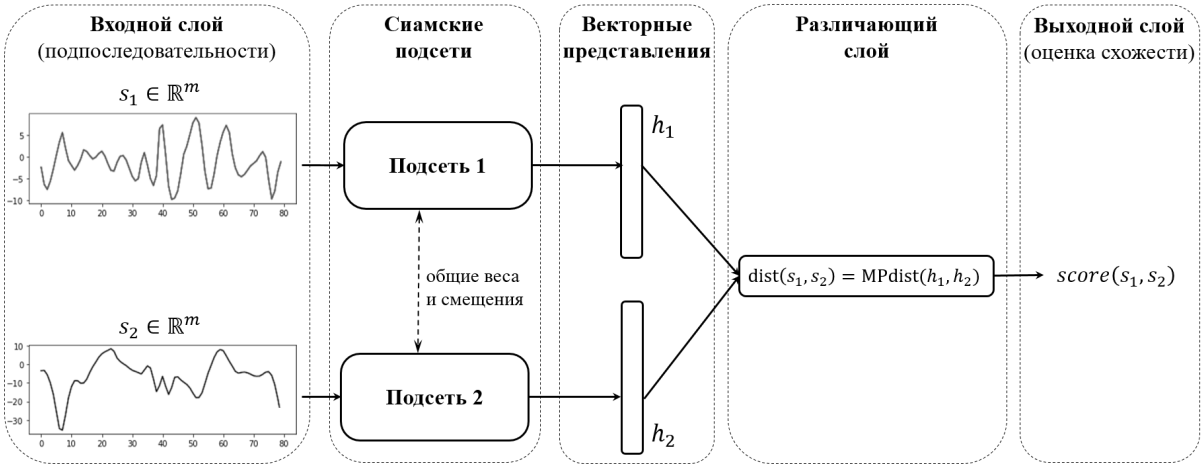



**Рис. 1.** Нейросетевая модель DiSSiD

Разработанная нейросетевая модель представлена на рис. 1. DiSSiD представляет собой сиамскую нейронную сеть (Siamese Neural Network, SNN) [8]. SNN объединяет в себе две подсети, которые имеют одинаковые архитектуру, конфигурацию (количество слоев, число нейронов в каждом слое, размерность входного и выходного слоев, функции активации и др.), а также наборы весов и смещений, полученных в результате обучения. Каждая из указанных подсетей формирует векторное представление (embedding) поданной на вход подпоследовательности, а на выходе модель выдает MPdist-расстояние между сформированными векторными представлениями. В качестве подсети фигурирует модификация нейросетевой модели ResNet [9]. Архитектура SNN по сравнению с традиционными нейросетевыми моделями лучше приспособлена к обучению в случае дисбаланса классов и позволяет добавить новый класс в уже развернутую модель без ее повторного обучения [8]. Архитек-

тура ResNet в настоящее время является одним из наиболее эффективных средств решения проблемы затухания градиента (vanishing gradient) при обучении нейросетевой модели [9].

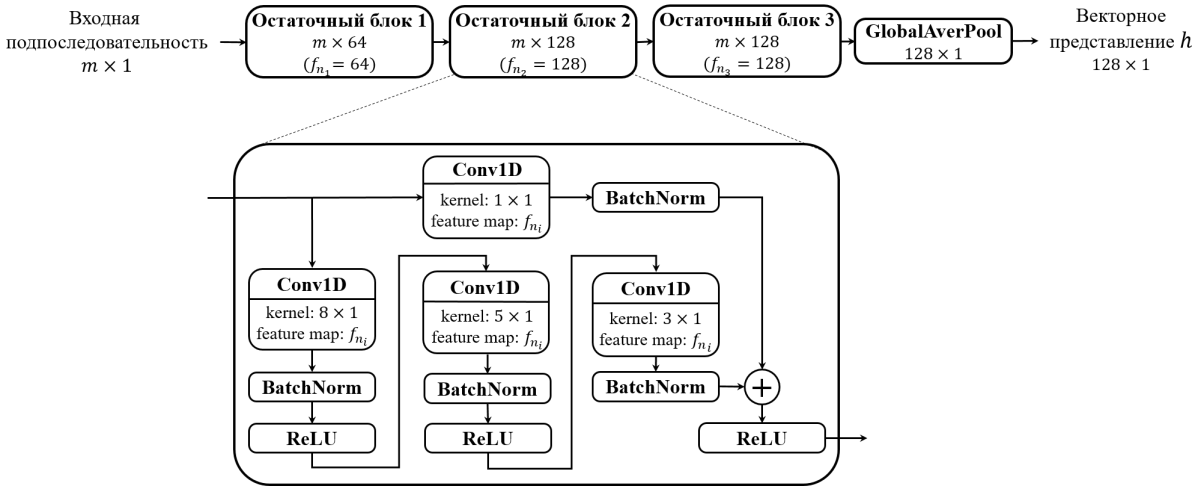



**Рис. 2.** Архитектура подсети ResNet

Подсеть на основе ResNet имеет следующую архитектуру (см. рис. 2). На входной слой поступает подпоследовательность временного ряда, имеющая длину $m$. Далее подсеть включает в себя три одинаковых остаточных блока (residual block) и за ними слой глобальной усредняющей агрегации (GlobalAveragePooling), формирующий векторное представление. Размерность итогового векторного представления определяется количеством карт признаков последнего слоя в последнем остаточном блоке.

Каждый остаточный блок включает в себя три сверточных слоя, на которых применяются фильтры (ядра) с размерами $8 \times 1$, $5 \times 1$ и $3 \times 1$ соответственно. Каждый сверточный слой чередуется со слоем пакетной нормализации (batch normalization) [27], к которому применяется функция активации Линейный выпрямитель (ReLU, Rectified linear unit). Пакетная нормализация преобразует набор входных данных таким образом, что его математическое ожидание обращается в ноль, а дисперсия — в единицу, и предназначена для ускорения сходимости обучения.

После прохождения трех сверточных слоев остаточный блок выдает карты признаков (feature maps): первый блок — 64 карты, остальные два блока — по 128 карт. Далее выполняется сложение входа остаточного блока, пропущенного через сверточный слой с ядром размера $1 \times 1$, с выходом этого остаточного блока. Однако входы не могут добавляться напрямую к выходам остаточного блока, поскольку они не имеют одинаковых размеров. Данный прием применяется как средство преодоления проблемы затухающего градиента (vanishing gradient) [28] между внутренними слоями нейронной сети.

### *3.1.2. Обучение модели*

Для обучения модели DiSSiD из заданного временного ряда формируется обучающая выборка, определяемая следующим образом:

$$\begin{aligned} \mathcal{L} =& \mathcal{L}_{\text{true}} \cup \mathcal{L}_{\text{false}} \setminus \mathcal{L}_{\text{anomaly}} \\ \mathcal{L}_{\text{true}} =& \{< s_1; s_2; 1 > |\ \ s_1, s_2 \in C_i.NN, \quad 1 \leqslant i \leqslant K\} \\ \mathcal{L}_{\text{false}} =& \{< s_1; s_2; 0 > |\ \ s_1 \in C_i.NN,\, s_2 \in C_j.NN, \quad i \neq j,\, 1 \leqslant i, j \leqslant K\}, \end{aligned} \tag{12}$$

где множество $\mathcal{L}_{\text{true}}$ включает в себя пары подпоследовательностей, входящих в множество ближайших соседей одного и того же сниппета, в множество $\mathcal{L}_{\text{false}}$ — пары подпоследовательностей из множеств ближайших соседей разных сниппетов, а множество $\mathcal{L}_{\text{anomaly}}$ содержит аномальные подпоследовательности, не включаемые в обучающую выборку. Алгоритм очистки исходного временного ряда от аномальных подпоследовательностей и формирования обучающей выборки рассмотрен далее в разделе 3.2.

Для обучения модели DiSSiD предлагается следующая модифицированная функция контрастных потерь (contrastive loss) [29]:

$$L(s_1, s_2, \delta_{s_1 s_2}) = \delta_{s_1 s_2} \cdot \text{MPdist}(h_1, h_2) + (1 - \delta_{s_1 s_2})\big(\max(\tau - \text{MPdist}(h_1, h_2), 0)\big)^2, \qquad (13)$$

где $s_1$ и $s_2$, $h_1$ и $h_2$ — исходные подпоследовательности и их векторные представления соответственно; кронекериан $\delta_{s_1 s_2}$ принимает значение 1, если исходные подпоследовательности являются ближайшими соседями одного и того же сниппета, и 0 в противном случае; $\tau$ — минимальное расстояние MPdist между векторными представлениями исходных подпоследовательностей, являющихся ближайшими соседями разных сниппетов (параметр модели). Указанная функция потерь обеспечивает обучение модели, в результате которого похожие подпоследовательности исходного ряда получат векторные представления, отстоящие друг от друга в смысле расстояния MPdist не более чем на $\tau$, а непохожие — более чем на $\tau$ соответственно.

Перед обучением элементы множества $\mathcal{L}$ случайным образом разделяются на два не пересекающихся подмножества: обучающую и валидационную выборки $\mathcal{L}_{\text{train}}$ и $\mathcal{L}_{\text{valid}}$, используемые для обучения модели и настройки ее гиперпараметров соответственно. Мощности указанных выборок находятся в традиционном соотношении 80% и 20% соответственно.

### *3.1.3. Применение модели*

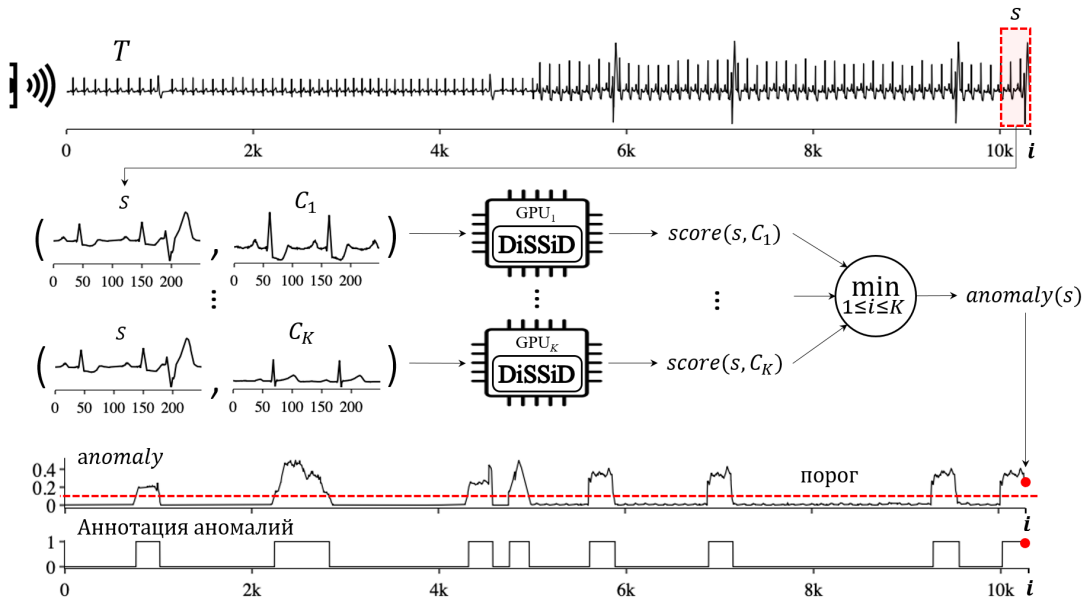



**Рис. 3.** Применение модели DiSSiD

Пусть имеется обученная модель DiSSiD, обучающая выборка которой содержит набор сниппетов $\{C_i\}_{i=1}^{K}$ и с помощью модели требуется определить, является ли входная подпо-

следовательность $s$ аномальной. Предполагается, что $K$ экземпляров модели запускаются каждый на отдельном графическом процессоре. Применение обученной модели выглядит следующим образом (см. рис. 3).

Сначала формируется набор пар $\{< s; C_i >\}_{i=1}^{K}$ «входная подпоследовательность, сниппет». Затем элементы данного набора параллельно подаются на вход экземплярам модели, каждая из которых выдает $score(s, C_i)$, соответствующую оценку схожести векторных представлений элементов входной пары в смысле расстояния MPdist. Далее оценка аномальности входной подпоследовательности получается как выполнение редукции по операции минимума $anomaly(s) = \min\limits_{1 \leqslant i \leqslant K} score(s, C_i)$. Подпоследовательность $s$ считается аномальной, если $anomaly(s)$ превышает значение наперед заданного аналитиком порога.

В качестве порога используется значение $k$-го процентиля по набору оценок схожести, которые выдает модель DiSSiD на кортежах валидационной выборки $\mathcal{L}_{\text{valid}}$, имеющих вид $< s_1; s_2; 1 >$ (иными словами, порог — это $k$-й процентиль подпоследовательностей валидационной выборки, входящих в множество ближайших соседей одного и того же сниппета). В данном исследовании в качестве порога применяется $k = 95$.

### 3.2. Алгоритм формирования обучающей выборки

Для формирования обучающей выборки $\mathcal{L}$ для представленной выше модели DiSSiD аналитик выбирает репрезентативный временной ряд, адекватно отражающий типичную деятельность субъекта (противоположную аномалиям, которые предполагается обнаруживать с помощью модели). Автоматизированное формирование обучающей выборки включает в себя два шага: очистка и генерация. Очистка подразумевает формирование множества подпоследовательностей ряда, имеющих заданную аналитиком длину, и удаление из указанного множества аномальных подпоследовательностей, которые не должны попасть в обучающую выборку. На шаге генерации из очищенного множества подпоследовательностей тривиальным образом формируются описанные выше множества $\mathcal{L}_{\text{true}}$ и $\mathcal{L}_{\text{false}}$.

