ISSN 2220-3397

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Программная инженерия

ПрИн
2
2022
Том 13

Рисунки к статье **А. М. Вульфина**
«ОБНАРУЖЕНИЕ СЕТЕВЫХ АТАК В ГЕТЕРОГЕННОЙ ПРОМЫШЛЕННОЙ СЕТИ НА ОСНОВЕ ТЕХНОЛОГИЙ МАШИННОГО ОБУЧЕНИЯ»

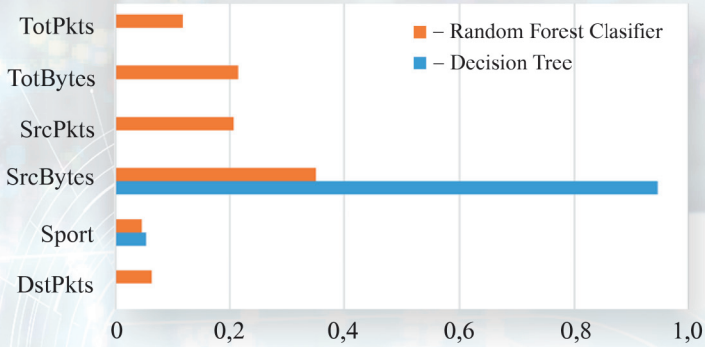


Рис. 6. Оценка значимости признаков с помощью классификатора на основе дерева решений и классификатора на основе комитета случайных деревьев

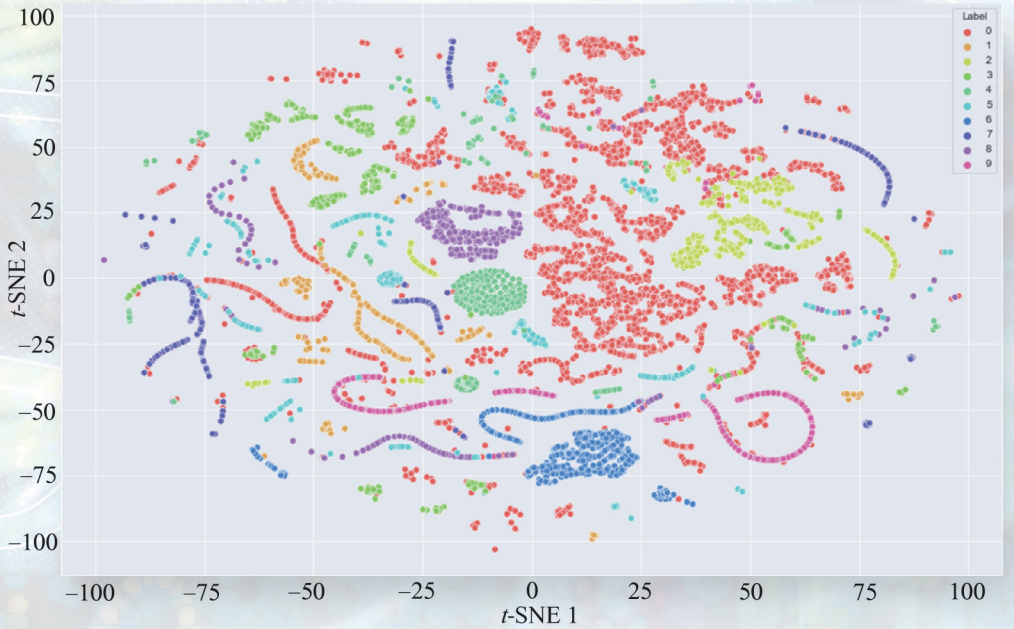


Рис. 8. Визуализация распределения примеров по классам *t*-распределением (по осям – компоненты *t*-SNE-разложения в диапазоне [–100, 100], 0–9 – метки классов)

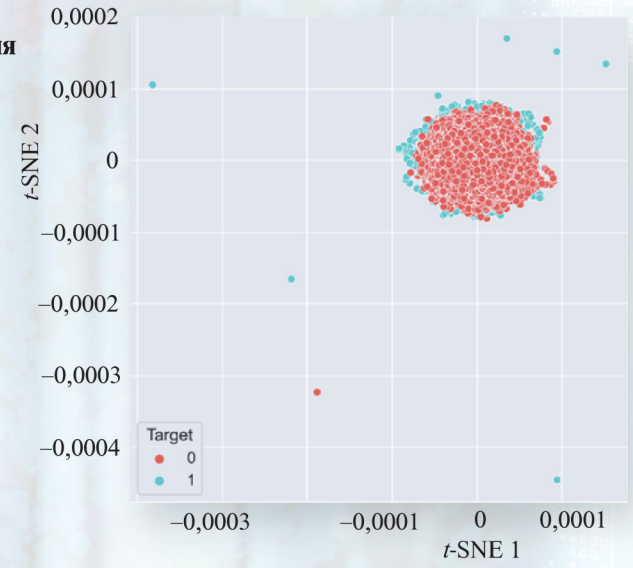


Рис. 9. Визуализация пространства признаков с помощью стохастического вложения соседей с *t*-распределением: 0 – «нормальная работа», 1 – «атака» (по осям – компоненты *t*-SNE-разложения в диапазоне [-100, 100])

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Бибило П. Н., Романов В. И. Экспериментальное исследование алгоритмов минимизации BDDI-представлений систем булевых функций с использованием алгебраических разложений кофакторов 51

Вульфин А. М. Обнаружение сетевых атак в гетерогенной промышленной сети на основе технологий машинного обучения 68

Читалов Д. И. О разработке модуля для работы с решателем buoyantSimpleFoam и утилитой postProcess платформы OpenFOAM 81

Коротышева А. А., **Жуков С. Н.** Реализация алгоритмов дополненной реальности в навигации автомобильного транспорта с использованием открытых сервисов 88

Капралов Н. С., Морозов А. Ю., Никулин С. П. Параллельная аппроксимация многомерных тензоров с использованием графических процессоров 94

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в подписных агентствах (индекс по Объединенному каталогу "Пресса России" — **22765**) или непосредственно в редакции (для юридических лиц).
Тел.: (499) 270-16-52.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования и базу данных RSCI на платформе Web of Science.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 13
N 2
2022

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

CONTENTS

Bibilo P. N., Romanov V. I. Experimental Study of Algorithms for Minimization of Binary Decision Diagrams using Algebraic Representations of Cofactors 51

Vulfin A. M. Detection of Network Attacks in a Heterogeneous Industrial Network Based on Machine Learning 68

Chitalov D. I. On the Development of a Module for Working with the buoyantSimpleFoam Solver and the postProcess Utility of the OpenFOAM Platform 81

Korotysheva A. A., Zhukov S. N. Implementation of Augmented Reality Algorithms in Road Transport Navigation Using Open Services . . . 88

Kapralov N. S., Morozov A. Yu., Nikulin S. P. Parallel Approximation of Multivariate Tensors using GPUs 94

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 519.714 DOI: 10.17587/prin.13.51-67

П. Н. Бибило, д-р техн. наук, проф., зав. лаб., bibilo@newman.bas-net.by,
В. И. Романов, вед. науч. сотр., доц., rom@newman.bas-net.by,
Объединенный институт проблем информатики Национальной академии наук Беларуси, Минск

Экспериментальное исследование алгоритмов минимизации BDDI-представлений систем булевых функций с использованием алгебраических разложений кофакторов

Рассматриваются многоуровневые разложения Шеннона систем полностью определенных булевых функций, графической формой задания которых являются модификации бинарных диаграмм решений (BDD - Binary Decision Diagram), называемые BDDI (Binary Decision Diagram with Inverse cofactors). В отличие от BDD, в BDDI могут быть пары взаимно инверсных подфункций (кофакторов). Описываются программно реализованные алгоритмы дополнительной логической оптимизации BDDI-представлений систем булевых функций на основе поиска алгебраических разложений кофакторов одного уровня BDDI в виде дизъюнкции либо конъюнкции других инверсных либо безынверсных кофакторов данного уровня BDDI. Приведены результаты применения соответствующих программ при синтезе логических схем в библиотеке проектирования заказных сверхбольших интегральных схем.

