# ВЕСТНИК
## НОВОСИБИРСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

Научный журнал
Основан в ноябре 1999 года

Серия: Информационные технологии 2025. Том 23, № 2


## СОДЕРЖАНИЕ




ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2025. Том 23, № 2
Vestnik NSU. Series: Information Technologies, 2025, vol. 23, no. 2

# V E S T N I K
## NOVOSIBIRSK STATE UNIVERSITY

Scientific Journal
Since 1999, November
In Russian

Series: Information Technologies 2025. Volume 23, № 2


## CONTENTS




ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2025. Том 23, № 2
Vestnik NSU. Series: Information Technologies, 2025, vol. 23, no. 2

*The journal is published quarterly in Russian since 1999*
*by Novosibirsk State University Press*

*The address for correspondence*
*Faculty of Information Technologies, Novosibirsk State University*
*1 Pirogov Street, Novosibirsk, 630090, Russia*
*Tel.* +7 (383) 363 42 46

*E-mail address:* inftech@vestnik.nsu.ru
*On-line version:* http://elibrary.ru

ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2025. Том 23, № 2
Vestnik NSU. Series: Information Technologies, 2025, vol. 23, no. 2

Научная статья

УДК 004.021, 004.032.24, 004.051, 004.272
DOI 10.25205/1818-7900-2025-23-2-5-17

# Эффективная реализация алгоритмов сортировки с помощью концепции *Q*-детерминанта

**Валентина Николаевна Алеева**
**Егор Константинович Кузнецов**

Южно-Уральский государственный университет
Челябинск, Россия

allevavn@susu.ru, https//orcid.org/0000-0002-4717-0045
i-irbis-i@mail.ru, https//orcid.org/0009-0009-1340-0269

*Аннотация*

Статья содержит исследование возможности эффективной реализации алгоритмов сортировки с помощью концепции *Q*-детерминанта. Для проведения исследования используются алгоритмы шейкерной сортировки, сортировки Шелла, быстрой сортировки и четно-нечетной сортировки слиянием Бэтчера. Для этих алгоритмов получены представления в форме *Q*-детерминантов для сортировки массива, состоящего из небольшого количества элементов. Проведен анализ структуры полученных представлений. На основе результатов анализа описано представление алгоритма сортировки в форме *Q*-детерминанта для общего случая. Рассмотрено применение для алгоритмов сортировки метода проектирования эффективных программ, использующего концепцию *Q*-детерминанта. Применение метода апробировано с помощью разработки эффективных программ, реализующих алгоритм сортировки Шелла на общей и распределенной памяти параллельных вычислительных систем.

*Ключевые слова*

повышение эффективности параллельных вычислений, *Q*-детерминант алгоритма, представление алгоритма в форме *Q*-детерминанта, *Q*-эффективная реализация алгоритма, ресурс параллелизма алгоритма, *Q*-эффективная программа

*Для цитирования*

*Алеева В. Н., Кузнецов Е. К.* Эффективная реализация алгоритмов сортировки с помощью концепции *Q*-детерминанта // Вестник НГУ. Серия: Информационные технологии. 2025. Т. 23, № 2. С. 5–17. DOI 10.25205/1818-7900-2025-23-2-5-17

# Effective Implementation of Sorting Algorithms Using the Concept of *Q*-determinant

**Valentina N. Aleeva, Egor K. Kuznetsov**

South Ural State University,
Chelyabinsk, Russian Federation

allevavn@susu.ru, https//orcid.org/0000-0002-4717-0045
i-irbis-i@mail.ru, https//orcid.org/0009-0009-1340-0269

*Abstract*

The present paper is devoted to the possibility of efficient implementation for sorting algorithms using the *Q*-determinant concept. We have investigated four algorithms: shaker sort, Shell sort, quicksort and Batcher's odd-even mergesort.

To sort small arrays, we obtained representations in the form of *Q*-determinants for these algorithms. Then the structures of the obtained representations were analyzed and, as a result, for the general case we have described the representations of the sorting algorithms in the form of a *Q*-determinant. Also, for sorting algorithms there was considered the application of the method of designing effective programs using the concept of *Q*-determinant. This application has been tested on shared and distributed memory of parallel computing systems by developing effective programs for the Shell sort.
Keywords
improving parallel computing efficiency, *Q*-determinant of algorithm, representation of algorithm in form of *Q*-determinant, *Q*-effective implementation of algorithm, parallelism resource of algorithm, *Q*-effective program
For citation
Aleeva V. N., Kuznetsov E. K. Effective implementation of sorting algorithms using the concept of *Q*-determinant. *Vestnik NSU. Series: Information Technologies*, 2025, vol. 23, no. 2, pp. 5–17 (in Russ.) DOI 10.25205/1818-7900-2025-23-2-5-17

## Введение

Одной из самых распространенных задач в программировании является сортировка. Для ее решения используются различные алгоритмы сортировки. Проблема эффективной реализации алгоритмов сортировки, как и алгоритмов в целом, является актуальной. Эффективная реализация алгоритмов предполагает их распараллеливание. Концепция *Q*-детерминанта – один из подходов к распараллеливанию численных алгоритмов. Она является теоретической основой данного исследования.

Результаты исследований, полученные в настоящее время 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).

Для создания эффективных программ с помощью концепции *Q*-детерминанта был специально разработан метод проектирования. Создаваемые эффективные программы называются *Q*-эффективными, а сам метод – методом проектирования *Q*-эффективных программ. В настоящее время существует коллекция алгоритмов, к которым был применен этот метод. Особенность данной статьи состоит в том, что она является первой статьей, посвященной исследованию возможности эффективной реализации алгоритмов сортировки с помощью концепции *Q*-детерминанта.

Цель данного исследования заключается в том, чтобы показать возможность эффективной реализации алгоритмов сортировки с помощью концепции *Q*-детерминанта. Для достижения цели должны быть решены следующие задачи.

1. Выявление структуры представлений в форме *Q*-детерминантов алгоритмов сортировки.
2. Описание применения метода проектирования *Q*-эффективных программ для алгоритмов сортировки.
3. Апробация метода проектирования *Q*-эффективных программ на конкретном алгоритме сортировки.

## Теоретические основы исследования

Концепция *Q*-детерминанта впервые была изложена в работе [1], а ее развитие и применение описаны в работах [2–9]. Здесь дадим краткое пояснение понятий, используемых в концепции *Q*-детерминанта, основным из которых является понятие *Q*-детерминанта алгоритма.

**Определение 1.** *Выражение* над множеством входных данных *B* алгоритма и множеством операций *Q*, используемых алгоритмом, определим как терм в стандартном смысле математической логики [10].

**Определение 2.** *Цепочкой длины n* будем называть выражение, представляющее собой результат применения некоторой ассоциативной операции из *Q* к *n* выражениям.

Пусть $N = n_1, \cdots, n_k$ – множество параметров размерности алгоритмической проблемы, решаемой алгоритмом. Тогда через $\overline{N} = \overline{n}_1, \cdots, \overline{n}_k$ обозначим кортеж, где $\overline{n}_i$ – некоторое заданное значение параметра $n_i$ для каждого $i \in \{1, \cdots, k\}$, а через $\{\overline{N}\}$ – множество всех кортежей $\overline{N}$.

Определим понятия *Q-термов*. *Q*-термы могут быть безусловными, условными и условными бесконечными.

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

*Условные Q-термы* состоят из конечного множества пар, а *условные бесконечные Q-термы* – из бесконечного множества пар безусловных *Q*-термов. Первые безусловные *Q*-термы пар принимают значения логического типа, поэтому называются *логическими Q-термами*.

**Определение 4.** Если алгоритм состоит в том, что для вычисления значения каждой выходной переменной $y_i \left(i \in \{1, ..., m\}\right)$ нужно вычислить значение соответствующего *Q*-терма $f_i \left(i \in \{1, ..., m\}\right)$, где $m$ – количество выходных переменных, то множество *Q*-термов $\{f_i\}_{i \in \{1, ..., m\}}$ называется *Q-детерминантом алгоритма*.

**Определение 5.** Система уравнений $y_i = f_i \left(i \in \{1, ..., m\}\right)$ называется *представлением алгоритма в форме Q-детерминанта*.

**Определение 6.** Процесс вычисления *Q*-термов $\{f_i\}_{i \in \{1, ..., m\}}$ называется *реализацией алгоритма*.

**Определение 7.** Реализация алгоритма называется *параллельной*, если существуют операции, которые выполняются одновременно.

**Определение 8.** Реализация алгоритма называется *Q-эффективной*, если *Q*-термы $\{f_i\}_{i \in \{1, ..., m\}}$ вычисляются одновременно, операции при их вычислении выполняются по мере готовности, при этом, если несколько операций цепочки готовы к выполнению, то они выполняются по схеме сдваивания.

**Замечание.** Определение *Q*-эффективной реализации показывает, что она полностью использует ресурс параллелизма алгоритма.

*Ресурс параллелизма алгоритма* $\alpha$ определяют *высота* $D_\alpha$ и *ширина* $P_\alpha$ *алгоритма*. $D_\alpha$ характеризует время выполнения *Q*-эффективной реализации алгоритма, а $P_\alpha$ – количество вычислителей (вычислительных ядер, процессоров) ПВС, необходимое для выполнения *Q*-эффективной реализации. Кроме того, $P_\alpha$ характеризует масштабируемость алгоритма $\alpha$. Подробно понятия высоты и ширины алгоритма рассматриваются в работах [2; 3; 7–9].

**Определение 9.** Реализация алгоритма называется *выполнимой*, если одновременно должно выполняться конечное (непустое) множество операций.

Метод проектирования *Q*-эффективных программ состоит из трех этапов:

1) построение *Q*-детерминанта алгоритма;
2) описание *Q*-эффективной реализации алгоритма;
3) разработка программы для выполнимой *Q*-эффективной реализации алгоритма.

**Определение 10.** Программу будем называть *Q-эффективной*, если она разработана с помощью данного метода.

Дадим еще одно определение *Q*-эффективной программы.

**Определение 11.** Программу будем называть *Q-эффективной*, если она выполняет *Q*-эффективную реализацию алгоритма.

Определениям 10 и 11 соответствует одно и то же множество программ. Таким образом, понятие *Q*-эффективной программы можно определять как с помощью определения 10, так и с помощью определения 11. Подробно метод проектирования *Q*-эффективных программ изложен в работах [2; 5].

## Представления алгоритмов сортировки в форме *Q*-детерминантов

Для исследования структуры представлений алгоритмов сортировки в форме *Q*-детерминантов использовались алгоритмы шейкерной сортировки, сортировки Шелла, быстрой сортировки и четно-нечетной сортировки слиянием Бэтчера.

Шейкерная сортировка [11, с. 130] является разновидностью пузырьковой сортировки. Она заключается в следующем. Элементы сортируемого массива разбиваются на левую, рабочую, где происходит движение, и правую части. Границы рабочей части массива устанавливаются в месте последнего обмена на каждой итерации. Обработка рабочей части состоит в прямом и обратном проходе. Прямой проход осуществляется слева направо и перемещает максимальный элемент из рабочей части к началу правой части. Обратный проход осуществляется справа налево и перемещает минимальный элемент из рабочей части к концу левой части.

Сортировка Шелла [11, с. 102] является усовершенствованным вариантом сортировки вставками. Идея метода заключается в сравнении разделенных на группы элементов сортируемого массива, находящихся друг от друга на некотором расстоянии. Изначально это расстояние равно $d$, например, $L/2$, где $L$ – общее число элементов массива. На первом шаге каждая группа содержит два элемента, расположенных друг от друга на расстоянии $L/2$, они сравниваются между собой и, если необходимо, меняются местами. На последующих шагах также происходят проверка и обмен, но расстояние $d$ сокращается в два раза, и количество групп, соответственно, уменьшается. На последнем шаге значение $d$ равно единице, и проход по массиву происходит в последний раз.

Алгоритмы быстрой сортировки [11, с. 134] и четно-нечетной сортировки слиянием Бэтчера [11, с. 249] часто используются на практике.

1) $A(1)A(2)A(3)A(4) * (A(1)\le A(2))\&(A(2)\le A(3))\&(A(3)\le A(4))\&(A(2)\le A(3))\&(A(1)\le A(2))\&(A(2)\le A(3))$
2) $A(1)A(2)A(4)A(3) * (A(1)\le A(2))\&(A(2)\le A(3))\&(A(3)>A(4))\&(A(2)\le A(4))\&(A(1)\le A(2))\&(A(2)\le A(4))$
3) $A(1)A(4)A(2)A(3) * (A(1)\le A(2))\&(A(2)\le A(3))\&(A(3)>A(4))\&(A(2)>A(4))\&(A(1)\le A(4))\&(A(4)\le A(2))$
4) $A(4)A(1)A(2)A(3) * (A(1)\le A(2))\&(A(2)\le A(3))\&(A(3)>A(4))\&(A(2)>A(4))\&(A(1)>A(4))\&(A(1)\le A(2))$
5) $A(1)A(3)A(2)A(4) * (A(1)\le A(2))\&(A(2)>A(3))\&(A(2)\le A(4))\&(A(3)\le A(2))\&(A(1)\le A(3))\&(A(3)\le A(2))$
6) $A(3)A(1)A(2)A(4) * (A(1)\le A(2))\&(A(2)>A(3))\&(A(2)\le A(4))\&(A(3)\le A(2))\&(A(1)>A(3))\&(A(1)\le A(2))$
7) $A(1)A(3)A(4)A(2) * (A(1)\le A(2))\&(A(2)>A(3))\&(A(2)>A(4))\&(A(3)\le A(4))\&(A(1)\le A(3))\&(A(3)\le A(4))$
8) $A(3)A(1)A(4)A(2) * (A(1)\le A(2))\&(A(2)>A(3))\&(A(2)>A(4))\&(A(3)\le A(4))\&(A(1)>A(3))\&(A(1)\le A(4))$
9) $A(3)A(4)A(1)A(2) * (A(1)\le A(2))\&(A(2)>A(3))\&(A(2)>A(4))\&(A(3)\le A(4))\&(A(1)>A(3))\&(A(1)>A(4))$
10) $A(1)A(4)A(3)A(2) * (A(1)\le A(2))\&(A(2)>A(3))\&(A(2)>A(4))\&(A(3)>A(4))\&(A(1)\le A(4))\&(A(4)\le A(3))$
11) $A(4)A(1)A(3)A(2) * (A(1)\le A(2))\&(A(2)>A(3))\&(A(2)>A(4))\&(A(3)>A(4))\&(A(1)>A(4))\&(A(1)\le A(3))$
12) $A(4)A(3)A(1)A(2) * (A(1)\le A(2))\&(A(2)>A(3))\&(A(2)>A(4))\&(A(3)>A(4))\&(A(1)>A(4))\&(A(1)>A(3))$
13) $A(2)A(1)A(3)A(4) * (A(1)>A(2))\&(A(1)\le A(3))\&(A(3)\le A(4))\&(A(1)\le A(3))\&(A(2)\le A(1))\&(A(1)\le A(3))$
14) $A(2)A(1)A(4)A(3) * (A(1)>A(2))\&(A(1)\le A(3))\&(A(3)>A(4))\&(A(1)\le A(4))\&(A(2)\le A(1))\&(A(1)\le A(4))$
15) $A(2)A(4)A(1)A(3) * (A(1)>A(2))\&(A(1)\le A(3))\&(A(3)>A(4))\&(A(1)>A(4))\&(A(2)\le A(4))\&(A(4)\le A(1))$
16) $A(4)A(2)A(1)A(3) * (A(1)>A(2))\&(A(1)\le A(3))\&(A(3)>A(4))\&(A(1)>A(4))\&(A(2)>A(4))\&(A(2)\le A(1))$
17) $A(2)A(3)A(1)A(4) * (A(1)>A(2))\&(A(1)>A(3))\&(A(1)\le A(4))\&(A(3)\le A(1))\&(A(2)\le A(3))\&(A(3)\le A(1))$
18) $A(3)A(2)A(1)A(4) * (A(1)>A(2))\&(A(1)>A(3))\&(A(1)\le A(4))\&(A(3)\le A(1))\&(A(2)>A(3))\&(A(2)\le A(1))$
19) $A(2)A(3)A(4)A(1) * (A(1)>A(2))\&(A(1)>A(3))\&(A(1)>A(4))\&(A(3)\le A(4))\&(A(2)\le A(3))\&(A(3)\le A(4))$
20) $A(3)A(2)A(4)A(1) * (A(1)>A(2))\&(A(1)>A(3))\&(A(1)>A(4))\&(A(3)\le A(4))\&(A(2)>A(3))\&(A(2)\le A(4))$
21) $A(3)A(4)A(2)A(1) * (A(1)>A(2))\&(A(1)>A(3))\&(A(1)>A(4))\&(A(3)\le A(4))\&(A(2)>A(3))\&(A(2)>A(4))$
22) $A(2)A(4)A(3)A(1) * (A(1)>A(2))\&(A(1)>A(3))\&(A(1)>A(4))\&(A(3)>A(4))\&(A(2)\le A(4))\&(A(4)\le A(3))$
23) $A(4)A(2)A(3)A(1) * (A(1)>A(2))\&(A(1)>A(3))\&(A(1)>A(4))\&(A(3)>A(4))\&(A(2)>A(4))\&(A(2)\le A(3))$
24) $A(4)A(3)A(2)A(1) * (A(1)>A(2))\&(A(1)>A(3))\&(A(1)>A(4))\&(A(3)>A(4))\&(A(2)>A(4))\&(A(2)>A(3))$

*Рис. 1.* Представление в форме *Q*-детерминанта алгоритма шейкерной сортировки для массива из четырех элементов
*Fig. 1.* Representation in the form of a *Q*-determinant of the shaker sort algorithm for a four-element array

Сначала было получено представление в форме *Q*-детерминанта алгоритма шейкерной сортировки для массива из четырех элементов, представленное на рис. 1. Для этого использовалась подробная блок-схема алгоритма шейкерной сортировки в формате JSON, изображенная на рис. 2. Описание блок-схемы алгоритма в формате JSON и алгоритм формирования представления алгоритма в форме *Q*-детерминанта по его блок-схеме приведены в работах [2; 8].


```
{"Vertices":[
        {"Id":1,"Type":0,"Content":"Start"},
        {"Id":2,"Type":4,"Content":"A(1)"},
        {"Id":3,"Type":4,"Content":"A(2)"},
        {"Id":4,"Type":4,"Content":"A(3)"},
        {"Id":5,"Type":4,"Content":"A(4)"},
        {"Id":6,"Type":2,"Content":"I(1)=1"},
        {"Id":7,"Type":2,"Content":"I(2)=2"},
        {"Id":8,"Type":2,"Content":"I(3)=3"},
        {"Id":9,"Type":2,"Content":"I(4)=4"},
        {"Id":10,"Type":3,"Content":"A(I(1))<=A(I(2))"},
        {"Id":11,"Type":2,"Content":"J(1)=I(1)"},
        {"Id":12,"Type":2,"Content":"I(1)=I(2)"},
        {"Id":13,"Type":2,"Content":"I(2)=J(1)"},
        {"Id":14,"Type":3,"Content":"A(I(2))<=A(I(3))"},
        {"Id":15,"Type":2,"Content":"J(2)=I(2)"},
        {"Id":16,"Type":2,"Content":"I(2)=I(3)"},
        {"Id":17,"Type":2,"Content":"I(3)=J(2)"},
        {"Id":18,"Type":3,"Content":"A(I(3))<=A(I(4))"},
        {"Id":19,"Type":2,"Content":"J(3)=I(3)"},
        {"Id":20,"Type":2,"Content":"I(3)=I(4)"},
        {"Id":21,"Type":2,"Content":"I(4)=J(3)"},
        {"Id":22,"Type":3,"Content":"A(I(2))<=A(I(3))"},
        {"Id":23,"Type":2,"Content":"J(2)=I(2)"},
        {"Id":24,"Type":2,"Content":"I(2)=I(3)"},
        {"Id":25,"Type":2,"Content":"I(3)=J(2)"},
        {"Id":26,"Type":3,"Content":"A(I(1))<=A(I(2))"},
        {"Id":27,"Type":2,"Content":"J(1)=I(1)"},
        {"Id":28,"Type":2,"Content":"I(1)=I(2)"},
        {"Id":29,"Type":2,"Content":"I(2)=J(1)"},
        {"Id":30,"Type":3,"Content":"A(I(2))<=A(I(3))"},
        {"Id":31,"Type":2,"Content":"J(2)=I(2)"},
        {"Id":32,"Type":2,"Content":"I(2)=I(3)"},
        {"Id":33,"Type":2,"Content":"I(3)=J(2)"},
        {"Id":34,"Type":4,"Content":"A(I(1))"},
        {"Id":35,"Type":4,"Content":"A(I(2))"},
        {"Id":36,"Type":4,"Content":"A(I(3))"},
        {"Id":37,"Type":4,"Content":"A(I(4))"},
        {"Id":38,"Type":1,"Content":"End"}],
"Edges":[
        {"From":1,"To":2,"Type":2},     {"From":2,"To":3,"Type":2},
        {"From":3,"To":4,"Type":2},     {"From":4,"To":5,"Type":2},
        {"From":5,"To":6,"Type":2},     {"From":6,"To":7,"Type":2},
        {"From":7,"To":8,"Type":2},     {"From":8,"To":9,"Type":2},
        {"From":9,"To":10,"Type":2},    {"From":10,"To":14,"Type":1},
        {"From":10,"To":11,"Type":0},   {"From":11,"To":12,"Type":2},
        {"From":12,"To":13,"Type":2},   {"From":13,"To":14,"Type":2},
        {"From":14,"To":18,"Type":1},   {"From":14,"To":15,"Type":0},
        {"From":15,"To":16,"Type":2},   {"From":16,"To":17,"Type":2},
        {"From":17,"To":18,"Type":2},   {"From":18,"To":22,"Type":1},
        {"From":18,"To":19,"Type":0},   {"From":19,"To":20,"Type":2},
        {"From":20,"To":21,"Type":2},   {"From":21,"To":22,"Type":2},
        {"From":22,"To":26,"Type":1},   {"From":22,"To":23,"Type":0},
        {"From":23,"To":24,"Type":2},   {"From":24,"To":25,"Type":2},
        {"From":25,"To":26,"Type":2},   {"From":26,"To":30,"Type":1},
        {"From":26,"To":27,"Type":0},   {"From":27,"To":28,"Type":2},
        {"From":28,"To":29,"Type":2},   {"From":29"To":30,"Type":2},
        {"From":30,"To":34,"Type":1},   {"From":30,"To":31,"Type":0},
        {"From":31,"To":32,"Type":2},   {"From":32"To":33,"Type":2},
        {"From":33,"To":34,"Type":2},   {"From":34,"To":35,"Type":2},
        {"From":35"To":36,"Type":2},    {"From":36,"To":37,"Type":2},
        {"From":37"To":38,"Type":2}]}
```


*Рис. 2.* Блок-схема алгоритма шейкерной сортировки в формате JSON для массива из четырех элементов
*Fig. 2.* Shaker sort algorithm flowchart in JSON format for a four-element array

Аналогично для массива из четырех элементов с помощью блок-схем в формате JSON были получены представления в форме $Q$-детерминантов алгоритмов сортировки Шелла, быстрой сортировки и четно-нечетной сортировки слиянием Бэтчера. Они показаны на рис. 3, 4 и 5. На всех пяти рисунках через $A(1)$, $A(2)$, $A(3)$, $A(4)$ обозначены элементы сортируемого массива.