**Алг. 1** CleanData (in: $T$, $m$, $\alpha$, $\varphi$, $K$; out: $\mathcal{L}$)

1: $C_T^m \leftarrow \text{PSF}(T, m, K)$ ▷ Поиск типичной активности
2: $C_{\text{weak}} \leftarrow \{C_i \in C_T^m \mid C_i.frac \leqslant \varphi,\ 1 \leqslant i \leqslant K\}$ ▷ Поиск нетипичной активности
3: $C_T^m \leftarrow C_T^m \setminus C_{\text{weak}}$
4: **for** $i \leftarrow 1$ **to** $|C_T^m|$ **do**
5: $\quad O \leftarrow \text{IsolationForest}(C_i.profile) \cup O$ ▷ Поиск шумов
6: $n_{\text{discord}} \leftarrow \lceil \alpha \cdot (n - m + 1) \rceil$
7: $D \leftarrow \text{PALMAD}(T, m, m, n_{\text{discord}})$ ▷ Поиск аномальной активности
8: $\mathcal{L}_{\text{anomaly}} \leftarrow C_{\text{weak}} \cup C_{\text{weak}}.NN \cup D \cup O$
9: $\mathcal{L} \leftarrow S_T^m \setminus \mathcal{L}_{\text{anomaly}}$ ▷ Очистка
10: **return** $\mathcal{L}$

Псевдокод шага очистки представлен в алг. 1. Данный алгоритм имеет следующие задаваемые аналитиком параметры: репрезентативный ряд $T$ ($|T| = n$), длина подпоследовательности $m$ ($m \ll n$), предполагаемая доля аномальных подпоследовательностей в ряде $\alpha$ ($0 < \alpha \ll 1$), предполагаемое количество сниппетов $K$ ($K > 1$), пороговая мощность сниппета $\varphi$ ($0 < \varphi < 1/K$).

Для очистки обучающей выборки из исходного ряда удаляются подпоследовательности, которые соответствуют аномальной и нетипичной активности субъекта, а также подпоследовательности-шумы. Подпоследовательности, отражающие аномальную активность, трактуются как диссонансы. Подпоследовательностям нетипичной активности субъекта сопоставляются сниппеты, имеющие мощность меньше заданного порога $\varphi$, с множеством своих ближайших соседей. Подпоследовательности-шумы трактуются как выбросы в рамках каждого сниппета.

Поиск сниппетов выполняется с помощью параллельного алгоритма PSF (Parallel Snippet-Finder) [30] (см. стр. 1 в алг. 1). Затем из найденного множества сниппетов исключаются маломощные сниппеты (см. стр. 2, 3 в алг. 1). Далее в множестве ближайших соседей каждого сниппета выполняется нахождение подпоследовательностей-шумов, реализуемое как поиск выбросов в профиле данного сниппета с помощью метода изолирующего леса (Isolation Forest) [31] (см. стр. 4, 5 в алг. 1). Наконец, с помощью разработанного автором параллельного алгоритма PALMAD [24] выполняется поиск диссонансов, реализующий нахождение подпоследовательностей аномальной активности (см. стр. 6, 7 в алг. 1). Мощность множества диссонансов вычисляется на основе параметра $\alpha$, долей аномальных подпоследовательностей в исходном временном ряде (типичным значением данного параметра является $\alpha = 0.05$). В завершении очистки из множества подпоследовательностей ряда исключаются найденные ранее маломощные сниппеты и их ближайшие соседи, а также выбросы и диссонансы (см. стр. 8, 9 в алг. 1), формируя тем самым множество, используемое для генерации обучающей выборки модели.

## 4. Вычислительные эксперименты

В данном разделе представлены результаты вычислительных экспериментов, проведенных на реальных временных рядах, которые имеют истинную разметку аномалий. В экспериментах выполняется сравнение точности предлагаемого метода DiSSiD с рассмотренными в обзоре аналогами (см. раздел 1), относящихся к методам с частичным привлечением учителя. Помимо этого, исследуется влияние функции расстояния между векторными представлениями входных подпоследовательностей (метрики L1 и предложенной в данной работе использование меры MPdist [25]) на эффективность обнаружения аномалий с помощью метода DiSSiD. В заключении данного раздела выполняется оценка времени обучения и тестирования DiSSiD и аналогов.

### 4.1. Наборы данных, аналоги и метрики сравнения

Временные ряды, использованные в экспериментах, взяты из реальных предметных областей и резюмированы в табл. 1. Данные взяты из общедоступного фреймворка TSB-UAD [12], предназначенного для проведения вычислительных экспериментов с алгоритмами обнаружения аномалий во временных рядах.

В экспериментах разработанная модель сравнивалась со следующими аналогами, принадлежащими, как и DiSSiD, к группе методов обнаружения аномалий с частичными привлечением учителя: LSTM-AD [22], OCSVM [39], AE [40], DeepAnT [23], IE-CAE [41], OceanWNN [42], Bagel [43], TAnoGAN [44]. Реализация указанных методов взята из работ [12, 45]. Кроме того, в сравнении участвовала версия DiSSiD, в которой модель выдает оценку схожести векторных представлений входных подпоследовательностей в виде евклидова расстояния вместо MPdist-расстояния.

**Таблица 1.** Временные ряды для вычислительных экспериментов

| № п/п | Временной ряд | Предметная область | Длина ряда $n$ | Длина сниппета $m$ | Длина значимого участка $\ell$ | Кол-во сниппетов $K$ | Доля аномалий $\alpha, \times 10^{-4}$ |
|---|---|---|---|---|---|---|---|
| 1 | SMD [32] | Показания потребления оперативной памяти сервером интернет-провайдера | 23 706 | 100 | 20 | 2 | 5 |
| 2 | OPP [33] | Показания носимого датчика движения при повседневной активности человека | 26 204 | 200 | 50 | 2 | 5 |
| 3 | Daphnet [34] | Показания носимого виброакселерометра, закрепленного на человеке с болезнью Паркинсона | 19 200 | 216 | 72 | 2 | 5 |
| 4 | ECG-2 (803, 805) [35] | Показания ЭКГ пациентов, страдающих синдромом преждевременного сокращения желудочков (конкатенация данных нескольких пациентов) | 200 000 | 250 | 75 | 2 | 8 |
| 5 | ECG-2 (803, 806) [35] | | 200 000 | 250 | 75 | 2 | 5 |
| 6 | ECG-3 (803, 805, 806) [35] | | 300 000 | 250 | 75 | 3 | 10 |
| 7 | MITDB [36] | Показания ЭКГ пациента с нарушенным сердечным ритмом | 200 000 | 520 | 75 | 2 | 2 |
| 8 | IOPS [37] | Показания производительности одного из серверов (операции ввода-вывода) компании Alibaba | 129 010 | 1000 | 500 | 2 | 1 |
| 9 | YAHOO [38] | Показания производительности одного из серверов (операции обращения к памяти) компании Yahoo | 1422 | 60 | 30 | 2 | 10 |

Для оценки качества обнаружения аномалий используется метрика VUS-PR [46], интегрирует в себе как стандартные метрики — точность (precision) и полноту (recall), так и величину смещения найденной аномальной подпоследовательности относительно истинной аномалии. Метрика VUS-PR принимает значения из отрезка $[0, 1]$, большему значению соответствует лучшее качество.

Вычислительные эксперименты выполнялись на вычислительном узле комплекса «Нейрокомпьютер» Суперкомпьютерного центра ЮУрГУ [47], который оснащен графическим процессором NVIDIA Tesla V100 SXM2 (5120 ядер @1.3 ГГц).

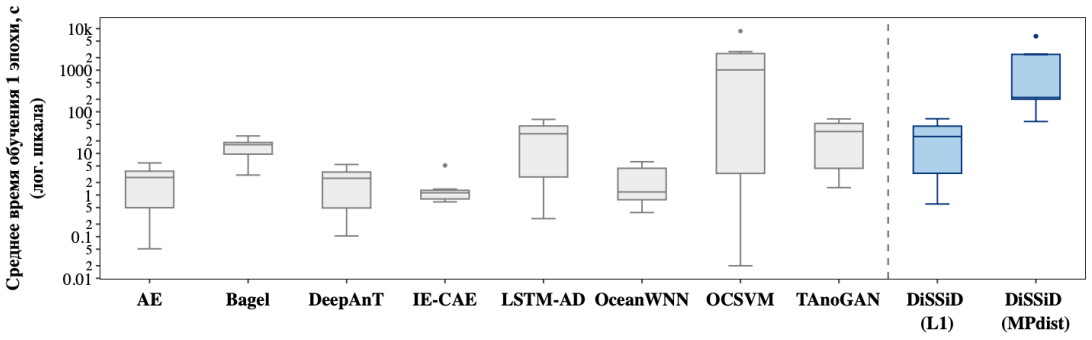
## 4.2. Результаты

**Таблица 2.** Сравнение точности метода DiSSiD с аналогами (метрика VUS-PR)

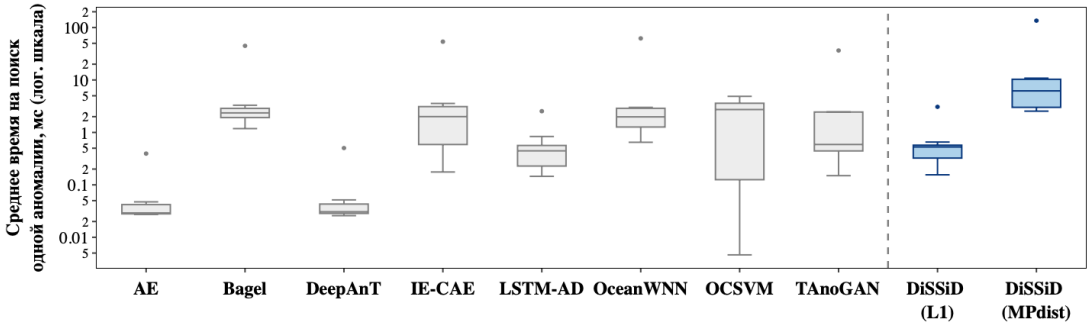
| Методы / Ряды | AE | Bagel | DeepAnT | IE-CAE | LSTM-AD | OceanWNN | OCSVM | TAnoGAN | DiSSiD (L1) | DiSSiD (MPdist) |
|---|---|---|---|---|---|---|---|---|---|---|
| **SMD** | 0.0767 (6) | 0.0559 (8) | 0.0522 (9) | 0.1297 (3) | 0.0653 (7) | 0.1075 (4) | 0.0119 (10) | 0.0965 (5) | 0.1543 (2) | **0.4889 (1)** |
| **OPP** | 0.1979 (5) | nan (10) | 0.0605 (9) | **0.9002 (1)** | 0.0650 (8) | 0.4678 (4) | 0.1795 (6) | 0.8090 (2) | 0.1222 (7) | 0.5340 (3) |
| **Daphnet** | 0.2160 (6) | 0.2269 (5) | 0.2573 (4) | 0.3079 (3) | 0.1711 (8) | 0.1812 (7) | 0.1388 (10) | 0.1609 (9) | **0.4124 (1)** | 0.3332 (2) |
| **ECG-2 (803, 805)** | 0.7758 (2) | 0.3302 (8) | 0.3350 (7) | 0.5234 (5) | 0.2897 (10) | 0.5544 (4) | 0.3548 (6) | 0.3002 (9) | 0.7477 (3) | **0.7801 (1)** |
| **ECG-2 (803, 806)** | 0.5589 (3) | 0.1878 (10) | 0.2346 (7) | 0.5397 (4) | 0.1934 (9) | 0.2003 (8) | 0.3069 (6) | 0.4635 (5) | **0.8008 (1)** | 0.7927 (2) |
| **ECG-3 (803, 805, 806)** | 0.7651 (2) | 0.2988 (7) | 0.2906 (8) | 0.4739 (4) | 0.2330 (9) | 0.3596 (5) | 0.3315 (6) | 0.1430 (10) | 0.7505 (3) | **0.8124 (1)** |
| **MITDB** | 0.0759 (8) | 0.0833 (5) | 0.0795 (7) | 0.1713 (3) | 0.0799 (6) | 0.1058 (4) | 0.0474 (10) | 0.0714 (9) | **0.3718 (1)** | 0.3544 (2) |
| **IOPS** | 0.3720 (7) | 0.2678 (8) | 0.1834 (10) | **0.9163 (1)** | 0.1595 (10) | 0.9085 (4) | 0.7533 (6) | 0.9130 (2) | 0.2464 (9) | 0.7922 (5) |
| **YAHOO** | 0.7238 (2) | 0.4871 (8) | 0.5659 (7) | 0.7050 (3) | 0.4478 (10) | 0.6126 (5) | 0.6639 (4) | 0.4591 (9) | 0.5961 (6) | **0.7306 (1)** |
| **Средний VUS-PR** | 0.4181 (4) | 0.2422 (8) | 0.2288 (9) | 0.5186 (2) | 0.1894 (10) | 0.3886 (5) | 0.3098 (7) | 0.3851 (6) | 0.4669 (3) | **0.6242 (1)** |
| **Средний ранг** | 4.56 (4) | 7.67 (9) | 7.56 (8) | 2.56 (2) | 8.56 (10) | 5 (5) | 7.11 (7) | 6.67 (6) | 3.67 (3) | **2 (1)** |

Результаты сравнения точности метода DiSSiD с аналогами представлены в табл. 2. В ячейке таблицы дано значение меры VUS-PR и в скобках — ранг метода, указанного в соответствующем столбце, среди всех аналогов на временном ряде, указанном в соответствующей строке. Полужирным шрифтом даны результат и место лучшего метода на заданном временном ряде. Две последние строки таблицы являются резюмирующими, в них

указаны соответственно средние значения метрики и ранга по всем рядам, а также среднее значение метрики и ранга метода в скобках. Можно видеть, что при применении евклидова расстояния в функции контрастных потерь метод DiSSiD в среднем входит в тройку лучших методов по точности обнаружения аномалий. Однако использование модифицированной функции контрастных потерь с расстоянием MPdist в формуле (13) позволяет добиться лучшей в среднем точности обнаружения аномалий.