Ключевые слова: *система булевых функций, дизъюнктивная нормальная форма (ДНФ), Binary Decision Diagram (BDD), разложение Шеннона, синтез логической схемы, VHDL, СБИС*

Введение

Автоматизация логического проектирования функциональных блоков комбинационной логики, входящих в состав заказных цифровых СБИС (сверхбольших интегральных схем), по прежнему остается актуальной научной проблемой, так как размерности решаемых задач возрастают, появляются более жесткие требования на энергопотребление логических схем, изменяются библиотеки логических элементов. Логическое проектирование комбинационных схем в библиотеках проектирования заказных СБИС традиционно разбивается на два этапа — логическую оптимизацию и технологическое отображение, на котором оптимизированные логические функциональные описания проектируемых логических схем покрываются функциональными описаниями логических элементов. Логическая оптимизация, часто называемая технологически независимой оптимизацией, обычно не ориентируется на используемый базис (библиотеку) логических элементов, а стремится уменьшить число литералов и число двухоперандных операций дизъюнкции и конъюнкции в минимизированных алгебраических представлениях систем полностью определенных булевых функций. В практике проектирования давно было замечено [1, с. 44], что сокращение числа литералов и логических операций в функциональных описаниях положительно влияет на сокращение в схеме числа элементов технологической библиотеки (базиса синтеза), т. е. по сути, на сокращение суммарного числа транзисторов, входящих во все элементы схемы. Уменьшение числа транзисторов позволяет уменьшать энергопотребление схем.

Для логической минимизации в настоящее время широко используются методы оптимизации многоуровневых представлений булевых функций на основе разложения Шеннона, графические формы таких представлений, часто используемые как структуры данных, называются бинарными диаграммами решений [2—7]. В работе [8] был предложен метод дополнительной минимизации BDD-представлений систем булевых функций, основанный на поиске дизъюнктивных, конъюнктивных и модулярных разложений подфункций, входящих в оптимизированные BDD-представления. Метод позволяет заменять формулы разложения Шеннона двухоперандными формулами дизъюнкции $g_3 = g_1 \vee g_2$, конъюнкции $g_3 = g_1 \,\&\, g_2$ либо модулярными формулами вида $g_3 = g_1 \oplus g_2$ (здесь g_1, g_2, g_3 — кофакторы одного и того же уровня BDDI). Такая замена уменьшает число литералов и логических операций в резуль-

тирующих логических формулах, по которым выполняется технологическое отображение, т. е. заключительный этап синтеза схемы. В настоящей работе метод [8] доведен до алгоритмов и программ, эффективность которых исследована на потоках примеров, входящих в известную библиотеку примеров, используемых для проверки и сравнения программ логической оптимизации и синтеза логических схем. Для разложений кофакторов используются только дизъюнктивные и конъюнктивные разложения, причем кофакторы в таких разложениях могут быть как в инверсной, так и безынверсной форме.

Бинарные диаграммы решений и алгебраические разложения кофакторов

Разложением Шеннона полностью определенной булевой функции $f = f(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, ..., x_n)$ по переменной x_i называется представление

$$f = f(\mathbf{x}) = \bar{x}_i f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \vee \vee x_i f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n). \quad (1)$$

Функции $f_0 = f(x_1, ..., x_{i-1}, 0, x_{i+1}, ... x_n)$, $f_1 = f(x_1, ..., x_{i-1}, 1, x_{i+1}, ... x_n)$ в правой части (1) называются *кофакторами* (*cofactors*, англ.) разложения по переменной x_i. Они получаются из функции f подстановкой вместо переменной x_i константы 0 и 1 соответственно. Каждый из кофакторов f_0 и f_1 может быть разложен по одной из переменных из множества $\{x_1, ..., x_{i-1}, x_{i+1}, ... x_n\}$. Процесс разложения кофакторов заканчивается, когда все n переменных будут использованы для разложения, либо когда все кофакторы выродятся до констант 0, 1. На каждом шаге разложения выполняется сравнение на равенство полученных кофакторов.

Пусть $\mathbf{f}(\mathbf{x})$ — упорядоченная система полностью определенных компонентных булевых функций $f_i(\mathbf{x})$, $i = 1, ..., m$ (векторная булева функция $\mathbf{f} = (f_1, ..., f_m)$): $\mathbf{f}(\mathbf{x}) = \left(f_1(\mathbf{x}), ... f_m(\mathbf{x})\right)$, $\mathbf{x} = (x_1, x_2, ..., x_n)$. Под BDD-представлением векторной булевой функции $\mathbf{f}(\mathbf{x})$ понимается ориентированный ациклический граф, задающий последовательные разложения Шеннона всех компонентных функций $f_i(\boldsymbol{x})$, $i = 1, ..., m$, по всем ее переменным $x_1, ..., x_n$ при одном и том же заданном порядке (перестановке) переменных, по которым проводятся разложения. Рассматриваемые в статье BDD соответствуют совместным (*shared*) сокращенным упорядоченным BDD для систем функций (*Reduced Ordered BDD*, ROBDD) [5, с. 18]. В совместных ROBDD функциональным вершинам BDD соответствуют кофакторы, общие для функций системы, при этом функциональные вершины лишь подразумеваются (отождествляются с вершинами-переменными). Описание OBDD (упорядоченных BDD) дано в работах [2, с. 90, 5, с. 16], ROBDD — в работе [3, с. 243]. Далее под BDD будем понимать совместные ROBDD для систем функций (векторных функций).