1) A(1)A(2)A(3)A(4) * (A(1)≤A(3))&(A(2)≤A(4))&(A(1)≤A(2))&(A(2)≤A(3))&(A(3)≤A(4))
2) A(1)A(3)A(2)A(4) * (A(1)≤A(3))&(A(2)≤A(4))&(A(1)≤A(2))&(A(2)>A(3))&(A(1)≤A(3))&(A(2)≤A(4))
3) A(2)A(1)A(3)A(4) * (A(1)≤A(3))&(A(2)≤A(4))&(A(1)>A(2))&(A(1)≤A(3))&(A(3)≤A(4))
4) A(1)A(4)A(3)A(2) * (A(1)≤A(3))&(A(2)>A(4))&(A(1)≤A(4))&(A(4)≤A(3))&(A(3)≤A(2))
5) A(1)A(3)A(4)A(2) * (A(1)≤A(3))&(A(2)>A(4))&(A(1)≤A(4))&(A(4)>A(3))&(A(1)≤A(3))&(A(4)≤A(2))
6) A(4)A(1)A(3)A(2) * (A(1)≤A(3))&(A(2)>A(4))&(A(1)>A(4))&(A(1)≤A(3))&(A(3)≤A(2))
7) A(3)A(2)A(1)A(4) * (A(1)>A(3))&(A(2)≤A(4))&(A(3)≤A(2))&(A(2)≤A(1))&(A(1)≤A(4))
8) A(3)A(1)A(2)A(4) * (A(1)>A(3))&(A(2)≤A(4))&(A(3)≤A(2))&(A(2)>A(1))&(A(3)≤A(1))&(A(2)≤A(4))
9) A(2)A(3)A(1)A(4) * (A(1)>A(3))&(A(2)≤A(4))&(A(3)>A(2))&(A(3)≤A(1))&(A(1)≤A(4))
10) A(3)A(4)A(1)A(2) * (A(1)>A(3))&(A(2)>A(4))&(A(3)≤A(4))&(A(4)≤A(1))&(A(1)≤A(2))
11) A(3)A(1)A(4)A(2) * (A(1)>A(3))&(A(2)>A(4))&(A(3)≤A(4))&(A(4)>A(1))&(A(3)≤A(1))&(A(4)≤A(2))
12) A(4)A(3)A(1)A(2) * (A(1)>A(3))&(A(2)>A(4))&(A(3)>A(4))&(A(3)≤A(1))&(A(1)≤A(2))
13) A(1)A(2)A(4)A(3) * (A(1)≤A(3))&(A(2)≤A(4))&(A(1)≤A(2))&(A(2)≤A(3))&(A(3)>A(4))&(A(2)≤A(4))
14) A(2)A(1)A(4)A(3) * (A(1)≤A(3))&(A(2)≤A(4))&(A(1)>A(2))&(A(1)≤A(3))&(A(3)>A(4))&(A(1)≤A(4))
15) A(1)A(4)A(2)A(3) * (A(1)≤A(3))&(A(2)>A(4))&(A(1)≤A(4))&(A(4)≤A(3))&(A(3)>A(2))&(A(4)≤A(2))
16) A(4)A(1)A(2)A(3) * (A(1)≤A(3))&(A(2)>A(4))&(A(1)>A(4))&(A(1)≤A(3))&(A(3)>A(2))&(A(1)≤A(2))
17) A(3)A(2)A(4)A(1) * (A(1)>A(3))&(A(2)≤A(4))&(A(3)≤A(2))&(A(2)≤A(1))&(A(1)>A(4))&(A(2)≤A(4))
18) A(2)A(3)A(4)A(1) * (A(1)>A(3))&(A(2)≤A(4))&(A(3)>A(2))&(A(3)≤A(1))&(A(1)>A(4))&(A(3)≤A(4))
19) A(3)A(4)A(2)A(1) * (A(1)>A(3))&(A(2)>A(4))&(A(3)≤A(4))&(A(4)≤A(1))&(A(1)>A(2))&(A(4)≤A(2))
20) A(4)A(3)A(2)A(1) * (A(1)>A(3))&(A(2)>A(4))&(A(3)>A(4))&(A(3)≤A(1))&(A(1)>A(2))&(A(3)≤A(2))
21) A(2)A(4)A(1)A(3) * (A(1)≤A(3))&(A(2)≤A(4))&(A(1)>A(2))&(A(1)≤A(3))&(A(3)>A(4))&(A(1)>A(4))&(A(2)≤A(4))
22) A(4)A(2)A(1)A(3) * (A(1)≤A(3))&(A(2)>A(4))&(A(1)>A(4))&(A(1)≤A(3))&(A(3)>A(2))&(A(1)>A(2))&(A(4)≤A(2))
23) A(2)A(4)A(3)A(1) * (A(1)>A(3))&(A(2)≤A(4))&(A(3)>A(2))&(A(3)≤A(1))&(A(1)>A(4))&(A(3)>A(4))&(A(2)≤A(4))
24) A(4)A(2)A(3)A(1) * (A(1)>A(3))&(A(2)>A(4))&(A(3)>A(4))&(A(3)≤A(1))&(A(1)>A(2))&(A(3)>A(2))&(A(4)≤A(2))

*Рис. 3.* Представление в форме $Q$-детерминанта алгоритма сортировки Шелла для массива из четырех элементов
*Fig. 3.* Representation in the form of a $Q$-determinant of the Shell sort algorithm for a four-element array

1) A(1)A(4)A(2)A(3) * (A(1)≤A(2))&(A(1)≤A(3))&(A(1)≤A(4))&(A(2)≤A(3))&(A(2)>A(4))
2) A(1)A(3)A(2)A(4) * (A(1)≤A(2))&(A(1)≤A(3))&(A(1)≤A(4))&(A(2)>A(3))&(A(2)≤A(4))
3) A(1)A(2)A(3)A(4) * (A(1)≤A(2))&(A(1)≤A(3))&(A(1)≤A(4))&(A(2)≤A(3))&(A(2)≤A(4))&(A(3)≤A(4))
4) A(1)A(2)A(4)A(3) * (A(1)≤A(2))&(A(1)≤A(3))&(A(1)≤A(4))&(A(2)≤A(3))&(A(2)≤A(4))&(A(3)>A(4))
5) A(1)A(3)A(4)A(2) * (A(1)≤A(2))&(A(1)≤A(3))&(A(1)≤A(4))&(A(2)>A(3))&(A(2)>A(4))&(A(3)≤A(4))
6) A(1)A(4)A(3)A(2) * (A(1)≤A(2))&(A(1)≤A(3))&(A(1)≤A(4))&(A(2)>A(3))&(A(2)>A(4))&(A(3)>A(4))
7) A(4)A(1)A(2)A(3) * (A(1)≤A(2))&(A(1)≤A(3))&(A(1)>A(4))&(A(2)≤A(3))
8) A(4)A(1)A(3)A(2) * (A(1)≤A(2))&(A(1)≤A(3))&(A(1)>A(4))&(A(2)>A(3))
9) A(3)A(1)A(2)A(4) * (A(1)≤A(2))&(A(1)>A(3))&(A(1)≤A(4))&(A(2)≤A(4))
10) A(3)A(1)A(4)A(2) * (A(1)≤A(2))&(A(1)>A(3))&(A(1)≤A(4))&(A(2)>A(4))
11) A(3)A(4)A(1)A(2) * (A(1)≤A(2))&(A(1)>A(3))&(A(1)>A(4))&(A(3)≤A(4))
12) A(4)A(3)A(1)A(2) * (A(1)≤A(2))&(A(1)>A(3))&(A(1)>A(4))&(A(3)>A(4))
13) A(2)A(1)A(3)A(4) * (A(1)>A(2))&(A(1)≤A(3))&(A(1)≤A(4))&(A(3)≤A(4))
14) A(2)A(1)A(4)A(3) * (A(1)>A(2))&(A(1)≤A(3))&(A(1)≤A(4))&(A(3)>A(4))
15) A(2)A(4)A(1)A(3) * (A(1)>A(2))&(A(1)≤A(3))&(A(1)>A(4))&(A(2)≤A(4))
16) A(4)A(2)A(1)A(3) * (A(1)>A(2))&(A(1)≤A(3))&(A(1)>A(4))&(A(2)>A(4))
17) A(2)A(3)A(1)A(4) * (A(1)>A(2))&(A(1)>A(3))&(A(1)≤A(4))&(A(2)≤A(3))
18) A(3)A(2)A(1)A(4) * (A(1)>A(2))&(A(1)>A(3))&(A(1)≤A(4))&(A(2)>A(3))
19) A(4)A(2)A(3)A(1) * (A(1)>A(2))&(A(1)>A(3))&(A(1)>A(4))&(A(2)≤A(3))&(A(2)>A(4))
20) A(3)A(2)A(4)A(1) * (A(1)>A(2))&(A(1)>A(3))&(A(1)>A(4))&(A(2)>A(3))&(A(2)≤A(4))
21) A(2)A(3)A(4)A(1) * (A(1)>A(2))&(A(1)>A(3))&(A(1)>A(4))&(A(2)≤A(3))&(A(2)≤A(4))&(A(3)≤A(4))
22) A(2)A(4)A(3)A(1) * (A(1)>A(2))&(A(1)>A(3))&(A(1)>A(4))&(A(2)≤A(3))&(A(2)≤A(4))&(A(3)>A(4))
23) A(3)A(4)A(2)A(1) * (A(1)>A(2))&(A(1)>A(3))&(A(1)>A(4))&(A(2)>A(3))&(A(2)>A(4))&(A(3)≤A(4))
24) A(4)A(3)A(2)A(1) * (A(1)>A(2))&(A(1)>A(3))&(A(1)>A(4))&(A(2)>A(3))&(A(2)>A(4))&(A(3)>A(4))

*Рис. 4.* Представление в форме $Q$-детерминанта алгоритма быстрой сортировки для массива из четырех элементов
*Fig. 4.* Representation in the form of a $Q$-determinant of quicksort algorithm for a four-element array

Поясним формат представления алгоритмов сортировки в форме $Q$-детерминантов, используемый на рис. 1, 3, 4 и 5. $Q$-детерминанты состоят из четырех условных $Q$-термов длины 4! = 24, равной количеству всех перестановок элементов массива $A(1)$, $A(2)$, $A(3)$, $A(4)$.

1) A(1)A(2)A(3)A(4) * (A(1)≤A(2))&(A(3)≤A(4))&(A(1)≤A(3))&(A(2)≤A(4))&(A(2)≤A(3))
2) A(1)A(3)A(2)A(4) * (A(1)≤A(2))&(A(3)≤A(4))&(A(1)≤A(3))&(A(2)≤A(4))&(A(2)>A(3))
3) A(1)A(3)A(4)A(2) * (A(1)≤A(2))&(A(3)≤A(4))&(A(1)≤A(3))&(A(2)>A(4))&(A(4)>A(3))
4) A(3)A(1)A(2)A(4) * (A(1)≤A(2))&(A(3)≤A(4))&(A(1)>A(3))&(A(2)≤A(4))&(A(2)>A(1))
5) A(3)A(4)A(1)A(2) * (A(1)≤A(2))&(A(3)≤A(4))&(A(1)>A(3))&(A(2)>A(4))&(A(3)≤A(1))
6) A(3)A(1)A(4)A(2) * (A(1)≤A(2))&(A(3)≤A(4))&(A(1)>A(3))&(A(2)>A(4))&(A(4)>A(1))
7) A(1)A(2)A(4)A(3) * (A(1)≤A(2))&(A(3)>A(4))&(A(1)≤A(4))&(A(2)≤A(3))&(A(2)≤A(4))
8) A(1)A(4)A(2)A(3) * (A(1)≤A(2))&(A(3)>A(4))&(A(1)≤A(4))&(A(2)≤A(3))&(A(2)>A(4))
9) A(1)A(4)A(3)A(2) * (A(1)≤A(2))&(A(3)>A(4))&(A(1)≤A(4))&(A(2)>A(3))&(A(3)>A(4))
10) A(4)A(1)A(2)A(3) * (A(1)≤A(2))&(A(3)>A(4))&(A(1)>A(4))&(A(2)≤A(3))&(A(2)>A(1))
11) A(4)A(3)A(1)A(2) * (A(1)≤A(2))&(A(3)>A(4))&(A(1)>A(4))&(A(2)>A(3))&(A(3)≤A(1))
12) A(4)A(1)A(3)A(2) * (A(1)≤A(2))&(A(3)>A(4))&(A(1)>A(4))&(A(2)>A(3))&(A(3)>A(1))
13) A(2)A(1)A(3)A(4) * (A(1)>A(2))&(A(3)≤A(4))&(A(2)≤A(3))&(A(1)≤A(4))&(A(1)≤A(3))
14) A(2)A(3)A(1)A(4) * (A(1)>A(2))&(A(3)≤A(4))&(A(2)≤A(3))&(A(1)≤A(4))&(A(1)>A(3))
15) A(2)A(3)A(4)A(1) * (A(1)>A(2))&(A(3)≤A(4))&(A(2)≤A(3))&(A(1)>A(4))&(A(4)>A(3))
16) A(3)A(2)A(1)A(4) * (A(1)>A(2))&(A(3)≤A(4))&(A(2)>A(3))&(A(1)≤A(4))&(A(1)>A(2))
17) A(3)A(4)A(2)A(1) * (A(1)>A(2))&(A(3)≤A(4))&(A(2)>A(3))&(A(1)>A(4))&(A(4)≤A(2))
18) A(3)A(2)A(4)A(1) * (A(1)>A(2))&(A(3)≤A(4))&(A(2)>A(3))&(A(1)>A(4))&(A(4)>A(2))
19) A(2)A(1)A(4)A(3) * (A(1)>A(2))&(A(3)>A(4))&(A(2)≤A(4))&(A(1)≤A(3))&(A(1)≤A(4))
20) A(2)A(4)A(1)A(3) * (A(1)>A(2))&(A(3)>A(4))&(A(2)≤A(4))&(A(1)≤A(3))&(A(1)>A(4))
21) A(2)A(4)A(3)A(1) * (A(1)>A(2))&(A(3)>A(4))&(A(2)≤A(4))&(A(1)>A(3))&(A(3)>A(4))
22) A(4)A(2)A(1)A(3) * (A(1)>A(2))&(A(3)>A(4))&(A(2)>A(4))&(A(1)≤A(3))&(A(1)>A(2))
23) A(4)A(3)A(2)A(1) * (A(1)>A(2))&(A(3)>A(4))&(A(2)>A(4))&(A(1)>A(3))&(A(3)≤A(2))
24) A(4)A(2)A(3)A(1) * (A(1)>A(2))&(A(3)>A(4))&(A(2)>A(4))&(A(1)>A(3))&(A(3)>A(2))

*Рис. 5.* Представление в форме $Q$-детерминанта алгоритма четно-нечетной сортировки слиянием Бэтчера для массива из четырех элементов

*Fig. 5.* Representation in the form of a $Q$-determinant of Batcher's odd-even mergesort algorithm for a four-element array

Рисунки содержат представления в форме $Q$-детерминантов всех четырех алгоритмов не в форме, используемой в определении 5, а в компактной форме. Каждая строка компактной формы включает:

1) порядковый номер строки, который соответствует значению номера пары $j\left(j \in \{1,\ldots,4!\}\right)$ условного $Q$-терма, то есть одной перестановке элементов сортируемого массива;

2) саму перестановку $w_j^i\left(i \in \{1,2,3,4\}\right)$ элементов сортируемого массива;

3) логический $Q$-терм $u_j$, отделенный от перестановки знаком «*».

Компактная запись представления в форме $Q$-детерминанта удобна для проведения анализа структуры логических $Q$-термов. Также для удобства проведения анализа специально использована небольшая длина сортируемого массива.

На рис. 1, 3–5 хорошо видно, что структура логических $Q$-термов зависит от алгоритма сортировки. Каждый логический $Q$-терм представляет собой цепочку, являющуюся результатом применения операции конъюнкции к различным попарным сравнениям сортируемых элементов.

На основе анализа сформированных по блок-схемам представлений в форме *Q*-детерминантов для четырех алгоритмов сортировки получаем следующее описание представления алгоритма сортировки в форме *Q*-детерминанта для общего случая. Обозначим через $L$ число элементов сортируемого массива, через $A(i)\left(i \in \{1,\ldots,L\}\right)$ – начальное расположение элементов сортируемого массива, а через $C(i)\left(i \in \{1,\ldots,L\}\right)$ – элементы отсортированного массива. Тогда, в соответствии с определением 5, представление алгоритма сортировки в форме *Q*-детерминанта имеет вид $C(i) = \left\{\left(u_j, w_j^i\right)\right\}$, где $i \in \{1,\ldots,L\}$, $j \in \{1,\ldots,L!\}$, $\left\{\left(u_j, w_j^i\right)\right\}$, – условный *Q*-терм длины L!. Для любого $j \in \{1,\ldots,L!\}$ логические *Q*-термы $u_j$ одинаковы во всех условных *Q*-термах, а $w_j^i\left(i \in \{1,\ldots,L\}\right)$ представляют собой одну из возможных перестановок элементов сортируемого массива $A(i)\left(i \in \{1,\ldots,L\}\right)$. Если логический *Q*-терм $u_j$ имеет значение *true*, то отсортированный массив будет являться перестановкой $w_j^i\left(i \in \{1,\ldots,L\}\right)$.

## Оценка ресурсов параллелизма алгоритмов сортировки

Оценим ресурсы параллелизма всех четырех проанализированных алгоритмов сортировки для массива из четырех элементов. Высота и ширина алгоритмов являются функциями от параметров размерности алгоритмической проблемы, которую решает алгоритм, если параметры размерности существуют. В нашем случае алгоритмическая проблема, заключающаяся в сортировке массива из чисел, имеет один параметр размерности – длину массива. По полученным представлениям в форме *Q*-детерминантов для параметра размерности 4 легко вычислить высоту и ширину всех четырех алгоритмов.

Для оценки высоты алгоритмов сортировки при любом значении длины массива нужно вычислить количество уровней вложенности каждого из выражений, описывающих логические *Q*-термы, и найти из них максимальное значение. При определении уровней вложенности необходимо учитывать, что цепочки операций в выражениях должны вычисляться по схеме сдваивания. Вычисление показывает, что высота проанализированных алгоритмов при длине массива 4 одинакова и равна четырем.

Для оценки ширины алгоритмов сортировки при любом значении длины массива нужно вычислить количество операций на каждом из уровней вложенности всех выражений, описывающих логические *Q*-термы, и найти из них максимальное значение. Максимальное количество операций имеет первый уровень вложенности. Вычисление показывает, что при длине массива 4 ширина алгоритма шейкерной сортировки равна 144, сортировки Шелла – 140, быстрой сортировки – 116, четно-нечетной сортировки слиянием Бэтчера – 120.

Высота рассмотренных алгоритмов при длине массива 4 одинакова, поэтому примерно одинаково время выполнения реализующих их *Q*-эффективных программ с одинаковыми вычислительными инфраструктурами, т. е. условиями разработки и выполнения программ. Ширина алгоритмов разная, поэтому для выполнения *Q*-эффективных реализаций алгоритмов потребуется разное количество вычислителей ПВС. Учитывая ширину алгоритмов, можно сделать вывод, что все они масштабируемые.

## Применение метода проектирования *Q*-эффективных программ для алгоритмов сортировки

Опишем, как применить метод проектирования *Q*-эффективных программ для алгоритмов сортировки. На первом этапе метода должен быть построен *Q*-детерминант алгоритма. Это было уже сделано ранее.

На втором этапе метода нужно описать *Q*-эффективную реализацию алгоритма. В случае алгоритмов сортировки описание заключается в том, что операции должны выполнять-

ся в соответствии с алгоритмом сортировки по мере их готовности к выполнению, при этом, если несколько операций цепочки готовы к выполнению, то они должны выполняться по схеме сдваивания. При такой реализации обеспечивается одновременное вычисление всех $Q$-термов $\{f_i\}_{i\in\{1,\ldots,L\}}$, составляющих $Q$-детерминант алгоритма сортировки. Более того, обеспечивается одновременное вычисление всех логических $Q$-термов $u_j\left(j\in\{1,\ldots,L!\}\right)$ и определение элементов соответствующих им перестановок сортируемого массива $w_j^i\left(i\in\{1,\ldots,L\}\right)$. Таким образом, описанная реализация алгоритма сортировки по определению 8 является $Q$-эффективной. Конкретное описание $Q$-эффективной реализации алгоритма сортировки зависит от алгоритма.

На третьем этапе метода должна быть разработана $Q$-эффективная программа, выполняющая $Q$-эффективную реализацию алгоритма, если $Q$-эффективная реализация выполнима. По определению 9 $Q$-эффективная реализация любого алгоритма сортировки выполнима, так как одновременно должно выполняться конечное (непустое) множество операций. При разработке $Q$-эффективной программы для общей памяти нужно использовать описание $Q$-эффективной реализации алгоритма, полученное на втором этапе. При разработке $Q$-эффективной программы для распределенной памяти нужно также выполнить распределение вычислений по узлам ПВС.

Примеры применения метода проектирования $Q$-эффективных программ к другим алгоритмам описаны в работах [2; 4; 5; 7; 9].

## Эффективная реализация алгоритма сортировки Шелла

Метод проектирования $Q$-эффективных программ был применен для $Q$-эффективной реализации алгоритма сортировки Шелла на общей и распределенной памяти ПВС. Исследования проводились на суперкомпьютере «Торнадо» Южно-Уральского государственного университета. При разработке программ применялся язык программирования C++. Для общей памяти использовался один вычислительный узел, который содержит два центральных процессора Intel Xeon X5680 с частотой 3,33 GHz, каждый из которых имеет 6 ядер и поддерживает 12 потоков, оперативная память узла 24 Гб ECC DDR3 Full buffered. При разработке программ применялась технология OpenMP для управления потоками программы. Для распределенной памяти использовалось несколько вычислительных узлов. При разработке программ применялись технологии OpenMP и MPI для управления процессами программы. Обе $Q$-эффективные программы разработаны так, что все операции над массивами при достаточных вычислительных ресурсах должны выполняться по мере их готовности.

$Q$-эффективная программа для реализации алгоритма на общей памяти в начале выполнения генерирует случайный, обратный (значения элементов массива не возрастают) и частично отсортированный массивы, затем выполняет их сортировку с применением алгоритма Шелла, после чего фиксирует время выполнения алгоритма для каждого из массивов и записывает эти данные в отдельный файл.

$Q$-эффективная программа для реализации алгоритма на распределенной памяти в начале выполнения инициализирует MPI, после этого определяет ранг текущего процесса и общее число процессов, затем главный процесс генерирует случайный, обратный и частично отсортированный массивы. После этого сгенерированные массивы разделяются на подмассивы равной длины для каждого процесса программы. Затем осуществляется локальная сортировка подмассивов каждым из процессов с применением алгоритма Шелла. После локальной сортировки главный процесс осуществляет сбор и слияние подмассивов в глобальный отсортированный массив, после чего программа фиксирует время выполнения алгоритма для каждого из массивов и записывает эти данные в отдельный файл.

Разработанные $Q$-эффективные программы полностью используют ресурс параллелизма алгоритмов. Однако ресурсов вычислительной системы было недостаточно для использования

всего параллелизма при вычислительных экспериментах. В этих условиях были оценены динамические характеристики программ. Ускорение программы вычислялось по формуле

$$S_p = T_1 / T_p,$$

где $T_1$ – время выполнения программы на одном вычислительном ядре; $T_p$ – время выполнения программы на  вычислительных ядрах. Затем для вычисления эффективности программы использовалась формула

$$E_p = S_p / p,$$

где $S_p$ – ускорение программы; $p$ – количество используемых вычислительных ядер.

Под последовательной программой мы понимаем параллельную программу, выполняемую на одном ядре одного вычислительного узла. В остальных случаях во время вычислительных экспериментов с программой для общей памяти количество процессорных ядер варьировалось от 2 до 12, а с программой для распределенной памяти на каждом вычислительном узле использовались все 12 физических ядер и максимально возможное количество одновременно выполняющихся нитей, равное 24.

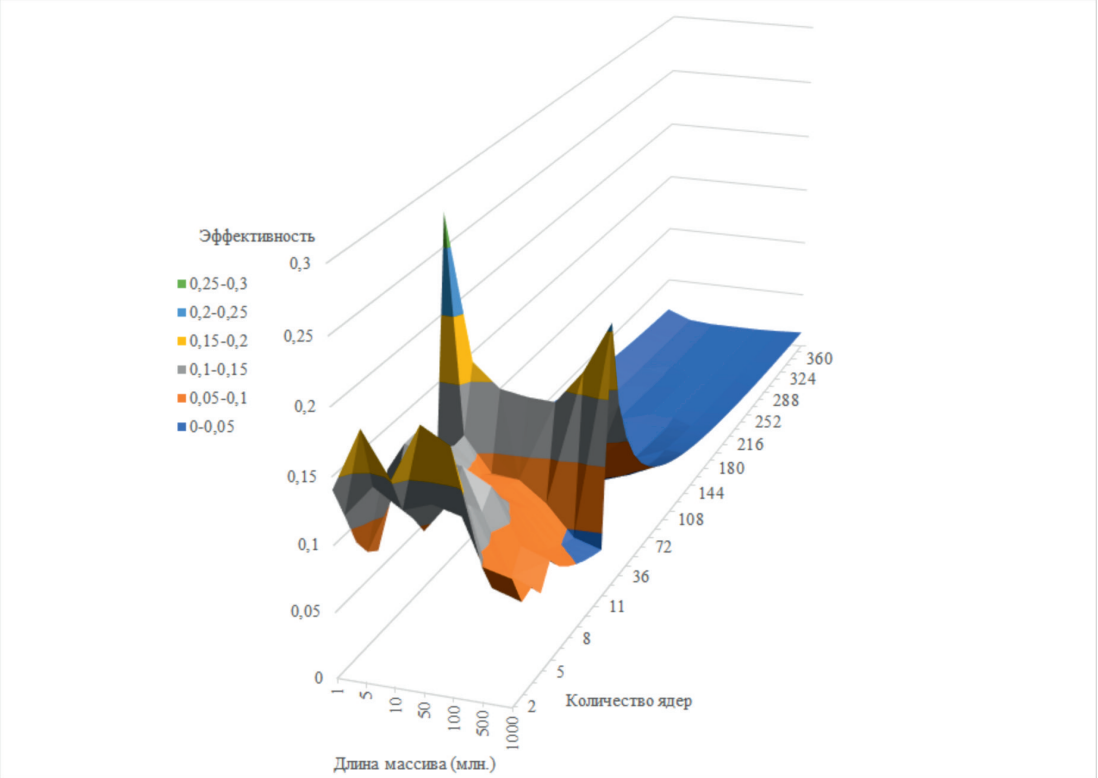


*Рис. 6.* Эффективность *Q*-эффективных программ при сортировке случайных массивов
*Fig. 6.* Efficiency of *Q*-effective programs in sorting random arrays

На рис. 6–8 представлены характеристики эффективности разработанных *Q*-эффективных программ при различном упорядочении элементов сортируемого массива. На каждом из рисунков совмещены характеристики *Q*-эффективных программ для общей и распределенной памяти.