а) обучение




б) тестирование

**Рис. 4.** Сравнение быстродействия метода DiSSiD с аналогами

Рисунок 4 показывает время работы метода DiSSiD по сравнению с аналогами. Можно видеть, что применение расстояния MPdist в функции контрастных потерь делает предложенную модель наиболее медленной как по времени ее обучения, так и по времени поиска аномалий. Причиной этого является применение в DiSSiD вычислительно затратной функции MPdist для нахождения расстояния между векторными представлениями входных подпоследовательностей (временная сложность указанной функции является кубической по отношению к длине подпоследовательности [11]). Низкое быстродействие является в данном случае обратной стороной высокой точности обнаружения аномалий (см. табл. 2).

Рассмотрим отдельно возможность применения модели DiSSiD в режиме реального времени. Из рис. 4б видно, что в экспериментах предложенная модель показывает среднее время поиска аномалий 0.1 с. Такое быстродействие модели допускает ее применение в режиме реального времени, что подтверждают следующие примеры. В системах автоматизации управления сеть передачи данных, обслуживающая датчики измерения температуры, влажности и давления, имеет цикл передачи данных до 100 мс [48]. Компания Emerson, один из ведущих мировых производителей измерительных систем, поставляет беспроводные температурные датчики, имеющие период обновления данных не менее 1 с [49].

## Заключение

В статье рассмотрена задача поиска аномальных подпоследовательностей временного ряда, решение которой в настоящее время востребовано в широком спектре предметных областей: Интернет вещей, умное управление зданиями и городом, персональная медицина и др. Предложен новый метод обнаружения аномальных подпоследовательностей временного ряда с частичным привлечением учителя. Теоретическую основу метода составляют концепции диссонанса [10] и сниппета [11], которые формализуют понятия аномальных и типичных подпоследовательностей временного ряда соответственно. Предложенный метод включает в себя нейросетевую модель, которая определяет степень аномальности входной подпоследовательности ряда, и алгоритм автоматизированного построения обучающей выборки для этой модели.

Нейросетевая модель представляет собой сиамскую нейронную сеть (Siamese Neural Network) [8], которая объединяет в себе две идентичные подсети: количество слоев, число нейронов в каждом слое, размерности входного и выходного слоев, функции активации и др., а также наборы весов и смещений, полученных в результате обучения, одинаковы. Подсеть формирует векторное представление (embedding) входной подпоследовательности. На выходе модель выдает близость между сформированными векторными представлениями в смысле расстояния MPdist [11], используемого для поиска сниппетов. В качестве подсети фигурирует модификация нейросетевой модели ResNet [9]. Для обучения модели предложена модифицированная функция контрастных потерь.

Входным данным для формирования обучающей выборки является репрезентативный временной ряд, адекватно отражающего типичную деятельность субъекта, противоположную аномалиям, которые предполагается обнаруживать с помощью модели. Формирование обучающей выборки включает в себя два шага: очистка и генерация. Очистка подразумевает формирование множества подпоследовательностей ряда, имеющих заданную аналитиком длину, и удаление из указанного множества подпоследовательностей, которые отражают аномальную, нетипичную активность субъекта и шум. Подпоследовательности, отражающие аномальную активность, трактуются как диссонансы. Подпоследовательностям нетипичной активности субъекта сопоставляются сниппеты с мощностью менее заданного аналитиком порога и их ближайшие соседи. Подпоследовательности-шумы трактуются как выбросы в рамках каждого сниппета. На шаге генерации из очищенных подпоследовательностей формируются два множества, объединение которых дает искомую выборку. Элементами первого из них являются пары подпоследовательностей-ближайших соседей одного и того же сниппета, второго — ближайших соседей разных сниппетов.

Применение модели происходит следующим образом. Сначала формируется набор пар «входная подпоследовательность, сниппет». Затем элементы данного набора последовательно подаются на вход модели, которая выдает набор соответствующих оценок схожести векторных представлений элементов входных пар. Далее оценка аномальности входной подпоследовательности получается как минимальное значение по указанному набору. Входная подпоследовательность считается аномальной, если ее оценка превышает значение наперед заданного аналитиком порога. В качестве порога используется значение $k$-го процентиля подпоследовательностей валидационной выборки, входящих в множество ближайших соседей одного и того же сниппета (в данном исследовании применяется значение порога при $k = 95$).

Вычислительные эксперименты на временных рядах из различных предметных областей показывают, что предложенная модель по сравнению с аналогами показывает в среднем наиболее высокую точность обнаружения аномалий по стандартной метрике VUS-PR. Обратной стороной высокой точности метода является большее по сравнению с аналогами время, которое затрачивается на обучение модели и распознавание аномалии. Тем не менее, в приложениях интеллектуального управления отоплением зданий метод обеспечивает быстродействие, достаточное для обнаружения аномальных подпоследовательностей в режиме реального времени.

В качестве направления будущих исследований можно рассматривать расширение разработанного метода для обнаружения аномальных подпоследовательностей в многомерных временных рядах.

*Работа выполнена при финансовой поддержке Российского научного фонда (грант № 23-21-00465).*

## Литература

1. Blázquez-García A., Conde A., Mori U., Lozano J.A. A Review on Outlier/Anomaly Detection in Time Series Data // ACM Comput. Surv. 2021. Vol. 54, no. 3. P. 56:1–56:33. DOI: 10.1145/3444690.
2. Kumar S., Tiwari P., Zymbler M.L. Internet of Things is a revolutionary approach for future technology enhancement: a review // J. Big Data. 2019. Vol. 6. P. 111. DOI: 10.1186/s40537-019-0268-2.
3. Цымблер М.Л., Краева Я.А., Латыпова Е.А. и др. Очистка сенсорных данных в интеллектуальных системах управления отоплением зданий // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 16–36. DOI: 10.14529/cmse210302.
4. Иванов С.А., Никольская К.Ю., Радченко Г.И. и др. Концепция построения цифрового двойника города // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 4. С. 5–23. DOI: 10.14529/cmse200401.
5. Volkov I., Radchenko G.I., Tchernykh A. Digital Twins, Internet of Things and Mobile Medicine: A Review of Current Platforms to Support Smart Healthcare // Program. Comput. Softw. 2021. Vol. 47, no. 8. P. 578–590. DOI: 10.1134/S0361768821080284.
6. Schmidl S., Wenig P., Papenbrock T. Anomaly Detection in Time Series: A Comprehensive Evaluation // Proc. VLDB Endow. 2022. Vol. 15, no. 9. P. 1779–1797. URL: https://www.vldb.org/pvldb/vol15/p1779-wenig.pdf.
7. Hodge V.J., Austin J. A Survey of Outlier Detection Methodologies // Artif. Intell. Rev. 2004. Vol. 22, no. 2. P. 85–126. DOI: 10.1023/B:AIRE.0000045502.10941.a9.
8. Chicco D. Siamese Neural Networks: An Overview // Artificial Neural Networks / ed. by H. Cartwright. New York, NY: Springer US, 2021. P. 73–94. DOI: 10.1007/978-1-0716-0826-5_3.
9. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. IEEE Computer Society, 2016. P. 770–778. DOI: 10.1109/CVPR.2016.90.

10. Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets // Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA. 2007. P. 381–390. DOI: 10.1109/ICDM.2007.61.

11. Imani S., Madrid F., Ding W., *et al.* Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining // 2018 IEEE International Conference on Big Knowledge, ICBK 2018, Singapore, November 17-18, 2018 / ed. by X. Wu, Y. Ong, C.C. Aggarwal, H. Chen. IEEE Computer Society, 2018. P. 382–389. DOI: 10.1109/ICBK.2018.00058.

12. Paparrizos J., Kang Y., Boniol P., *et al.* TSB-UAD: An End-to-End Benchmark Suite for Univariate Time-Series Anomaly Detection // Proc. VLDB Endow. 2022. Vol. 15, no. 8. P. 1697–1711. URL: https://www.vldb.org/pvldb/vol15/p1697-paparrizos.pdf.

13. Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: finding unusual time series in terabyte sized datasets // Knowl. Inf. Syst. 2008. Vol. 17, no. 2. P. 241–262. DOI: 10.1007/s10115-008-0131-9.

14. Yeh C.M., Zhu Y., Ulanova L., *et al.* Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile // Data Min. Knowl. Discov. 2018. Vol. 32, no. 1. P. 83–123. DOI: 10.1007/s10618-017-0519-9.

15. Nakamura T., Imamura M., Mercer R., Keogh E.J. MERLIN: Parameter-free discovery of arbitrary length anomalies in massive time series archives // 20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020 / ed. by C. Plant, H. Wang, A. Cuzzocrea, *et al.* 2020. P. 1190–1195. DOI: 10.1109/ICDM50108.2020.00147.

16. Lu Y., Wu R., Mueen A., *et al.* DAMP: accurate time series anomaly detection on trillions of datapoints and ultra-fast arriving data streams // Data Min. Knowl. Discov. 2023. Vol. 37, no. 2. P. 627–669. DOI: 10.1007/s10618-022-00911-7.

17. Boniol P., Linardi M., Roncallo F., *et al.* Unsupervised and scalable subsequence anomaly detection in large data series // VLDB J. 2021. Vol. 30, no. 6. P. 909–931. DOI: 10.1007/s00778-021-00655-8.

18. Boniol P., Linardi M., Roncallo F., *et al.* Correction to: Unsupervised and scalable subsequence anomaly detection in large data series // VLDB J. 2023. Vol. 32, no. 2. P. 469. DOI: 10.1007/s00778-021-00678-1.

19. Li J., Pedrycz W., Jamal I. Multivariate time series anomaly detection: A framework of Hidden Markov Models // Appl. Soft Comput. 2017. Vol. 60. P. 229–240. DOI: 10.1016/j.asoc.2017.06.035.

20. Marteau P., Soheily-Khah S., Béchet N. Hybrid Isolation Forest - Application to Intrusion Detection // CoRR. 2017. Vol. abs/1705.03800. arXiv: 1705.03800. URL: http://arxiv.org/abs/1705.03800.

21. Ryzhikov A., Borisyak M., Ustyuzhanin A., Derkach D. Normalizing flows for deep anomaly detection // CoRR. 2019. Vol. abs/1912.09323. arXiv: 1912.09323. URL: http://arxiv.org/abs/1912.09323.

22. Malhotra P., Vig L., Shroff G., Agarwal P. Long Short Term Memory Networks for Anomaly Detection in Time Series // 23rd European Symposium on Artificial Neural Networks, ESANN 2015, Bruges, Belgium, April 22-24, 2015. 2015. URL: https://www.esann.org/sites/default/files/proceedings/legacy/es2015-56.pdf.

23. Munir M., Siddiqui S.A., Dengel A., Ahmed S. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series // IEEE Access. 2019. Vol. 7. P. 1991–2005. DOI: 10.1109/ACCESS.2018.2886457.

24. Zymbler M., Kraeva Y. High-Performance Time Series Anomaly Discovery on Graphics Processors // Mathematics. 2023. Vol. 11, no. 14. P. 3193. DOI: 10.3390/math11143193.

25. Gharghabi S., Imani S., Bagnall A.J., *et al.* An ultra-fast time series distance measure to allow data mining in more complex real-world deployments // Data Min. Knowl. Discov. 2020. Vol. 34, no. 4. P. 1104–1135. DOI: 10.1007/s10618-020-00695-8.

26. Yeh C.M., Zhu Y., Ulanova L., *et al.* Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets // IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain / ed. by F. Bonchi, J. Domingo-Ferrer, R. Baeza-Yates, *et al.* IEEE Computer Society, 2016. P. 1317–1322. DOI: 10.1109/ICDM.2016.0179.

27. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. Vol. 37 / ed. by F.R. Bach, D.M. Blei. JMLR.org, 2015. P. 448–456. JMLR Workshop and Conference Proceedings. URL: http://proceedings.mlr.press/v37/ioffe15.html.

28. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions // Int. J. Uncertain. Fuzziness Knowl. Based Syst. 1998. Vol. 6, no. 2. P. 107–116. DOI: 10.1142/S0218488598000094.

29. Hadsell R., Chopra S., LeCun Y. Dimensionality Reduction by Learning an Invariant Mapping // 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA. IEEE Computer Society, 2006. P. 1735–1742. DOI: 10.1109/CVPR.2006.100.

30. Zymbler M., Goglachev A. Fast Summarization of Long Time Series with Graphics Processor // Mathematics. 2022. Vol. 10, no. 10. P. 1781. DOI: 10.3390/math10101781.

31. Liu F.T., Ting K.M., Zhou Z. Isolation Forest // Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy. IEEE Computer Society, 2008. P. 413–422. DOI: 10.1109/ICDM.2008.17.

32. Su Y., Zhao Y., Niu C., *et al.* Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network // Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019. ACM, 2019. P. 2828–2837. DOI: 10.1145/3292500.3330672.

33. Roggen D., Calatroni A., Rossi M., *et al.* Collecting complex activity datasets in highly rich networked sensor environments // Seventh International Conference on Networked Sensing Systems, INSS 2010, Kassel, Germany, June 15-18, 2010. IEEE, 2010. P. 233–240. DOI: 10.1109/INSS.2010.5573462.