BDD-представлению соответствуют формулы разложения Шеннона, каждой функциональной вершине BDD [6, с. 18] соответствует своя формула, в которой оба кофактора выступают в безынверсной форме. По BDD-представлению можно найти задание каждой из компонентных функций $f_i(\mathbf{x})$ в виде двух ортогонализованных ДНФ (дизъюнктивных нормальных форм): одна из таких ДНФ задает область $M^1_{f_i}$ единичных значений функции $f_i(\mathbf{x})$, другая ДНФ — область $M^0_{f_i}$ нулевых значений. Аналогично можно найти представления в виде ДНФ каждого из кофакторов. Данные переходы подробно описаны в работе [6, с. 37]. На рис. 1, *а* изображена BDD, которой соответствуют следующие взаимосвязанные уравнения разложения Шеннона:

$$f_1 = \bar{x}_3 g_1 \vee x_3 x_1; \quad f_2 = \bar{x}_3 g_2 \vee x_3 g_4;$$

$$f_3 = \bar{x}_3 g_5 \vee x_3 g_3; \quad f_4 = \bar{x}_3 \vee x_3 g_6;$$

$$g_1 = \bar{x}_1 s_1 \vee x_1 s_2; \quad g_2 = \bar{x}_1 s_1; \quad g_3 = x_1 s_2;$$

$$g_4 = \bar{x}_1 s_2; \quad g_5 = x_1 s_1; \quad g_6 = \bar{x}_1 s_2 \vee x_1 s_1;$$

$$s_1 = x_2; \quad s_2 = \bar{x}_2.$$

Кофакторы $g_1, ..., g_6$ находятся на втором уровне BDD и зависят от двух переменных x_1, x_2, кофакторы на первом уровне зависят от переменной x_2: $s_1 = x_2$, $s_2 = \bar{x}_2$ (см. рис. 1, *а*). На нижнем уровне BDD всегда находятся константы 0, 1, которые могут дублироваться для упрощения изображения графа BDD на рисунке. На рис. 1, *б* показан граф той же BDD в символике, принятой в зарубежной литературе: штриховые линии, имеющиеся на рис. 1, соответствуют пометкам 0, сплошные — пометкам 1. По графу BDD легко найти кофакторы, находящиеся на одном уровне BDD, а от задания кофактора (либо функции исходной системы) в виде ДНФ можно перейти к его заданию в виде СДНФ (совершенной ДНФ) либо таблицы истинности. Рассмотрим кофакторы $g_1, g_2, g_3, g_4, g_5, g_6$ второго уровня BDD (рис. 1, *а*), элиминируем промежуточные переменные, получим совершенные ДНФ:

$$g_1 = \bar{x}_1 s_1 \vee x_1 s_2 = \bar{x}_1 x_2 \vee x_1 \bar{x}_2;$$

$$g_2 = \bar{x}_1 s_1 = \bar{x}_1 x_2; \quad g_3 = x_1 s_2 = x_1 \bar{x}_2;$$

$$g_4 = \bar{x}_1 s_2 = \bar{x}_1 \bar{x}_2; \quad g_5 = x_1 s_1 = x_1 x_2;$$

$$g_6 = \bar{x}_1 s_2 \vee x_1 s_1 = \bar{x}_1 \bar{x}_2 \vee x_1 x_2.$$

Зададим СДНФ кофакторов и их инверсий таблицей истинности (табл. 1).

По таблицам истинности легко проверить представления кофакторов одного и того же уровня BDD в виде *алгебраических разложений*:

дизъюнктивного

$$g_p = g_i \vee g_j \quad (2)$$

либо *конъюнктивного*

$$g_r = g_i \,\&\, g_j. \quad (3)$$

Можно также использовать инверсии $\bar{g}_i$, $\bar{g}_j$ кофакторов g_i, g_j в разложениях вида (2), (3). Для кофакто-

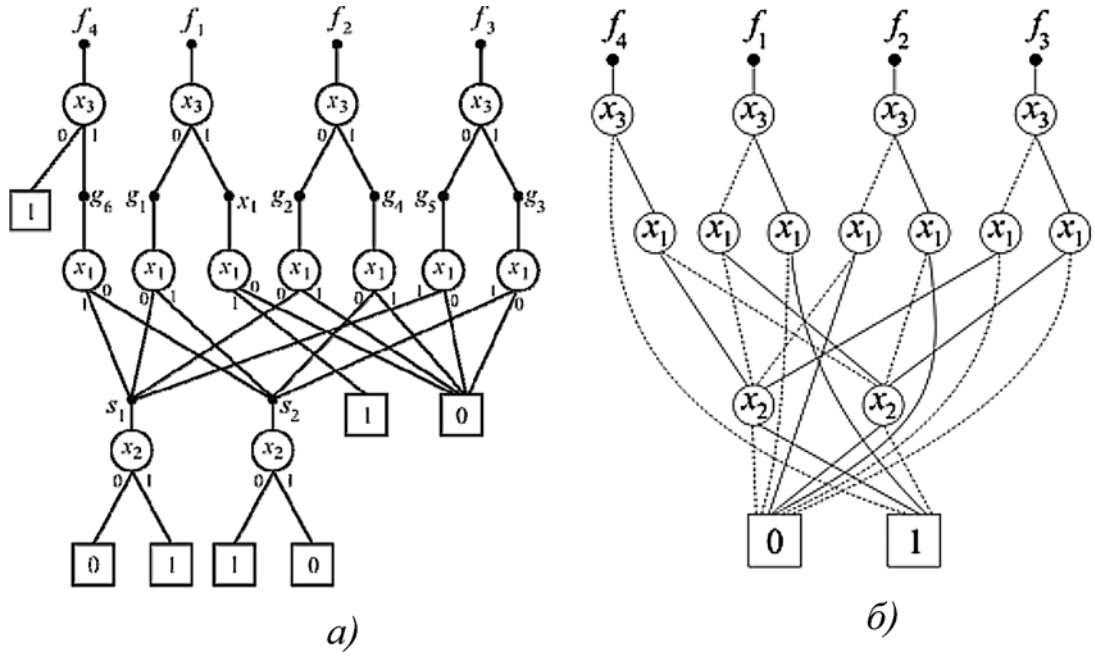


Рис. 1. BDD:
а — с выделенными кофакторами; *б* — традиционное изображение

Таблица 1

Кофакторы второго уровня BDD (рис. 1)

$x_1\ x_2$	g_1	g_2	g_3	g_4	g_5	g_6	$\overline{g}_1$	$\overline{g}_2$	$\overline{g}_3$	$\overline{g}_4$	$\overline{g}_5$	$\overline{g}_6$
0 0	0	0	0	1	0	1	1	1	1	0	1	0
0 1	1	1	0	0	0	0	0	0	1	1	1	1
1 0	1	0	1	0	0	0	0	1	0	1	1	1
1 1	0	0	0	0	1	1	1	1	1	1	0	0

ров, которые имеют в своей записи четыре литерала (это кофакторы g_1, g_6), запишем уравнения их алгебраических разложений через другие кофакторы:

$$g_1 = \overline{g}_4 \,\&\, \overline{g}_5;\quad g_1 = g_2 \vee g_3;\quad g_6 = \overline{g}_2 \,\&\, \overline{g}_3;$$

$$g_6 = g_4 \vee g_5$$

и представим полученные уравнения в виде орграфа (рис. 2).

Легко видеть (рис. 3), что кофакторы $g_1 = \overline{x}_1 s_1 \vee x_1 s_2$, $g_6 = \overline{x}_1 s_2 \vee x_1 s_1$ могут быть выражены формулами алгебраических разложений через два кофактора g_2, g_3: $g_1 = g_2 \vee g_3$; $g_6 = \overline{g}_2 \,\&\, \overline{g}_3$. Замена формул $g_1 = \overline{x}_1 s_1 \vee x_1 s_2$, $g_6 = \overline{x}_1 s_2 \vee x_1 s_1$ более простыми позволяет сокращать число литералов и логических операций в функциональном описании BDD (см. рис. 1, *а*).