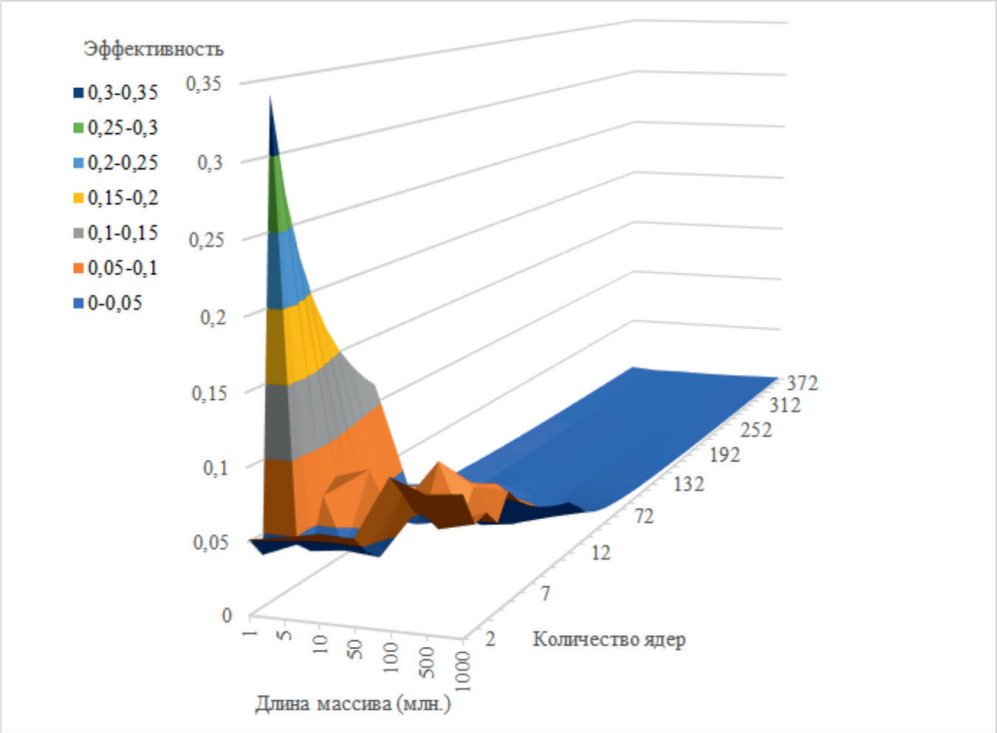


*Рис. 7.* Эффективность *Q*-эффективных программ при сортировке обратных массивов
*Fig. 7.* Efficiency of *Q*-effective programs when sorting inverse arrays

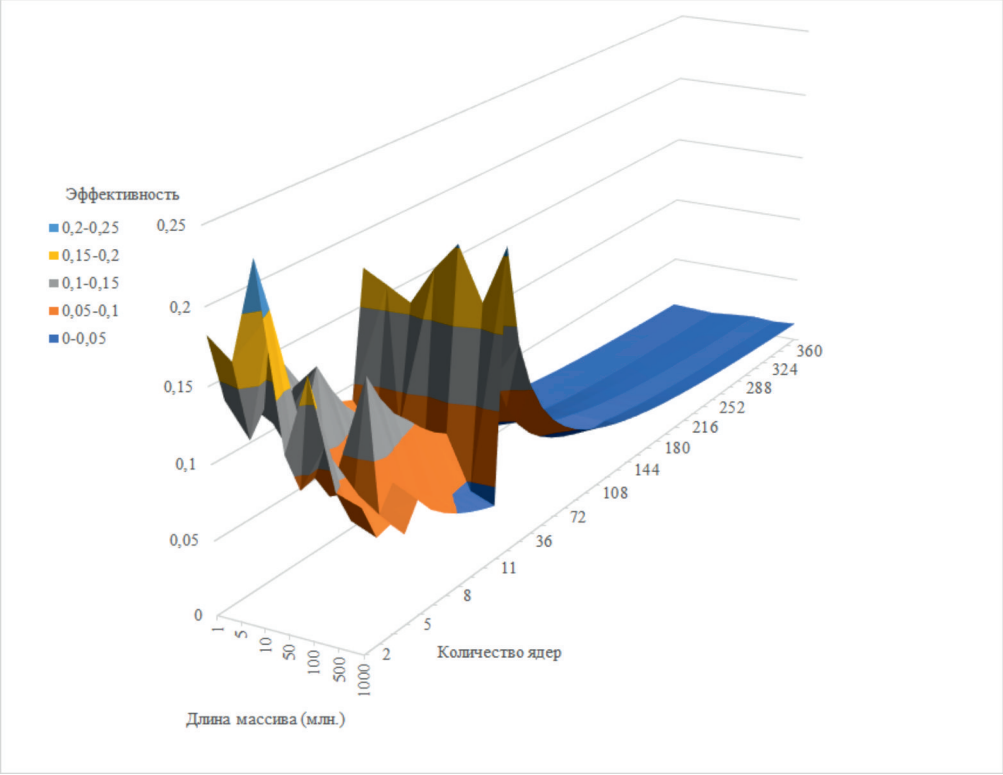


*Рис. 8.* Эффективность *Q*-эффективных программ при сортировке частично упорядоченных массивов
*Fig. 8.* Efficiency of *Q*-effective programs in sorting partially ordered arrays

## Заключение

Эта статья является первой, в которой концепция *Q*-детерминанта применяется к алгоритмам сортировки. В ней исследована возможность эффективной реализации алгоритмов сортировки с помощью концепции *Q*-детерминанта. При этом выявлена структура представлений в форме *Q*-детерминантов алгоритмов сортировки для общего случая, описано применение метода проектирования *Q*-эффективных программ для алгоритмов сортировки, выполнена апробация метода проектирования *Q*-эффективных программ на конкретном алгоритме сортировки.

## Список литературы

1. **Алеева В. Н.** Анализ параллельных численных алгоритмов. Препринт № 590. Новосибирск: ВЦ СО АН СССР, 1985. 23 с.
2. **Aleeva V., Aleev R.** Investigation and Implementation of Parallelism Resources of Numerical Algorithms // ACM Transactions on Parallel Computing. 2023. Vol. 10. No. 2. P. 1–64. DOI: 10.1145/3583755
3. **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
4. **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
5. **Aleeva V.** Designing a Parallel Programs on the Base of the Conception of *Q*-Determinant // Supercomputing. RuSCDays 2018. Communications in Computer and Information Science. 2019. Vol. 965. P. 565–577. DOI: 10.1007/978-3-030-05807-4_48
6. **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). Lecture Notes in Computer Science. 2015. Vol. 9251. P. 3–9. DOI: 10.1007/978-3-319-21909-7_1
7. **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
8. **Aleeva V., Bogatyreva E., Skleznev A. et al.** Software *Q*-system for the Research of the Resource of Numerical Algorithms Parallelism // Supercomputing. RuSCDays 2019. Communications in Computer and Information Science. 2019. Vol. 1129. P. 641–652. DOI: 10.1007/978-3-030-36592-9_52
9. **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
10. **Ершов Ю. Л., Палютин Е. А.** Математическая логика. М.: Наука, 1987. 336 с.
11. **Кнут Д.** Искусство программирования. Т. 3. Сортировка и поиск, 2-е изд. М.: Вильямс, 2018. 832 с.

## 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 Russ.)

2. **Aleeva V., Aleev R.** Investigation and Implementation of Parallelism Resources of Numerical Algorithms. *ACM Transactions on Parallel Computing*, 2023, vol. 10, no. 2, pp. 1–64. DOI: 10.1145/3583755
3. **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, pp. 66–81. (in Russ.) DOI: 10.14529/cmse210205
4. **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, pp. 56–71. (in Russ.) DOI: 10.14529/cmse210304
5. **Aleeva V.** Designing a Parallel Programs on the Base of the Conception of Q-Determinant. Supercomputing. RuSCDays 2018. *Communications in Computer and Information Science*, 2019, vol. 965, pp. 565–577. DOI: 10.1007/978-3-030-05807-4_48
6. **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). *Lecture Notes in Computer Science*, 2015, vol. 9251, pp. 3–9. DOI: 10.1007/978-3-319-21909-7_1
7. **Aleeva V. N., Aleev R. Zh.** High-Performance Computing Using Application of Q-determinant of Numerical Algorithms. In: *Proceedings – 2018 Global Smart Industry Conference, GloSIC 2018. IEEE*, 2018, 8 p. Article number 8570160. DOI: 10.1109/GloSIC.2018.8570160
8. **Aleeva V., Bogatyreva E., Skleznev A. et al.** Software Q-system for the Research of the Resource of Numerical Algorithms Parallelism. Supercomputing. RuSCDays 2019. *Communications in Computer and Information Science*, 2019, vol. 1129, pp. 641–652. DOI: 10.1007/978-3-030-36592-9_52
9. **Aleeva V. N.** Improving Parallel Computing Efficiency. In: *Proceedings – 2020 Global Smart Industry Conference, GloSIC 2020. IEEE*, 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828
10. **Ershov Yu. L., Palyutin E. A.** Mathematical Logic. Moscow, Mir publ., 1984, 303 p. (in Russ.)
11. **Knuth D.** The Art of Computer Programming. Vol. 3. Sorting and Searching, Second Edition. Moscow, LLC “I.D. Williams”, 2018, 832 p. (in Russ.)

## Информация об авторах

**Алеева Валентина Николаевна**, кандидат физико-математических наук

**Кузнецов Егор Константинович,** студент магистратуры

## Information about the Authors

**Valentina N. Aleeva,** Candidate of Science in Physics and Mathematics

**Egor K. Kuznetsov,** Master’s Degree Student

*Статья поступила в редакцию 04.03.2025;*
*одобрена после рецензирования 03.04.2025; принята к публикации 03.04.2025*

*The article was submitted 04.03.2025;*
*approved after reviewing 03.04.2025; accepted for publication 03.04.2025*

Научная статья

УДК 004.827
DOI 10.25205/1818-7900-2025-23-2-18-28

# Интеграция тесных кванторов в систему проверки согласованности экспертных оценок

**Елена Дмитриевна Малаева**

Новосибирский государственный университет
Новосибирск, Россия

e.malaeva@g.nsu.ru

*Аннотация*

В статье рассматривается интеграция тесных кванторов в систему проверки согласованности экспертных оценок, что позволяет значительно расширить возможности анализа различных ситуаций. Кванторы существования и всеобщности обеспечивают обработку более сложных логических выражений, увеличивая точность и гибкость системы. Такой подход позволяет учитывать как индивидуальные вероятности факторов, так и их комбинации, что особенно важно при решении комплексных задач, связанных с оценкой рисков и стратегическим управлением. В работе подробно описаны алгоритмы проверки согласованности, включая построение графов для визуализации взаимосвязей между экспертными оценками и методы исправления несогласованностей. Также обсуждается модульный подход к разработке программной системы, обеспечивающей автоматизацию анализа экспертных оценок. Предложенная система демонстрирует высокую универсальность и применимость в задачах управления рисками, прогнозирования и принятия решений, что делает ее важным инструментом для повышения точности прогнозов и разработки более эффективных стратегий управления.

*Ключевые слова*

экспертные оценки, субъективная вероятность, проверка согласованности, тесные кванторы

*Для цитирования*

*Малаева Е. Д.* Интеграция тесных кванторов в систему проверки согласованности экспертных оценок // Вестник НГУ. Серия: Информационные технологии. 2025. Т. 23, № 2. С. 18–28. DOI 10.25205/1818-7900-2025-23-2-18-28

# Integration of Tight Quantifiers into the System for Checking Consistency of Expert Evaluations

**Elena D. Malaeva**

Novosibirsk State University,
Novosibirsk, Russian Federation

e.malaeva@g.nsu.ru

*Abstract*

This paper discusses the integration of tight quantifiers into a system for checking the consistency of expert evaluations, significantly enhancing the system's ability to analyze various scenarios. Existential and universal quantifiers enable the processing of more complex logical expressions, increasing both accuracy and flexibility. This approach allows for consideration of both individual probabilities of factors and their combinations, which is particularly important for solving

complex problems related to risk assessment and strategic management. The paper provides a detailed description of consistency-checking algorithms, including graph-based visualization of relationships between expert evaluations and methods for resolving inconsistencies. Additionally, a modular approach to developing a software system for automating expert evaluation analysis is discussed. The proposed system demonstrates high versatility and applicability in risk management, forecasting, and decision-making tasks, making it a valuable tool for improving prediction accuracy and developing more effective management strategies.

*Keywords*

expert evaluations, subjective probability, checking consistency, tight quantifiers

*For citation*

Malaeva E. D. Integration of tight quantifiers into the system for checking consistency of expert evaluations. *Vestnik NSU. Series: Information Technologies*, 2025, vol. 23, no. 2, pp. 18–28 (in Russ.) DOI 10.25205/1818-7900-2025-23-2-18-28

## Введение

Оценка рисков занимает ключевое место в процессе принятия решений, особенно в условиях возрастающей сложности современных социально-экономических систем. Основной целью оценки рисков является идентификация, анализ и управление факторами, способными повлиять на успех или провал различных проектов и процессов. В последние десятилетия значительное внимание уделяется разработке методов оценки рисков, которые способны учитывать как количественные, так и качественные аспекты. Одним из наиболее широко применяемых инструментов является метод экспертных оценок, который используется в ситуациях, где невозможно или нецелесообразно полагаться исключительно на статистические данные. Методы экспертных оценок предоставляют возможность учитывать мнение высококвалифицированных специалистов для анализа сложных проблем. Экспертные оценки применяются в различных сферах, включая промышленность, энергетику, финансы и управление проектами. Эти методы особенно важны, когда отсутствуют достаточные статистические данные или когда требуются интуитивно-логические выводы. Эксперты привлекаются для определения ключевых факторов рисков, их вероятности и последствий, что позволяет формировать обоснованные стратегии минимизации угроз [1; 2].

Экспертные методы оценки можно разделить на индивидуальные и коллективные. Индивидуальные оценки основываются на мнении одного специалиста, тогда как коллективные включают обработку мнений группы экспертов с использованием таких техник, как метод Дельфи, мозговой штурм или анкетирование. Коллективные методы позволяют снизить субъективность, выявляя более объективные тенденции и оценки [4]. Применение экспертных методов обладает рядом преимуществ. Во-первых, они обеспечивают гибкость и возможность учитывать уникальные аспекты рассматриваемых ситуаций. Экспертные методы не ограничиваются жесткими алгоритмами или моделями. Они позволяют адаптироваться к специфике задачи, будь то промышленный проект, энергетическая система или финансовый анализ. Во-вторых, экспертные оценки позволяют анализировать не только количественные, но и качественные параметры, такие как неопределенность и многокритериальность. Однако, несмотря на свои достоинства, данные методы имеют и ограничения, связанные с субъективностью экспертов, возможными конфликтами интересов и сложностью формирования однородной экспертной группы [2; 4].

Для повышения надежности экспертных оценок необходимы стандартизированные процедуры подбора и обучения экспертов, а также четкие регламенты проведения исследований. Например, методология, представленная А. И. Орловым, включает структурированный подход к организации экспертных опросов, обработке данных и интерпретации результатов, что снижает вероятность ошибок и повышает объективность итоговых выводов [4]. В энергетике экспертные оценки используются для анализа рисков, связанных с надежностью оборудования,

устойчивостью поставок и безопасностью производства. Например, в работе Е. П. Куркиной и Д. Г. Шуваловой подчеркивается значимость разработки анкет для экспертов, позволяющих учитывать специфику отрасли и особенности проектов [1]. В финансовом секторе методы экспертных оценок применяются для анализа кредитных рисков, где объективность играет ключевую роль из-за высокой волатильности рынка [3].

Особую роль экспертные оценки играют при управлении сложными проектами, включающими множество взаимосвязанных рисков. В таких ситуациях коллективные методы, сочетающие статистический анализ с интуитивными выводами экспертов, предоставляют наиболее полное и надежное представление о возможных угрозах и путях их минимизации [3; 4]. Современные вызовы, такие как нестабильность рынков, геополитические риски и климатические изменения, требуют совершенствования существующих методов анализа рисков. Интеграция экспертных оценок в системы управления рисками позволяет учитывать сложные и многомерные зависимости, которые трудно формализовать в рамках традиционных методов. Это делает экспертные оценки незаменимым инструментом для поддержки принятия решений в условиях неопределенности [2; 3].

Таким образом, развитие и применение методов экспертных оценок является важным шагом на пути повышения эффективности управления рисками. Современные подходы, включающие использование методов нечеткой логики, позволяют значительно расширить возможности анализа и принимать более обоснованные решения в сложных ситуациях.

## 1. Обоснование и цели разработки

Рассмотрим пример, связанный с оценкой рисков для компании, планирующей запуск нового производственного объекта. Проект требует оценки нескольких критически важных факторов, таких как надежность оборудования, стабильность поставок и финансовая устойчивость компании. Для получения объективной картины компания обращается к трем экспертам: инженеру по оборудованию, логисту и финансовому аналитику, чтобы каждый из них оценил вероятность успешной реализации проекта со своей стороны.

Инженер по оборудованию оценивает вероятность успеха в 30 %, принимая во внимание возможные технические сложности и необходимость регулярного обслуживания оборудования. Логист, отвечающий за поставки, дает более высокую оценку – 40 %, ссылаясь на риски, связанные с возможными задержками и ограничениями в цепочке поставок. В то же время финансовый аналитик оценивает вероятность успешного завершения проекта в 90 %, так как компания имеет устойчивое финансовое положение и может покрыть непредвиденные расходы.

На первый взгляд, такие оценки кажутся несогласованными из-за значительных расхождений в них. Но каждый эксперт использует свои критерии: инженер фокусируется на технических аспектах, логист – на устойчивости поставок, а финансовый аналитик – на общей платежеспособности компании. Компания проводит анализ согласованности этих вероятностей. В ходе анализа выясняется, что, несмотря на различия в оценках, все эксперты согласны в одном – для успешного запуска проекта необходимо усилить контроль над поставками и повысить надежность оборудования. А финансовая устойчивость компании является важным преимуществом, которое можно использовать для реализации этих мер. Таким образом, компания принимает решение инвестировать в проект, но с учетом дополнительных мер, направленных на повышение устойчивости поставок и надежности оборудования.

Какой же из этого вывод? Использование анализа согласованности позволяет увидеть полную картину и учесть разнообразные факторы, что помогает принимать более обоснованные решения.

Ранее была разработана система для проверки согласованности оценок, предназначенная для анализа экспертных данных, выявления потенциальных противоречий и визуализации

этих несоответствий [5]. В данной программной системе обрабатывались только предложения логики предикатов без кванторов. Расширение входных данных системы за счет использования тесных кванторов значительно увеличит выразительность логической модели, что позволит более точно формулировать и анализировать утверждения о влиянии различных факторов на конечный результат. Кванторы существования и всеобщности позволяют моделировать намного более разноплановые ситуации, и это значительно расширяет возможности системы, делая ее более гибкой и способной учитывать сложные связи между условиями.

Использование кванторов, например в контексте оценки рисков, будет способствовать более точному и полному анализу, учитывая как отдельные факторы рисков, так и их комбинации. Это сделает прогнозы более надежными, так как система будет способна учитывать более широкий спектр взаимосвязанных рисков. Кванторы дадут возможность анализировать не только вероятность каждого отдельного события, но и взаимодействие между ними, что повысит точность и полезность результатов.

Таким образом, расширение системы тесными кванторами создаст более мощную, гибкую и эффективную среду для анализа сложных ситуаций, улучшая не только точность вычислений, но и качество принимаемых решений, будь то в области оценки рисков или других задачах, требующих комплексного анализа.

Исходя из этого, было принято решение об интеграции тесных кванторов в ранее разработанную систему проверки согласованности экспертных оценок. Опишем общие принципы работы ранее разработанных модулей и модуля работы с тесными кванторами.

## 2. Алгоритм проверки согласованности экспертных оценок

Алгоритм проверки согласованности экспертных оценок основывается на теоретико-модельной формализации знаний экспертов в виде нечеткой модели, которая является консервативным расширением понятия модели в классической логике предикатов [6]. Проверка согласованности экспертных знаний заключается в определении нечеткой модели, в которую эти знания вкладываются. Если такой модели не существует, то полученные знания считаются некорректными с логической точки зрения [7].

Набор экспертных оценок конечен и часто бывает неполным. Поэтому не всегда удается однозначно сформулировать нечеткую модель, которая бы формализовала эти знания. В результате возникает класс нечетких моделей, соответствующих данному набору экспертных оценок. Это приводит к проблеме неопределенности при генерации новых оценочных знаний о рассматриваемой области [8].

Существующие решения не позволяют оценивать события несколькими значениями, что является серьезным ограничением [9]. Обычно системы оценки используют одно числовое значение вероятности для каждого события, однако эксперты могут иметь разные оценки, основанные на их опыте и восприятии. Это приводит к несовместимости систем, когда одно и то же событие имеет разные вероятностные оценки, что затрудняет принятие решений. Одним из способов решения этой проблемы является использование интервала вероятностей, который охватывает все оценки от экспертов. Такой подход учитывает разнообразие мнений и предоставляет более полное представление о вероятности события [10].

Система проверки согласованности состоит из трех модулей, каждый из которых выполняет определенную задачу, позволяя наглядно отобразить и проанализировать данные.

1. Модуль парсинга введенных в программную систему предложений.
2. Модуль проверки согласованности оценок, выявляющий и отображающий несогласованности, строя граф взаимосвязей между оценками.
3. Модуль исправления несогласованностей, предлагающий два метода корректировки оценок: метод ближайшего верного значения и метод равномерного распределения нагрузки.

Для проверки согласованности оценочных знаний необходимо привести все введенные предложения к виду СДНФ. Это возможно, потому что все введенные формулы являются предложениями. По предложениям и субъективным вероятностям их наступления строится система линейных уравнений, а далее по теореме Кронекера – Капелли [11] проверяется совместность системы. Затем система решается, но на нее накладывается ограничение: вероятность каждого конъюнкта СДНФ должна принадлежать отрезку от 0 до 1, так как по определению вероятность принимает именно такие значения.

При решении системы уравнений может возникнуть три ситуации:

1. Система имеет решения при заданных ограничениях.
2. Система не имеет решений при заданных ограничениях, но имеет решения, выходящие за пределы ограничений.
3. Система не имеет решений.

В первом и втором случаях для визуализации данных производится отрисовка графа. Для примера из табл. 1 граф изображен на рис. 1.

*Таблица 1*

Входные данные

*Table 1*

Input data

| | |
|---|---|
| $\varphi_1 = P(a) \vee !Q(b);$ | 0,65 |
| $\varphi_2 = \left(P(a) \vee Q(b)\right) \& \left(P(a) \vee !Q(b)\right).$ | 0,8 |

В графе в качестве последнего уровня вершин представлены конъюнкты из СДНФ предложений. Эти конъюнкты постепенно собираются в предложения, которые были поданы на вход, с помощью дизъюнкции между собой, а ребра представляют собой отношение частичного порядка между ними.

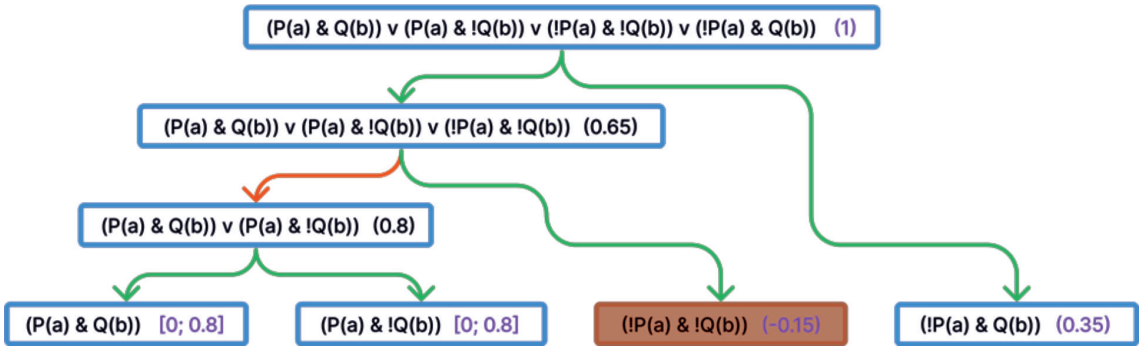


*Рис. 1.* Граф для визуализации согласованности экспертных оценок
*Fig. 1.* Graph for visualizing the consistency of expert evaluations

У каждой вершины есть свое оценочное значение. Вероятность события, которое является дизъюнкцией нескольких условий, всегда будет больше либо равна вероятности любого из его подмножеств из-за монотонности вероятности.

Происходит проверка всех ребер построенного графа. Вес вершин на пути от последнего уровня до корня должен монотонно нестрого возрастать из-за монотонности вероятности. В том месте, где это правило нарушается, происходит рассогласование оценок.

## 3. Модуль работы с тесными кванторами

Формулой с тесными кванторами будем называть формулу логики предикатов, где область действия квантора представлена одним предикатом.

Так как множество объектов в поданных на вход программе предложениях конечно, каждое предложение, содержащее квантор, эквивалентно некоторому бескванторному предложению. Кванторные предложения можно развернуть в конечное число бескванторных предложений путем подстановки всех возможных значений из этого множества объектов. Квантор существования можно заменить на конечную дизъюнкцию предложений, а квантор всеобщности – на конечную конъюнкцию предложений:

$$\exists x P(x) \equiv P(a_1) \vee P(a_2) \vee \ldots \vee P(a_n);$$
$$\forall x P(x) \equiv P(a_1) \wedge P(a_2) \wedge \ldots \wedge P(a_n).$$

Поэтому сначала программная система формирует множество констант, которые она берет из поданных на вход предложений. А затем заменяет все предложения с тесными кванторами на бескванторные предложения, подставляя в них все возможные значения из множества объектов.

После этого производится вычисление оценочных значений все по тому же алгоритму: сначала все предложения приводятся к виду СДНФ, строится система линейных уравнений, проверяется ее совместность и решается система линейных уравнений. Следует отметить, что при подстановке констант в предложение с квантором могут получиться предикаты с константами, которые изначально не присутствовали в исходных предложениях, поданных на вход программы. Это значит, что программная система не только определяет согласованность экспертных оценок, но и вычисляет вероятность наступления событий для ранее не встречавшихся предложений, порожденных в процессе подстановки, что, безусловно, является преимуществом.

После всего этого также строится граф. Для демонстрации работы возьмем формализованный пример, который был приведен в разделе 1.

Компания планирует внедрение нового продукта, и для этого она оценивает различные риски, связанные с успешностью проекта. В частности, компания анализирует, как отдельные риски и общая вероятность рисков могут повлиять на успех проекта.

На успех проекта потенциально влияет множество различных рисков. Рассмотрим несколько конкретных аспектов, по которым известны оценки:

1. Надежность оборудования (обозначим это как $a$).
2. Стабильность поставок (обозначим как $b$).
3. Финансовая устойчивость компании (обозначим как $c$).

И рассмотрим предикаты, заданные на этом множестве рисков:

1. $P(x)$ – это предикат, который означает, что «условие $x$ способствует успеху проекта».
2. $Q(y)$ – это предикат, который означает, что «условие $y$ несет в себе риск для проекта».

На вход подаются предложения из табл. 2 с вероятностными оценками.

*Таблица 2*

Экспертные оценки рисков

*Table 2*

Expert risk evaluations

| | |
|---|---|
| $\varphi_1 = P(a) \vee !Q(b);$ | 0,3 |
| $\varphi_2 = P(c);$ | 0,9 |
| $\varphi_3 = \exists x P(x).$ | 0,4 |

Первое предложение отражает вероятность того, что проект будет успешен, если оборудование надежное (условие $P(a)$) или поставки будут стабильными (отсутствие риска $Q(b)$). Его вероятность равна 0,3.

Второе предложение – вероятность того, что финансовая устойчивость компании способствует успеху проекта, и она равна 0,9.

Третье предложение – существует хотя бы одно условие $x$ из множества всех рисков, которое способствует успеху проекта, вероятность этого равна 0,4.