34. Bächlin M., Plotnik M., Roggen D., *et al.* Wearable assistant for Parkinson's disease patients with the freezing of gait symptom // IEEE Trans. Inf. Technol. Biomed. 2010. Vol. 14, no. 2. P. 436–446. DOI: 10.1109/TITB.2009.2036165.

35. Goldberger A.L., Amaral L.A.N., Glass L., *et al.* PhysioBank, PhysioToolkit, and PhysioNet components of a new research resource for complex physiologic signals // Circulation. 2000. Vol. 101, no. 23. P. 215–220. DOI: 10.1161/01.CIR.101.23.e215.

36. Moody G., Mark R. The impact of the MIT-BIH Arrhythmia Database // IEEE Engineering in Medicine and Biology Magazine. 2001. Vol. 20, no. 3. P. 45–50. DOI: 10.1109/51.932724.

37. KPI Anomaly Detection Dataset. 2018. URL: http://iops.ai/dataset_detail/?id=10 (дата обращения: 15.08.2023).

38. Laptev N., Amizadeh S., Billawala Y. S5 - A Labeled Anomaly Detection Dataset, version 1.0(16M). 2015. URL: https://webscope.sandbox.yahoo.com/catalog.php?%20datatype=s%5C&did=70 (дата обращения: 15.08.2023).

39. Schölkopf B., Williamson R.C., Smola A.J., *et al.* Support Vector Method for Novelty Detection // Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999] / ed. by S.A. Solla, T.K. Leen, K. Müller. The MIT Press, 1999. P. 582–588. URL: http://papers.nips.cc/paper/1723-support-vector-method-for-novelty-detection.

40. Sakurada M., Yairi T. Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction // Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, Gold Coast, Australia, QLD, Australia, December 2, 2014 / ed. by A. Rahman, J.D. Deng, J. Li. ACM, 2014. P. 4. DOI: 10.1145/2689746.2689747.

41. Garcia G.R., Michau G., Ducoffe M., *et al.* Time Series to Images: Monitoring the Condition of Industrial Assets with Deep Learning Image Processing Algorithms // CoRR. 2020. Vol. abs/2005.07031. arXiv: 2005.07031. URL: https://arxiv.org/abs/2005.07031.

42. Wang Y., Han L., Liu W., *et al.* Study on wavelet neural network based anomaly detection in ocean observing data series // Ocean Engineering. 2019. Vol. 186. P. 106129. DOI: 10.1016/j.oceaneng.2019.106129.

43. Li Z., Chen W., Pei D. Robust and Unsupervised KPI Anomaly Detection Based on Conditional Variational Autoencoder // 37th IEEE International Performance Computing and Communications Conference, IPCCC 2018, Orlando, FL, USA, November 17-19, 2018. IEEE, 2018. P. 1–9. DOI: 10.1109/PCCC.2018.8710885.

44. Bashar M.A., Nayak R. TAnoGAN: Time Series Anomaly Detection with Generative Adversarial Networks // 2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020, Canberra, Australia, December 1-4, 2020. IEEE, 2020. P. 1778–1785. DOI: 10.1109/SSCI47803.2020.9308512.

45. Wenig P., Schmidl S., Papenbrock T. TimeEval: A Benchmarking Toolkit for Time Series Anomaly Detection Algorithms // Proc. VLDB Endow. 2022. Vol. 15, no. 12. P. 3678–3681. URL: https://www.vldb.org/pvldb/vol15/p3678-schmidl.pdf.

46. Paparrizos J., Boniol P., Palpanas T., *et al.* Volume Under the Surface: A New Accuracy Evaluation Measure for Time-Series Anomaly Detection // Proc. VLDB Endow. 2022. Vol. 15, no. 11. P. 2774–2787. URL: https://www.vldb.org/pvldb/vol15/p2774-paparrizos.pdf.

47. Биленко Р.В., Долганина Н.Ю., Иванова Е.В., Рекачинский А.И. Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университета // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 15–30. DOI: 10.14529/cmse220102.

48. Лопухов И. Сети Real-Time Ethernet: от теории к практической реализации // СТА: Современные технологии автоматизациии. 2010. Т. 10, № 3. С. 8–15.

49. Каталог 2021. Датчики температуры Emerson. URL: https://www.c-o-k.ru/library/catalogs/emerson/110477.pdf (дата обращения: 03.09.2021).

Краева Яна Александровна, старший преподаватель, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

**DOI: 10.14529/cmse230304**

# DETECTION OF TIME SERIES ANOMALIES BASED ON DATA MINING AND NEURAL NETWORK TECHNOLOGIES

**© 2023 Ya.A. Kraeva**

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

*E-mail: kraevaya@susu.ru*

Received: 20.05.2023

The article touches upon the problem of discovering subsequence anomalies in time series, which is currently in demand in a wide range of subject domains. We propose a new semi-supervised method to detect subsequence anomalies in time series. The method is based on the concepts of discord and snippet, which formalize, respectively, the concepts of anomalous and typical time series subsequences. The proposed method includes a neural network model that calculates the anomaly score of the input subsequence and an algorithm to automatically construct the model's training set. The model is implemented as a Siamese neural network, where we employ a modification of ResNet as a subnet. To train the model, we proposed a modified contrast loss function. The training set is formed as a representative fragment of the time series from which discords, low-fraction snippets with their nearest neighbors, and outliers within each snippet are removed since they are interpreted as abnormal, atypical activity of the subject, and noise, respectively. Computational experiments over time series from various subject domains showed that the proposed model, compared with analogues, has on average the highest accuracy of anomaly detection with respect to the standard VUS-PR metric. The downside of the high accuracy of the method is the longer time spent on model training and anomaly detection compared to analogues. Nevertheless, in applications of intelligent building heating control, the method provides a speed sufficient to detect subsequence anomalies in real time.

*Keywords: time series, anomaly detection, discord, snippet, Siamese neural network.*

## FOR CITATION

Kraeva Ya.A. Detection of Time Series Anomalies Based on Data Mining and Neural Network Technologies. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 3. P. 50–71. (in Russian) DOI: 10.14529/cmse230304.

## References


1. Blázquez-García A., Conde A., Mori U., Lozano J.A. A Review on Outlier/Anomaly Detection in Time Series Data. ACM Comput. Surv. 2021. Vol. 54, no. 3. P. 56:1–56:33. DOI: 10.1145/3444690.
2. Kumar S., Tiwari P., Zymbler M.L. Internet of Things is a revolutionary approach for future technology enhancement: a review. J. Big Data. 2019. Vol. 6. P. 111. DOI: 10.1186/s40537-019-0268-2.
3. Zymbler M.L., Kraeva Y.A., Latypova E.A., *et al.* Cleaning Sensor Data in Intelligent Heating Control System. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 3. P. 16–36. (in Russian) DOI: 10.14529/cmse210302.
4. Ivanov S.A., Nikolskaya K.Y., Radchenko G.I., *et al.* Digital Twin of a City: Concept Overview. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2020. Vol. 9, no. 4. P. 5–23. (in Russian) DOI: 10.14529/cmse200401.
5. Volkov I., Radchenko G.I., Tchernykh A. Digital Twins, Internet of Things and Mobile Medicine: A Review of Current Platforms to Support Smart Healthcare. Program. Comput. Softw. 2021. Vol. 47, no. 8. P. 578–590. DOI: 10.1134/S0361768821080284.
6. Schmidl S., Wenig P., Papenbrock T. Anomaly Detection in Time Series: A Comprehensive Evaluation. Proc. VLDB Endow. 2022. Vol. 15, no. 9. P. 1779–1797. URL: https://www.vldb.org/pvldb/vol15/p1779-wenig.pdf.
7. Hodge V.J., Austin J. A Survey of Outlier Detection Methodologies. Artif. Intell. Rev. 2004. Vol. 22, no. 2. P. 85–126. DOI: 10.1023/B:AIRE.0000045502.10941.a9.
8. Chicco D. Siamese Neural Networks: An Overview. Artificial Neural Networks / ed. by H. Cartwright. New York, NY: Springer US, 2021. P. 73–94. DOI: 10.1007/978-1-0716-0826-5_3.
9. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. IEEE Computer Society, 2016. P. 770–778. DOI: 10.1109/CVPR.2016.90.
10. Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA. 2007. P. 381–390. DOI: 10.1109/ICDM.2007.61.
11. Imani S., Madrid F., Ding W., *et al.* Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining. 2018 IEEE International Conference on Big Knowledge, ICBK 2018, Singapore, November 17-18, 2018 / ed. by X. Wu, Y. Ong, C.C. Aggarwal, H. Chen. IEEE Computer Society, 2018. P. 382–389. DOI: 10.1109/ICBK.2018.00058.
12. Paparrizos J., Kang Y., Boniol P., *et al.* TSB-UAD: An End-to-End Benchmark Suite for Univariate Time-Series Anomaly Detection. Proc. VLDB Endow. 2022. Vol. 15, no. 8. P. 1697–1711. URL: https://www.vldb.org/pvldb/vol15/p1697-paparrizos.pdf.

13. Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: finding unusual time series in terabyte sized datasets. Knowl. Inf. Syst. 2008. Vol. 17, no. 2. P. 241–262. DOI: 10.1007/s10115-008-0131-9.

14. Yeh C.M., Zhu Y., Ulanova L., *et al.* Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. Data Min. Knowl. Discov. 2018. Vol. 32, no. 1. P. 83–123. DOI: 10.1007/s10618-017-0519-9.

15. Nakamura T., Imamura M., Mercer R., Keogh E.J. MERLIN: Parameter-free discovery of arbitrary length anomalies in massive time series archives. 20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020 / ed. by C. Plant, H. Wang, A. Cuzzocrea, *et al.* IEEE, 2020. P. 1190–1195. DOI: 10.1109/ICDM50108.2020.00147.

16. Lu Y., Wu R., Mueen A., *et al.* DAMP: accurate time series anomaly detection on trillions of datapoints and ultra-fast arriving data streams. Data Min. Knowl. Discov. 2023. Vol. 37, no. 2. P. 627–669. DOI: 10.1007/s10618-022-00911-7.

17. Boniol P., Linardi M., Roncallo F., *et al.* Unsupervised and scalable subsequence anomaly detection in large data series. VLDB J. 2021. Vol. 30, no. 6. P. 909–931. DOI: 10.1007/s00778-021-00655-8.

18. Boniol P., Linardi M., Roncallo F., *et al.* Correction to: Unsupervised and scalable subsequence anomaly detection in large data series. VLDB J. 2023. Vol. 32, no. 2. P. 469. DOI: 10.1007/s00778-021-00678-1.

19. Li J., Pedrycz W., Jamal I. Multivariate time series anomaly detection: A framework of Hidden Markov Models. Appl. Soft Comput. 2017. Vol. 60. P. 229–240. DOI: 10.1016/j.asoc.2017.06.035.

20. Marteau P., Soheily-Khah S., Béchet N. Hybrid Isolation Forest - Application to Intrusion Detection. CoRR. 2017. Vol. abs/1705.03800. arXiv: 1705.03800. URL: http://arxiv.org/abs/1705.03800.

21. Ryzhikov A., Borisyak M., Ustyuzhanin A., Derkach D. Normalizing flows for deep anomaly detection. CoRR. 2019. Vol. abs/1912.09323. arXiv: 1912.09323. URL: http://arxiv.org/abs/1912.09323.

22. Malhotra P., Vig L., Shroff G., Agarwal P. Long Short Term Memory Networks for Anomaly Detection in Time Series. 23rd European Symposium on Artificial Neural Networks, ESANN 2015, Bruges, Belgium, April 22-24, 2015. 2015. URL: https://www.esann.org/sites/default/files/proceedings/legacy/es2015-56.pdf.

23. Munir M., Siddiqui S.A., Dengel A., Ahmed S. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. IEEE Access. 2019. Vol. 7. P. 1991–2005. DOI: 10.1109/ACCESS.2018.2886457.

24. Zymbler M., Kraeva Y. High-Performance Time Series Anomaly Discovery on Graphics Processors. Mathematics. 2023. Vol. 11, no. 14. P. 3193. DOI: 10.3390/math11143193.

25. Gharghabi S., Imani S., Bagnall A.J., *et al.* An ultra-fast time series distance measure to allow data mining in more complex real-world deployments. Data Min. Knowl. Discov. 2020. Vol. 34, no. 4. P. 1104–1135. DOI: 10.1007/s10618-020-00695-8.

26. Yeh C.M., Zhu Y., Ulanova L., *et al.* Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain / ed. by F. Bonchi, J. Domingo-Ferrer, R. Baeza-Yates, *et al.* IEEE Computer Society, 2016. P. 1317–1322. DOI: 10.1109/ICDM.2016.0179.

27. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. Vol. 37 / ed. by F.R. Bach, D.M. Blei. JMLR.org, 2015. P. 448–456. JMLR Workshop and Conference Proceedings. URL: http://proceedings.mlr.press/v37/ioffe15.html.

28. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. Int. J. Uncertain. Fuzziness Knowl. Based Syst. 1998. Vol. 6, no. 2. P. 107–116. DOI: 10.1142/S0218488598000094.

29. Hadsell R., Chopra S., LeCun Y. Dimensionality Reduction by Learning an Invariant Mapping. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA. IEEE Computer Society, 2006. P. 1735–1742. DOI: 10.1109/CVPR.2006.100.

30. Zymbler M., Goglachev A. Fast Summarization of Long Time Series with Graphics Processor. Mathematics. 2022. Vol. 10, no. 10. P. 1781. DOI: 10.3390/math10101781.