Заметим, что кофакторы g_1, g_6 являются взаимно инверсными, поэтому далее в качестве исходных рассматриваются формулы разложений Шеннона, соответствующие BDDI. Под BDDI (***B**inary **D**ecision **D**iagram with **I**nverse cofactors*) далее понимается ориентированный ациклический граф, задающий последовательные разложения Шеннона булевой функции $f(x_1, ..., x_n)$ по всем ее переменным $x_1, ..., x_n$ при заданном порядке (перестановке) переменных, по которым проводятся разложения, при условии нахождения пар взаимно инверсных кофакторов [9].

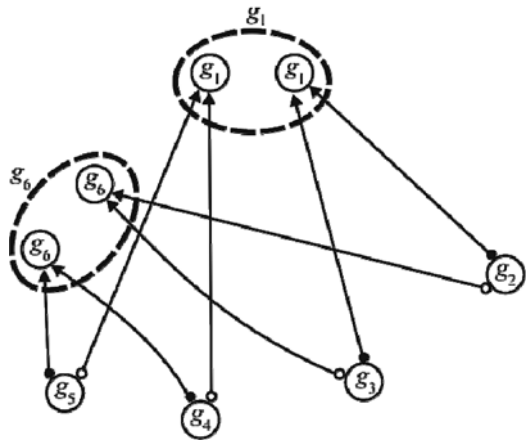


Рис. 2. Задание уравнений в виде орграфа

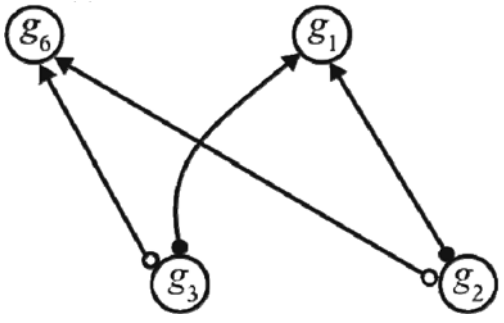
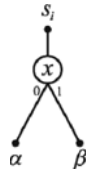


Рис. 3. Сокращенный бесконтурный орграф

Таблица 2

Соответствие графических элементов BDDI и логических формул

Графический элемент	Формула
	$s_i = \bar{x}\alpha \vee x\beta$
	$s_i = \bar{x}\alpha \vee x\beta$ $s_j = \bar{s}_i$

Графы BDDI содержат четыре вида вершин: корневые вершины, соответствующие функциям; функциональные вершины, соответствующие парам разлагаемых взаимно инверсных кофакторов (один из элементов пары может отсутствовать, если такового нет среди кофакторов на данном уровне BDDI); вершины-переменные; листовые вершины, соответствующие константам 0, 1. Функциональная вершина BDDI (табл. 2) реализует один кофактор s_i либо два кофактора — s_i и инверсию $\bar{s}_i$. Формула $s_i = \bar{x}\alpha \vee x\beta$ в правом столбце табл. 2 является формулой (1) разложения Шеннона кофактора s_i по переменной x: $\alpha = f_0$, $\beta = f_1$.

Заметим, что в BDD функциональная вершина реализуют один кофактор, взаимно инверсные кофакторы представляются парой вершин. BDDI соответствуют совместным ROBDD, в которых пара взаимно инверсных вершин-кофакторов задается одной функциональной вершиной. В уравнениях разложений Шеннона, соответствующих BDDI, кофакторы могут быть как со знаком инверсии, так и в безынверсной форме.

Далее предлагаются алгоритмы нахождения максимальных множеств алгебраических разложений вида (2), (3) кофакторов, находящихся на одном уровне BDDI. Основная проблема заключается в том, что при задании алгебраических уравнений в виде орграфа могут возникать контуры, что недопустимо при схемной реализации комбинационной логики. В рассматриваемом простом примере орграфа (см. рис. 2), иллюстрирующем задание уравнений, контуров нет. Переход к результирующему бесконтурному орграфу (см. рис. 3), в котором каждый кофактор представляется только одним уравнением алгебраического разложения, является сложной комбинаторной задачей для графов, содержащих сотни и тысячи вершин, и в которых может быть огромное число контуров, от которых надо избавляться. Предварительное нахождение пар взаимно инверсных кофакторов (использование BDDI вместо BDD) сокращает комбинаторный перебор при нахождении вариантов алгебраических разложений кофакторов, далее это будет пояснено.

Алгоритм 1 нахождения алгебраических разложений кофакторов в BDDI

Исходными данными для предлагаемого алгоритма 1 нахождения алгебраических представлений кофакторов являются граф BDDI, представляющий исходную векторную полностью определенную булеву функцию, и логические уравнения (формулы разложения Шеннона), соответствующие функциональным вершинам BDDI и задающие многоуровневое описание компонентных функций. Данные уравнения легко могут быть записаны по графу BDDI.

Предлагаемый алгоритм включает этапы, выполняемые для кофакторов каждого уровня BDDI, за исключением корневого, листового и первого. На первом уровне BDDI располагаются кофакторы, получаемые в результате разложения по последней переменной в заданной перестановке переменных, по которым ведется разложение Шеннона. Кофакторы первого уровня BDDI зависят от одной переменной, кофакторы второго уровня зависят от не более чем двух переменных и т. д. В алгоритме уровни BDDI рассматриваются "сверху вниз", т. е. от кофакторов, получаемых при разложении по первой переменной, и заканчивая кофакторами второго уровня BDD. Алгоритм ориентирован на замену наибольшего числа формул разложения Шеннона формулами дизъюнкции либо конъюнкции, что позволяет сокращать общее число литералов в многоуровневом задании системы булевых функций. Для рассматриваемого уровня BDDI решается задача нахождения наибольшего числа кофакторов, которые представимы в виде дизъюнкции либо конъюнкции других кофакторов данного уровня (ранее такие представления были названы алгебраическими разложениями).

Этап 1. Разбить множество уравнений по уровням BDDI.

Этап 2. Для каждого уровня BDDI (за исключением корневого, листового и первого) найти максимальное множество реализуемых уравнений (реализумых кофакторов).

Выполнение этапа 2 сводится к следующим шагам.

Шаг 2.1. Элиминировать промежуточные переменные и построить таблицу истинности кофакторов заданного (рассматриваемого) уровня BDDI.

Шаг 2.2. Добавить в таблицу истинности инверсии кофакторов.

Шаг 2.3. Найти все варианты алгебраических представлений безынверсных кофакторов в виде двухоперандных дизъюнкций либо конъюнкций других кофакторов (либо их инверсий) рассматриваемого уровня BDDI. В результате один и тот же кофактор, назовем его *представимым*, может быть представлен различными уравнениями.

Шаг 2.4. Найти множество реализуемых кофакторов.

Из множества представимых кофакторов выделяется максимальное по мощности подмножество *реализуемых* кофакторов, для этого из множества уравнений алгебраического представления одного и того же кофактора выбирается одно уравнение,

либо уравнение не выбирается. Этот выбор определяется решением задачи 1, которая сформулирована в терминах теории графов.

Этап 3. Скорректировать исходное многоуровневое BDDI-представление системы булевых функций.