Чтобы работать с третьим предложением на практике, заменим квантор существования на конечную дизъюнкцию, используя конкретные риски $a$, $b$ и $c$, для которых у нас есть данные. Таким образом, мы переходим к бесквантерному предложению:

$$\varphi_3 = P(a) \vee P(b) \vee P(c).$$

Теперь у нас есть три бесквантерных предложения, для которых можно проверить согласованность с помощью системы линейных уравнений и построить соответствующий граф (рис. 2).

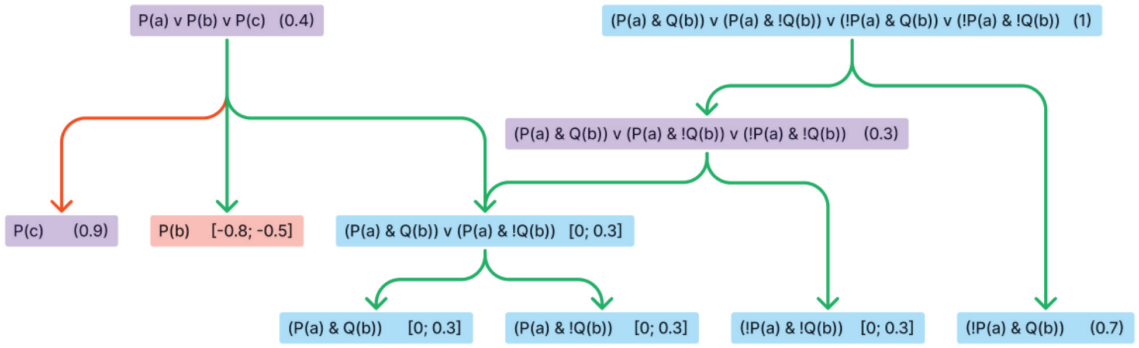


*Рис. 2.* Граф для визуализации согласованности экспертных оценок рисков
*Fig. 2.* Graph for visualizing the consistency of expert risk evaluations

Красная стрелка указывает на место, где монотонность нарушена, т. е. вероятность более общего события оказалась меньше вероятности частного события.

Что же может предпринять компания, увидев такой результат?

Первый вариант. Вероятность для финансовой устойчивости компании $P(c)$ равна 0,9, а общая вероятность для хотя бы одного условия, способствующего успеху, равна 0,4. Это может говорить о необходимости пересмотра оценки для $P(c)$.

Второй вариант. Вероятность 0,4 для общего события показывает, что даже при высокой оценке для финансовой устойчивости (0,9) вероятность успеха при учете всех условий остается низкой. Это говорит о том, что надежность оборудования $P(a)$ и стабильность поставок $P(b)$ оцениваются недостаточно высоко, что снижает общую вероятность успеха проекта. В таком контексте компания может усилить надежность оборудования или снизить риски, связанные с поставками, что увеличит общую вероятность успеха.

Учитывая нашу прошлую разработку – модуль исправления несогласованностей – программная система также может помочь компании понять, как именно и насколько им нужно скорректировать свои оценочные высказывания.

Если построенный граф оказался бы согласованным, при добавлении аспектов в рассмотрение экспертами важно учитывать, что их оценочное значение должно быть не меньше оценочного значения предложения с квантором, иначе согласованность может быть нарушена.

Когда вероятности согласованы, компания может быть уверена, что оценки экспертов по ключевым аспектам проекта согласуются друг с другом. Это дает более ясное представление о рисках. Такое согласованное видение помогает команде избежать ситуаций, когда одно направление работы настраивается на успех, а другое остается уязвимым. Проведение анализа согласованности помогает компании не только устранить противоречия, но и правильно расставить приоритеты, направив ресурсы на те области, которые требуют усиления. В конечном итоге, согласованность вероятностей позволяет компании принять более обоснованное решение: запускать проект, откладывать его или вносить корректировки. Если согласованная вероятность успеха проекта высока, компания может смело приступить к реализации. Если же согласованность показала, что успех маловероятен, проект можно пересмотреть, чтобы снизить риски или изменить стратегию.

## 4. Интерфейс программной системы

Интерфейс программной системы разработан с использованием библиотеки JavaFX. Реализовано разделение на Model, View и Controller с применением FXML, что позволяет привязывать методы контроллеров к элементам пользовательского интерфейса.

Пользователь может задавать события через диалоговое окно, формулируя их в виде предложений логики предикатов, используя операторы конъюнкции, дизъюнкции, импликации, отрицания, кванторы существования и всеобщности, а также указывать вероятности наступления этих событий в диапазоне от 0 до 1. Пользователь может вводить любое количество таких формул с соответствующими вероятностными оценками. Входные данные для примера из раздела 2 представлены на рис. 3.

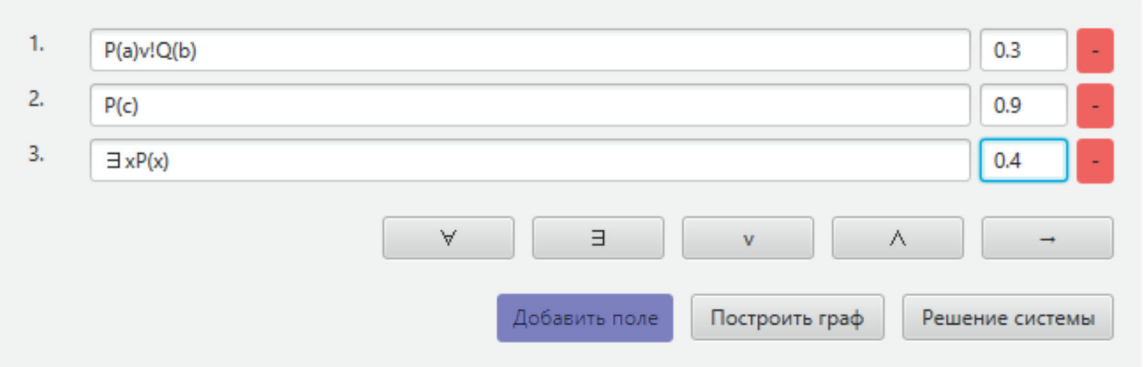


*Рис. 3.* Диалоговое окно
*Fig. 3.* Dialog window

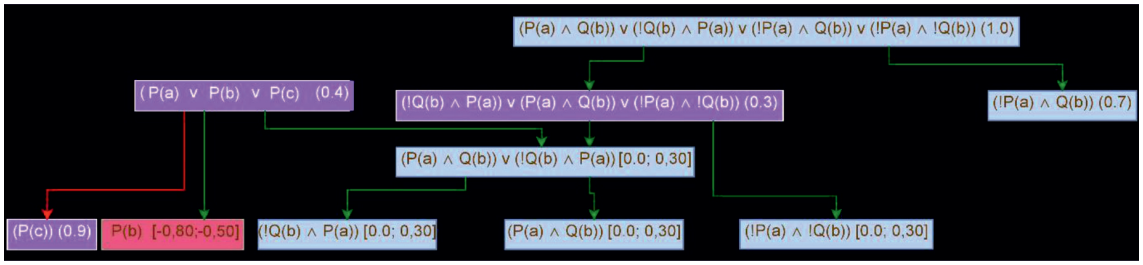


*Рис. 4.* Результат работы программной системы
*Fig. 4.* Result of the software system's operation

После нажатия кнопки «Построить граф» система визуализирует введенные данные. Узлы, представляющие программно-рассчитанные вероятности, отображаются голубым цветом, а те, которые определены пользователем, – фиолетовым. Если введенные данные противоречат друг другу, места несогласованности подсвечиваются в интерфейсе. Розовым цветом отображаются узлы с отрицательной вероятностью, а красные ребра графа обозначают нарушение монотонности вероятностных оценок. Скриншот, на котором отражен результат работы программной системы для примера из раздела 2, приведен на рис. 4.

## Заключение

Интеграция тесных кванторов в систему проверки согласованности экспертных оценок значительно расширяет возможности анализа сложных ситуаций. Это позволяет более точно моделировать взаимодействие различных факторов, учитывая как их индивидуальные вероятности, так и их комбинации. Кванторы существования и всеобщности позволяют обрабатывать более сложные логические выражения, что увеличивает гибкость и точность системы. Такой подход улучшает способность системы учитывать разнообразные сценарии, что особенно важно для комплексных задач, таких как оценка рисков или принятие стратегических решений, где необходимо учитывать множество факторов и их взаимные зависимости.

Кроме того, использование тесных кванторов помогает не только выявлять несогласованности в оценках, но и предсказывать новые, ранее не учтенные обстоятельства, что повышает общую надежность полученных результатов. Расширение системы с учетом кванторов делает ее более универсальной и мощной, что способствует принятию более обоснованных решений. В контексте оценки рисков для бизнеса или других задач, требующих анализа экспертных оценок, такая интеграция играет ключевую роль в повышении точности прогнозов и поддержке более эффективных стратегий управления рисками.

## Список литературы

1. **Куркина Е. П., Шувалова Д. Г.** Оценка риска: экспертный метод // Проблемы науки. 2017. № 1 (14). С. 63–69.
2. **Павленко А. В., Ковалева Е. Г., Радоуцкий В. Ю.** Анализ подходов к оценке риска // Вестник БГТУ им. В. Г. Шухова. 2015. №3. С. 106–109.
3. **Жуков М. С., Орлов А. И., Фалько С. Г.** Экспертные оценки в рисках // Контроллинг. 2017. № 66. С. 24–27.
4. **Орлов А. И.** Организационно-экономическое моделирование. Ч. 2. Экспертные оценки. М.: Моск. гос. техн. ун-т им. Н. Э. Баумана, 2011. 488 с.
5. **Малаева Е. Д., Яхъяева Г. Э.** Программная система визуализации и проверки согласованности оценочных знаний экспертов // Вестник НГУ. Серия: Информационные технологии. 2023. Т. 21. № 1. С. 32–45.
6. **Yakhyaeva G.** Fuzzy model truth values // Proceedings of the 6-th International Conference Aplimat, February 6–9, 2007. Bratislava, Slovak Republic, 2007. P. 423–431.
7. **Яхъяева Г. Э., Ясинская О. В.** Методы согласования знаний по компьютерной безопасности, извлеченных из различных документов // Вестник НГУ. Серия: Информационные технологии. 2013. Т. 11, № 3. С. 63–73.
8. **Yakhyaeva G., Skokova V.** Subjective Expert Evaluations in the Model-Theoretic Representation of Object Domain Knowledge. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2021. P. 152–165.

9. **Карасев О. И., Китаев А. Е., Миронова И. И., Шинкаренко Т. В.** Экспертные процедуры в форсайте: особенности взаимодействия с экспертами в проектах по долгосрочному прогнозированию // Вестник Санкт-Петербург. ун-та. Социология. 2017. № 2. С. 169–184.
10. **Дымонт Н. А., Малаева Е. Д., Яхъяева Г. Э.** Алгоритмы проверки корректности интервальных экспертных оценок. // Вестник НГУ. Серия: Информационные технологии. 2024. Т. 22. № 2.
11. **Ильин В. А., Ким Г. Д.** Линейная алгебра и аналитическая геометрия. М.: Проспект, 2007. 400 с.

## References

1. **Kurkina E. P., Shuvalova D. G.** Otsenka riska: ekspertnyi metod [Risk assessment: expert method]. *Problems of Science*, 2017, no. 1 (14), pp. 63–69. (In Russ.)
2. **Pavlenko A. V., Kovaleva E. G., Radoutskiy V. Yu.** Analiz podkhodov k otsenke riska [Analysis of approaches to risk assessment]. *Vestnik BSTU named after V. G. Shukhov*, 2015, no. 3, pp. 106–109. (In Russ.)
3. **Zhukov M. S., Orlov A. I., Falko S. G.** Ekspertnye otsenki v riskakh [Expert evaluations in risks]. *Controlling*, 2017, no. 66, pp. 24–27. (In Russ.)
4. **Orlov A. I.** Organizatsionno-ekonomicheskoe modelirovanie. Chast' 2. Ekspertnye otsenki [Organizational and economic modeling. Part 2. Expert evaluations]. Moscow, Bauman Moscow State Technical University, 2011, 488 p. (In Russ.).
5. **Malaeva E. D., Yakhyaeva G. E.** Programmnaya sistema vizualizatsii i proverki soglasovannosti otsenochnykh znaniy ekspertov [Software system for visualization and verification of expert evaluative knowledge consistency]. *Vestnik NSU. Series: Information Technologies*, 2023, vol. 21, no. 1, pp. 32–45. (In Russ.)
6. **Yakhyaeva G.** Fuzzy model truth values. // Proceedings of the 6-th International Conference Aplimat, February 6-9, 2007, Bratislava, Slovak Republic, pp. 423–431.
7. **Yakhyaeva G. E., Yasinskaya O. V.** Metody soglasovaniya znaniy po komp'yuternoy bezopasnosti, izvlechennykh iz razlichnykh dokumentov [Methods for aligning knowledge on computer security extracted from various documents]. *Vestnik NSU. Series: Information Technologies*, 2013, vol. 11, no. 3, pp. 63–73. (In Russ.)
8. **Yakhyaeva G., Skokova V.** Subjective Expert Evaluations in the Model-Theoretic Representation of Object Domain Knowledge. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2021, pp. 152–165.
9. **Karasev O. I., Kitaev A. E., Mironova I. I., Shinkarenko T. V.** Ekspertnye protsedury v forsayte: osobennosti vzaimodeystviya s ekspertami v proyektakh po dolgosrochnomu prognozirovaniyu [Expert procedures in foresight: features of interaction with experts in long-term forecasting projects]. *Vestnik of Saint Petersburg University. Sociology*, 2017, no. 2, pp. 169–184. (In Russ.)
10. **Dymont N. A., Malaeva E. D., Yakhyaeva G. E.** Algoritmy proverki korrektnosti interval'nykh ekspertnykh otsenok [Algorithms for verifying the correctness of interval expert evaluations]. *Vestnik NSU. Series: Information Technologies*, 2024, vol. 22, no. 2. (In Russ.)
11. **Ilyin V. A., Kim G. D.** Lineynaya algebra i analiticheskaya geometriya [Linear algebra and analytic geometry]. Moscow, Prospekt publ., 2007, 400 p. (In Russ.)

## Информация об авторе

**Малаева Елена Дмитриевна**, студентка

## Information about the Author

**Elena D. Malaeva,** Student

*Статья поступила в редакцию 04.02.2025;*
*одобрена после рецензирования 17.04.2025; принята к публикации 17.04.2025*

*The article was submitted 04.02.2025;*
*approved after reviewing 17.04.2025; accepted for publication 17.04.2025*

Научная статья

УДК 004.658.6
DOI 10.25205/1818-7900-2025-23-2-29-42

# Алгоритм проверки предусловий корректности запуска репликации в отказоустойчивом кластере PostgreSQL

**Андрей Сергеевич Рудометов**
**Михаил Валерьевич Рутман**

Новосибирский государственный университет
Новосибирск, Россия

a.rudometov@g.nsu.ru
m.rutman@g.nsu.ru

*Аннотация*

Традиционно отказоустойчивые кластеры СУБД на базе PostgreSQL строятся с использованием механизма потоковой репликации, передающего файлы журнала предзаписи между узлами. При запуске репликации выполняются проверки лишь на целостность файлов журнала. Поэтому при вводе в кластер узлов после отработки отказа и настройки репликации можно получить резервный узел с данными, отличными от основного, либо выходящий из строя при попытке запуска или перезапуска. Существующие системы высокой доступности вынуждены требовать длительного процесса пересоздания узла в данных сценариях.

В работе предложен алгоритм, блокирующий запуск репликации в сценариях, когда это заведомо приведет к потере идентичности данных или сбоям при восстановлении узла. Для этого перед запуском выполняется проверка предусловия корректности, основанных на сборе и сравнении информации о состоянии журналов предзаписи узлов кластера. После блокировки возможна автоматическая синхронизация узлов для последующего корректного запуска репликации.

Предложен способ встраивания алгоритма в СУБД, проведено тестирование на различных конфигурациях с эмуляцией сбоев. При минимальных накладных расходах алгоритм предотвращает последствия некорректного запуска репликации в большинстве случаев.

*Ключевые слова*

репликация, отказоустойчивый кластер СУБД, высокая доступность, предсказание сбоев, PostgreSQL

*Для цитирования*

*Рудометов А. С., Рутман М. В.* Алгоритм проверки предусловий корректности запуска репликации в отказоустойчивом кластере PostgreSQL // Вестник НГУ. Серия: Информационные технологии. 2025. Т. 23, № 2. С. 29–42. DOI 10.25205/1818-7900-2025-23-2-29-42

# Preconditions-Based Algorithm for Safe Start of Replication in Fault-Tolerant PostgreSQL Cluster

**Andrey S. Rudometov, Mikhail V. Rutman**

Novosibirsk State University,
Novosibirsk, Russian Federation

a.rudometov@g.nsu.ru
m.rutman@g.nsu.ru

*Abstract*

Traditionally, fault-tolerant DBMS clusters using PostgreSQL or derivatives are built on replication machinery, operated via write-ahead log shipping. Default checks are aimed only at preserving the integrity of received records. In certain conditions replication start can lead to standby cluster node having data different from other nodes, or being unable to finish startup procedures. Existing high availability systems are forced to cope with the problem through recreating such nodes from backups, which is usually costly in terms of recovery time.

To address this issue, we propose an algorithm to prevent replication start when it is guaranteed to lead to data differences or node startup failure. For detection of such cases node collects information about write-ahead logs in the cluster and performs additional checks. If replication was blocked, automatic node synchronization for consequent replication start is available.

We have tested the algorithm on various real-world cluster configurations with simulated failures, and the experimental results indicate that algorithm substantially reduces the chance of nodes being non-eligible to restart.

*Keywords*

replication, fault-tolerant DBMS cluster, high availability, failure prediction, PostgreSQL

*For citation*

Aleeva V. N., Kuznetsov E. K. Effective implementation of sorting algorithms using the concept of Q-determinant. *Vestnik NSU. Series: Information Technologies*, 2025, vol. 23, no. 2, pp. 29–42 (in Russ.) DOI 10.25205/1818-7900-2025-23-2-29-42

## Введение

Репликация широко используется в распределенных системах как простой способ обеспечения избыточности с минимальной задержкой. Распределенные СУБД, в частности, основанные на PostgreSQL решения, не являются исключением; большинство программного обеспечения для управления отказоустойчивыми кластерами СУБД на базе PostgreSQL, например: patroni, pg_auto_failover, stolon, PostgresPro BiHA, pgEdge, полагается на встроенный в PostgreSQL механизм репликации для поддержания актуальной копии данных на резервных узлах кластера [1].

Потоковая репликация в PostgreSQL передает между узлами кластера файлы журнала предзаписи, работая при этом параллельно процессу, который применяет их содержимое в процессе запуска узла или его функционирования как резервного. Поскольку ожидается, что узлы имеют полностью идентичные журналы и данные, никаких дополнительных проверок при копировании и возможной перезаписи файлов журнала не выполняется.

С другой стороны, при функционировании в составе отказоустойчивого кластера и после реакции на отказы или изменения топологии кластера условие идентичности журналов узлов может нарушаться. Запуск репликации между узлами с разными журналами и/или разными текущими позициями применения записей из них может привести к различным сбоям узла. Сбои могут быть как легко наблюдаемые, например, резервный узел, на котором не применяются реплицированные данные, или узел, не обрабатывающий любые запросы к нему из-за невозможности окончить процедуру запуска, так и практически не диагностируемые. В качестве примера последнего можно привести возможность получить резервный узел с отличными от остальных данными, но нормально применяющий реплицированный журнал – подобная

ситуация будет обнаружена только при нарушении какой-либо проверки целостности при обращении к некорректным данным, если таковая проверка вообще существует.

Аппаратные и программные сбои могут приводить к выходу узлов из отказоустойчивого кластера. После нейтрализации некоторых сбоев, например, потери сетевой связности между узлами или отключения питания, узел может немедленно либо после минимального вмешательства войти в состав кластера и участвовать в резервировании данных. Для других видов сбоев, например, безвозвратной потери хранимых данных или полного выхода из строя всего центра обработки данных, восстановление и возврат узла в кластер будут невозможны.

Сбои, вызванные запуском репликации между узлами с разными журналами, сейчас тоже требуют пересоздания узла, поскольку не существует универсального механизма исправления последствий таких сбоев – пример с отличными данными это хорошо показывает. Поскольку полное пересоздание подразумевает восстановление узла из резервной копии, т. е. как минимум копирование всей базы данных и ожидание завершения репликации разницы с актуальными данными, то очевидно, что данный способ ввода узла в кластер много дольше обычного запуска узла. Чем дольше восстановление после отказа, тем выше риск потери данных из-за повторных отказов [2].

С другой стороны, на узлах кластера содержится вся необходимая информация для предсказания и недопущения сбоев после запуска репликации. Эта информация не используется ни встроенным механизмом репликации, из-за предположения об идентичности журналов, ни внешними решениями для построения кластеров, из-за трудности доступа к внутренней информации СУБД.

В работе предложен алгоритм принятия решения о безопасности запуска репликации. Алгоритм блокирует автоматический запуск репликации; после получения информации о состояниях журнала других узлов кластера и проверки набора предусловий либо разрешает репликацию, либо оставляет корректно работающий, но не подключенный к остальному кластеру узел. Во втором случае предлагается либо вмешательство системного администратора, либо автоматическая синхронизация узла с одним из узлов кластера посредством утилиты pg_rewind. Такая синхронизация занимает больше времени, чем репликация, но много меньше, чем пересоздание узла.

Предложенный алгоритм не зависит от конкретной архитектуры какого-либо решения для построения отказоустойчивого кластера и может использоваться в уже существующих системах после внесения правок в ядро СУБД и обновления узлов.

## Сильное зацепление репликации и запуска узла

Надежное функционирование СУБД PostgreSQL обеспечивается использованием журнала предзаписи. Для описания предлагаемого алгоритма и решаемых им проблем достаточной абстракцией будет пронумерованная последовательность записей, хранимая в последовательности сегментных файлов. Номер следующей записи на единицу больше номера предыдущей. Новые записи добавляются в последний сегментный файл последовательности. При заполнении сегментного файла создается новый, последовательность записей продолжается в нем (рис. 1).

Исторически репликация не проектировалась частью сервера баз данных (или узла, в терминологии данной работы) PostgreSQL; вместо этого архитектура была сфокусирована на надежном механизме восстановления после сбоев, позволяя продолжать обслуживание клиентов после неожиданных остановов или перезапусков СУБД [3–5]. Когда была признана необходимость данного механизма и начали появляться сторонние расширения, реализующие логику передачи журнала предзаписи, например Slony-I [6], реализация встроенного механизма ре-

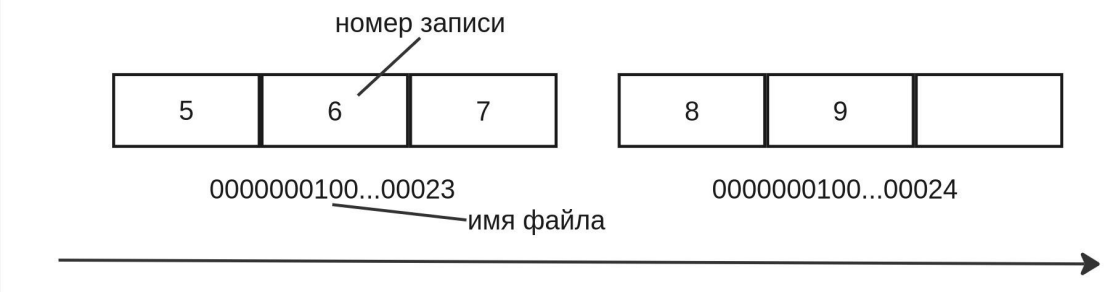


*Рис. 1.* Записи журнала предзаписи в сегментных файлах
*Fig. 1.* Write-ahead log records in segment files

пликации была добавлена в PostgreSQL 9.0 как расширение к существующей логике восстановления после сбоя [7].

Сервер PostgreSQL работает в одном из двух режимов: восстановления (режиме резерва) либо обычном. В обычном режиме обрабатываются все запросы клиентов, в режиме восстановления – только запросы на чтение и только после достижения СУБД целостного состояния.

После каждого запуска узел PostgreSQL переходит в режим восстановления и последовательно читает записи журнала предзаписи и применяет их согласно протоколу REDO. Чтение останавливается при достижении записи с целевым номером, после этого узел может перейти из режима восстановления в обычный режим и начать обрабатывать клиентские запросы.

Если узел не был остановлен с помощью команды администратора или используемых механизмов автоматизации, т. е. останов был неожиданным и не оставил записи в управляющих файлах СУБД, целевым номером записи считается номер последней из всех присутствующих в журнале узла на момент запуска.

Механизм репликации расширяет данную логику, добавляя процесс, способный получать записи журнала с других узлов и сохранять как часть журнала узла, в то время как узел продолжает оставаться в режиме восстановления и применять новые записи. В случае достижения конца журнала и отсутствия новых записей узел ожидает их появления.

Данный процесс, *walreceiver*, является частью довольно простого механизма – соответствующий ему процесс *walsender* на узле-источнике репликации в цикле проверяет разницу между номером последней отправленной записи и номером последней созданной и при необходимости пересылает составляющие разницу записи на узел-получатель, где они записываются в соответствующий сегментный файл журнала узла. Если в журнале узла-получателя уже были файлы или записи с такими же номерами, они будут перезаписаны, поскольку при нормальном функционировании репликации ожидается, что узлы имеют одинаковые журналы с единственной разницей в отставании содержимого одного от другого.

Если при попытке применения перезаписанной записи возникает ошибка, то такая ошибка заблокирует применение последующих записей, а при перезапуске узла не даст ему окончить восстановление – оно остановится на той же самой ошибке, оставляя узел недоступным для управления посредством SQL-интерфейса.

Хотя теоретически возможно вручную исправить содержимое записей журнала, отсутствие готовых инструментов для этого потребует сложного анализа бинарных файлов с высокой вероятностью ошибки, что делает метод неприменимым на практике.

## Линии времени

В качестве части механизма встроенной репликации PostgreSQL предлагает возможность восстановления на момент времени (PITR). Если необходимые записи журнала еще не были

удалены или были восстановлены из архива, процесс восстановления может закончиться на заданной администратором целевой записи, соответствующей заданному времени, идентификатору транзакции и т. д. Поскольку записи, добавленные после выбранной целевой, все еще остаются в каталоге с файлами журнала, а после восстановления до заданной цели узел может перейти в обычный режим и начать добавлять с этого момента новые записи, то для дифференциации «старых» и «новых» записей была введена концепция линий времени (*timeline*) [8]. Линия времени с определенным номером объединяет записи, сделанные после последнего перехода узла в обычный режим.