31. Liu F.T., Ting K.M., Zhou Z. Isolation Forest. Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy. IEEE Computer Society, 2008. P. 413–422. DOI: 10.1109/ICDM.2008.17.

32. Su Y., Zhao Y., Niu C., *et al.* Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019. ACM, 2019. P. 2828–2837. DOI: 10.1145/3292500.3330672.

33. Roggen D., Calatroni A., Rossi M., *et al.* Collecting complex activity datasets in highly rich networked sensor environments. Seventh International Conference on Networked Sensing Systems, INSS 2010, Kassel, Germany, June 15-18, 2010. IEEE, 2010. P. 233–240. DOI: 10.1109/INSS.2010.5573462.

34. Bächlin M., Plotnik M., Roggen D., *et al.* Wearable assistant for Parkinson's disease patients with the freezing of gait symptom. IEEE Trans. Inf. Technol. Biomed. 2010. Vol. 14, no. 2. P. 436–446. DOI: 10.1109/TITB.2009.2036165.

35. Goldberger A.L., Amaral L.A.N., Glass L., *et al.* PhysioBank, PhysioToolkit, and PhysioNet components of a new research resource for complex physiologic signals. Circulation. 2000. Vol. 101, no. 23. P. 215–220. DOI: 10.1161/01.CIR.101.23.e215.

36. Moody G., Mark R. The impact of the MIT-BIH Arrhythmia Database. IEEE Engineering in Medicine and Biology Magazine. 2001. Vol. 20, no. 3. P. 45–50. DOI: 10.1109/51.932724.

37. KPI Anomaly Detection Dataset. 2018. URL: http://iops.ai/dataset_detail/?id=10 (accessed: 15.08.2023).

38. Laptev N., Amizadeh S., Billawala Y. S5 - A Labeled Anomaly Detection Dataset, version 1.0(16M). 2015. URL: https://webscope.sandbox.yahoo.com/catalog.php?%20datatype=s&did=70 (accessed: 15.08.2023).

39. Schölkopf B., Williamson R.C., Smola A.J., *et al.* Support Vector Method for Novelty Detection. Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999] / ed. by S.A. Solla, T.K. Leen, K. Müller. The MIT Press, 1999. P. 582–588. URL: http://papers.nips.cc/paper/1723-support-vector-method-for-novelty-detection.

40. Sakurada M., Yairi T. Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, Gold Coast, Australia, QLD, Australia, December 2, 2014 / ed. by A. Rahman, J.D. Deng, J. Li. ACM, 2014. P. 4. DOI: 10.1145/2689746.2689747.

41. Garcia G.R., Michau G., Ducoffe M., *et al.* Time Series to Images: Monitoring the Condition of Industrial Assets with Deep Learning Image Processing Algorithms. CoRR. 2020. Vol. abs/2005.07031. arXiv: 2005.07031. URL: https://arxiv.org/abs/2005.07031.

42. Wang Y., Han L., Liu W., *et al.* Study on wavelet neural network based anomaly detection in ocean observing data series. Ocean Engineering. 2019. Vol. 186. P. 106129. DOI: 10.1016/j.oceaneng.2019.106129.

43. Li Z., Chen W., Pei D. Robust and Unsupervised KPI Anomaly Detection Based on Conditional Variational Autoencoder. 37th IEEE International Performance Computing and Communications Conference, IPCCC 2018, Orlando, FL, USA, November 17-19, 2018. IEEE, 2018. P. 1–9. DOI: 10.1109/PCCC.2018.8710885.

44. Bashar M.A., Nayak R. TAnoGAN: Time Series Anomaly Detection with Generative Adversarial Networks. 2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020, Canberra, Australia, December 1-4, 2020. IEEE, 2020. P. 1778–1785. DOI: 10.1109/SSCI47803.2020.9308512.

45. Wenig P., Schmidl S., Papenbrock T. TimeEval: A Benchmarking Toolkit for Time Series Anomaly Detection Algorithms. Proc. VLDB Endow. 2022. Vol. 15, no. 12. P. 3678–3681. URL: https://www.vldb.org/pvldb/vol15/p3678-schmidl.pdf.

46. Paparrizos J., Boniol P., Palpanas T., *et al.* Volume Under the Surface: A New Accuracy Evaluation Measure for Time-Series Anomaly Detection. Proc. VLDB Endow. 2022. Vol. 15, no. 11. P. 2774–2787. URL: https://www.vldb.org/pvldb/vol15/p2774-paparrizos.pdf.

47. Bilenko R.V., Dolganina N.Y., Ivanova E.V., Rekachinsky A.I. High-performance Computing Resources of South Ural State University. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 1. P. 15–30. (in Russian) DOI: 10.14529/cmse220102.

48. Lopukhov I. Real-Time Ethernet network: from theory to practical implementation. MAT: Modern automation technologies. 2010. Vol. 10, no. 3. P. 8–15.

49. Catalogue 2021. Emerson temperature sensors. URL: https://www.c-o-k.ru/library/catalogs/emerson/110477.pdf (accessed: 03.09.2021).

УДК 519.766, 519.1, 519.713.1, 519.713.2 DOI: 10.14529/cmse230305

# ОБ ОДНОЙ ГИПОТЕЗЕ ТЕОРИИ ФОРМАЛЬНЫХ ЯЗЫКОВ. ЧАСТЬ I

© 2023 Б.Ф. Мельников

*Совместный университет МГУ – ППИ в Шэньчжэне*
*(Китай, 518172, провинция Гуандун, Шэньчжэнь, район Лунган,*
*Даюньсиньчэн, ул. Гоцзидасюеюань, д. 1)*
*E-mail: bormel@mail.ru*

Поступила в редакцию: 20.06.2023

Основной предмет статьи — рассмотрение задач, возникающих при исследовании необходимых условий равенства бесконечных итераций конечных языков. В предыдущих публикациях автором рассматривались примеры применения соответствующего этому равенству специального бинарного отношения эквивалентности на множестве конечных языков, причем рассматривались как примеры, описывающие необходимые условия его выполнения, так и примеры его использования. К одному из таких необходимых условий применены два варианта сведéния рассматриваемой задачи: к конечным автоматам и к бесконечным итерационным деревьям. Также в статье приведены несколько вариантов важной гипотезы, формулируемой для множества конечных языков; ее исследование дает и иные варианты сведéния рассматриваемой задачи к специальным задачам для недетерминированных конечных автоматов. При этом в случае выполнения сформулированной гипотезы некоторые из таких задач решаются за полиномиальное время, а некоторые не решаются; при продолжении работ по данной тематике последний факт может дать возможность переформулировки проблемы $\mathsf{P} = \mathsf{NP}$ в виде специальной задачи теории формальных языков.

*Ключевые слова: формальные языки, итерации языков, бинарные отношения, морфизмы, инверсные морфизмы, алгоритмы, полиномиальные алгоритмы.*

## ОБРАЗЕЦ ЦИТИРОВАНИЯ

Мельников Б.Ф. Об одной гипотезе теории формальных языков. Часть I // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 3. С. 72–86. DOI: 10.14529/cmse230305.

## Введение

Некоторые результаты предлагаемой работы уже были опубликованы, причем относительно недавно, в 2019 г. и позднее — см. [1–12]. Однако их желательно повторить (с добавлениями и изменениями) — по следующим причинам.

- Во-первых, необходима публикация на эту тематику именно в журнале, тематика которого в большей степени связана с прикладной математикой: в упомянутых выше публикациях большее внимание уделялось связи с теоретической информатикой [1].
- Во-вторых, приводятся подробные обозначения для применяемой в статье версии недетерминированных конечных автоматов (соответствующие определения и простейшие утверждения). Эти обозначения для конечных автоматов на русском языке уже были опубликованы — но в полном объеме лишь в монографии, [13].
- В-третьих, немного по-другому описываются результаты статьи [14] (1993 г.) — где было приведено *неполное* доказательство одного из рассматривавшихся там утверждений.

[1] Хотя, конечно, провести четкую границу между собственно математикой и теоретической информатикой вряд ли возможно.

- В-четвертых, хотя настоящую работу можно назвать обобщающей, — но в нее автор постарался включить только такой минимум из материала ранее опубликованных статей, который необходим для понимания рассматриваемых проблем. Здесь под этим минимумом в первую очередь понимается то, что необходимо для плана сведения равенства $\mathsf{P} = \mathsf{NP}$ (гипотезы $\mathsf{P} = \mathsf{NP}$) к специальной гипотезе теории формальных языков.
- Однако при этом («в-пятых»), такой минимально возможный объем материала отражает не только сами результаты работы, но и связь с другими ранее рассматривавшимися задачами. Более того, как автор надеется, при такой подаче, которая применена в настоящей обобщающей статье, более понятна мотивация к рассмотрению вспомогательных задач — а эту мотивацию при простом чтении в хронологическом порядке статей [1–12] можно и не увидеть.
- В-шестых, в настоящей статье добавлены некоторые новые результаты, полученные в последнее время.
- Однако самой важной является следующая причина («в-седьмых»): существенно изменяется сама структура изложения. А именно, в упомянутых выше работах основной целью было решение соответствующих вспомогательных задач, в частности — описание возможных полурешеток, полученных на основе двух заданных конечных языков, а также описание применения результатов, связанных с этими полурешетками, в прикладных задачах теории языков. А у настоящей статьи цель более глобальная, и поэтому приводится:
  - подробное описание самих рассматриваемых задач;
  - описание нескольких гипотез, возникающих для специальных множеств формальных языков (иными словами — для подмоноидов глобального надмоноида свободного моноида, [15]);
  - а также связь этих задач с упомянутой выше проблемой $\mathsf{P} = \mathsf{NP}$.

Все упомянутые выше задачи связаны с исследованием бинарного отношения эквивалентности в бесконечности, заданного на множестве конечных языков, — отношения $\Diamond\!\!\!\!\!-$. В предыдущих публикациях уже были рассмотрены примеры применения как этого отношения, так и его частного случая — отношения $\leqslant$; при этом примеры можно разделить на две группы:

- примеры, описывающие необходимые условия его выполнения;
- и примеры его использования — в различных областях теории формальных языков, дискретной математики и абстрактной алгебры.

Из сказанного выше следует, что основным предметом статьи можно считать рассмотрение разных задач, возникающих при исследовании необходимых условий равенства бесконечных итераций конечных языков (иными словами — условий выполнения бинарного отношения $\Diamond\!\!\!\!\!-$). И, по-видимому, главной «составляющей» такой мотивации является план доказательства возможности сведения равенства $\mathsf{P} = \mathsf{NP}$ к специальной гипотезе теории формальных языков, сформулированной ранее в предыдущих публикациях. Стоит отметить по этому поводу, что первая версия этого плана была опубликована в 2011 г., см. [16] — а здесь приведена измененная версия, причем со значительно большим числом комментариев. Однако основная идея при этом та же самая — но здесь она уже оформлена конкретно, и решения многих вспомогательных подзадач, необходимых для осуществления указанного плана, за прошедшее время уже опубликованы.

Кратко этот план можно сформулировать следующим образом. По некоторому недетерминированному конечному автомату специальным образом строится пара конечных языков — и для этой пары показывается, что если бы была выполнена вышеупомянутая гипотеза, то существовал бы и алгоритм проверки выполнения отношения эквивалентности в бесконечности за полиномиальное время. Но, с другой стороны, взяв произвольный недетерминированный конечный автомат и рассматривая его как определенный в предыдущих публикациях т. н. автомат NSPRI (мы его строго определим и будем рассматривать начиная с раздела 6 настоящей статьи), мы за полиномиальное время могли бы построить соответствующую этому автомату пару конечных языков — причем такую пару, которая удовлетворяет отношению $\lessdot\!\!\!\cdot\!\!\!\gtrdot$ тогда и только тогда, когда язык автомата NSPRI совпадает с универсальным языком над заданным алфавитом (т. е. языком, содержащим все возможные слова). Таким образом, в случае осуществления плана, кратко описанного здесь, мы покажем, что выполнение вышеупомянутой основной гипотезы влечет выполнение равенства $\mathsf{P} = \mathsf{NP}$.

В конце данного раздела отметим следующее. Мы всюду пишем «сведение *равенства* $\mathsf{P} = \mathsf{NP}$ к одной из гипотез теории формальных языков» (и т. п.) – но, по-видимому, еще точнее было бы говорить «сведение *гипотезы* $\mathsf{P} = \mathsf{NP}$ к одной из гипотез теории формальных языков»; однако последний вариант звучит хуже (из-за двух вхождений слова «гипотеза»). На сказанное здесь можно было бы возразить, что любая (или почти любая) вычислительная проблема может рассматриваться именно как проблема теории формальных языков (см. [17, 18] и мн. др.); однако «возражение на возражение» заключается в том, что для получения необходимой информации нужно решить *только одну* проблему, или, другими словами, здесь речь идет о формулировке задачи про задачи, можно сказать — «о формулировке метазадачи».

Следует также особо отметить, что даже после возможного завершения реализации плана, приведенного вышу, о возможности решения проблемы $\mathsf{P} = \mathsf{NP}$ ничего утверждаться не будет: будет показана только возможность сведения этой проблемы к проблеме теории формальных языков.

Приведем содержание статьи по разделам.

*В разделе 1* приводятся основные обозначения и обсуждаются соглашения об их использовании; некоторые из этих обозначений являются нестандартными. Одним из наиболее важных понятий, определенных в этом разделе, является бинарное отношение $\lessdot\!\!\!\cdot\!\!\!\gtrdot$, определенное на множестве конечных языков над рассматриваемым алфавитом; можно сказать, что с этим бинарным отношением связан весь материал настоящей статьи.