Исходное множество формул разложений Шеннона изменяется — формулы разложения Шеннона реализуемых кофакторов заменяются найденными формулами дизъюнкции либо конъюнкции.

Оценим число перебираемых вариантов на шаге 2.3. Пусть k — число безынверсных кофакторов g_1, ..., g_k. Тогда требуется рассмотреть $12 \times C_k^3$ (C_k^3 — число сочетаний из k кофакторов по 3, $k \geqslant 3$) вариантов проверки дизъюнктивных разложений: число всех различных неупорядоченных троек кофакторов g_p, g_i, g_j равно C_k^3, а для каждой тройки $\{g_p, g_i, g_j\}$ надо проверять 12 дизъюнктивных разложений:

$$g_p = \overline{g}_i \vee g_j;\ \ g_p = g_i \vee \overline{g}_j;\ \ g_p = \overline{g}_i \vee \overline{g}_j;\ \ g_p = g_i \vee g_j;$$

$$g_i = \overline{g}_p \vee g_j;\ \ g_i = g_p \vee \overline{g}_j;\ \ g_i = \overline{g}_p \vee \overline{g}_j;\ \ g_i = g_p \vee g_j;$$

$$g_j = \overline{g}_i \vee g_p;\ \ g_j = g_i \vee \overline{g}_p;\ \ g_j = \overline{g}_i \vee \overline{g}_p;\ \ g_j = g_i \vee g_p.$$

Аналогично, для каждой из C_k^3 троек кофакторов надо перебирать 12 вариантов для построения конъюнктивных разложений.

Теперь легко объяснить, что использование BDDI вместо BDD сокращает комбинаторный перебор вариантов при нахождении формул дизъюнктивных и конъюнктивных разложений. Например, если предварительно не установлено, что кофакторы g_p, g_s являются взаимно инверсными, то при поиске разложений и нахождении формулы $g_p = g_i \vee g_j$ для инверсии $\overline{g}_p = g_s$ всегда будет найдена двойственная формула $g_s = \overline{g}_i \,\&\, \overline{g}_j$. Таким образом, для каждой формулы алгебраического разложения безынверсного кофактора будет найдена двойственная формула для инверсного кофактора и перебор троек кофакторов станет более трудоемким.

Нахождение множества реализуемых кофакторов

Построим ориентированный граф G, задающий формулы алгебраических представлений кофакторов (будем называть их логическими уравнениями либо просто уравнениями). Вершинам орграфа G соответствуют кофакторы в прямой форме, которые упоминаются в уравнениях. Обозначим это множество кофакторов K_Y, а множество кофакторов, для которых не построены уравнения, — K_Z. Дуги (ориентированные ребра), соответствующие операндам алгебраически представленного кофактора (результату логической операции), помечены одним и тем же числом — номером уравнения. Каждое уравнение задается подграфом с тремя вершинами и двумя помеченными дугами: вершина, соответствующая представимому кофактору, имеет две заходящие дуги (с одной и той же пометкой — номером уравнения). Исходящие из вершины представимого кофактора дуги (см. рис. 3) могут соответствовать прямой форме кофактора (черный кружок) либо инверсной форме кофактора (светлый кружок).

Введем понятие *кластера* (подмножества вершин графа G): в кластер входят вершины, соответствующие уравнениям с одним и тем же кофактором в левой части. Именем одинаковых кофакторов и обозначается кластер.

С использованием орграфа G задача нахождения множества реализуемых кофакторов сводится к следующей задаче.

Задача 1. В орграфе G требуется оставить в каждом кластере только одну вершину (условие A) и удалить из графа G такое подмножество пар одинаково помеченных дуг, чтобы орграф G стал бесконтурным (условие B) и содержал наибольшее число неизолированных вершин, в каждую из которых заходит только одна пара одинаково помеченных дуг (условие C).

Смысл оставления в кластере одной вершины (условие A) очевиден — кофактор при схемной реализации достаточно представить только одним уравнением. Требование отсутствия контуров (условие B) продемонстрируем на примере двух уравнений $g_6 = g_1 \vee g_4$, $g_1 = g_6 \,\&\, g_7$ из рассматриваемого далее примера. Данные уравнения в орграфе G создают контур, т. е. логическое противоречие в требовании алгебраического (и схемного) представления кофакторов, заключающееся в следующем. Чтобы выразить g_6 в виде конъюнкции $g_6 = g_1 \vee g_4$, требуется схема, реализующая кофактор g_1. Однако чтобы получить g_1 в виде $g_1 = g_6 \,\&\, g_7$, требуется схема, реализующая g_6. Условие C требует, чтобы как можно больше кофакторов было представлено в виде конъюнкции либо дизъюнкции.

После решения задачи 1 множество K_Y вершин орграфа G разбивается на три попарно непересекающихся подмножества. Вершины, не имеющие заходящих дуг, но имеющие исходящую дугу, соответствуют нереализуемым кофакторам. Вершины, имеющие одну пару заходящих дуг (с одной и той же пометкой), соответствуют реализуемым кофакторам. Вершины, которые оказались изолированными, например, при удалении дуг для удовлетворения условия B, не будут участвовать в уравнениях и зачисляются в множество K_Z также нереализуемых кофакторов. Реализуемый кофактор будет записан в виде уравнения в результирующее многоуровневое представление системы булевых функций.

Алгоритм решения задачи 1

Этап А1. Выбор одного уравнения в каждом кластере.

Шаг А1.1. Упорядочить кластеры по возрастанию (неубыванию) числа содержащихся в них вершин и рассматривать кластеры в этом порядке.

Шаг А1.2. Оставить только одну вершину в каждом кластере.

Положить пустым текущее множество T вершин: $T = \varnothing$. Каждая оставленная вершина g_r в кластере, в которую заходят одноименно помеченные дуги, инцидентные вершинам g_i, g_j, пополняет множество T элементами g_i, g_j, g_r.

Для выбора оставляемой в первом кластере вершины применяется *эвристика 1.*

Эвристика 1. Каждая вершина g_r кластера, имеющая заходящие дуги из вершин g_i, g_j, оценивается суммарным числом W_{q_r} исходящих дуг из вершин g_i, g_j. Оставляется в кластере та вершина g_s, которой соответствует максимальное значение W_{q_r}.

Вершины, которые не остались в кластере, удаляются из графа вместе с заходящими в них дугами. В множество T заносятся три вершины — это вершина g_s и две вершины, инцидентные заходящим в вершину g_s дугам.

Для следующих рассматриваемых кластеров используется *эвристика 2.*

Эвристика 2. В кластере оставляется та вершина, которая имеет две заходящие дуги из вершин множества T; если такой вершины нет, то оставляется та вершина, которая имеет одну заходящую дугу из вершин множества T. Если нет вершин, для которых эвристика 2 выполняется, то для оставления вершины в кластере используется эвристика 1. Вершины, которые не остались в кластере, удаляются из графа вместе с заходящими в них дугами.

Шаг А1.2 и этап А1 считаются выполненным, когда в каждом кластере останется по одной вершине.

Этап А2. Приведение орграфа G к бесконтурному графу.

Итеративно выполняются шаги А2.1—А2.3, пока орграф не станет бесконтурным.

Шаг А2.1. Проверка рассматриваемого орграфа на отсутствие контуров.