Номер линии времени используется при именовании сегментных файлов журнала, что позволяет легко отличать записи, сделанные в разные линии времени. После каждого окончания восстановления номер линии времени инкрементируется, генерируя копию последнего сегментного файла журнала, но с увеличенным номером линии времени в имени. Информация о новой линии времени сохраняется в записи об окончании режима восстановления (*END_OF_RECOVERY, EOR* на схеме). Все новые записи будут записаны в этот и последующие файлы (рис. 2).

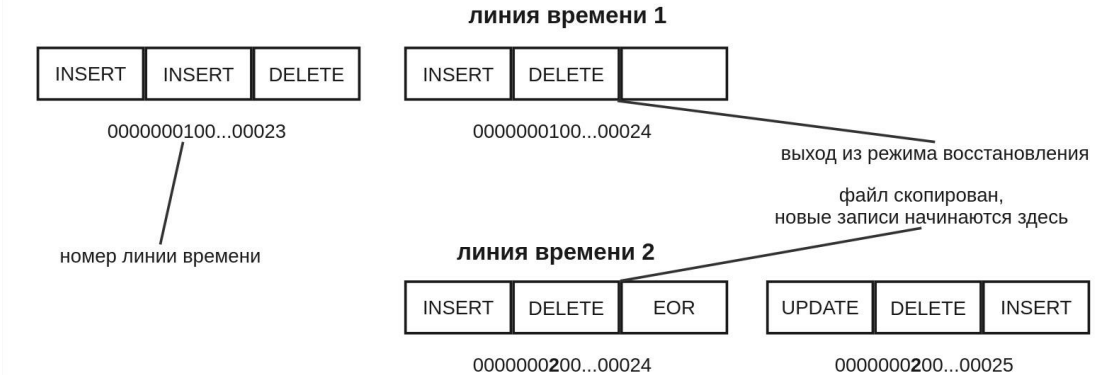


*Рис. 2.* Сегментные файлы после увеличения номера линии времени
*Fig. 2.* Segment files after timeline increment

Кроме того, механизм репликации поддерживает *файлы истории* – по одному для каждой линии времени вплоть до текущей. Файл содержит информацию о номерах записей журнала, на которых происходили все переходы на новую линию времени от первой до соответствующей файлу.

Анализ этих файлов для последней линии времени узла позволяет проводить сравнение журналов узлов без их чтения.

## Синхронизация узлов: pg_rewind

Пусть синхронизация данных на узлах PostgreSQL при помощи репликации недоступна: из-за сбоев на уровне кластера и возможных split-brain [9] ситуаций история линий времени узлов различается, а значит, различается и содержимое журналов. В таком случае все еще остается способ синхронизации узлов более быстрой, чем инициализация нового из резервной копии [10].

Узел может быть синхронизирован с другим при помощи стандартной утилиты pg_rewind. Утилита ищет последнюю одинаковую запись в двух журналах, затем составляет список измененных последующими записями журнала файлов на узле-получателе и копирует соответствующие файлы с узла-источника [11]. После копирования для восстановления целостности данных необходимо применить все записи журнала, начиная с последней одинаковой.

Кроме того, при запуске узла-получателя после синхронизации необходимо реплицировать и применить все записи журнала, созданные на узле-источнике между моментами начала и окончания процесса копирования. Очевидно, что в противном случае целостность данных может быть нарушена: пусть изменение затронуло уже скопированный файл A и еще не скопированный файл B, тогда после синхронизации узел-получатель будет содержать только часть изменения в файле B.

## Пример некорректного запуска репликации

Приведем пример сценария, который приводит к расхождению данных на узлах. Пусть узел A обрабатывает запросы клиентов (является ведущим узлом), узел B – его резервный, получает реплицированные записи и применяет их, находясь в режиме восстановления (рис. 3). Указатели replay_lsn, flush_lsn соответствуют номерам последней примененной и последней реплицированной с A записи. Пусть теперь связь между узлами теряется, причем на A остаются еще не реплицированные записи (на схеме выделены темно-серым фоном).

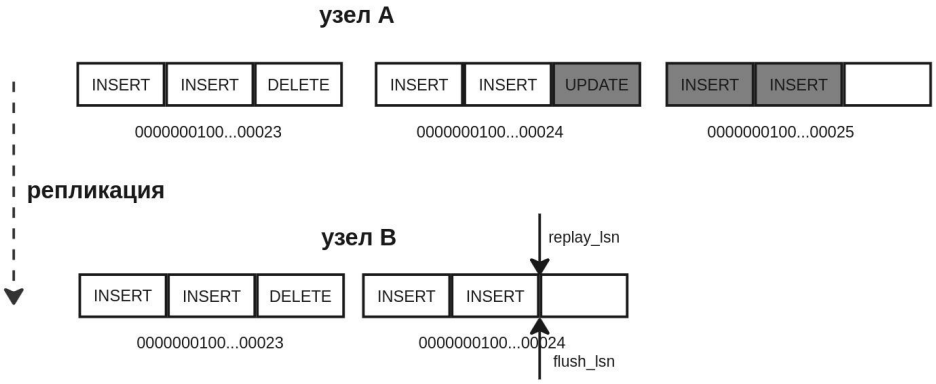


*Рис. 3.* Состояние журналов узлов до некорректного запуска репликации
*Fig. 3.* Node's logs before incorrect replication start

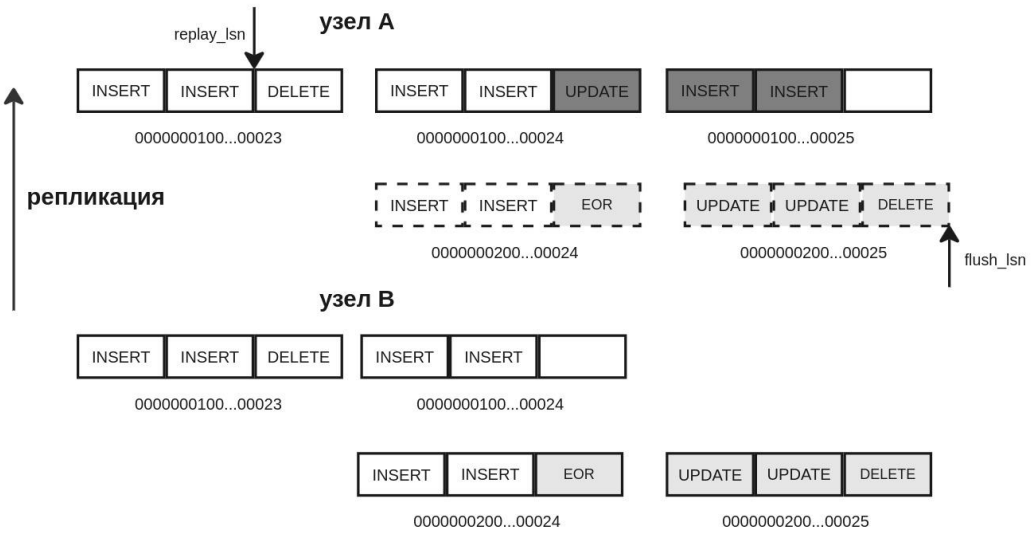


*Рис. 4.* Состояние журналов узлов после некорректного запуска репликации
*Fig. 4.* Node's logs after incorrect replication start

Если узлы A и B были частью отказоустойчивого кластера, то вместо A ведущим будет назначен один из резервных узлов. Пусть, не ограничивая общности, новым ведущим оказался B. Теперь, если мы хотим вернуть A в кластер как резервный узел, необходимо перезапустить его в режиме восстановления и настроить репликацию с B на A (рис. 4). На схеме светло-серым фоном выделены записи, созданные на B во время его работы ведущим. При запуске репликации с B на A будут переданы и записаны в журнал записи из файлов 24 и 25 второй линии времени – результат записи выделен пунктиром.

Тогда набор данных узла A будет зависеть от взаимного положения указателей применения и репликации записей. В данном примере предполагается, что перезапись происходит до того, как было начато применение перезаписываемых записей (replay_lsn < flush_lsn). Нетрудно заметить, что на узле A будет содержаться результат применения записей с темно-серым фоном, в частности файла 25 первой линии времени, а на всех остальных узлах кластера его не будет.

## Обзор литературы

СУБД PostgreSQL не проектировалась с учетом использования в отказоустойчивых кластерах: наглядным примером является использование примитивного последовательного REDO при запуске. Решения для построения отказоустойчивых кластеров работают с предоставляемыми СУБД интерфейсами и не вмешиваются в логику работы с данными и журналом предзаписи, поэтому решения данной проблемы со стороны СУБД ранее не предлагалось. Однако область ускорения восстановления данных уже исследовалась ранее, и некоторые идеи могут быть применены в данной работе.

В работе [12] предлагается новый алгоритм мгновенного восстановления, основная идея которого заключается в поиске изменений, REDO которых может быть выполнено «ленивым» образом. Таким образом, эти изменения могут быть применены по необходимости (при доступе к соответствующим им данным) после приведения узла в состояние доступности. В то время как данная идея не может быть легко интегрирована в существующую логику восстановления СУБД, отложить применение реплицированных с других узлов изменений относительно применения уже имеющихся полезно для достижения поставленной цели.

В исследованиях онлайн-восстановления (online recovery [13]), «способности восстановления узла после сбоя, пока другие узлы исполняют запросы клиентов», авторы в основном фокусируются на использовании существующих внешних интерфейсов PostgreSQL, таких как восстановление их архива или восстановление на момент времени (PITR) [13; 14]. В то время как данные методы могут помочь избежать части сценариев, приводящих к невозможности запуска после сбоя репликации, при правильной конфигурации – они не универсальны, и такие побочные эффекты не являлись целью исследований.

Пока инкрементальное копирование, добавленное в PostgreSQL 17, было в стадии разработки, было предложено и реализовано несколько внешних решений, обычно полностью основанных на управлении WAL для снижения накладных расходов на чтение и запись – пример такого подхода демонстрируется в работе [15]. Хотя общая идея исследования подходит как инструмент решения обозначенной проблемы, внешние относительно СУБД решения не имеют доступа к полной информации о ее состоянии и способны использовать меньше методов управления этим состоянием.

## Предлагаемый алгоритм

Основная идея алгоритма заключается в запрете запуска репликации через блокировку функций, приводящих к запуску процесса *walreceiver*. Запрет поддерживается, как минимум,

пока СУБД не будет уверена, что репликация с узла-источника согласно ее конфигурации не приведет к сбою.

Обычно *walreceiver* запускается без каких-либо проверок того, что именно будет записано в каталог, содержащий файлы журнала.

После установки блокировки алгоритм ожидает получения информации о состоянии журнала узла-источника, затем выполняет проверку предусловий запуска репликации и соответствующие действия согласно матрице решений.

Необходимая для проверки предусловий информация о состоянии журналов узла-источника и узла-получателя:

1) файл истории текущей линии времени узла-источника;
2) файл истории текущей линии времени узла-получателя;
3) номера последней примененной и последней записанной на диск записи журнала узла-источника;
4) номера последней примененной и последней записанной на диск записи журнала узла-получателя;
5) точка расхождения историй линий времени узла-источника и узла-получателя.

В решениях для построения отказоустойчивых кластеров для передачи информации о состоянии часто используется механизм heartbeat-сообщений, отправляемых периодически с заданным интервалом и содержащих краткий статус узла для принятия решений об изменении конфигурации кластера какими-либо внешними модулями управления.

История и номера транзакций узла-источника (пп. 1, 3 списка необходимой информации) могут быть встроены в такие сообщения, и тогда узлу-получателю нужно только дождаться установки соединения с узлом-источником и получения первого такого сообщения. Для эталонной реализации алгоритма реализовано расширение PostgreSQL, обеспечивающее соединение узлов кластера «каждый-с-каждым» и аналогичную рассылку heartbeat-сообщений с необходимыми параметрами и настраиваемой периодичностью. Расширение запрашивает у СУБД номер последней записи перед каждой отправкой, а содержимое файла истории текущей линии времени кэшируется и обновляется только при смене линии времени.

Для узла-получателя известен номер текущей линии времени, поэтому файл истории (п. 2) достаточно прочитать из каталога файлов журнала.

Получение информации для п. 4 сложно, поскольку во время принятия решения узел обязан находиться в состоянии восстановления, а в таком состоянии СУБД не имеет штатных средств для определения номера последней записи в журнале, т. е. последней записанной на диск. Состояние восстановления подразумевает, что выполнение записей последовательно и бесконечно, и в любой момент они могут быть добавлены в конец журнала – процессами репликации, при восстановлении из архива журнала или вручную администратором. Поэтому узлом в состоянии восстановления отслеживается только номер последней примененной записи.

В то время как возможна разработка метода надежного хранения необходимого номера, в предлагаемом алгоритме выбран более простой подход – узел должен завершить восстановление после сбоя до принятия решения согласно алгоритму, т. е применить все записи, содержащиеся в каталоге журнала, и перейти к ожиданию появления новых. Таким образом, номера последней примененной и последней записанной на диск записи совпадут.

Пусть в обеих историях линий времени двух узлов присутствует запись об окончании времени $t$, причем номера последних записей в них различаются на двух узлах. Тогда назовем точкой расхождения (п. 3) пару ($t$, $elsn$) такую, где $t$ – наименьшая из удовлетворяющих условию, а $elsn$ – наименьший из двух различных номеров последней записи, соответствующих $t$. Из существования точки расхождения двух историй следует неодинаковость записей после точки расхождения, что приведет к сбоям после запуска репликации между узлами с такими историями.

Точка расхождения может быть найдена путем сопоставления обеих линий времени. Достаточно начать с линии времени 1 и сравнивать номера последних записей в линиях

времени с одинаковым номером, пока не будет достигнут конец одного или обоих файлов истории.

## Матрица решений

Обозначим узел-получатель, стремящийся закончить процесс восстановления и войти в состав кластера СУБД, как узел A, а узел-источник, с которого ему необходимо реплицировать недостающие данные, как узел B.

Соответствующие номера текущей линии времени обозначим *a_tli* и *b_tli; m_tli = min(a_tli, b_tli*). Определим *elsn(tli)* как номер последней записи в линии времени.

Значение *elsn* для предыдущих линий времени может быть получено посредством чтения файлов истории. Для текущей линии времени в ее файле истории нет номера последней записи, только номер записи, на котором закончилась предыдущая линия времени. Поэтому доопределим *elsn* для текущей линии времени как номер последней примененной или записанной на диск записи, в зависимости от режима, в котором находится узел B.

Отметим также, что такое доопределение не делает корректным понятие точки расхождения для текущей линии времени (если *a_tli = b_tli*), поскольку она еще не завершена посредством перехода на следующую. В таком случае есть опасность посчитать точкой расхождения согласно ее определению сценарий, когда один из журналов имеет больше записей, но общие записи журналов совпадают, т. е. после запуска репликации журналы станут одинаковыми.

Зная обе текущие линии времени и их *elsn*, можно выделить ситуации, в которых запускать репликацию точно нельзя:

1. Если номер линии времени узла A больше, то очевидно, что на нем есть данные, которых нет на узле B. Если мы разрешим репликацию, то какие-то записи журнала будут перезаписаны, что с высокой вероятностью приведет к сбою.
2. Аналогично, если линии времени узлов совпадают, но *elsn(a_tli) > elsn(b_tli)*, существует хотя бы одна запись, встречающаяся только в журнале узла A.
3. Если номер линии времени A меньше такового у B, а *elsn(a_tli)*, наоборот, больше *elsn(b_tli)*, значит, *(a_tli, elsn(b_tli))* – точка расхождения.
4. Для линий времени, предшествующих *m_tli*, существует точка расхождения.

В данных случаях запуск репликации возможен, только если синхронизировать данные и журнал узла-получателя с узлом-источником, в предлагаемом алгоритме используется синхронизация при помощи pg_rewind.

В остальных случаях матрица принятия решений допускает запуск репликации, как показано на рис. 5.

| | **a_tli < b_tli** | **a_tli = b_tli** | **a_tli > b_tli** |
|---|---|---|---|
| **elsn(a_tli) < elsn(m_tli)** | можно | можно | нельзя/ синхронизация |
| **elsn(a_tli) < elsn(m_tli)** | можно | можно | нельзя/ синхронизация |
| **elsn(a_tli) < elsn(m_tli)** | нельзя/ синхронизация | нельзя/ синхронизация | нельзя/ синхронизация |

*Рис. 5.* Матрица принятия решений
*Fig. 5.* Decision matrix

В случае совпадения номеров линий времени нельзя говорить о полной уверенности отсутствия расхождения журналов узлов.

Поскольку алгоритм сравнивает только номера позиций, но не содержимое записей журнала, возможны ситуации, когда на узле A меньше или столько же записей, но их содержимое отлично от такового в части записей узла B.

Во время тестирования алгоритма наблюдались редкие случаи, приводившие к подобному состоянию журналов. Такое может произойти, например, если в процессе работы кластера из-за внутренних задержек изменения состояния некоторое время допускалась запись на два изолированных по сети друг от друга узла, причем на тот узел, который в схеме алгоритма будет соответствовать A, данных было записано меньше.

Для полного избавления от подобных эффектов необходимы более вычислительно затратные решения, например, последовательное чтение и сравнение записей журнала или попытки вычисления какого-либо рода хеш-функций для сравнения содержимого, с учетом разного количества записей на узлах. В данной работе такие решения не рассматриваются в силу достаточной доли сценариев, в которых алгоритм достигает поставленной цели.

## Результаты

Описанный выше подход был реализован в виде набора модификаций ядра PostgreSQL 16.1 и расширения для той же версии. Модификации и расширение написаны на языке C. Для создания нагрузки при тестировании производительности использовалась нагрузочная утилита pgbench со стандартным TPC-C-like сценарием и соответствующим масштабированием.

Эталонная реализация алгоритма состоит из двух частей: расширения для передачи heartbeat-сообщений и принятия решений; и модификаций ядра СУБД для управления процессом репликации.

В расширении, помимо установки соединений и передачи сообщений, извлекаются и поддерживаются значения номеров последних записей журнала и файла истории последней линии времени. Состав кластера задается пользователем в конфигурационном файле. При запуске СУБД расширение немедленно, до запуска логики восстановления после сбоя, стартует один процесс-обработчик, периодически отправляющий сообщения.

Когда применение всех доступных записей журнала завершено и необходимая информация получена с узла-источника, в расширении принимается решение о дальнейшем поведении репликации согласно матрице принятия решения.

Расширение и ядерные изменения взаимодействуют посредством выделяемой в PostgreSQL для вспомогательных процессов разделяемой памяти.

В случае если решением, которое принял алгоритм, оказалось «нельзя / синхронизация», поведение зависит от параметров конфигурации. Выполнение синхронизации при помощи pg_rewind вызывает перезагрузку СУБД и автоматически производит синхронизацию, если пользователь установил соответствующий параметр конфигурации. Если этот параметр отсутствует, запуск репликации блокируется.

Для тестирования использовались два способа создания кластеров, с помощью кластерных утилит patroni версии 4.0.5 [16] и Postgres Pro BiHA в составе PostgresPro Enterprise 16.1 [17].

Все три варианта подразумевают кластер из трех узлов, с одним ведущим и двумя ведомыми, получающими данные посредством асинхронной физической репликации. Ведущий узел работает в обычном режиме и доступен клиенту на чтение и запись. При выходе ведущего узла из строя или изоляции относительно ведомых один из ведомых становится новым ведущим узлом. Все узлы запускаются внутри виртуальных машин на одном физическом сервере.

Критерий оценки эффективности алгоритма – способность узла, бывшего ведущим, вернуться в кластер и продолжить работу в качестве резервного узла, имея корректный набор данных и корректно применяя новые изменения с нового ведущего. В качестве типовых сце-

нариев, когда ожидается подобное поведение, использовалось переключение ведущего узла (switchover [18]), и возникновение split-brain ситуаций вследствие сетевого разделения кластера. Корректность набора данных проверялась посредством попарного сравнения записей в тестовой таблице между введенным узлом и новым ведущим узлом.

Сетевое разделение считается частой причиной сбоев в распределенных системах [19]. В работе сетевое разделение эмулировалось посредством изменения правил *iptables* на узлах для отклонения всех входящих и исходящих пакетов и через настраиваемый временной интервал – отмены таких правил. Такой подход позволяет блокировать любое сетевое взаимодействие между выбранными узлами кластера, эффективно заставляя их считать друг друга вышедшими из строя. После разделения выполнялись записи на оба ведущих узла – на старый, пока узел еще не запретил запись, и на новый, после того как он был избран кластерным ПО.

Переключение ведущего узла обычно определяется как контролируемое аварийное переключение ведущего узла с балансировкой нагрузки. Если кластер СУБД построен с использованием асинхронной репликации, то отказы во временном интервале между принятием решения о переключении и моментом, когда узел-кандидат на роль нового ведущего получит все необходимые данные посредством репликации, могут привести к расхождению данных старого и нового ведущего. Например, если текущий ведущий становится недоступным во время процесса переключения, а кандидат не получил все необходимые данные, то кандидат будет вынужден стать новым ведущим с неполным комплектом данных.

При тестировании во время переключений эмулировались внешние сбои: останов узла или сетевое разделение. Для эмуляции замедления из-за пиковой нагрузки на случайных временных интервалах запускались дополнительные требовательные к процессорному времени процессы и/или урезались доступные узлу ресурсы. Если выполнялись разделение или останов узла, после завершения переключения восстанавливалось исходное состояние: правила iptables удалены, все узлы запущены.

Каждый из сценариев воспроизводился по 2000 раз для кластера с загруженным расширением, реализующим алгоритм, и без него. Для сценария переключения лидера случайным образом, но не менее одного во время каждого переключения генерировались события выключения или сетевого разделения узлов кластера.

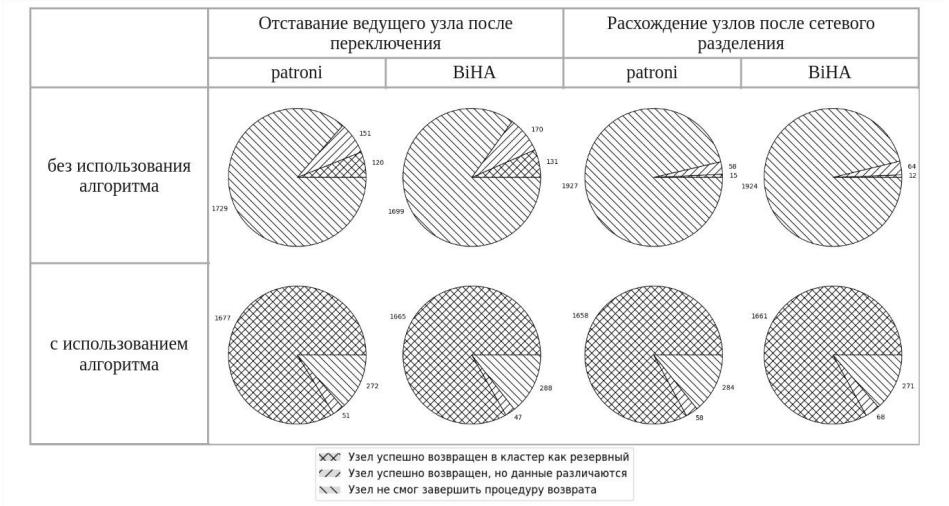


*Рис. 6.* Число успешных возвратов бывшего ведущего узла в кластер как резервного
*Fig. 6.* Amount of successful former primary node's readditions as standby

Результаты приведены на рис. 6. Стоит отметить, что без использования алгоритма вероятность успешного ввода в строй после эмулируемого сценария невысока. Использование

предложенных проверок повышает вероятность успешного ввода до приемлемых значений. Остальные случаи относятся к краевым и требуют дополнительного сравнения содержимого журналов, что описано в разделе о матрице решений.

Использование реализованного в расширении алгоритма незначительно увеличивает время восстановления после сбоя. Основные факторы – отказ от записи реплицированных одновременно с восстановлением и ожидание получения информации от узла-источника. В работе не приводятся исчерпывающие измерения, поскольку влияние данных факторов зависит от объемов журнала при восстановлении, сетевой инфраструктуры, используемого внешнего ПО для обслуживания кластера (в частности, использование такого ПО тоже привносит задержки из-за обмена информацией о состоянии узлов). На тестовых стендах разница среднего времени восстановления для сценариев без и с использованием расширения не превысила 3 % от абсолютного значения.

## Заключение

В работе предложен и реализован алгоритм проверки предусловий корректности запуска репликации с минимальными накладными расходами. На типовых для отказоустойчивого кластера сценариях алгоритм обнаружил большинство некорректных ситуаций и предотвратил сбой узла из-за запуска репликации. Предлагаемая алгоритмом автоматическая синхронизация узлов при помощи утилиты pg_rewind позволила запустить репликацию на узел корректным образом и вернуть узел в кластер без вмешательства администратора.

Таким образом, в данных сценариях при реальном сбое использование алгоритма позволит ввести узел в состав кластера без необходимости создания нового, что значительно сокращает время ввода узла в кластер.

В работе также описаны не детектируемые алгоритмом пограничные сценарии, которые требуют разработки дополнительных, более требовательных к ресурсам проверок перед запуском репликации.

Для автоматической синхронизации узлов в работе используется утилита pg_rewind. Из-за архитектурных особенностей утилиты для завершения синхронизации требуется априорный запуск репликации при запуске узла. Для этого требуется единоразовое отключение логики предлагаемого алгоритма. Проектирование более подходящего способа синхронизации узлов, между которыми нельзя запустить репликацию, может быть направлением дальнейшего исследования.