*В разделе 2* рассматриваются *бесконечные деревья особого вида*, которые возникают в различных задачах теории формальных языков. Отличительной особенностью этих деревьев является то, что все их ребра отмечены буквами заданного алфавита, а вершины делятся на некоторые классы эквивалентности, причем (бесконечные) *поддеревья, соответствующие вершинам одного и того же класса, эквивалентны.* Разделом 2 заканчивается часть I настоящей статьи, последующие разделы составят части II–IV.

*В разделе 3* обсуждается несложное *преобразование*, строящее по заданному бесконечному итерационному дереву специальный детерминированный конечный автомат. При этом с точки зрения алгебры мы переходим от работы с элементами определенного множества *к работе с классами эквивалентности*, на которые это множество делится с помощью бинарного отношения, фактически рассмотренного в предыдущем разделе 2.

Таким образом, в разделе 3 рассматривается общий (абстрактный) вариант преобразования произвольного бесконечного итерационного дерева в детерминированный конечный автомат; а *в разделах 4 и 5* — конкретное дерево, получаемое не «абстрактно», как в разделе 3 и ранее, а специально описанным образом по двум заданным конечным языкам; соответственно, получается связанный с этими языками детерминированный автомат. Эти дерево и автомат предназначены *для проверки упомянутого выше бинарного отношения* $\trianglelefteqslant$; точнее, используется другое бинарное отношение $\trianglelefteq$, двойное применение которого и дает требуемое отношение $\trianglelefteqslant$.

*В разделе 6* описывается совершенно иной объект (теперь — недетерминированный конечный автомат) — хотя при этом стоит отметить, что построение этого автомата может быть выполнено с использованием тех же самых алгоритмов (и соответствующих компьютерных программ, [5, 6]), что и для предыдущего раздела. Итак, в разделе 6 рассматривается недетерминированный автомат, *отвечающий на тот же самый вопрос*. Важно отметить, что последний автомат определен на конечном множестве слов — а не на конечном множестве языков (конечных множеств слов).

*В разделе 7* и последующих разделах рассматривается *специальная гипотеза теории формальных языков*, которую автор считает очень важной; по ней и названа вся статья. Очень кратко эту гипотезу можно сформулировать следующим образом: бесконечные итерации конечных языков одинаковы тогда и только тогда, когда эти языки могут быть представлены как один и тот же морфизм расширенных максимальных префиксных кодов, определенных над некоторым вспомогательным алфавитом (едином для обоих языков). Непосредственно в разделе 7 рассматриваются результаты предыдущих публикаций, относящиеся к упомянутой гипотезе в частности и к материалу настоящей статьи вообще. Далее *в разделе 8* рассматриваются несколько *основных вариантов формулировки* этой гипотезы — а именно, такие варианты, которые не использующие автоматы и деревья.

*В разделе 9* кратко обсуждаются недетерминированные конечные автоматы особого вида — так называемые лепестковые автоматы (иногда в английской литературе встречается термин “semiflower automata”). Мы приводим результаты одной из предыдущих работ, эти результаты можно сформулировать следующим образом: для любого регулярного языка и его таблицы бинарного отношения # (подробнее об этом отношении см. раздел 1) существует алгоритм построения лепесткового автомата, имеющего либо ту же самую таблицу, либо ее же с ровно одним добавленным столбцом (т. е. с ровно одним добавленным состоянием канонического автомата для зеркального языка). Конкретно в настоящей статье лепестковые автоматы используются для еще двух переформулировок основной рассматриваемой в статье гипотезы — см. *раздел 10*. Поэтому название раздела 10 таково: «Вспомогательные варианты формулировки гипотезы — с использованием автоматов и деревьев». Для таких вариантов и используются лепестковые конечные автоматы, а также бесконечные итерационные деревья.

Упомянутый выше план сведения равенства $\mathsf{P} = \mathsf{NP}$ к более простой формулировке приведен *в разделе 11*. Более детальное название этого раздела могло бы быть таким: «О плане доказательства возможности сведения равенства $\mathsf{P} = \mathsf{NP}$ к гипотезе теории формальных языков».

Последний раздел — заключение; в нем формулируются возможные направления дальнейших работ по тематике настоящей статьи.

## 1. Предварительные сведения

В этом разделе приводятся основные обозначения и соглашения об их использовании. Следует отметить, что некоторые из этих обозначений являются нестандартными. Одним из наиболее важных понятий, приводимых в этом разделе, является бинарное отношение $\leqslant\!\!\!\!\diamond\!\!\!\!\geqslant$, определенное на множестве конечных языков над рассматриваемым алфавитом, с ним связан весь материал этой статьи. Перейдем к самим обозначениям.

Чаще всего слова и языки будут рассматриваться над главным алфавитом $\Sigma$; вспомогательным же алфавитом обычно будет $\Delta$, иногда с нижними индексами; все алфавиты всегда конечны.

Для заданного слова $u \in \Sigma^*$ и языка $A \subseteq \Sigma^*$:

- язык $\mathrm{pref}(u)$ – это множество префиксов слова $u$ (включая само $u$);
- $\mathrm{opref}(u) = \mathrm{pref}(u) \setminus \{u\}$;
- $\mathbf{pref}(A) = \bigcup\limits_{u \in A} \mathbf{pref}(u)$;
- $\mathrm{opref}(A)$ определяется аналогично.

Если для двух конечных языков $A$ и $B$ (в предыдущих работах они чаще всего рассматриваются над главным алфавитом $\Sigma$) выполняется условие

$$(\forall u \in A^*)\ (\exists v \in B^*)\ (u \in \mathbf{opref}(v)),$$

то будем писать $A \trianglelefteqslant B$ (либо $B \trianglerighteqslant A$). Если условия $A \trianglelefteqslant B$ и $A \trianglerighteqslant B$ выполняются одновременно, то будем писать $A \leqslant\!\!\!\!\diamond\!\!\!\!\geqslant B$.

(Специально отметим еще раз разницу в обозначениях: с одной стороны, в недавних работах [1–12], и с другой стороны, в [14] и некоторых других работах 1990-х годов. Конечно, в настоящей статье будут использоваться только обозначения, приведенные в этом разделе.)

Теперь приведем некоторые обозначения, связанные с недетерминированными конечными автоматами. Пусть

$$K = \big(\, Q,\, \Sigma,\, \delta,\, S,\, F \,\big) \tag{1}$$

— некоторый автомат. $\delta$ — его функция переходов вида $\delta : Q \times \Sigma \to \mathcal{P}(Q)$, где обозначение $\mathcal{P}(Q)$ обозначает надмножество (степень множества, множество подмножеств) множества $Q$; таким образом, будут рассматриваться только автоматы без $\varepsilon$-переходов. Иногда некоторая дуга графа переходов автомата $\delta(q, a) \ni r$ будет записываться в виде $q \xrightarrow[\delta]{a} r$, или, если это не вызывает неоднозначности, просто в виде $q \xrightarrow{a} r$.

Зеркальный автомат для автомата (1), т. е. $\big(Q, \Sigma, \delta^R, F, S\big)$, где $q' \xrightarrow[\delta^R]{a} q''$ тогда и только тогда, когда $q'' \xrightarrow[\delta]{a} q'$, будем обозначать $K^R$; заметим, что $K^R$ всегда определяет зеркальный язык $L^R$.

Для рассматриваемого языка $L$ его канонический автомат будет обозначаться записью $\widetilde{L}$. Пусть автоматы $\widetilde{L}$ и $\widetilde{L^R}$ для заданного регулярного языка $L$ таковы:

$$\widetilde{L} = \big(Q_\pi, \Sigma, \delta_\pi, \{s_\pi\}, F_\pi\big) \quad \text{и} \quad \widetilde{L^R} = \big(Q_\rho, \Sigma, \delta_\rho, \{s_\rho\}, F_\rho\big).$$

Более того, для языков конечных автоматов не будет допускаться возможность $L = \varnothing$, поэтому оба этих автомата действительно имеют стартовые состояния (т. е. приведенные обозначения не могут содержать противоречия).

Для автомата (1) выходной язык состояния $q$ (он обозначается записью $\mathcal{L}^{out}_K(q)$) — это язык автомата $\big(Q, \Sigma, \delta, \{q\}, F\big)$. Аналогично, входной язык состояния $q$ (обозначается $\mathcal{L}^{in}_K(q)$) — это язык автомата $\big(Q, \Sigma, \delta, S, \{q\}\big)$.

Также напомним определение бинарного отношения $\#$, подробнее см. [19] и др. Отношение $\# \subseteq Q_\pi \times Q_\rho$ определяется для пар состояний автоматов $\widetilde{L}$ и $\widetilde{L^R}$ следующим образом: $A \# X$ тогда и только тогда, когда

$$\big(\exists uv \in L\big)\ \big(u \in \mathcal{L}^{in}_{\widetilde{L}}(A)\,,\, v^R \in \mathcal{L}^{in}_{\widetilde{L^R}}(X)\big)$$

(в последней записи для каждого из канонических автоматов применено обозначение входного языка состояния). Важно отметить, что именно такой вариант определения неконструктивен; однако, например, [19] содержит эквивалентный конструктивный вариант подобного определения (однако, как обычно в подобных ситуациях, более сложно формулируемый).

Важно также отметить, что детерминированные конечные автоматы всегда рассматриваются как *частный случай недетерминированных автоматов*, определенных в соответствие с (1), — аналогично большинству источников, но в отличие, например, от [20]. В статье не приводятся более подробные определения для детерминированных автоматов, они являются более-менее стандартными; отметим только, что будут специально использоваться так называемые всюду определенные автоматы (“total automata”), т. е. такие, для которых выполняется следующее условие:

$$\big(\forall q \in Q\big)\ \big(\forall a \in \Sigma\big)\ \big(|\delta(q, a)| = 1\big).$$

В то же время во многих публикациях подобная такая «полнота» даже рассматривается в качестве обязательного свойства детерминированного автомата — однако в настоящей статье применяется другой подход, т. е. всюду определенные автоматы рассматриваются в качестве собственного подмножества автоматов детерминированных. В настоящей статье будут использоваться и так называемые лепестковые конечные автоматы (“semiflower automata”); подробности см. в разделе 10, где кроме них определены еще и автоматы типа $\mathcal{K}(A)$ для заданного конечного языка $A \subseteq \Sigma^*$. Дополнительная информация о связанных с такими автоматами обозначениях приведена в [21].

Перейдем к специальным обозначениям, связанным с морфизмами языков. Для этого сначала специально отметим, что все рассматриваемые конечные языки не являются пустыми и не содержат пустого слова $\varepsilon$. Для некоторого языка

$$A \subseteq \Sigma^*, \quad A = \big\{\, u_1, u_2, \ldots u_n \,\big\}$$

(не ограничивая общности можно предполагать, что слова языка каким-то образом упорядочены), мы рассматриваем алфавит $\Delta_A = \big\{\, d_1, d_2, \ldots d_n \,\big\}$ (если это не вызывает неоднозначности, обычно пишем просто $\Delta$). Для этого алфавита рассматривается морфизм вида $h_A : \Delta_A^* \to \Sigma^*$, определенный следующим образом:

$$h_A(d_1) = u_1\,,\ h_A(d_2) = u_2\,,\ \ldots\,,\ h_A(d_n) = u_n\,.$$

При этом мы, как обычно, для каждого слова $d_1 d_2 \ldots d_n \in \Delta^*$ предполагается, что

$$h(d_1 d_2 \ldots d_n) = h(d_1)\, h(d_2) \ldots h(d_n).$$

Однако, как может быть понятно из введения, предмет настоящей статьи связан (частично) с более важной и более сложной проблемой (т. е. более сложной, чем построение морфизма), а именно — с проблемой построения *инверсного* морфизма. Отметим, что в [8–10] приведено только начало рассмотрения этой проблемы.

Сформулируем его определение более подробно. Во-первых, дадим такое естественное определение. Для данного конечного языка $A$, морфизма $h_A$ и некоторого слова $u \in \Sigma^*$ рассмотрим *язык*

$$h_A^{-1}(u) = \left\{ u_\Delta \in \Delta^* \,\middle|\, h_A(u_\Delta) = u \right\};$$

специально подчеркнем, что этот объект представляет собой множество (а не единственный элемент), причем, возможно, $\varnothing$. Такую же конструкцию можно рассмотреть для некоторого языка (вместо слова $u$; пусть это язык $B$), нас будут интересовать только конечные языки. А именно,

$$h_A^{-1}(B) = \bigcup_{u \in B} h_A^{-1}(u).$$

Теперь рассмотрим другое определение, практически не связанное с предыдущим. Причем, в отличие от других определений настоящей статьи, алфавит $\Delta$ в нем используется в качестве главного. Множество максимальных префиксных кодов (как множество языков) над этим алфавитом будет обозначаться $\mathfrak{mp}(\Delta)$.