На данном шаге осуществляется построение множества простых контуров графа. Для этого реализуется последовательный перебор всех вершин графа. Для очередной выбранной вершины выполняется поиск в глубину для построения всевозможных путей в графе, начинающихся с этой вершины. Такая процедура является модификацией программной реализации [10] в языке C# известного алгоритма Джонсона [11].

Если в результате проверки выявлены контуры, то выполняется шаг А2.2. Если орграф не имеет контуров, то осуществляется переход на шаг А2.4.

Шаг А2.2. Нахождение уравнения, удаляемого из графа, т. е. пары дуг с одинаковыми пометками, соответствующих уравнению и заходящих в одну и ту же вершину графа.

Каждая пара дуг с одинаковой пометкой (номером уравнения) p оценивается числом C_p контуров, которые могут быть разорваны при удалении из графа данной пары дуг.

Шаг А2.3. Удаляется пара дуг (уравнение) с пометкой p, которой соответствует максимальное значение числа C_p. Переход на шаг А2.1.

Шаг А2.4. Конец.

Пример выполнения алгоритма 1

Проиллюстрируем предложенный алгоритм 1 на примере BDDI, изображенной на рис. 4. Будем рассматривать третий уровень BDDI, кофакторы которого зависят от переменных x_3, x_4, x_5.

Таблица 3

Уравнения для BDDI (рис. 4)

Номер уровня BDDI	Уравнения
5	$f_1 = \bar{x}_1 h_1 \vee x_1 h_2$; $f_2 = \bar{x}_1 h_2 \vee x_1 h_3$; $f_3 = \bar{x}_1 h_4 \vee x_1 h_5$; $f_4 = \bar{x}_1 h_4 \vee x_1 h_6$; $f_5 = \bar{x}_1 h_6 \vee x_1 h_7$;
4	$h_1 = \bar{x}_2 \bar{g}_1 \vee x_2 \bar{g}_3$; $h_2 = \bar{x}_2 g_2 \vee x_2 \bar{g}_8$; $h_3 = \bar{x}_2 \bar{g}_5 \vee x_2 g_6$; $h_4 = \bar{x}_2 g_6 \vee x_2 \bar{g}_7$; $h_5 = \bar{x}_2 g_6 \vee x_2 g_{10}$; $h_6 = \bar{x}_2 g_4 \vee x_2 g_9$; $h_7 = \bar{x}_2 g_8 \vee x_2 g_{10}$;
3	$g_1 = \bar{x}_4 s_2 \vee x_4 s_3$; $g_2 = \bar{x}_4 \bar{x}_5 \vee x_4 s_1$; $g_3 = \bar{x}_4 s_0 \vee x_4 s_1$; $g_4 = \bar{x}_4 x_5 \vee x_4 s_2$; $g_5 = \bar{x}_4 s_0 \vee x_4 x_5$; $g_6 = \bar{x}_4 \bar{s}_0 \vee x_4 s_4$; $g_7 = \bar{x}_4 \bar{x}_5 \vee x_4 x_5$; $g_8 = \bar{x}_4 s_3 \vee x_4 \bar{s}_1$; $g_9 = \bar{x}_4 s_2 \vee x_4 \bar{x}_5$; $g_{10} = \bar{x}_4 x_3 \vee x_4 x_5$;
2	$s_0 = \bar{x}_5 x_3$; $s_1 = x_5 \bar{x}_3$; $s_2 = \bar{x}_5 \bar{x}_3$; $s_3 = x_5 x_3$; $s_4 = \bar{x}_5 \bar{x}_3 \vee x_5 x_3$.
1	Литералы x_3, $\bar{x}_3$
0	Константы 0, 1

Этап 1. Разбить множество уравнений по уровням BDDI.

Уравнения для каждого из уровней BDDI заданы в табл. 3.

Этап 2. Нахождение множества реализуемых кофакторов.

Шаги 2.1—2.3. С помощью программы [12] элиминации промежуточных переменных построим таблицу истинности кофакторов третьего уровня BDDI и найдем инверсии кофакторов (табл. 4). После этого проведем полный комбинаторный перебор вариантов нахождения дизъюнктивных и конъюнктивных разложений неинверсных кофакторов из табл. 4. Полученные алгебраические формулы запишем в табл. 5.

Шаг 2.4.

Этап А1 решения задачи 1. Выбор уравнений в кластерах (сокращение кластеров).

Шаг А1.1. Порядок рассмотрения кластеров согласно возрастанию (неубыванию) весов кластеров: $<g_6, g_2, g_4, g_1, g_5, g_7, g_3>$.

Шаг А1.2.

1. Рассматривается кластер g_6, оставляется уравнение $g_6 = g_1 \vee g_4$, множество $T = \{g_1, g_4, g_6\}$.

2. Рассматривается кластер g_2, оставляется уравнение $g_2 = \bar{g}_4 \,\&\, g_8$, реализуется эвристика 1, множество T пополняется вершинами из оставляемого уравнения: $T = \{g_1, g_2, g_4, g_6, g_8\}$.

3. Рассматривается кластер g_4, максимальный вес $W_{q_4} = 18$ имеет вершина g_4 для уравнения $g_4 = g_6 \,\&\, \bar{g}_1$, оставляется уравнение $g_4 = g_6 \,\&\, \bar{g}_1$, реализуется эвристика 1, множество не изменяется: $T = \{g_1, g_2, g_4, g_6, g_8\}$.

Аналогично оставляются уравнения в других кластерах g_1, g_5, g_7, g_3. Результат выполнения этапа А1:

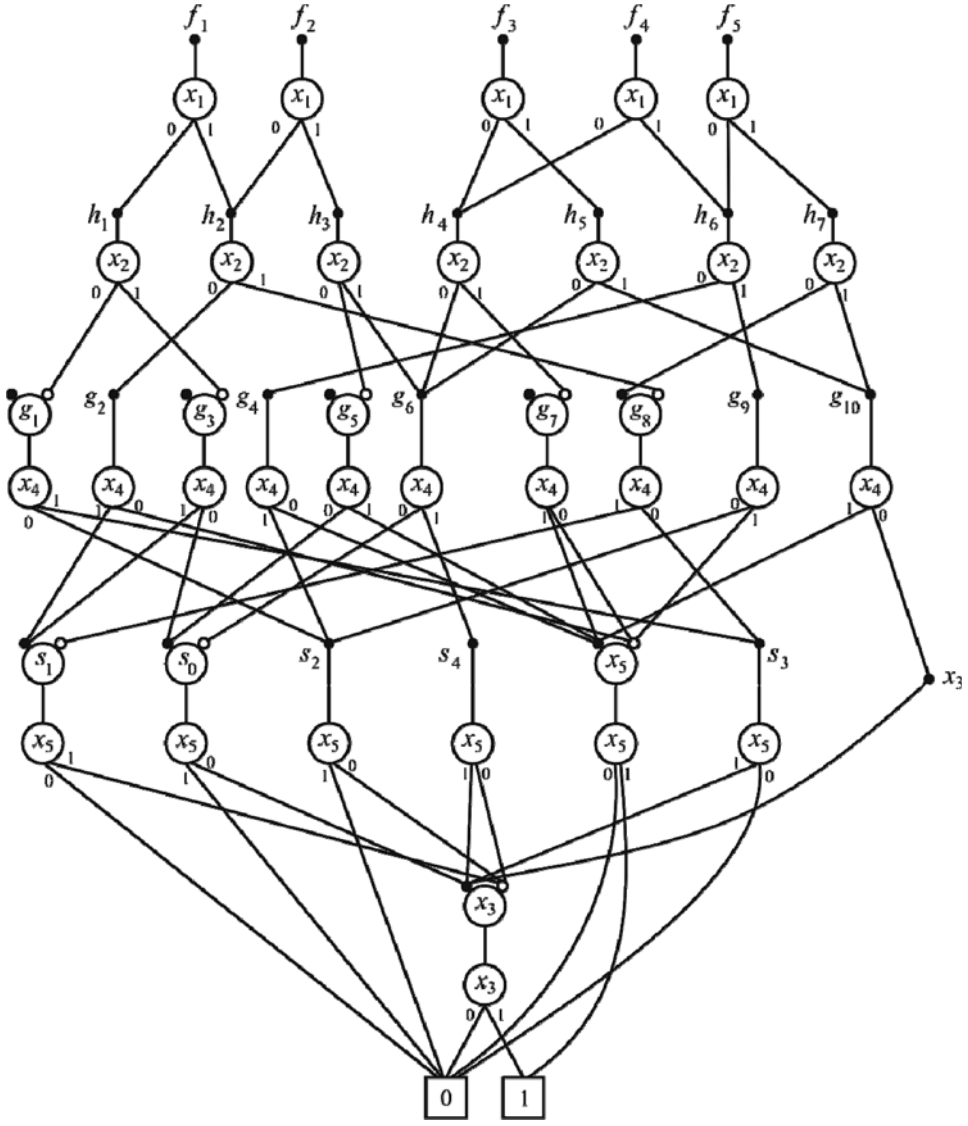


Рис. 4. Граф BDDI

Таблица 4

Кофакторы и их инверсии третьего уровня BDDI (рис. 4)

x_3 x_4 x_5	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	$\overline{g}_1$	$\overline{g}_2$	$\overline{g}_3$	$\overline{g}_4$	$\overline{g}_5$	$\overline{g}_6$	$\overline{g}_7$	$\overline{g}_8$	$\overline{g}_9$	$\overline{g}_{10}$
0 0 0	1	1	0	0	0	1	1	0	1	0	0	0	1	1	1	0	0	1	0	1
0 0 1	0	0	0	1	0	1	0	0	0	0	1	1	1	0	1	0	1	1	1	1
0 1 0	0	0	0	1	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1
0 1 1	0	1	1	0	1	0	1	0	0	1	1	0	0	1	0	1	0	1	1	0
1 0 0	0	1	1	0	1	0	1	0	0	1	1	0	0	1	0	1	0	1	1	0
1 0 1	0	0	0	1	0	1	0	1	0	1	1	1	1	0	1	0	1	0	1	0
1 1 0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	0	0	1
1 1 1	1	0	0	0	1	1	1	1	0	1	0	1	1	1	0	0	0	0	1	0

Таблица 5

Уравнения алгебраических разложений кофакторов третьего уровня BDDI

Номер уравнения	Уравнение	Кластер	Вес кластера
1	$g_1 = g_6 \,\&\, \overline{g}_4$	g_1	3
2	$g_1 = g_6 \,\&\, g_7$		
3	$g_1 = g_7 \,\&\, \overline{g}_3$		
4	$g_2 = \overline{g}_4 \,\&\, \overline{g}_8$	g_2	2
5	$g_2 = g_7 \,\&\, \overline{g}_8$		
6	$g_3 = \overline{g}_6 \,\&\, \overline{g}_8$	g_3	14
7	$g_3 = \overline{g}_6 \,\&\, \overline{g}_9$		
8	$g_3 = g_{10} \,\&\, \overline{g}_6$		
9	$g_3 = g_{10} \,\&\, \overline{g}_8$		
10	$g_3 = g_2 \,\&\, \overline{g}_1$		
11	$g_3 = g_2 \,\&\, \overline{g}_6$		
12	$g_3 = g_2 \,\&\, \overline{g}_9$		
13	$g_3 = g_2 \,\&\, g_{10}$		
14	$g_3 = g_2 \,\&\, g_5$		
15	$g_3 = g_5 \,\&\, \overline{g}_1$		
16	$g_3 = g_5 \,\&\, \overline{g}_6$		
17	$g_3 = g_5 \,\&\, \overline{g}_8$		
18	$g_3 = g_7 \,\&\, \overline{g}_1$		
19	$g_3 = g_7 \,\&\, \overline{g}_6$		
20	$g_4 = g_6 \,\&\, \overline{g}_1$	g_4	2
21	$g_4 = g_6 \,\&\, \overline{g}_7$		
22	$g_5 = \overline{g}_4 \,\&\, \overline{g}_9$	g_5	4
23	$g_5 = g_{10} \,\&\, \overline{g}_4$		
24	$g_5 = g_7 \,\&\, \overline{g}_9$		
25	$g_5 = g_7 \,\&\, g_{10}$		
26	$g_6 = g_1 \vee g_4$	g_6	1
27	$g_7 = g_1 \vee g_2$	g_7	4
28	$g_7 = g_1 \vee g_3$		
29	$g_7 = g_1 \vee g_5$		
30	$g_7 = g_2 \vee g_5$		

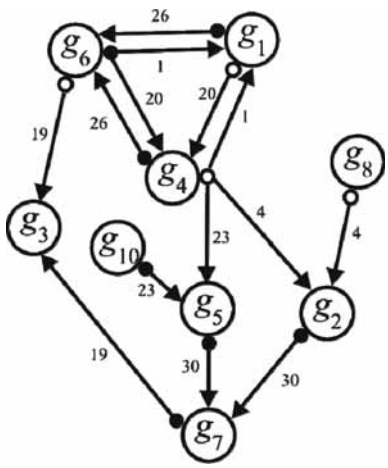


Рис. 5. Орграф *G* после сокращения кластеров

орграф (рис. 5) и соответствующие ему уравнения (табл. 6).

Таблица 6

Уравнения, реализованные в графе *G*, после сокращения кофакторов

Номер уравнения	Уравнение	Кластер
1	$g_1 = g_6 \,\&\, \overline{g}_4$	g_1
4	$g_2 = \overline{g}_4 \,\&\, \overline{g}_8$	g_2
19	$g_3 = g_7 \,\&\, \overline{g}_6$	g_3
20	$g_4 = g_6 \,\&\, \overline{g}_1$	g_4
23	$g_5 = g_{10} \,\&\, \overline{g}_4$	g_5
26	$g_6 = g_1 \vee g_4$	g_6
30	$g_7 = g_2 \vee g_5$	g_7

Этап A2 решения задачи 1. Приведение орграфа *G* к бесконтурному графу.

В качестве исходного рассматривается орграф на рис. 5.

Итерация 1 (далее будут выполняться итерации 2 и 3).