## Список литературы

1. **Thomas S. M.** PG Phriday: Redefining Postgres High Availability // BonesMoses.org: сайт. 2024. URL: https://bonesmoses.org/2024/pg-phriday-redefining-postgres-high-availability/
2. **Kassema J. J.** Disaster Recovery Plan for Business Continuity: Case Study in a Business Sector. In: SSRN. 2016. DOI: 10.2139/ssrn.2796601
3. **Stonebraker M., Rowe L. A.** The design of POSTGRES. In: Proceedings of the 1986 ACM SIGMOD international conference on Management of data (SIGMOD '86) (June 1986). Association for Computing Machinery, New York, NY, USA, 1986. P. 340–355. DOI 10.1145/16856.16888
4. **Cecchet E., Candea G., Ailamaki A.** Middleware-based database replication: the gaps between theory and practice // Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08). Association for Computing Machinery, New York, NY, USA, 2008. P. 739–752. DOI 10.1145/1376616.1376691
5. **Stonebraker M., Rowe L., Hirohama M.** The Implementation Of Postgres // Knowledge and Data Engineering, IEEE Transactions on. 2. 1990. P. 125–142. DOI 10.1109/69.50912
6. **Wieck J.** Slony-I. A replication system for PostgreSQL // Slony-I. URL: https://slony.info/images/Slony-I-concept.pdf

7. PostgreSQL: Documentation 9.0: Release 9.0 // PostgreSQL: Documentation. URL: https://www.postgresql.org/docs/9.0/release-9-0.html
8. **Linnakangas H.** Understanding PostgreSQL timelines // FOSDEM 2013. URL: https://wiki.postgresql.org/images/e/e5/FOSDEM2013-Timelines.pdf
9. **Davidson S. B., Garcia-Molina H.; Skeen D.** Consistency In A Partitioned Network: A Survey // ACM Computing Surveys. 1985. Vol. 17, iss. 3. P. 341–370. DOI 10.1145/5505.5508
10. **Панченко И.** PostgreSQL: вчера, сегодня, завтра // Открытые системы. СУБД. 2015. № 3. С. 34–37. URL: https://www.osp.ru/os/2015/03/13046900
11. PostgreSQL: Documentation 17.0: pg_rewind // PostgreSQL: Documentation. URL: https://www.postgresql.org/docs/17/app-pgrewind.html
12. **Härder T., Sauer C., Graefe G. et al.** Instant recovery with write-ahead logging // Datenbank Spektrum. 2015. Vol. 15. P. 235–239. DOI 10.1007/s13222-015-0204-3.
13. **Bárbaro P., Pedroso M.** High Availability and Load Balancing for Postgresql Databases: Designing and Implementing. In: International Journal of Database Management Systems. 2016, vol. 8, pp. 27–34. DOI 10.5121/ijdms.2016.8603
14. **Md. Anower H., Md. Imrul H., Dr. MD Rashedul I., Nadeem A.** A Novel Recovery Process in Timelagged Server using Point in Time Recovery (PITR). In: 24th International Conference on Computer and Information Technology (ICCIT). 2021. DOI: 10.1109/ICCIT54785.2021.9689808.
15. **Kim H., Yeom H. Y, Son Y.** An Efficient Database Backup and Recovery Scheme using Write-Ahead Logging // 2020 IEEE 13th International Conference on Cloud Computing (CLOUD). 2020. P. 405–413. DOI 10.1109/CLOUD49709.2020.00062
16. Introduction – patroni 3.3 documentation // Patroni documentation. https://patroni.readthedocs.io/en/rel_3_3/
17. Postgres Pro Enterprise: Документация: 16: F.8: biha – встроенный отказоустойчивый кластер // Документация PostgreSQL и Postgres Pro: компания Postgres Professional. URL: https://postgrespro.ru/docs/enterprise/16/biha
18. **Meng-Lai Y.** Assessing availability impact caused by switchover in database failover // 2009 Annual Reliability and Maintainability Symposium. Fort Worth, TX, USA. 2009. P. 401–406. DOI 10.1109/RAMS.2009.4914710.
19. **Coan B. A., & Oki B. M., Kolodner E. K.** Limitations on Database Availability when Networks Partition // PODC ‘86: Proceedings of the fifth annual ACM symposium on Principles of distributed computing. 1986. P. 187–194. DOI: 10.1145/10590.10606.

## References

1. **Thomas S. M.** PG Phriday: Redefining Postgres High Availability. In: BonesMoses.org: сайт. 2024. URL: https://bonesmoses.org/2024/pg-phriday-redefining-postgres-high-availability/
2. **Kassema J. J.** Disaster Recovery Plan for Business Continuity: Case Study in a Business Sector. In: SSRN, 2016, DOI: 10.2139/ssrn.2796601
3. **Stonebraker M., Rowe L. A.** The design of POSTGRES. In: Proceedings of the 1986 ACM SIGMOD international conference on Management of data (SIGMOD ‘86) (June 1986). Association for Computing Machinery, New York, NY, USA, 1986, pp. 340–355. DOI 10.1145/16856.16888
4. **Cecchet E., Candea G., Ailamaki A.** Middleware-based database replication: the gaps between theory and practice. In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD ‘08)*. Association for Computing Machinery, New York, NY, USA, 2008, pp. 739–752. DOI 10.1145/1376616.1376691
5. **Stonebraker M., Rowe L., Hirohama M.** The Implementation Of Postgres. In: *Knowledge and Data Engineering, IEEE Transactions on. 2.* 1990, pp. 125–142. DOI 10.1109/69.50912

6. **Wieck J.** Slony-I. A replication system for PostgreSQL. In: Slony-I. URL: https://slony.info/images/Slony-I-concept.pdf
7. PostgreSQL: Documentation 9.0: Release 9.0. In: PostgreSQL: Documentation. URL: https://www.postgresql.org/docs/9.0/release-9-0.html
8. **Linnakangas H.** Understanding PostgreSQL timelines. In: FOSDEM 2013. URL: https://wiki.postgresql.org/images/e/e5/FOSDEM2013-Timelines.pdf
9. **Davidson S. B., Garcia-Molina H.; Skeen D.** Consistency In A Partitioned Network: A Survey. In: *ACM Computing Surveys*, 1985, vol. 17, iss. 3, pp. 341–370. DOI 10.1145/5505.5508
10. **Panchenko I.** PostgreSQL: yesterday, today, tomorrow. In: Open systems. DBMS, 2015, no. 3, pp. 34–37. URL: https://www.osp.ru/os/2015/03/13046900
11. PostgreSQL: Documentation 17.0: pg_rewind. In: PostgreSQL: Documentation. URL: https://www.postgresql.org/docs/17/app-pgrewind.html
12. **Härder, T., Sauer, C., Graefe, G. et al.** Instant recovery with write-ahead logging. *Datenbank Spektrum* 15. 2015, pp. 235–239. DOI 10.1007/s13222-015-0204-3.
13. **Bárbaro P., Pedroso M.** High Availability and Load Balancing for Postgresql Databases: Designing and Implementing. *International Journal of Database Management Systems*, 2016, vol. 8, pp. 27–34. DOI 10.5121/ijdms.2016.8603
14. **Md. Anower H., Md. Imrul H., Dr. MD Rashedul I., Nadeem A.** A Novel Recovery Process in Timelagged Server using Point in Time Recovery (PITR). In: *24th International Conference on Computer and Information Technology (ICCIT)*. 2021. DOI 10.1109/ICCIT54785.2021.9689808.
15. **Kim H., Yeom H. Y, Son Y.** An Efficient Database Backup and Recovery Scheme using Write-Ahead Logging. In: *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, 2020, pp. 405–413, DOI: 10.1109/CLOUD49709.2020.00062
16. Introduction – patroni 3.3 documentation. In: patroni documentation. https://patroni.readthedocs.io/en/rel_3_3/
17. Postgres Pro Enterprise: Documentation: 16: F.8: biha – built-in high-availability cluster // Documentation PostgreSQL и Postgres Pro: Postgres Professional: site. URL: https://postgrespro.ru/docs/enterprise/16/biha
18. **Meng-Lai Y.** Assessing availability impact caused by switchover in database failover. In: *2009 Annual Reliability and Maintainability Symposium*. Fort Worth, TX, USA, 2009, pp. 401–406. DOI: 10.1109/RAMS.2009.4914710.
19. **Coan B. A., & Oki B. M., Kolodner E. K.** Limitations on Database Availability when Networks Partition. In: *PODC '86: Proceedings of the fifth annual ACM symposium on Principles of distributed computing*. 1986, pp. 187–194. DOI: 10.1145/10590.10606.

## Информация об авторах

**Рудометов Андрей Сергеевич,** студент магистратуры; ассистент

**Рутман Михаил Валерьевич,** доцент

## Information about the Authors

**Andrey S. Rudometov,** Master's Student

**Mikhail V. Rutman,** Associate Professor

*Статья поступила в редакцию 10.03.2025;*
*одобрена после рецензирования 07.04.2025; принята к публикации 07.04.2025*

*The article was submitted 10.03.2025;*
*approved after reviewing 07.04.2025; accepted for publication 07.04.2025*

Научная статья

УДК 004.046
DOI 10.25205/1818-7900-2025-23-2-43-52

# Система управления скриптами в графическом видеоредакторе Adobe After Effects

**Вадим Игоревич Саранин**

Новосибирский государственный университет
Новосибирск, Россия

v.saranin@g.nsu.ru

*Аннотация*

В статье предложен метод управления и персонализированных рекомендаций скриптов для Adobe After Effects, реализованный с помощью расширения CSXS и веб-платформы. Проведен анализ существующих методов поиска, установки и запуска скриптов, а также рассмотрены их ограничения, включая необходимость ручного перемещения файлов, отсутствие персонализированного подбора и интеллектуального поиска. Исследована архитектура предложенного решения, включающая использование ExtendScript для выполнения скриптов без перезапуска After Effects, а также реализация системы рекомендаций на базе векторного анализа текста с применением SentenceTransformers и хранения векторных данных в Qdrant. Обоснована целесообразность применения метода k-nearest neighbors для подбора релевантных скриптов и установлен порог схожести для фильтрации нерелевантных результатов. Рассмотрен алгоритм персонализированных рекомендаций, основанный на анализе избранных скриптов и интересов схожих пользователей. Описаны механизмы синхронизации между веб-платформой и расширением, обеспечивающие доступ к сохраненным скриптам с любого устройства.

*Ключевые слова*

Adobe After Effects, CSXS, ExtendScript, автоматизация, Qdrant, векторизация текста, поиск скриптов, k-nearest neighbors, персонализированные рекомендации

*Для цитирования*

*Саранин В. И.* Система управления скриптами в графическом видеоредакторе Adobe After Effects // Вестник НГУ. Серия: Информационные технологии. 2025. Т. 23, № 2. С. 43–52. DOI 10.25205/1818-7900-2025-23-2-43-52

# Script Management System in Adobe After Effects Graphic Video Editor

**Vadim I. Saranin**

Novosibirsk State University,
Novosibirsk, Russian Federation

v.saranin@g.nsu.ru

*Abstract*

The article proposes a method for managing and personalized script recommendations for Adobe After Effects, implemented using the CSXS extension and the web platform. The analysis of existing methods of searching, installing and running scripts is carried out, as well as their limitations, including the need for manual file movement, the lack of personalized selection and intelligent search. The architecture of the proposed solution is investigated, including the use of ExtendScript to execute scripts without restarting After Effects, as well as the implementation of a recommendation

system based on vector text analysis using SentenceTransformers and vector data storage in Qdrant. The expediency of using the k-nearest neighbors method for selecting relevant scripts is substantiated, and a similarity threshold is set for filtering irrelevant results. The algorithm of personalized recommendations based on the analysis of selected scripts and interests of similar users is considered. Synchronization mechanisms between the web platform and the extension are described, providing access to saved scripts from any device.

*Keywords*

Adobe After Effects, CSXS, ExtendScript, automation, Qdrant, text vectorization, script search, k-nearest neighbors, personalized recommendations

*For citation*

Saranin V. I. Script management system in Adobe After Effects graphic video editor. *Vestnik NSU. Series: Information Technologies*, 2025, vol. 23, no. 2, pp. 43–52 (in Russ.) DOI 10.25205/1818-7900-2025-23-2-43-52

## Введение

В современных условиях развития цифровых инструментов для анимации и видеомонтажа особенно важным становится повышение эффективности рабочих процессов. Для выполнения сложных задач и упрощения работы монтажеры и аниматоры часто используют Adobe After Effects[1]. Эта программа дает много возможностей для создания эффектов и анимаций, однако ее стандартных функций не всегда хватает для реализации сложных идей. Поэтому многие дизайнеры используют скрипты [1].

Скрипты – это программы, написанные на языке ExtendScript, предназначенные для автоматизации различных процессов в Adobe After Effects. Они позволяют выполнять повторяющиеся задачи, изменять параметры проекта, генерировать сложные анимации и выполнять действия, которые вручную заняли бы значительно больше времени. Их использование позволяет повысить эффективность работы монтажеров, аниматоров и дизайнеров, минимизируя ручной труд и ускоряя процесс создания видеоконтента. Они полезны при создании сложных анимаций, работе с данными и управлении проектами. Однако пользователи часто сталкиваются с проблемами: необходимо самостоятельно устанавливать скрипты, следом перезагружать приложение, а также не хватает персонализированных рекомендаций, чтобы найти наиболее подходящий инструмент. Все это замедляет рабочий процесс.

Для решения этих проблем разработано расширение для After Effects с серверной частью и веб-сайтом. Такая архитектура позволяет пользователям легко находить и скачивать скрипты прямо из программы и сразу же их использовать без лишних шагов. Вместе с этим внедрена система рекомендаций, которая подбирает скрипты в зависимости от интересов и истории использования. Рекомендации работают по трем принципам: поиск по описанию, подбор на основе избранных пользователем скриптов и на основе интересов похожих пользователей.

Все эти функции доступны как на сайте, так и в самом расширении, обеспечивает удобную и быструю работу со скриптами.

## Анализ существующих решений

В настоящее время существует несколько способов установки, поиска и запуска скриптов в Adobe After Effects:

1. Ручной запуск скрипта через интерфейс File > Scripts > Run Script... . Этот метод требует от пользователя предварительного хранения скриптов в локальной системе и выбора нужного файла при каждом запуске.

2. Ручная установка скрипта в Adobe After Effects. Пользователю необходимо закрыть программу, переместить файл скрипта по пути C:\Program Files\Adobe\Adobe After Effects

[1] См.: Adobe After Effects. URL: https://www.adobe.com/products/aftereffects.html

2023\Support Files\Scripts, запустить программу и найти в панели Windows название нужного инструмента.

Недостатки таких решений:

1. Затраты времени – запуск или установка требует нескольких одинаковых действий, что тормозит работу.
2. Отсутствие поиска – пользователю необходимо заранее найти в Интернете нужный инструмент, помнить его местоположение и название в файловой системе или самостоятельно перемещать его в нужное место.

На рынке представлено несколько инструментов, предназначенных для управления и запуска скриптов. Например:

1. KBar[2]. Данный инструмент позволяет создавать панели с кнопками, запускающими конкретные скрипты, которые указал пользователь.
2. aescripts+aeplugins[3]. Данное приложение позволяет устанавливать скрипт, не требуя от пользователя самостоятельно перемещать файл в нужное место в локальной системе для корректной работы инструмента.

Однако и в таких решениях есть ряд ограничений:

1. Ручная настройка – пользователю все еще нужно самостоятельно добавлять скрипты, что требует времени и знаний.
2. Отсутствие персонализации – такие менеджеры не анализируют предпочтения пользователей и не предлагают подходящих скриптов.
3. Отсутствие поиска – ни один из этих инструментов не предоставляет возможности искать скрипты.

Существует веб-сайт aescripts.com, который позволяет находить нужные скрипты. Однако он ограничивается поиском, не использующим умных подходов, и предоставлением самых популярных инструментов.

Анализ существующих методов показал, что они либо требуют значительных временных затрат на настройку, либо не обеспечивают интеллектуальный поиск и рекомендации. Многие инструменты, предлагающие интеграцию скриптов в After Effects, ориентированы на пользователей с техническими навыками. Это делает существующие решения сложными для тех, кто хочет просто находить и запускать скрипты без лишних действий. Предложенный в рамках проекта метод направлен на решение этих проблем за счет интеграции единой системы управления скриптами, умного поиска и персонализированных рекомендаций. Это позволит значительно повысить эффективность работы видеомонтажеров и аниматоров, снизив время на запуск и поиск нужных инструментов.

## Предложенное решение

Для решения выявленных проблем был разработан метод, который объединяет удобное управление скриптами, интеллектуальный поиск и систему персонализированных рекомендаций [2]. Он основан на запуске скриптов в Adobe After Effects через ExtendScript, использовании алгоритмов векторного поиска и хранении данных в базе данных Qdrant[4].

Разработанное расширение значительно упрощает взаимодействие с пользовательскими скриптами. Оно позволяет скачивать, запускать и удалять инструменты без необходимости вручную перемещать файлы в системные папки или перезапускать программу. Для этого нужно открыть расширение, найти кнопку «Поиск», выбрать из списка рекомендаций или найти через поисковую строку скрипт, выбрать понравившийся, добавить в избранное и нажать

[2] См.: KBar. URL: https://aescripts.com/kbar/

[3] См.: AE Scripts. URL: https://aescripts.com/

[4] См.: Qdrant Documentation. URL: https://qdrant.tech/documentation/

на кнопку «Скачать», чтобы скрипт автоматически добавился в список доступных. Возможность запустить появляется мгновенно, без необходимости открывать файловый менеджер или искать путь к нужному инструменту. Это значительно ускоряет рабочий процесс и устраняет лишние действия, характерные для традиционных методов работы со скриптами.

## Общий обзор архитектуры

Архитектура системы построена на многоуровневой структуре, объединяющей клиентские и серверные компоненты, обеспечивающей взаимодействие расширения для Adobe After Effects, веб-платформы и backend-сервера.

Система состоит из следующих ключевых компонентов:

1. Клиентская часть:
   1.1. Расширение CSXS для Adobe After Effects, работающее внутри программы и взаимодействующее с сервером.
   1.2. Веб-платформа, предоставляющая интерфейс для поиска, добавления в избранное и загрузки скриптов.
2. Серверная часть (Backend):
   2.1. Java Spring – феймворк, который управляет бизнес-логикой, обрабатывает запросы клиентов и взаимодействует с базами данных.
   2.2. FastAPI (Python) – феймворк, отвечающий за векторизацию данных, работу с рекомендациями и семантический поиск.
   2.3. PostgreSQL – система управления базами данных, используемая для хранения общей информации о скриптах и пользователях.
   2.4. Qdrant – база данных, обеспечивающая поиск похожих скриптов и пользователей на основе векторных представлений данных.

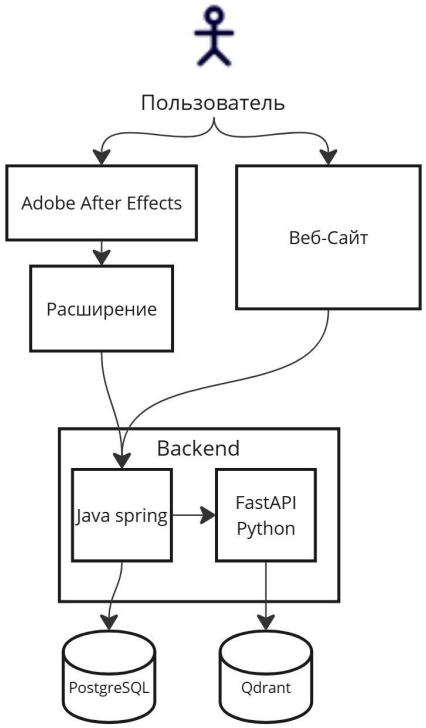


*Рис. 1.* Схема архитектуры системы
*Fig. 1.* System architecture diagram

## Метод векторного анализа текстовых данных для рекомендаций скриптов

В системе разработаны персонализированные рекомендации, которые используют векторизацию текста для определения наиболее подходящих скриптов. Алгоритм написан на отдельном микросервисе во фреймворке FastAPI с использованием следующих библиотек:

1. **qdrant_client** – предоставляет возможность управлять базой данных Qdrant для хранения векторных данных. Она предназначена для хранения и быстрого поиска схожих объектов. В системе данная библиотека используется для хранения векторных представлений скриптов и для их сравнения между собой.

2. **SentenceTransformers** – предоставляет возможность векторизовать текстовые данные на основе обученных моделей. В проекте применяется модель sentence-transformers/distiluse-base-multilingual-cased-v2[5], которая преобразует текстовые описания скриптов в векторы фиксированной размерности (512 измерений). Эта модель содержит в векторах семантическое содержание преобразованного текста [3].

Вся информация о скриптах хранится в базе данных PostgreSQL. При добавлении нового скрипта в PostgreSQL его текстовое описание проходит процесс векторизации, и результат записывается в базу данных Qdrant.

Система рекомендаций работает по трем ключевым принципам: поиск по текстовому описанию, рекомендации на основе списка избранных скриптов и рекомендации, основанные на интересах схожих пользователей.

## Поиск по текстовому описанию

Когда пользователь выполняет поиск, введенный текст векторизуется с помощью модели sentence-transformers/distiluse-base-multilingual-cased-v2 и становится основой для сравнения с уже имеющимися векторами в базе данных Qdrant. Для поиска наиболее подходящих результатов используется метод k-nearest neighbors (k-NN), который позволяет найти скрипты с максимальным сходством к введенному запросу. В качестве метрики близости применяется косинусная мера сходства, что позволяет анализировать не только совпадение ключевых слов, но и общий смысл запроса.

Однако не все результаты могут быть уместны. Если косинусная мера сходства между вектором запроса и найденными векторами скриптов слишком велика, то такие результаты автоматически отсеиваются. Это ограничение предотвращает выдачу совсем неподходящих скриптов, которые могут попасть в список рекомендаций, но не имеют достаточно смыслового сходства с запросом пользователя. Для фильтрации вводится пороговое значение схожести (score_threshold) = 0.1, которое определяет минимально допустимый уровень близости между векторами. Это число является косинусным сходством между двумя векторами. Оно показывает, насколько два вектора близки по смыслу. Когда используется библиотека SentenceTransformers, текст переводится в многомерный вектор с размерностью 512, и каждая строка текста превращается в массив из 512 чисел. Следом высчитывается косинусное сходство, и система отсеивает все результаты, у которых оно меньше 0.1.

Выбор 0.1 был сделан экспериментально.

[5] См.: distiluse-base-multilingual-cased-v2. URL: https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v2

## Генерация персонализированных рекомендаций

Система рекомендаций использует векторизацию текстовых данных и анализ избранных пользователем скриптов для формирования персонализированных предложений. Каждый добавленный в избранные пользователем скрипт проходит процесс векторизации с помощью модели sentence-transformers/distiluse-base-multilingual-cased-v2 и сохраняется в базе данных Qdrant. При генерации рекомендаций система сравнивает векторы купленных скриптов с векторами всех доступных в базе инструментов, используя тот же метод k-nearest neighbors (k-NN). Это позволяет определить скрипты, наиболее схожие по смыслу описания с уже добавленными в список избранных.

Чтобы подбор был более точным и вариативным, дополнительно анализируются интересы пользователей с похожими предпочтениями. Для этого векторизируются сами пользователи. Система соединяет все описания скриптов в один текст и векторизирует его. Всякий раз, когда пользователь добавляет новый инструмент в список избранных, его вектор, который хранится в базе данных Qdrant, обновляется. Если система обнаружила похожего пользователя, его скрипты из списка избранных, отсутствующие у текущего пользователя, включаются в рекомендации.

Таким образом, чтобы сделать рекомендации разнообразными, система смешивает результаты двух методов, чередуя их через раз: один скрипт предлагается на основе избранных пользователем инструментов, следующий – на основе предпочтений схожих пользователей. Это позволяет сбалансировать рекомендации, сочетая персонализированный подбор с дополнительными предложениями, которые могли бы заинтересовать пользователя. Такой метод позволяет находить полезные инструменты не только на основе прямого сходства с уже сохраненными скриптами, но и с учетом опыта других пользователей.

Как и в поиске, не все найденные результаты могут быть уместными. Для предотвращения выдачи случайных и нерелевантных рекомендаций используется пороговое значение схожести (score_threshold) = 0.1 для рекомендаций на основе добавленных в избранные скриптов и (score_threshold) = 0.4 для рекомендаций на основе схожих пользователей по интересам. Принцип работы отсеивания такой же, как был описан выше.

Таким образом, система персонализированных рекомендаций учитывает как индивидуальные предпочтения пользователя, так и интересы схожих пользователей, исключая нерелевантные результаты и предлагая только те скрипты, которые действительно понравятся пользователю.

## Язык программирования ExtendScript для приложений Adobe

ExtendScript[6] – это специальный язык программирования, разработанный компанией Adobe, с помощью которого создаются программы, позволяющие управлять их приложениями, включая After Effects. Он основан на JavaScript, но имеет расширенные возможности к внутренним функциям Adobe, благодаря чему может управлять слоями, анимацией, эффектами и другими элементами проекта, изменять их свойства, запускать эффекты или даже экспортировать видео без необходимости кликать мышкой. Также ExtendScript позволяет создавать простые интерфейсы, значительно упрощающие работу с инструментами автоматизации. На этом языке и пишутся скрипты для After Effects.

## Технология CSXS в разработке расширений для Adobe

Разработанное расширение для Adobe After Effects использует CSXS[7] (Common Extensibility Platform). CSXS – это технология от Adobe, которая позволяет создавать для их приложений

[6] Adobe Inc. ExtendScript Toolkit. URL: https://www.adobe.com/devnet/scripting.html

[7] CSXS SDK Documentation. URL: https://www.adobe.io/apis/creativecloud/aftereffects.html

специальные расширения. Расширение предназначено для управления файлами, выполнения пользовательских команд и взаимодействия с сервером, обеспечивая более гибкую работу с Adobe After Effects. В отличие от скриптов, которые запускаются внутри программы и выполняют определенный набор команд, расширение представляет собой отдельную среду, которая может взаимодействовать с внешними ресурсами, получать данные из сети и предоставлять пользователю отдельный удобный интерфейс для управления процессами. CSXS позволяет взаимодействовать с внутренними API Adobe.

CSXS предоставляет разработчикам инструменты для интеграции веб-технологий (HTML, CSS, JavaScript) в приложения от Adobe. Это позволяет создавать гибкие и мощные расширения с настраиваемым интерфейсом, что дает возможность взаимодействовать с внешними серверами и локальными файлами.

CSXS поддерживает JavaScript API, с помощью которого можно:

1) вызывать ExtendScript для запуска скриптов After Effects;
2) отправлять HTTP-запросы на сервер;
3) работать с локальными файлами, сохраняя и загружая ресурсы напрямую в системные каталоги.