(Не будем давать подробного определения максимального префиксного кода. Вполне достаточно сказать, что это — код, который является максимальным и префиксным, причем все эти понятия присутствуют в «обычных студенческих курсах», [22] и мн. др. То есть можно сказать, что *само название уже содержит определение.*)

Множество языков, каждый из которых содержит некоторый максимальный префиксный код в качестве подмножества (возможно, несобственного), будет обозначаться записью $\mathfrak{mp}^+(\Delta)$. Таким образом, $\mathfrak{mp}(\Delta) \subset \mathfrak{mp}^+(\Delta)$, и, конечно же, для любого алфавита такое включение является собственным. Мы будем называть каждый из этих языков *расширенным максимальным префиксным кодом.*

Вернемся к рассмотрению главного алфавита $\Sigma$, а также морфизмов вида $h_A : \Delta_A^* \to \Sigma^*$. Пусть у нас будет какой-нибудь язык $A_\Delta \in \mathfrak{mp}(\Delta)$; тогда считаем, что выполняется следующее условие: $h_A(A_\Delta) \in \mathfrak{mp}(A)$; именно так определяется множество языков $\mathfrak{mp}(A)$ над алфавитом $\Sigma$. Аналогично, для некоторого языка $A_\Delta \in \mathfrak{mp}^+(\Delta)$ считаем, что $h_A(A_\Delta) \in \mathfrak{mp}^+(A)$. Следовательно, выполняются такие альтернативные варианты определений:

- $\mathfrak{mp}^+(\Delta)$ — это множество языков, каждый из которых содержит некоторый максимальный префиксный код над $\Delta$ в качестве подмножества;
- $\mathfrak{mp}^+(\Delta)$ — это множество языков, каждый из которых является $A$-морфизмом некоторого языка множества $\mathfrak{mp}^+(\Delta)$; т. е. каждый из таких языков является специальным морфизмом некоторого расширенного максимального префиксного кода.

## 2. Бесконечные итерационные деревья

В этом разделе рассматриваются *бесконечные деревья специального вида*, возникающие в различных прикладных задачах теории формальных языков. Отличительной особенностью этих деревьев является то, что все их ребра помечены буквами заданного алфавита, вершины разделены на некоторые классы эквивалентности, а бесконечные поддеревья, соответствующие вершинам одного и того же класса, эквивалентны. Число полученных классов эквивалентности обычно (но не всегда) конечно.

Бесконечные итерационные деревья обсуждались в нескольких процитированных выше предыдущих статьях; они, как уже было отмечено, возникают во многих моделях теории формальных языков. Среди очень большого числа возможных примеров отметим так называемые ограниченно-детерминированные функции, которые являются предметом изучения студентов-математиков уже на первом курсе (см. уже процитированный выше учебник [22] и др.); часто это — «первое знакомство» студентов с подобными объектами. Однако, несмотря на это, у автора нет каких-либо ссылок на публикации, где были бы введены обобщающие определения для таких объектов. Мы не будем делать этого и в данной статье (т.е. не будем давать строгих формальных определений), а ограничимся только кратким их описанием.

В предыдущих публикациях (см. [3–6] и др.) рассматривались несколько вариантов таких бесконечных деревьев — здесь приводится один из этих вариантов, имеющий отношение к предмету статьи, т. е. к специальной гипотезе теории формальных языков и к некоторым различным эквивалентным вариантам ее описания, а также к ее применению. Начинаем изложение с достаточно простого описания — которое легко можно довести до строгого определения; в таком описании определяется бесконечное упорядоченное корневое дерево.

В его простейшем варианте рассматривается бесконечное число вершин, помеченных всеми возможными словами над рассматриваемым алфавитом $\Sigma$ (т. е. всеми возможными словами множества $\Sigma^*$); для каждого слова $u \in \Sigma^*$ и каждой буквы $a \in \Sigma$ слово $u$ соединено ребром (иногда обозначаемым $a$) со словом $ua$. Ясно, что в этом случае корнем дерева является пустое слово $\varepsilon$, и для каждой из других вершин $u$ возможно представление вида $u = va$ (где $u, v \in \Sigma^*$, $a \in \Sigma$), и поэтому для каждого такого $u$ также существует соответствующая дуга из $v$. Иногда будем рассматривать ориентированное (а не корневое) дерево, понятно, что разницы нет; обычно тип дерева (один из этих двух) выбирается просто в зависимости от того, что будет «более удачно» выглядеть на рисунках.

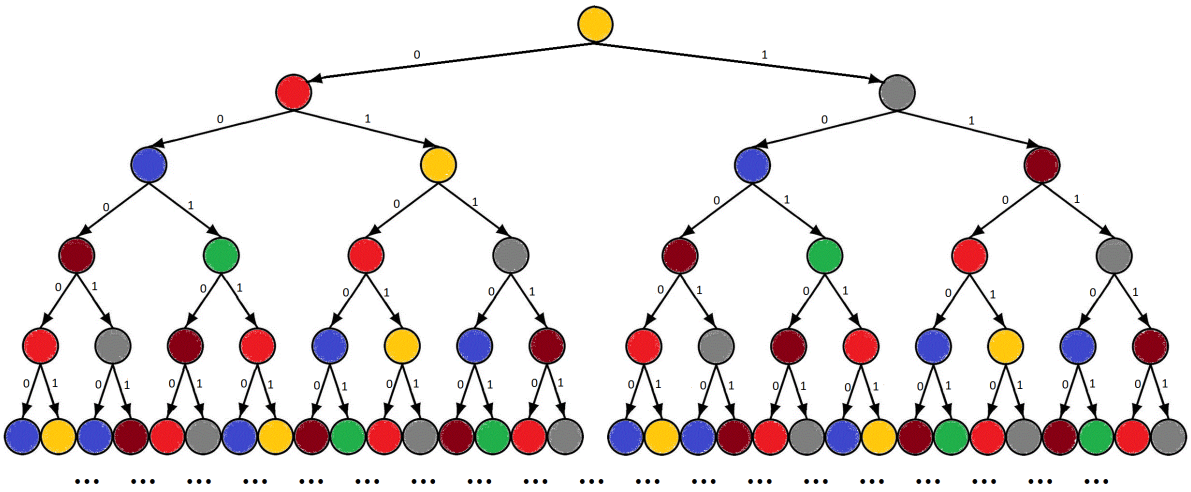


**Рис. 1.** Пример *абстрактного* бесконечного итерационного дерева морфизма для алфавита $\Delta = \{0, 1\}$ (показаны уровни от 0-го до 5-го)

Однако из всего сформулированного выше еще нельзя получить полное определение бесконечного итерационного дерева; для такого полного определения необходимо добавить, что все вершины дерева каким-то образом разделены на некоторые классы эквивалентности, и бесконечные (под)деревья с корнями, принадлежащими одному и тому же классу эквивалентности, эквивалентны. (Эквивалентность (под)деревьев определяется естествен-

ным образом.) См. рис. 1, на приведенном на нем примере разными цветами показаны вершины, принадлежащие разным классам эквивалентности. Отметим также, что условие, сформулированное выше в этом абзаце, эквивалентно другому, более простому: достаточно потребовать, чтобы для каждого ребра, принадлежащего одному и тому же классу эквивалентности, и для всех букв $a \in \Sigma$ ребра, выходящие из этих вершин, также вели к вершинам одного и того же класса эквивалентности (вообще говоря, к вершинам какого-то нового класса).

Заранее отметим, что во всех рассматриваемых примерах количество классов эквивалентности будет конечным. Немного более подробные комментарии по этому поводу будут приведены в следующем разделе. На рис. 1 такие классы эквивалентности расположены более или менее произвольно, хотя важно отметить, что все свойства и условия, сформулированные выше, выполняются.

Далее рассмотрим пример алфавита из 2 букв (будем алфавит обозначать $\{a, b\}$ вместо приведенного на рисунке $\{0, 1\}$). Конкретный пример также может быть получен следующим образом (будем условно называть такие деревья деревьями простейшего уровня). Во-первых, начинаем с дерева, уже рассмотренного на рис. 1, у которого, однако, вершины еще не выделены цветами. Для дальнейших построений задаем язык (пусть это $A = \{ a, ab, bb \}$), после чего для этого языка среди вершин дерева отмечаем те, которые соответствуют словам языка-итерации $A^*$. Более того, на рис. 2 используются следующие цвета для выделения слов:

- красный — для тех слов языка $A^*$, для которых — если их представить именно в виде итераций слов языка $A$, — последним словом может быть $a$;
- аналогично синий — для возможного последнего слова $ab$;
- и зеленый — для возможного последнего слова $bb$.

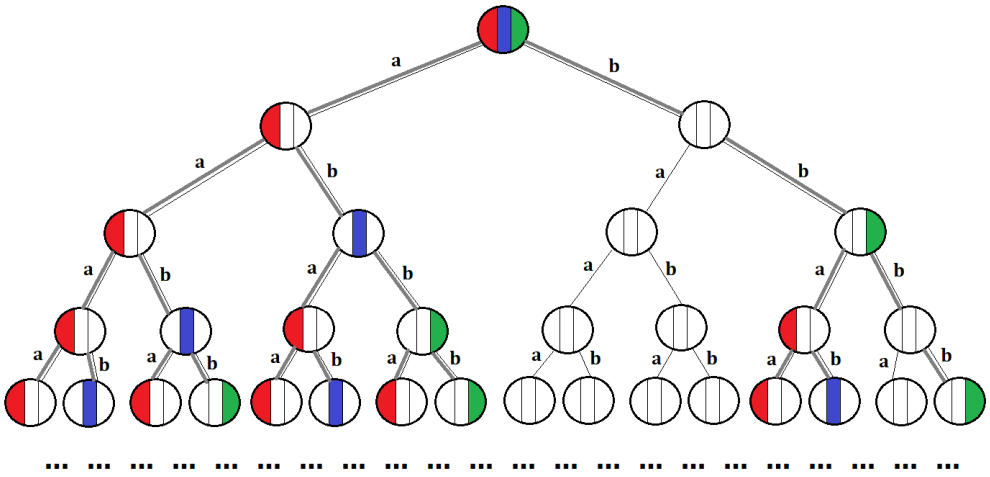


**Рис. 2.** Пример *конкретного* бесконечного итерационного дерева морфизма для алфавита $\Sigma = \{a, b\}$ и языка $A = \{ a, ab, bb \}$ (показаны уровни от 0-го до 4-го)

Отметим, что в нашем примере рассматриваемый язык $A$ является (однозначным) кодом, поэтому все выделенные слова — за исключением пустого слова $\varepsilon$ — выделены только одним цветом; однако в общем случае такое выделение одного слова только одним цветом, вообще говоря, необязательно.

Итоговое бесконечное итерационное дерево показано на рис. 2.

Таким образом на дереве отмечены все вершины языка $A^*$. Однако, вообще говоря, при таком способе отметки не все вершины бесконечного дерева попадают в множество отмеченных: на рисунке выше они не отмечены ни одним из 3 цветов. Но, несмотря на это, можно считать, что последний факт согласуется с определением бесконечного итерационного дерева; более того, «подгонка» описания определения к набору неотмеченных вершин может быть выполнена любым из рассмотренных далее двух способов. Оба способа проиллюстрированы на рис. 3, обе части рисунка показывают изменения, внесенные в правую часть исходного бесконечного дерева; при этом внесенные изменения выделены фиолетовым цветом.

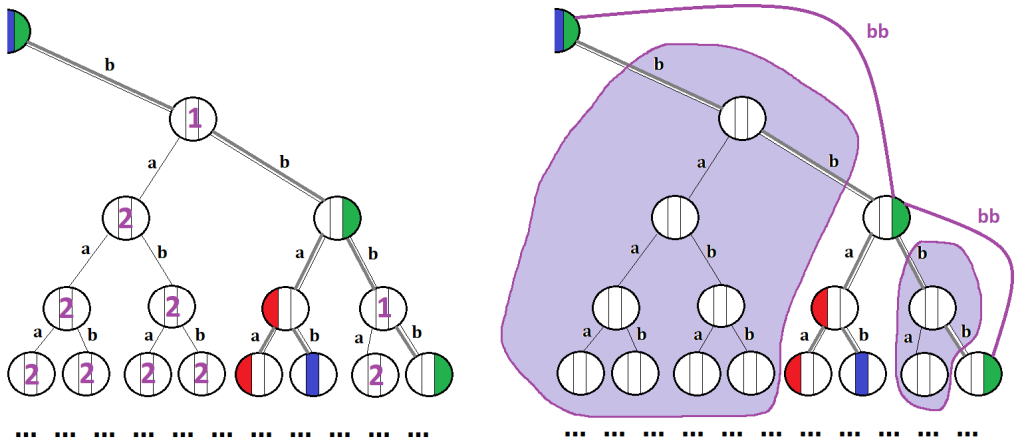



**Рис. 3.** Два метода, модифицирующих правое поддерево предыдущего дерева. В обоих случаях получается конечное число таких классов эквивалентности

- В первом методе (показанном в левой части рис. 3) мы добавляем новые классы эквивалентности для немаркированных вершин. Названия этих классов отмечены фиолетовыми цифрами 1 и 2. Можно строго доказать, что такое добавление новых классов эквивалентности для немаркированных вершин всегда возможно — после чего получается дерево, полностью удовлетворяющее приведенному выше описанию. Но строгих доказательств приводить не будем: это выходит за рамки настоящей статьи.
- А второй метод состоит:
  - во-первых, в том, что разрешается помечать ребра дерева не буквами алфавита $\Sigma$, а словами из $\Sigma^*$ (при этом уже существующие буквы рассматриваются как соответствующие однобуквенные слова),
  - и, во-вторых, в удалении ранее немаркированных вершин дерева и одновременной замене путей, проходящих через них, ребрами, помеченными необходимыми словами (т. е. сочетаниями удаляемых букв).

  Такая замена (исключение вершин и старых пометок, а также добавление пометок новых) показана в правой части рис. 3, причем также фиолетовым цветом: удаляемые объекты (вершины и ребра) отмечены фиолетовыми «пятнами» (их показано 2), также легко определить и добавляемые объекты.