Таким образом, CSXS – это фреймворк, который объединяет веб-технологии и внутренние механизмы Adobe, позволяя интегрировать сложные решения в рабочий процесс пользователей.

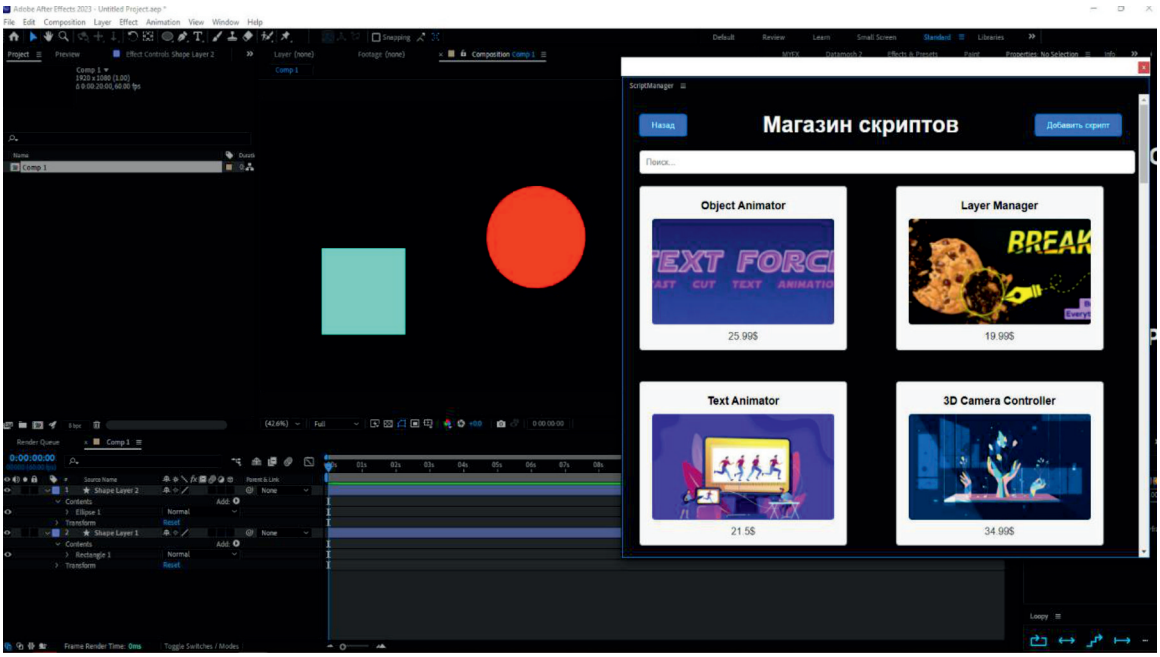


*Рис. 2.* Пример списка скриптов в расширении
*Fig. 2.* Example of the list of scripts in the extension

## Архитектура и работа расширения

Разработанное расширение для Adobe After Effects использует CSXS (Common Extensibility Platform), предоставляющий для API взаимодействия с программами Adobe.

Когда необходимо установить новый скрипт, пользователь вводит описание желаемого инструмента для определенных целей в поисковую строку. Сервер возвращает найденные скрип-

ты и выводит их список в окне интерфейса. После выбора нужного инструмента пользователь должен добавить его в избранное, нажать на кнопку «Скачать», и система автоматически отправляет запрос на сервер для загрузки файла, который сохраняется в директорию, где находятся все скрипты Adobe After Effects: C:\Program Files\Adobe\Adobe After Effects\Support Files\Scripts.

Как только скрипт установлен, он сразу появляется в списке доступных инструментов в интерфейсе расширения. Нет необходимости вручную обновлять интерфейс или перезапускать программу. Система мгновенно обновляет список, делая новый скрипт сразу доступным для запуска. Когда пользователь нажимает на него, расширение через CSXS запускает нужный файл, который выполняет команды в After Effects и, если скрипт имеет интерфейс, показывает его окно прямо внутри приложения. Это происходит автоматически, без необходимости открывать меню и выбирать файл вручную. CSXS использует метод evalScript() из библиотеки CSInterface, которая позволяет расширению передавать команды в After Effects и взаимодействовать с его внутренней средой выполнения через ExtendScript. Сам процесс выглядит так: сначала система определяет путь к файлу скрипта, который уже установлен в нужную папку, затем этот путь передается в app.executeScript(File(path)), и After Effects загружает скрипт. Этот процесс обеспечивает такой же функционал, как если бы пользователь открыл меню File > > Scripts > Run Script…, выбрал файл вручную и запустил его, но здесь все происходит без участия пользователя.

## Веб-платформа для управления и поиска скриптов

Помимо расширения для Adobe After Effects, разработан и веб-сайт, который предоставляет пользователям удобный интерфейс для поиска, управления и загрузки скриптов. Он является частью общей системы и синхронизирован с расширением, позволяя пользователю взаимодействовать с инструментами как в браузере, так и непосредственно в After Effects через расширение.

Основная функциональность веб-сайта заключается в поиске скриптов, управлении аккаунтом и добавлении скриптов в избранное. Пользователь может войти в свою учетную запись и получить доступ ко всей библиотеке инструментов, применяя поиск по описанию или просматривая список доступных скриптов. В отличие от расширения сайт не позволяет сразу запускать скрипты в After Effects, но предоставляет возможность добавить понравившиеся инструменты в избранное. После этого, при открытии After Effects, все скрипты, добавленные в избранное, автоматически появятся в расширении и будут доступны для скачивания и использования без необходимости повторного поиска.

Дополнительно веб-сайт поддерживает загрузку пользовательских скриптов. Разработчики и студии могут добавлять свои инструменты в систему, загружая файлы скриптов и описания, после чего они становятся доступными для других пользователей. Эти скрипты проходят процесс индексации, после чего попадают в базу данных, а их текстовые описания векторизуются и интегрируются в систему рекомендаций. Это позволяет другим пользователям находить новые инструменты через интеллектуальный поиск и рекомендации на основе интересов.

Синхронизация между веб-сайтом и расширением делает процесс работы со скриптами удобным: пользователь может искать и управлять своими инструментами на сайте, а затем мгновенно загружать их в After Effects через расширение. Такой подход позволяет сочетать возможности браузерного поиска с оперативным применением скриптов в профессиональной среде.

Преимущество данного метода в том, что независимо от того, решил ли дизайнер переустановить Windows, расширение, программу или поменять компьютер, все скрипты, ко-

торые он сохранил в избранное, никуда не пропадут, и можно с легкостью установить все инструменты за считанные секунды с помощью разработанного расширения. Пользователь может не переживать, что после переустановки After Effects или смены компьютера у него все исчезнет.

## Заключение

Разработанное расширение для Adobe After Effects предоставляет удобный способ поиска, установки и запуска скриптов, устраняя ограничения традиционных методов работы. Оно интегрируется с программой через CSXS (Common Extensibility Platform), используя связку технологий CSInterface, ExtendScript и evalScript(), что позволяет пользователю мгновенно загружать и применять инструменты без перезапуска After Effects.

Ключевым компонентом системы является метод интеллектуального подбора скриптов, который использует векторизацию текстовых данных и хранение в Qdrant, что позволяет находить инструменты по смысловому соответствию. Внедренная система персонализированных рекомендаций учитывает не только избранные пользователем скрипты, но и анализирует предпочтения схожих пользователей, повышая точность подбора.

Особенностью метода является фильтрация нерелевантных скриптов с помощью порогового значения схожести (score_threshold), что предотвращает выдачу случайных рекомендаций.

Автоматизация процессов с помощью ExtendScript и прямого взаимодействия с After Effects делает работу с расширением интуитивно понятной и быстрой. Система устраняет необходимость ручного перемещения файлов и сложных настроек, позволяя пользователю просто выбрать скрипт, нажать кнопку «Скачать» и сразу начать его использовать.

Дополнением к расширению является веб-платформа, которая предоставляет пользователям возможность управлять скриптами, добавлять их в избранное и загружать собственные инструменты. Благодаря синхронизации между веб-сайтом и расширением, пользователь может искать скрипты в браузере, сохранять их в избранное, а затем мгновенно загружать их в After Effects через расширение на любом персональном компьютере.

Таким образом, разработанное решение значительно ускоряет и упрощает работу видеомонтажеров, аниматоров и дизайнеров, работающих в After Effects, повышая их продуктивность. Интеллектуальная система рекомендаций, глубокая интеграция с Adobe и удобный интерфейс делают его полезным инструментом, который помогает находить и запускать нужные скрипты всего в один клик.

## Список литературы

1. **Саранин В. И.** Система управления расширениями и скриптами в графическом видеоредакторе After Effects // Информационные технологии. Научный инжиниринг: материалы 62-й Международной научной студенческой конференции. Новосибирск, 2024. С. 236.
2. **Саранин В. И.** Персонализированная система рекомендаций скриптов в Adobe After Effects // Алгебро-логические методы в информационных технологиях: материалы Международной конференции «Мальцевские чтения». Новосибирск, 2024. С. 84.
3. **Reimers N., Gurevych I.** Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv.org, 2019. URL: https://arxiv.org/abs/1908.10084

## References

1. **Saranin V. I.** Sistema upravleniya rasshireniyami i skriptami v graficheskom videoredaktore After Effects [Extension and Script Management System in Adobe After Effects]. *Informatsion-*

*nye tekhnologii. Nauchnyi inzhiniring: materialy 62-i Mezhdunarodnoi nauchnoi studencheskoi konferentsii*, Novosibirsk, 2024, p. 236. (in Russ.)
2. **Saranin V. I.** Personalizirovannaya sistema rekomendatsii skriptov v Adobe After Effects [Personalized Script Recommendation System in Adobe After Effects]. *Algebro-logicheskie metody v informatsionnykh tekhnologiyakh: materialy Mezhdunarodnoi konferentsii "Maltsevskie chteniya"*, Novosibirsk, 2024, p. 84. (in Russ.)
3. **Reimers N., Gurevych I.** Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv.org, 2019. URL: https://arxiv.org/abs/1908.10084

## Информация об авторе

**Саранин Вадим Игоревич,** студент

## Information about the Author

**Vadim I. Saranin,** Student

*Статья поступила в редакцию 14.03.2025;*
*одобрена после рецензирования 28.04.2025; принята к публикации 28.04.2025*

*The article was submitted 14.03.2025;*
*approved after reviewing 28.04.2025; accepted for publication 28.04.2025*

Научная статья

УДК 004.021
DOI 10.25205/1818-7900-2025-23-2-53-63

# Методика оценки результатов решения задачи отбора признаков

**Артем Дмитриевич Черемухин**
**Андрей Давыдович Рейн**

Нижегородский государственный инженерно-экономический университет
Княгинино, Россия

ngieu.cheremuhin@yandex.ru, https//orcid.org/0000-0003-4076-5916
ndr18@yandex.ru, https//orcid.org/0000-0002-1921-483X

*Аннотация*

Данная статья представляет собой исследование методов оценки эффективности алгоритмов отбора признаков и предлагает новую методику их оценки. Отмечено, что существующие методы и подходы к оценке не всегда способны адекватно отразить действительную эффективность алгоритмов, особенно при применении к реальным задачам. В рамках статьи подробно рассматриваются различные мнения исследователей по проблемам внутренней и внешней валидности существующих методов оценки, оценивается влияние разных параметров, включая объем данных, различия в реализациях алгоритмов и другие факторы. Авторы предлагают новый комплексный подход к оценке эффективности алгоритмов, включающий ряд показателей, среди которых затраты ресурсов, стабильность и качество решения задачи. Одной из особенностей предлагаемого подхода является использование искусственно сгенерированных данных, что позволяет учесть специфические характеристики реальных данных и оценить алгоритмы в контролируемых условиях. Это дает возможность более точно определить их эффективность и надежность. Статья также содержит результаты предварительного тестирования предложенной методики на примере искусственных данных. Анализ этих результатов демонстрирует преимущества нового подхода над традиционными методами оценки. В частности, было установлено, что новая методика позволяет более точно оценивать стабильность и качество работы алгоритмов, что имеет важное значение для принятия решений о выборе подходящего алгоритма для конкретных задач.

В заключение, авторы подчеркивают необходимость дальнейшего развития и расширения предложенной методики, а также ее адаптации для решения различных типов задач. Они также указывают на перспективы интеграции этой методики в специализированные программные пакеты, что сделает ее доступной для широкого круга пользователей и ускорит внедрение инновационных алгоритмов в практику.

*Ключевые слова*

отбор признаков, методика, стабильность отбора признаков, эффективность отбора признаков, вычислительные ресурсы

*Для цитирования*

*Черемухин А. Д., Рейн А. Д.* Методика оценки результатов решения задачи отбора признаков // Вестник НГУ. Серия: Информационные технологии. 2025. Т. 23, № 2. С. xx–xx. DOI 10.25205/1818-7900-2025-23-2-53-63

# Methodology for Evaluating the Results of Feature Selection Task Solving

**Artem D. Cheremuhin, Andrey D. Rein**

Nizhny Novgorod State Engineering and Economic University
Knyaginino, Russian Federation

ngieu.cheremuhin@yandex.ru, https//orcid.org/0000-0003-4076-5916
ndr18@yandex.ru, https//orcid.org/0000-0002-1921-483X

*Abstract*

Article is a study of methods for evaluating the effectiveness of feature selection algorithms and proposes a new methodology for their evaluation. It is noted that existing methods and approaches to assessment do not always adequately reflect the actual efficiency of algorithms, especially when applied to real problems. The article comprehensively discusses various opinions of researchers on the issues of internal and external validity of existing evaluation methods, assesses the impact of different parameters, including data volume, differences in algorithm implementations, and other factors. The authors propose a new integrated approach to evaluating the effectiveness of algorithms, which includes a set of indicators such as resource costs, stability, and task solution quality. One of the peculiarities of the proposed approach is the use of artificially generated data, which allows considering specific characteristics of real data and evaluating algorithms under controlled conditions. This makes it possible to more accurately determine their efficiency and reliability. The article also contains the results of preliminary testing of the proposed methodology using artificial data. The analysis of these results demonstrates the advantages of the new approach over traditional evaluation methods. In particular, it was found that the new methodology allows more accurately evaluating the stability and quality of algorithm performance, which is crucial for making decisions about choosing an appropriate algorithm for specific tasks. In conclusion, the authors emphasize the need for further development and expansion of the proposed methodology, as well as its adaptation for solving various types of tasks. They also point out the prospects for integrating this methodology into specialized software packages, which will make it accessible to a wide range of users and accelerate the implementation of innovative algorithms in practice.

*Keywords*

feature selection, methodology, feature selection stability, feature selection efficiency, computational resources

*For citation*

Cheremuhin A. D., Rein A. D. Methodology for Evaluating the Results of Feature Selection Task Solving. *Vestnik NSU. Series: Information Technologies*, 2025, vol. 23, no. 2, pp. 53–63 (in Russ.) DOI 10.25205/1818-7900-2025-23-2-53-63

## Введение

Машинное обучение с учителем – один из наиболее распространенных видов машинного обучения, характеризующийся наличием известных значений зависимой переменной. В практических задачах оно применяется при решении задач регрессии (если зависимая переменная числовая) или классификации (если зависимая переменная факторная); при этом очень часто перед исследователями встает задача определения точного состава признаков, определяющих значение выбранной переменной. Особенно это важно, например, при решении задачи по определению влияющих генов, где необходимо выбрать 1–2 влияющих признака из сотен или тысяч. Эта задача известна как задача отбора признаков (feature selection).

## Постановка задачи

В настоящее время разработано и реализовано в программном коде больше 100 различных алгоритмов отбора признаков, что ставит вопрос об определении наиболее эффективных методов. Классическим подходом для доказательства результативности разработанных методов и их пригодности для реальной деятельности является подход с применением бенчмарков:

оценивается эффективность алгоритма (как правило, одним показателем) на основе общедоступного набора датафреймов и сравнивается с результатами прошлых алгоритмов.

Однако практика использования бенчмарков подвергается обоснованной критике – в работе [1] описываются многочисленные проблемы внутренней и внешней валидности, которые возникают при подобном подходе. Во-первых, указывается на то, что лучшие результаты на бенчмарке не гарантируют лучшего результата на задачах «реального мира». Во-вторых, существуют проблемы воспроизводимости алгоритмов – разные реализации одних и тех же алгоритмов зачастую ведут себя как разные алгоритмы. В-третьих, тестируемые алгоритмы в реальных задачах вынуждены работать в более широком диапазоне сценариев (больше/меньше наблюдений, признаков, загрязненные данные), чем реализуемые при тестировании бенчмарками. В-четвертых, когда алгоритм разрабатывается для достижения высоких результатов на каких-то данных, можно уверенно говорить о наличии эффекта переобучения.

Кроме того, как отмечается в работе [2], «даже в самых простых сценариях одного общего показателя недостаточно для точного отражения производительности машинного обучения», соответственно, даже использование показателя величины ошибки (для регрессии) или точности (для классификации) не гарантирует эффективность метода. В работе [3] обращается внимание на наличие разных объектов измерения: точность алгоритма, его устойчивость, вычислительную сложность, объяснимость результатов, ресурсоемкость алгоритмов. При этом обзор литературы показал, что в большинстве исследований, как в [4], игнорируется многообразие объектов оценки, что ставит под сомнения получаемые выводы об эффективности/неэффективности алгоритмов в контексте решения задачи.

Таким образом, можно подтвердить, что в сфере отбора признаков, как и вообще в сфере машинного обучения, сейчас «разработка теории измерений результатов машинного обучения отстает от достижений самого машинного обучения» [1]. Этот факт определил тему исследования, его целью является разработка и апробация методики оценки результатов решения задачи отбора признаков.

Отбор признаков (feature selection) формально является подвидом машинного обучения без учителя и методом сокращения размерности [5], который позволяет из множества переменных отобрать только значимые/влияющие и убрать незначимые/невлияющие. Он применяется в процессе использования как методов обучения с учителем (при решении задач классификации и регрессии), так и методов обучения без учителя (при решении задач кластеризации).

Существуют три основные стратегии решения задачи отбора признаков:

– прямая (на первом шаге множество значимых признаков является пустым, и на каждой итерации происходит добавление в него одного самого важного признака до достижения критерия остановки) [6];

– обратная (на первом шаге множество значимых признаков равно множеству исследуемых признаков, и на каждой итерации происходит удаление самого незначимого признака);

– двусторонняя (на каждом шаге происходит добавление самого значимого из невключенных и удаление самого незначимого из включенных в состав значимых признаков) [7].

Все алгоритмы отбора признаков условно можно разделить на 4 большие группы:

– алгоритмы-фильтры. В них выбирается некоторая метрика «важности» признака (как правило, она является метрикой качества связи между зависимой переменной и переменной-кандидатом на включение в состав влияющих), и решением задачи отбора признаков некоторое является подмножество наиболее важных признаков, выбранных по критерию топ-N или иным подобным признакам [8]. Преимущество данного класса алгоритмов – они быстры и просты в вычислительном отношении и масштабируются для данных любого размера [9], недостаток – рассмотрение признаков «по одному» игнорирует случаи взаимодействия признаков [10], что зачастую приводит к игнорированию важных закономерностей и снижения качества решения задачи;

– «оберточные» алгоритмы. В них выбирается некоторое подмножество признаков, сразу оценивается его качество (в отличие от алгоритмов-фильтров, которые оценивают качество полученного набора признаков только один раз, в конце), и на основе этого принимается решение об изменении подмножества. Примерами таких признаков являются генетические алгоритмы;

– встроенные алгоритмы. К этому семейству алгоритмов относятся те, в которых процесс отбора признаков тем или иным образом встроен в метод определения качества решения, например алгоритм LASSO-регрессии;

– гибридные алгоритмы – алгоритмы, реализующие в себе два или более рассматриваемых подхода.

## Методы исследования

Поскольку в общем случае задача отбора признаков является задачей обучения без учителя (потому что правильный набор влияющих признаков неизвестен), а используется он как составляющая часть решения задачи обучения с учителем, то наиболее часто применяемый подход – качеством решения задачи отбора признаков объявить качество решения исходной задачи (например, величину среднеквадратичной ошибки для задачи регрессии и величину точности классификации для задачи классификации). Однако, как показывают современные исследования в области медицины и биоинформатики, важна не только точность метода отбора признаков, но и его стабильность – он должен обеспечивать выбор максимального одинакового подмножества признаков при добавлении или удалении обучающих образцов [11].

Стоит отметить, что в современных исследованиях [3] используется три подхода к оценке стабильности: стабильность по записи (record-stability of a feature selection; оценивает стабильность подмножества при удалении/добавлении наблюдений в датафрейме), стабильность по признакам (feature-stability of a feature selection; оценивает стабильность подмножества при удалении/добавлении признаков в датафрейме) и стабильность по Ляпунову, которая объединяет стабильность и по записи, и по признакам; при этом конкретная мера оценки сходства двух подмножеств может быть любой.

Необходимое для корректного решения практических задач усложнение подхода к оценке результатов алгоритма отбора признаков еще больше актуализирует необходимость разработки понятной методики данного процесса, основой которой должна стать соответствующая система показателей.

Однако есть нюанс, дополнительно усложняющий задачу. Наряду с использованием классических бенчмарков популярным является подход с использованием синтетических и искусственного сгенерированных датафреймов с заранее заданными свойствами [12]. Он обладает рядом преимуществ, главным из которых является знание правильного подмножества признаков; однако при использовании синтетических датафреймов задача отбора признаков превращается в задачу обучения с учителем, что диктует иной подход к оценке ее точности.

По мнению автора, оценка результатов решения задачи отбора признаков должна включать:

– меры затрат ресурсов (количество операций, объем занимаемой памяти, время работы алгоритма) [13];

– меры стабильности (рекомендуется применять все три ранее рассмотренных оценки);

– меры эффективности работы алгоритма.

Дискуссионным является вопрос и о содержании мер стабильности – в работе [5] изучены наиболее часто рассматриваемые расстояния между подмножествами, при этом особо указывается на отсутствие четких теоретических рекомендаций по выбору метрик.

Таким образом, при работе с данными, по которым неизвестно верное подмножество признаков, предлагается использовать следующую систему показателей:

– количество операций, совершенное алгоритмом;

– объем оперативной памяти, затраченной алгоритмом;
– время работы алгоритма;
– индексы Танимото [14] стабильности по записи, признакам и Ляпунову;
– показатели качества решения задачи: среднеквадратичная ошибка, скорректированный коэффициент детерминации в случае решения задачи регрессии; общий показатель точности с учетом дисбаланса классов, F1-мера при решении задачи классификации.

При работе с данными, по которым известно верное подмножество признаков, предлагается дополнить данную систему на основе подхода, представленного автором в [15]:
– количество операций, совершенное алгоритмом;
– объем оперативной памяти, затраченной алгоритмом;
– время работы алгоритма;
– индексы Танимото стабильности по записи, признакам и Ляпунову;
– индекс Танимото расстояния между вычисленным и правильным подмножеством признаков;
– процент правильно определенных влияющих признаков;
– процент правильно определенных невлияющих признаков;
– процент невключенных влияющих признаков;
– процент включенных невлияющих признаков.

Отдельным является вопрос о порядках расчета индекса Танимото для вычисления показателей стабильности – сколько необходимо переменных/наблюдений изменить? По мнению авторов, целесообразно не фокусироваться на одном показателе, а построить график в системе координат «процент сохраненной изначальной выборки – показатель стабильности», что позволит более точно оценивать устойчивость метода отбора признаков.

Таким образом, для определения эффективности некоторого алгоритма отбора признаков предлагается следующая методика:
– генерируется по одним начальным условиям (количество переменных, количество значимых переменных, количество наблюдений, форма и параметры зависимости между переменными, распределение ошибок и переменных) 10 датафреймов;
– в датафрейме выделяется тренировочная часть (50 %), на которой решается задача отбора признаков, идентифицируется выделенное подмножество признаков;
– по выделенному и правильному набору признаков рассчитывается индекс Танимото и 4 рассмотренных выше процентных показателя;
– фиксируются три показателя затрат ресурсов;
– путем увеличения тренировочной базы данных до изначально сгенерированной с шагом в 5 % и уменьшения до 10 % от исходной рассчитывается динамика индекса Танимото стабильности по записи;
– путем добавления в тренировочную базу случайно сгенерированных признаков с шагом в 1 до двукратного увеличения базы данных или снижения числа незначимых признаков до 0 рассчитывается динамика индекса Танимото стабильности по признакам;
– путем одновременного увеличения/уменьшения базы данных и количества незначимых признаков рассчитывается динамика индекса Танимото стабильности по Ляпунову;
– далее показатели усредняются по каждому из 10 сгенерированных датафреймов.

## Результаты

Для апробации представленной методики был поставлен следующий эксперимент: в R был сгенерирован искусственный датафрейм (полный код доступен на https://github.com/acheremuhin/FS_methodics_experiment), в котором сгенерированные переменные $X$ и $Y$, распределенные по нормальному закону, влияют на переменную $Z$ с некоторой нормально распреде-

ленной аддитивной ошибкой, а также присутствуют три нормально распределенные шумовые переменные; и с помощью пакета FSinR [16] решается задача отбора признаков 6 способами: методом муравьиной колонии [17], генетическим алгоритмом [18], методом «Лас-Вегас» [19], методом имитации отжига [20], методом поиска с запретами [21], методом «китовой охоты» [22].

Показатели результативности и затрат ресурсов представлены в табл. 1.

*Таблица 1*

Показатели ресурсоемкости и результативности решения задачи отбора признаков разными алгоритмами

*Table 1*

Indicators of resource intensity and effectiveness of solving the problem of feature selection by different algorithms

| Метод | Время работы алгоритма, с | Объем потребляемой памяти, Кб | Индекс Танимото | Процент правильно выбранных значимых переменных, % | Процент правильно невыбранных незначимых переменных, % | Процент ошибочно невыбранных значимых переменных, % | Процент ошибочно выбранных незначимых переменных, % |
|---|---|---|---|---|---|---|---|
| Муравьиная колония | 1,59 | 1097357,6 | 0,4 | 100 | 0 | 0 | 100 |
| Генетический алгоритм | 3,03 | 1012 | 0,4 | 100 | 0 | 0 | 100 |
| «Лас-Вегас» | 0,95 | 738,4 | 0,41 | 100 | 3 | 0 | 97 |
| Имитация отжига | 0,9 | 764 | 0,5 | 100 | 30 | 0 | 70 |
| Поиск с запретами | 1,94 | 784 | 0,4 | 100 | 0 | 0 | 100 |
| «Китовая охота» | 1,07 | 781,6 | 0,45 | 100 | 17 | 0 | 83 |

Далее была оценена стабильность алгоритмов по базе данных на основании индексов Танимото (табл. 2).

После этого была оценена стабильность алгоритмов по признакам на основании индексов Танимото (табл. 3).

И в целом была оценена стабильность алгоритмов по Ляпунову на основании индексов Танимото (табл. 4).

## Обсуждение

На основе анализа таблиц 1–4 можно сделать следующие выводы:

– самым долгим по времени работы является алгоритм генетического алгоритма, наиболее затратным по объему памяти – алгоритм муравьиной колонии;

*Таблица 2*

Динамика индекса Танимото устойчивости по наблюдениям алгоритма методом отбора признаков

*Table 2*

Dynamics of the Tanimoto stability index based on observations of the algorithm by feature selection method

| Размер датафрейма от естественного, % | Муравьиная колония | Генетический алгоритм | «Лас-Вегас» | Имитация отжига | Поиск с запретами | «Китовая охота» |
|---|---|---|---|---|---|---|
| 10 | 1,00 | 1,00 | 0,96 | 0,75 | 1,00 | 0,79 |
| 20 | 1,00 | 1,00 | 1,00 | 0,73 | 1,00 | 0,83 |
| 30 | 1,00 | 1,00 | 1,00 | 0,77 | 1,00 | 0,86 |
| 40 | 1,00 | 1,00 | 0,96 | 0,72 | 1,00 | 0,76 |
| 50 | 1,00 | 1,00 | 0.94 | 0,77 | 1.00 | 0,79 |
| 60 | 1,00 | 1,00 | 0,98 | 0,76 | 1,00 | 0,76 |
| 70 | 1,00 | 1,00 | 1,00 | 0,65 | 1,00 | 0,76 |
| 80 | 1,00 | 1.00 | 1,00 | 0,67 | 1,00 | 0,76 |
| 90 | 1,00 | 1,00 | 0,98 | 0,68 | 1,00 | 0,81 |
| 110 | 1,00 | 1,00 | 0,96 | 0,73 | 1,00 | 0,77 |
| 120 | 1,00 | 1,00 | 1,00 | 0,75 | 1,00 | 0,74 |
| 130 | 1,00 | 1,00 | 0,96 | 0,67 | 1,00 | 0,76 |
| 140 | 1,00 | 1,00 | 1,00 | 0,68 | 1,00 | 0,74 |
| 150 | 1,00 | 1,00 | 0,92 | 0,76 | 1,00 | 0,76 |
| 160 | 1,00 | 1,00 | 1,00 | 0,71 | 1,00 | 0,78 |
| 170 | 1,00 | 1,00 | 1,00 | 0,62 | 1,00 | 0,72 |
| 180 | 1,00 | 1,00 | 0,94 | 0,75 | 1,00 | 0,77 |
| 190 | 1,00 | 1,00 | 0,98 | 0,73 | 1,00 | 0,82 |
| 200 | 1,00 | 1,00 | 0,96 | 0,67 | 1,00 | 0,75 |

*Таблица 3*

Динамика индекса Танимото устойчивости по признакам алгоритма методом отбора признаков

*Table 3*

Dynamics of the Tanimoto index of stability according to the characteristics of the algorithm by the method of feature selection

| Количество шумовых переменных в датафрейме по сравнению с исходным | Муравьиная колония | Генетический алгоритм | «Лас-Вегас» | Имитация отжига | Поиск с запретами | «Китовая охота» |
|---|---|---|---|---|---|---|
| –3 | 0,80 | 0,80 | 0,80 | 0,74 | 0,80 | 0,70 |
| –2 | 0,60 | 0,60 | 0,60 | 0,71 | 0,60 | 0,56 |
| –1 | 0,40 | 0,40 | 0,40 | 0,46 | 0,40 | 0,46 |
| +1 | 0,83 | 0,83 | 0,78 | 0,55 | 0,83 | 0,59 |
| +2 | 0,71 | 0,71 | 0,68 | 0,55 | 0,71 | 0,56 |
| +3 | 0,63 | 0,63 | 0,58 | 0,41 | 0,63 | 0,54 |
| +4 | 0,56 | 0,56 | 0,59 | 0,43 | 0,56 | 0,50 |
| +5 | 0,50 | 0,50 | 0,46 | 0,42 | 0,50 | 0,42 |

*Таблица 4*

Динамика индекса Танимото устойчивости алгоритма
методом отбора признаков

*Table 4*

Dynamics of the Tanimoto stability index according to the algorithm
by feature selection method

| Шаг изменения датафрейма | Муравьиная колония | Генетический алгоритм | «Лас-Вегас» | Имитация отжига | Поиск с запретами | «Китовая охота» |
|---|---|---|---|---|---|---|
| 1 | 0,83 | 0,83 | 0,77 | 0,64 | 0,83 | 0,66 |
| 2 | 0,71 | 0,71 | 0,72 | 0,54 | 0,71 | 0,57 |
| 3 | 0,63 | 0,63 | 0,63 | 0,40 | 0,63 | 0,53 |
| 4 | 0,56 | 0,56 | 0,58 | 0,47 | 0,56 | 0,53 |
| 5 | 0,50 | 0,50 | 0,45 | 0,37 | 0,50 | 0,49 |
| 6 | 0,45 | 0,45 | 0,44 | 0,34 | 0,45 | 0,39 |
| 7 | 0,42 | 0,42 | 0,44 | 0,29 | 0,42 | 0,39 |
| 8 | 0,38 | 0,38 | 0,42 | 0,33 | 0,38 | 0,33 |
| 9 | 0,36 | 0,36 | 0,39 | 0,29 | 0,36 | 0,33 |
| 10 | 0,33 | 0,33 | 0,37 | 0,29 | 0,33 | 0,33 |

– почти все алгоритмы включают все 5 переменных (две влияющие и три шумовые) в состав факторов, влияющих на зависимую переменную, при этом все алгоритмы правильно включили все действительно влияющие переменные;

– метод «имитации отжига» является самым перспективным, он в некоторых случаях отсекал одну шумовую переменную;

– методы муравьиной колонии, генетического алгоритма, поиска с запретами показали абсолютную стабильность по наблюдениям и не меняли результаты вычислений при сокращении/увеличении выборки;

– все методы показали неустойчивость к включению/удалению шумовых признаков, что объясняется низкой различительной способностью алгоритмов в данных условиях. Этим также объяснены низкие значения стабильности по Ляпунову.

В целом, можно констатировать, что в условиях проведенного эксперимента наиболее эффективным стал метод поиска с запретами – он не тратит больших временных и вычислительных ресурсов, не отстает от остальных по показателям результативности и показывает стабильность по признакам.

## Заключение

В данной статье была апробирована методика оценки результатов решения задачи отбора признаков для задачи регрессии. Дальнейшими направлениями исследований будет:

– разработка и апробация аналогичной методики для задач классификации;

– доработка и апробация аналогичной методики на реальных задачах;

– реализация данной методики в виде программного пакета на языке R с оптимизированным кодом;

– продолжение экспериментов и разработка рекомендаций по выбору алгоритмов отбора признаков в разных ситуациях.

## Список литературы


1. **Liao T. et al.** Are we learning yet? a meta review of evaluation failures across machine learning // Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2). 2021.
2. **Flach P.** Performance evaluation in machine learning: the good, the bad, the ugly, and the way forward // Proceedings of the AAAI conference on artificial intelligence. 2019. Vol. 33, № 01. P. 9808–9814.
3. **Lazebnik T., Rosenfeld A.** A new definition for feature selection stability analysis // Annals of Mathematics and Artificial Intelligence. 2024. P. 1–18.
4. **Sağbaş E. A.** Performance Evaluation of Feature Selection Methods for Sentiment Classification in Amazon Product Reviews // International Artificial Intelligence And Data Science Congress (ICADA'23). 2023. P. 2.
5. **Mahendran N. et al.** Machine learning based computational gene selection models: a survey, performance evaluation, open issues, and future research directions // Frontiers in genetics. 2020. Vol. 11. P. 603808.
6. **Mohapatra P., Chakravarty S., Dash P. K.** Microarray medical data classification using kernel ridge regression and modified cat swarm optimization based gene selection system // Swarm and Evolutionary Computation. 2016. Vol. 28. P. 144–160.
7. **Abinash M. J., Vasudevan V.** A study on wrapper-based feature selection algorithm for leukemia dataset // Intelligent Engineering Informatics: Proceedings of the 6th International Conference on FICTA. Springer Singapore, 2018. P. 311–321.
8. **Hancer E., Xue B., Zhang M.** Differential evolution for filter feature selection based on information theory and feature ranking // Knowledge-Based Systems. 2018. Vol. 140. P. 103–119.
9. **Ang J. C. et al.** Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection // IEEE/ACM transactions on computational biology and bioinformatics. 2015. Vol. 13. № 5. P. 971–989.
10. **Saeys Y., Inza I., Larranaga P.** A review of feature selection techniques in bioinformatics // Bioinformatics. 2007. Vol. 23, № 19. P. 2507–2517.
11. **Xin B. et al.** Stable feature selection from brain sMRI // Proceedings of the AAAI Conference on Artificial Intelligence. 2015. Vol. 29, № 1.
12. **Bolón-Canedo V., Sánchez-Maroño N., Alonso-Betanzos A.** A review of feature selection methods on synthetic data // Knowledge and information systems. 2013. Vol. 34. P. 483–519.
13. **Sulistiani H. et al.** Performance evaluation of feature selections on some ML approaches for diagnosing the narcissistic personality disorder // Bulletin of Electrical Engineering and Informatics. 2024. Vol. 13, № 2. P. 1383–1391.
14. **Mohammadi M. et al.** Robust and stable gene selection via maximum–minimum correntropy criterion // Genomics. 2016. Vol. 107, № 2–3. P. 83–87.
15. **Черемухин А. Д.** Устойчивость алгоритмов отбора признаков к ошибкам второго рода // Вестник кибернетики. 2021. № 4 (44). С. 78–82. DOI 10.34822/1999-7604-2021-4-78-82. EDN JRYVAF.
16. **Aragón-Royón F. et al.** FSinR: an exhaustive package for feature selection // arXiv preprint arXiv:2002.10330. 2020.
17. **Kashef S., Nezamabadi-pour H.** An advanced ACO algorithm for feature subset selection // Neurocomputing. 2015. Vol. 147. P. 271–279.
18. **Yang J., Honavar V.** Feature subset selection using a genetic algorithm //IEEE Intelligent Systems and their Applications. 1998. Vol. 13, № 2. P. 44–49.

19. **Liu H., Setiono R.** Feature selection and classification–a probabilistic wrapper approach // Industrial and engineering applications or artificial intelligence and expert systems. CRC Press, 2022. P. 419–424.
20. **Posario F., Thangadurai K.** Simulated Annealing Algorithm for Feature Selection // International Journal of Computers & Technology. 2016. Vol. 15, № 2. P. 6471–6479.
21. **Glover F.** Tabu search—part I // ORSA Journal on computing. 1989. Vol. 1, № 3. P. 190–206.
22. **Zamani H., Nadimi-Shahraki M. H.** Feature selection based on whale optimization algorithm for diseases diagnosis // International Journal of Computer Science and Information Security. 2016. Vol. 14, № 9. P. 1243.

## References

1. **Liao T. et al.** Are we learning yet? a meta review of evaluation failures across machine learning. *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021.
2. **Flach P.** Performance evaluation in machine learning: the good, the bad, the ugly, and the way forward. *Proceedings of the AAAI conference on artificial intelligence*, 2019, vol. 33, no. 1, pp. 9808–9814.
3. **Lazebnik T., Rosenfeld A.** A new definition for feature selection stability analysis. *Annals of Mathematics and Artificial Intelligence*, 2024, pp. 1–18.
4. **Sağbaş E. A.** Performance Evaluation of Feature Selection Methods for Sentiment Classification in Amazon Product Reviews. *International Artificial Intelligence And Data Science Congress (ICADA'23)*, 2023, p. 2.
5. **Mahendran N. et al.** Machine learning based computational gene selection models: a survey, performance evaluation, open issues, and future research directions. *Frontiers in genetics*, 2020, vol. 11, p. 603808.
6. **Mohapatra P., Chakravarty S., Dash P. K.** Microarray medical data classification using kernel ridge regression and modified cat swarm optimization based gene selection system. *Swarm and Evolutionary Computation*, 2016, vol. 28, pp. 144–160.
7. **Abinash M. J., Vasudevan V.** A study on wrapper-based feature selection algorithm for leukemia dataset. In: *Intelligent Engineering Informatics: Proceedings of the 6th International Conference on FICTA*. Springer Singapore, 2018, pp. 311–321.
8. **Hancer E., Xue B., Zhang M.** Differential evolution for filter feature selection based on information theory and feature ranking. *Knowledge-Based Systems,* 2018, vol. 140, pp. 103–119.
9. **Ang J. C. et al.** Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. *IEEE/ACM transactions on computational biology and bioinformatics*, 2015, vol. 13, no. 5, pp. 971–989.
10. **Saeys Y., Inza I., Larranaga P.** A review of feature selection techniques in bioinformatics. *Bioinformatics*, 2007, vol. 23, no. 19, pp. 2507–2517.
11. **Xin B. et al.** Stable feature selection from brain sMRI. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015, vol. 29, no. 1.
12. **Bolón-Canedo V., Sánchez-Maroño N., Alonso-Betanzos A.** A review of feature selection methods on synthetic data. *Knowledge and information systems,* 2013, vol. 34, pp. 483–519.
13. **Sulistiani H. et al.** Performance evaluation of feature selections on some ML approaches for diagnosing the narcissistic personality disorder. *Bulletin of Electrical Engineering and Informatics*, 2024, vol. 13, no. 2, pp. 1383–1391.
14. **Mohammadi M. et al.** Robust and stable gene selection via maximum–minimum correntropy criterion. *Genomics*, 2016, vol. 107, no. 2–3, pp. 83–87.

15. **Cheremuhin, A. D.** Stability of Algorithms for Feature Selection to Type II Errors. *Cybernetics Bulletin*. 2021, no. 4(44), pp. 78–82. DOI 10.34822/1999-7604-2021-4-78-82. EDN JRYVAF.
16. **Aragón-Royón F. et al.** FSinR: an exhaustive package for feature selection. In: arXiv preprint arXiv:2002.10330. 2020.
17. **Kashef S., Nezamabadi-pour H.** An advanced ACO algorithm for feature subset selection. *Neurocomputing*, 2015, vol. 147, pp. 271–279.
18. **Yang J., Honavar V.** Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems and their Applications*, 1998, vol. 13, no. 2, pp. 44–49.
19. **Liu H., Setiono R.** Feature selection and classification–a probabilistic wrapper approach. In: *Industrial and engineering applications or artificial intelligence and expert systems*. CRC Press, 2022, pp. 419–424.
20. **Posario F., Thangadurai K.** Simulated Annealing Algorithm for Feature Selection. *International Journal of Computers & Technology*, 2016, vol. 15, no. 2, pp. 6471–6479.
21. **Glover F.** Tabu search—part I. *ORSA Journal on computing*, 1989, vol. 1, no. 3, pp. 190–206.
22. **Zamani H., Nadimi-Shahraki M. H.** Feature selection based on whale optimization algorithm for diseases diagnosis. *International Journal of Computer Science and Information Security*, 2016, vol. 14, no. 9, pp. 1243.

## Информация об авторе

**Черемухин Артем Дмитриевич,** кандидат экономических наук, доцент

**Рейн Андрей Давыдович,** кандидат экономических наук, доцент

## Information about the Authors

**Artem D. Cheremuhin,** Ph.D. in Economics, Associate Professor

**Andrey D. Rein,** Ph.D. in Economics, Associate Professor

*Статья поступила в редакцию 26.02.2025;*
*одобрена после рецензирования 05.04.2025; принята к публикации 05.04.2025*

*The article was submitted 26.02.2025;*
*approved after reviewing 05.04.2025; accepted for publication 05.04.2025*

## Правила оформления текста рукописи

Авторы представляют статьи на русском или английском языке объемом от 0,5 авторского листа (20 тыс. знаков) до 1 авторского листа (40 тыс. знаков), включая иллюстрации (1 иллюстрация форматом 190 × 270 мм = 1/6 авторского листа, или 6,7 тыс. знаков). Публикации, превышающие указанный объем, допускаются к рассмотрению только после индивидуального согласования с редакцией журнала.

Текст рукописи должен быть представлен в редколлегию в виде файла MS Word (.doc, .docx). Гарнитура Times New Roman, размер шрифта 11, межстрочный интервал 1, размеры полей – стандартные значения текстового редактора. Форматирование – выравнивание по ширине страницы, переносы слов включены, каждый новый абзац начинается с красной строки. Не допускается ручное форматирование абзацев (пробелами, лишними переводами строк, разрывами страниц).

## Структура статьи

- Индекс УДК (универсальной десятичной классификации). Выравнивание по левому краю
- Название статьи. Выравнивание по центру, полужирный шрифт
- ФИО авторов (полностью). Выравнивание по центру, полужирный шрифт
- Места работы всех авторов. Выравнивание по центру, курсив
- Адреса электронной почты, ORCID авторов
- Аннотация статьи
- Ключевые слова, не более 10
- Благодарности, сведения о финансовой поддержке
- Название статьи **на английском языке**. Выравнивание по центру, полужирный шрифт
- ФИО авторов **на английском языке** (полностью). Выравнивание по центру, полужирный шрифт
- Места работы авторов **на английском язык**е. Выравнивание по центру, курсив
- Аннотация статьи **на английском языке (Abstract)**, 200–250 слов
- Ключевые слова **на английском языке (Keywords)**, не более 10
- Благодарности, сведения о финансовой поддержке **на английском язык**е, если есть соответствующий раздел на русском языке **(Acknowledgements)**
- Основной текст
- Список литературы / **References**
- Сведения об авторах

## Требования к оформлению основного текста и иллюстративных материалов

Основной текст должен быть представлен в структурированном виде, рекомендуется использовать подзаголовки – например: Введение, Методика..., Выводы, Результаты, Заключение.

Подзаголовки отделяются и набираются полужирным шрифтом. В целях выделения частей текста и отдельных слов и словосочетаний допускается использование курсива или полужирного шрифта. Подчеркивание, разрядка, изменение основного кегля и выделение цветом не используются.

Иллюстрации к рукописи статьи должны быть приложены в виде отдельных файлов. При этом в тексте должно содержаться включенное изображение с указанием имени файла. Все иллюстрации, содержащие схемы, графики, алгоритмы и т. п., должны быть представлены в векторном виде (.ai, .eps, .cdr). Скриншоты и другие растровые изображения должны быть представлены в максимально высоком качестве, без каких-либо потерь и искажений (.jpg, .tif). Все иллюстрации должны иметь подрисуночную подпись – свое название. Надписи к таблицам и подписи к иллюстрациям приводятся **на двух языках (русском и английском)**.

Примеры:

*Рис. 1.* Диаграмма производительности...
*Fig. 1.* Performance diagram...

*Таблица 1*

Сравнение алгоритмов...

*Table 1*

Comparison of algorithms...

Нумерация последовательная и неразрывная от начала статьи. Не допускается использование других наименований, кроме «Рис.» / «Fig.», «Таблица» / «Table», и усложнение нумерации (например, «Рис. 3.2.»). Ссылка на иллюстрацию в тексте должна быть приведена в круглых скобках, например: (рис. 1), (табл. 1).

Формулы должны быть набраны с использованием редактора MathType либо встроенного редактора формул MS Word. Кегль основных символов – 11, греческие символы набираются прямым шрифтом, латинские – курсивом. Нумеруются только те формулы, на которые автор ссылается в тексте.

## Abstract

Аннотация статьи на английском языке (Abstract) не должна быть дословным переводом русскоязычной аннотации. Раздел Abstract, как и основной текст, должен быть структурирован, в нем должно содержаться описание цели работы, методов исследования, научной значимости, выводов / результатов. Требуется качественный перевод на английский язык (при необходимости просим авторов обращаться к профессиональным переводчикам). **Объем Abstract 200–250 слов.**

## Список литературы / References

Список литературы и список литературы на английском языке (References) размещаются в общем разделе. Рекомендуемое количество цитируемых в статье источников – не менее 10, в список желательно включать ссылки на актуальные работы по теме исследования, особенно в иностранных периодических изданиях.

В тексте статьи ссылки на литературу указываются цифрами в квадратных скобках, при необходимости указываются номера страниц, например: [2; 3. С. 15].

Список литературы нумеруется в порядке цитирования и оформляется в соответствии с ГОСТ Р 7.0.5-2008 на библиографическое описание (знаки тире в описании опускаются). Ссылки на неопубликованные работы, а также на интернет-ресурсы (кроме электронных изданий, поддающихся библиографическому описанию) оформляются в виде сноски.

В Список литературы ссылки на источники следует включать на оригинальном языке опубликования. Каждый источник должен быть также оформлен на английском языке (References) по международному стандарту для публикаций в области информатики IEEE Style со следующими отличиями:

- инициалы авторов указываются после фамилии;
- название статьи не берется в кавычки, отделяется точкой;

- отсутствует союз «and» перед фамилией последнего автора;
- в диапазоне страниц – удвоенная «p» (например, «pp. 2–9»);
- год издания указывается после места издания (для книг) и сразу после названия журнала (для периодики).
- Перевод источника на английский язык:
- если источник имеет выходные данные на английском языке, то для формирования References **следует использовать именно эти данные**;
- если оригинальная публикация не содержит выходных данных на английском языке, то допускается транслитерация названия материала на латинский алфавит в сочетании с переводом на английский язык в квадратных скобках. В конце описания указывается, на каком языке написана эта работа, например, (in Russ.). При транслитерации можно воспользоваться интернет-ресурсом http://ru.translit.ru/, рекомендуется выбрать стандарт BSI. Место издания не транслитерируется, указывается полностью на английском языке, например: Moscow. Название издательства / издателя, как правило, транслитерируется. Для журналов, у которых есть официальное название на английском языке, – использовать его (проверить на сайте журнала, или, например, в библиотеке WorldCat), если названия на английском языке нет, использовать транслитерацию по системе BSI. Не следует самостоятельно переводить названия журналов.

Если у цитируемого источника есть **цифровой идентификатор DOI** (https://search. crossref. org), его требуется обязательно указывать в конце библиографической ссылки.

Примеры оформления ссылок. Каждый источник в том же пункте дублируется на английском языке (References).

***Источник на русском языке, перевод на английский доступен в метаданных статьи***

1. **Журавлев С. С., Рудометов С. В., Окольнишников В. В., Шакиров С. Р.** Применение модельно-ориентированного проектирования к созданию АСУ ТП опасных промышленных объектов // Вестник НГУ. Серия: Информационные технологии. 2018. Т. 16, № 4. С. 56–67. DOI 10.25205/1818-7900-2018-16-4-56-67

**Zhuravlev S. S., Rudometov S. V., Okolnishnikov V. V., Shakirov S. R.** Model-Based Design Approach for Development Process Control Systems of Hazardous Industrial Facilities. *Vestnik NSU. Series: Information Technologies*, 2018, vol. 16, no. 4, pp. 56–67. (in Russ.) DOI 10.25205/ 1818-7900-2018-16-4-56-67

***Источник на английском языке. Оформляем согласно требованиям для References. Приводим только 1 раз.***

2. **Telnov V. I.** Optimization of the Beam Crossing Angle at the ILC for E + e- and yy Collisions. *Journal of Instrumentation*, 2018, vol. 13, no. 03, pp. P03020–P03020. DOI 10.1088/1748-0221/13/03/ p03020

***Метаданные источника доступны только на русском языке***

3. **Жижимов О. Л., Федотов А. М., Шокин Ю. И.** Технологическая платформа массовой интеграции гетерогенных данных // Вестник НГУ. Серия: Информационные технологии. 2013. Т. 11, вып. 1. С. 24–41.

**Zhizhimov O. L., Fedotov A. M., Shokin Yu. I.** Tekhnologicheskaya platforma massovoi integratsii geterogennykh dannykh [Technology Platform for the Mass Integration of Heterogeneous Data]. *Vestnik NSU. Series: Information Technologies*, 2013, vol. 11, no. 1, pp. 24–41. (in Russ.)

## Сведения об авторах

Последний раздел статьи – информация об авторе / авторах **на русском и английском языках**:

- ФИО полностью, ученая степень, ученое звание;
- идентификаторы автора, такие как ResearcherID (всем авторам рекомендуется использовать данные сервисы для ведения актуального списка своих публикаций);
- контактный телефон (не публикуется).

Если статья представляется на английском языке, необходимо приложить перевод на русский язык названия, аннотации, ключевых слов, сведений об авторе.

## Доставка материалов

Материалы предоставляются в редакцию по электронной почте inftech@vestnik.nsu.ru.

## Порядок рецензирования

Все статьи сначала проходят проверку на заимствование и только после этого отправляются на рецензирование. Редакционный совет не допускает к публикации материал, если имеется достаточно оснований полагать, что он является плагиатом.

Тип рецензирования статей – двухуровневое, одностороннее анонимное («слепое»).

Для каждой статьи редколлегией выбираются рецензенты, научная деятельность которых связана с темой представленного материала. Ответственный секретарь журнала обращается к ним с просьбой дать экспертную оценку статье либо помочь организовать рецензирование.

Рецензии для журнала «Вестник НГУ. Серия: Информационные технологии» составляются по единой схеме и подразумевают оценку по следующим критериям: соответствие тематике журнала, оригинальность и значимость результатов, качество изложения материала.

Заполненный бланк рецензии высылается на электронный адрес редакции. В зависимости от экспертных заключений статья может быть принята редакционным советом к опубликованию, рекомендована автору к доработке (с последующим повторным рецензированием либо без него) или отклонена (с предоставлением автору мотивированного отказа). Автору на электронный адрес высылается текст рецензии без указания ФИО рецензента и его контактных данных.

Все рецензии хранятся в редакции журнала не менее 5 лет. Редколлегия журнала обязуется при поступлении соответствующего запроса направлять копии рецензий в Министерство науки и высшего образования Российской Федерации.