Однако для материала настоящей статьи все приведенные варианты бесконечных итерационных деревьев могут рассматриваться только как вспомогательные. Для статьи более важными являются более сложные деревья, построенные в соответствии с несколько более

сложным алгоритмом, который в то же время удовлетворяет приведенному выше описанию-определению. В этом разделе такие деревья рассматриваются очень кратко (более подробно — в следующих разделах), однако при этом приводится начало существенно более сложного примера — который будет далее рассматриваться для нескольких разных целей. Для подобных примеров нам нужны два непустых конечных языка (т. е. упорядоченная пара языков) над заданным алфавитом $\Sigma$ (а не только один язык, как в примерах, рассматривавшихся ранее); в предыдущих публикациях такие два языка почти всегда обозначались как A (первый язык) и B (второй). Вершины такого дерева соответствуют не словам алфавита $\Sigma$ (в предыдущих примерах $\varepsilon$ соответствует слову 0-го уровня, все 1-буквенные слова соответствуют словам 1-го уровня и так далее), а словам нового, специально введенного алфавита $\Delta$; значение этого нового алфавита такое же, как и при введении алфавита того же название «для организации морфизмов». (В нашей ситуации это A-морфизм; его определение см. в разделе 1, а его использование — далее, например в разделе 9.)

При таком способе определения бесконечного дерева предполагается, что пометкой каждой его вершины является подмножество множества префиксов языка B. О методе выбора такой разметки, а также о разметке ребер этого дерева см. ниже. Отметим лишь, что «выходящие» ребра каждого куста дерева желательно обозначать числами от 0 до 3, и эти ребра можно считать соответствующими A-морфизму из алфавита $\Delta = \{0, 1, 2, 3\}$ в алфавит $\Sigma = \{a, b\}$.

Итак, в этом разделе приводится только начало примера (см. рис. 4); пример полностью (т. е. описание всего бесконечного итерационного дерева, причем не только одно описание, но и подробные варианты применения) будет рассматриваться в дальнейших частях статьи – применительно к нескольким разным дискретным моделям.

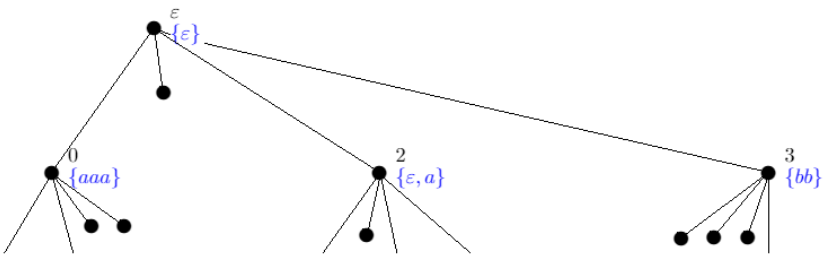



**Рис. 4.** Начало построения бесконечного итерационного дерева морфизма для пары заданных языков $A = \{aaa, aabba, abba, bb\}$ и $B = \{aaaa, abb, abba, bbb\}$

Как уже было отмечено, само *применение* бесконечных итерационных деревьев для рассматриваемых в статье задач, а также сведение этих деревьев к конечным автоматам будет рассмотрено в частях II–IV настоящей статьи.

*Работа частично поддержана грантом научной программы китайских университетов "Higher Education Stability Support Program" (раздел "Shenzhen 2022 — Science, Technology and Innovation Commission of Shenzhen Municipality").*

## Литература

1. Melnikov B.F., Korabelshchikova S.Yu., Dolgov V.N. On the task of extracting the root from the language // International Journal of Open Information Technologies. 2019. Vol. 7, no. 3. P. 1–6.

2. Melnikov B.F., Melnikova A.A. Some more on ω-finite automata and ω-regular languages. Part I: The main definitions and properties // International Journal of Open Information Technologies. 2020. Vol. 8, no. 8. P. 1–7.

3. Мельников Б.Ф., Мельникова А.А. Бесконечные деревья в алгоритме проверки условия эквивалентности итераций конечных языков. Часть I // International Journal of Open Information Technologies. 2021. Vol. 9, no. 4. P. 1–11.

4. Мельников Б.Ф., Мельникова А.А. Бесконечные деревья в алгоритме проверки условия эквивалентности итераций конечных языков. Часть II // International Journal of Open Information Technologies. 2021. Vol. 9, no. 5. P. 1–11.

5. Мельников Б.Ф. Варианты конечных автоматов, соответствующих бесконечным итерационным деревьям морфизмов. Часть I // International Journal of Open Information Technologies. 2021. Vol. 9, no. 7. P. 5–13.

6. Мельников Б.Ф. Варианты конечных автоматов, соответствующих бесконечным итерационным деревьям морфизмов. Часть II // International Journal of Open Information Technologies. 2021. Vol. 9, no. 10. P. 1–8.

7. Мельников Б.Ф., Мельникова А.А. Полиномиальный алгоритм построения конечного автомата для проверки равенства бесконечных итераций двух конечных языков // International Journal of Open Information Technologies. 2021. Vol. 9, no. 11. P. 1–10.

8. Мельников Б.Ф. Полурешетки подмножеств потенциальных корней в задачах теории формальных языков. Часть I. Извлечение корня из языка // International Journal of Open Information Technologies. 2022. Vol. 10, no. 4. P. 1–9.

9. Мельников Б.Ф. Полурешетки подмножеств потенциальных корней в задачах теории формальных языков. Часть II. Построение инверсного морфизма // International Journal of Open Information Technologies. 2022. Vol. 10, no. 5. P. 1–8.

10. Мельников Б.Ф. Полурешетки подмножеств потенциальных корней в задачах теории формальных языков. Часть III. Условие существования решетки // International Journal of Open Information Technologies. 2022. Vol. 10, no. 7. P. 1–9.

11. Мельников Б.Ф. Лепестковые конечные автоматы: основные определения, примеры и их связь с полными автоматами. Часть I // International Journal of Open Information Technologies. 2022. Vol. 10, no. 9. P. 1–11.

12. Мельников Б.Ф. Лепестковые конечные автоматы: основные определения, примеры и их связь с полными автоматами. Часть II // International Journal of Open Information Technologies. 2022. Vol. 10, no. 10. P. 1–10.

13. Мельников Б.Ф. Регулярные языки и недетерминированные конечные автоматы (монография). М.: Изд-во Российского государственного социального университета, 2018. 179 с.

14. Melnikov B.F. The equality condition for infinite catenations of two sets of finite words // International Journal of of Foundations of Computer Science. 1993. Vol. 4, no. 3. P. 267–273.

15. Мельников Б.Ф. Описание специальных подмоноидов глобального надмоноида свободного моноида // Известия высших учебных заведений. Математика. 2004. № 3. С. 46–56.

16. Алексеева А.Г., Мельников Б.Ф. Итерации конечных и бесконечных языков и недетерминированные конечные автоматы // Вектор науки Тольяттинского государственного университета. 2011. № 3 (17). С. 30–33.

17. Hopcroft R., Motwani R., Ullman J. Introduction to Automata Theory, Languages, and Computation. Massachusetts: Addison-Wesley Publishing Company Reading, 2006. 530 p.

18. Hromkovič J. Theoretical Computer Science. Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography. Berlin: Springer, 2003. 323 p.

19. Melnikov B.F. Once more on the edge-minimization of nondeterministic finite automata and the connected problems // Fundamenta Informaticae. 2010. Vol. 104, no. 3. P. 267–283.

20. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Том 1. М.: Мир, 1978. 617 с.

21. Melnikov B.F. The complete finite automaton // International Journal of Open Information Technologies. 2017. Vol. 5, no. 10. P. 9–17.

22. Яблонский С.В. Введение в дискретную математику. М.: Высшая школа, 2003. 384 с.

Мельников Борис Феликсович, д.ф.-м.н., профессор, факультет Вычислительной математики и кибернетики, Совместный университет МГУ–ППИ в Шэньчжэне (Шэньчжэнь, Китай)

**DOI: 10.14529/cmse230305**

# ON A HYPOTHESIS OF THE THEORY OF FORMAL LANGUAGES. PART I

**© 2023 B.F. Melnikov**

*Shenzhen MSU – BIT University (No. 1, International University Park Road, Dayun New Town, Longgang District, Shenzhen, Guangdong Province, 518172, China)*

*E-mail: bormel@mail.ru*

Received: 20.06.2023

The main subject of the article is the consideration of problems arising in the study of the necessary conditions for the equality of infinite iterations of finite languages. In previous publications, the author considered examples of the application of a special binary equivalence relation corresponding to this equality in a set of finite languages, and considered both examples describing the necessary conditions for its implementation and examples of its use. Further reducing the problem under consideration is applied to one of such necessary conditions: to finite automata and to infinite iterative trees. In addition, the article presents several variants of the formulation of an important hypothesis on the set of finite languages, its study also provides other options for reducing the problem under consideration to special problems of studying nondeterministic finite automata. At the same time, if the formulated hypothesis is fulfilled, some of these problems are solved in polynomial time, and some are not; with the continuation of work on this topic, the last fact may make it possible to reformulate the problem P=NP in the form of a special problem of the theory of formal languages.

*Keywords: formal languages, iterations of languages, binary relations, morphisms, inverse morphisms, algorithms, polynomial algorithms.*

## FOR CITATION

Melnikov B.F. On a Hypothesis of the Theory of Formal Languages. Part I. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 3. P. 72–86. (in Russian) DOI: 10.14529/cmse230305.

## References


1. Melnikov B.F., Korabelshchikova S.Yu., Dolgov V.N. On the task of extracting the root from the language. International Journal of Open Information Technologies. 2019. Vol. 7, no. 3. P. 1–6.
2. Melnikov B.F., Melnikova A.A. Some more on ω-finite automata and ω-regular languages. Part I: The main definitions and properties. International Journal of Open Information Technologies. 2020. Vol. 8, no. 8. P. 1–7.
3. Melnikov B.F., Melnikova A.A. Infinite trees in the algorithm for checking the equivalence condition of iterations of finite languages. Part I. International Journal of Open Information Technologies. 2021. Vol. 9, no. 4. P. 1–11. (in Russian).
4. Melnikov B.F., Melnikova A.A. Infinite trees in the algorithm for checking the equivalence condition of iterations of finite languages. Part II. International Journal of Open Information Technologies. 2021. Vol. 9, no. 5. P. 1–11. (in Russian).
5. Melnikov B.F. Variants of finite automata corresponding to infinite iterative morphism trees. Part I. International Journal of Open Information Technologies. 2021. Vol. 9, no. 7. P. 5–13. (in Russian).
6. Melnikov B.F. Variants of finite automata corresponding to infinite iterative morphism trees. Part II. International Journal of Open Information Technologies. 2021. Vol. 9, no. 10. P. 1–8. (in Russian).
7. Melnikov B.F., Melnikova A.A. A polynomial algorithm for constructing a finite automaton to check the equality of infinite iterations of two finite languages. International Journal of Open Information Technologies. 2021. Vol. 9, no. 11. P. 1–10. (in Russian).
8. Melnikov B.F. Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part I. Extracting the root from the language. International Journal of Open Information Technologies. 2022. Vol. 10, no. 4. P. 1–9. (in Russian).
9. Melnikov B.F. Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part II. Constructing an inverse morphism. International Journal of Open Information Technologies. 2022. Vol. 10, no. 5. P. 1–8. (in Russian).
10. Melnikov B.F. Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part III. The condition for the existence of a lattice. International Journal of Open Information Technologies. 2022. Vol. 10, no. 7. P. 1–9. (in Russian).
11. Melnikov B.F. Petal finite automata: basic definitions, examples and their relation to complete automata. Part I. International Journal of Open Information Technologies. 2022. Vol. 10, no. 9. P. 1–11. (in Russian).
12. Melnikov B.F. Petal finite automata: basic definitions, examples and their relation to complete automata. Part II. International Journal of Open Information Technologies. 2022. Vol. 10, no. 10. P. 1–10. (in Russian).
13. Melnikov B.F. Regular languages and nondeterministic finite automata (monograph). Moscow: Russian State Social University Ed., 2018. (in Russian). 179 p.
14. Melnikov B.F. The equality condition for infinite catenations of two sets of finite words. International Journal of of Foundations of Computer Science. 1993. Vol. 4, no. 3. P. 267–273. (in Russian).

15. Melnikov B.F. Description of special submonoids of the global supermonoid of the free monoid. News of higher educational institutions. Mathematics. 2004. No. 3. P. 46–56. (in Russian).

16. Alekseeva A.G., Melnikov B.F. Iterations of finite and infinite languages and nondeterministic finite automata. Vektor nauki of Togliatti State University. 2011. No. 3 (17). P. 30–33. (in Russian).

17. Hopcroft R., Motwani R., Ullman J. Introduction to Automata Theory, Languages, and Computation. Massachusetts: Addison-Wesley Publishing Company Reading, 2006. 530 p.

18. Hromkovič J. Theoretical Computer Science. Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography. Berlin: Springer, 2003. 323 p.

19. Melnikov B.F. Once more on the edge-minimization of nondeterministic finite automata and the connected problems. Fundamenta Informaticae. 2010. Vol. 104, no. 3. P. 267–283.

20. Aho A., Ullman J. The Theory of Parsing, Translation, and Compiling. Vol. 1: Parsing NJ: Prentice Hall, 1972. 560 p.

21. Melnikov B.F. The complete finite automaton. International Journal of Open Information Technologies. 2017. Vol. 5, no. 10. P. 9–17.

22. Yablonskiy S.V. Introduction to Discrete Mathematics. Moscow: Vysshaya shkola, 2003. 384 p. (in Russian).

## СВЕДЕНИЯ ОБ ИЗДАНИИ

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

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

*Главный редактор — Л.Б. Соколинский.*

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

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

*Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по научным специальностям и соответствующим им отраслям науки: 1.2.3 – Теоретическая информатика, кибернетика (физико-математические науки), 2.3.5 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки).*

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

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

## ПРАВИЛА ДЛЯ АВТОРОВ

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

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, кафедра СП,
   зам. главного редактора Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**