На *шаге A2.1* находятся все простые контуры: $\{g_1, g_6, g_1\}$, $\{g_1, g_4, g_1\}$, $\{g_1, g_4, g_6, g_1\}$, $\{g_1, g_6, g_4, g_1\}$, $\{g_4, g_6, g_4\}$.

На *шаге A2.2* для каждого из уравнений (пар одинаково помеченных дуг) 1, 20, 26 подсчитывается число разрываемых контуров, если удалить данную пару дуг из графа. Легко видеть, что удаление каждой из пар дуг приводит к разрыву четырех контуров.

На *шаге A2.3* выбирается пара дуг с пометкой 1 (рис. 6), которые удаляются из графа.

Итерация 2. На *шаге A2.1* находится один простой контур $\{g_4, g_6, g_4\}$, который образуют две дуги с пометками 20, 26.

На *шаге A2.2* выясняется, что удаление любой из пар дуг с пометками 20, 26 разрывает единственный простой контур $\{g_4, g_6, g_4\}$.

На *шаге A2.3* удаляется пара дуг с пометкой 20, получается граф, представленный на рис. 7.

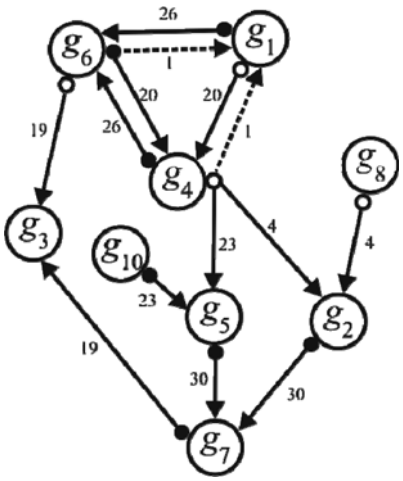


Рис. 6. Шаг 1 приведения графа к бесконтурному (удаляются штриховые дуги, соответствующие уравнению 1)

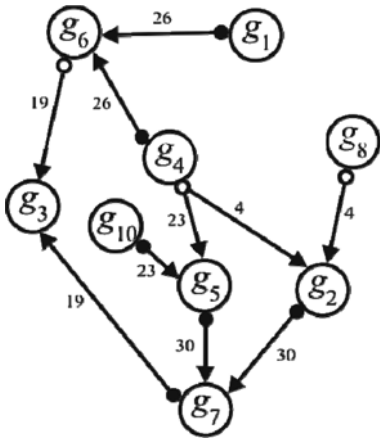


Рис. 7. Результирующий бесконтурный орграф, задающий реализуемые кофакторы

Итерация 3 до конца не выполняется, так как на шаге А2.1 выясняется, что граф (рис. 7) является бесконтурным.

Результат выполнения этапа А2: уравнения для найденных реализуемых кофакторов (табл. 7). На рис. 8 показана логическая схема, соответствующая найденному множеству реализуемых кофакторов, она не имеет обратных связей, так как соответствует ориентированному бесконтурному графу.

Таблица 7

Уравнения, соответствующие реализуемым кофакторам

Номер уравнения	Уравнение	Кластер
4	$g_2 = \bar{g}_4 \,\&\, \bar{g}_8$	g_2
19	$g_3 = g_7 \,\&\, \bar{g}_6$	g_3
23	$g_5 = g_{10} \,\&\, \bar{g}_4$	g_5
26	$g_6 = g_1 \vee g_4$	g_6
30	$g_7 = g_2 \vee g_5$	g_7

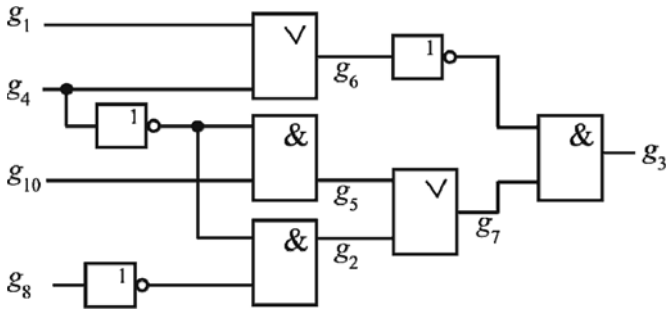


Рис. 8. Логическая схема, соответствующая результирующему бесконтурному орграфу

Этап 3. Корректировка исходного многоуровневого BDDI-представления векторной булевой функции.

На этапе 3 алгоритма 1 в функциональном описании (см. табл. 3) уравнения разложения Шеннона для кофакторов g_2, g_3, g_5, g_6, g_7 заменяются формулами алгебраических разложений из табл. 7 и получается функциональное описание:

$$f_1 = \bar{x}_1 h_1 \vee x_1 h_2; \quad f_2 = \bar{x}_1 h_2 \vee x_1 h_3; \quad f_3 = \bar{x}_1 h_4 \vee x_1 h_5;$$
$$f_4 = \bar{x}_1 h_4 \vee x_1 h_6; \quad f_5 = \bar{x}_1 h_6 \vee x_1 h_7; \quad h_1 = \bar{x}_2 \bar{g}_1 \vee x_2 \bar{g}_3;$$
$$h_2 = \bar{x}_2 g_2 \vee x_2 \bar{g}_8; \quad h_3 = \bar{x}_2 \bar{g}_5 \vee x_2 g_6; \quad h_4 = \bar{x}_2 g_6 \vee x_2 \bar{g}_7;$$
$$h_5 = \bar{x}_2 g_6 \vee x_2 g_{10}; \quad h_6 = \bar{x}_2 g_4 \vee x_2 g_9;$$
$$h_7 = \bar{x}_2 g_8 \vee x_2 g_{10}; \quad g_1 = \bar{x}_4 s_2 \vee x_4 s_3; \quad g_2 = \bar{g}_4 \,\&\, \bar{g}_8; \quad (4)$$
$$g_3 = g_7 \,\&\, \bar{g}_6; \quad g_4 = \bar{x}_4 x_5 \vee x_4 s_2; \quad g_5 = g_{10} \,\&\, \bar{g}_4;$$
$$g_6 = g_1 \vee g_4; \quad g_7 = g_2 \vee g_5; \quad g_8 = \bar{x}_4 s_3 \vee x_4 \bar{s}_1;$$
$$g_9 = \bar{x}_4 s_2 \vee x_4 \bar{x}_5; \; g_{10} = \bar{x}_4 x_3 \vee x_4 x_5;$$
$$s_1 = x_5 \bar{x}_3; \quad s_2 = \bar{x}_5 \bar{x}_3; \quad s_3 = x_5 x_3.$$

После замены формул выяснилось, что уравнения

$$s_0 = \bar{x}_5 x_3; \quad s_4 = \bar{x}_5 \bar{x}_3 \vee x_5 x_3$$

оказались неиспользуемыми (лишними), они отсутствуют в составе взаимосвязанных формул (4). В результате применения предложенного метода логической оптимизации в функциональном BDDI-описании сократилось число литералов и были получены лучшие по площади результаты логического синтеза (табл. 8). Информация об используемой библиотеке логических элементов будет дана далее при описании вычислительного эксперимента.

Таблица 8

Результаты логического синтеза

Функциональное описание	Число литералов	Логическая схема		
		Число логических элементов	Площадь S_{ASIC}	Задержка τ, нс
Исходное (формулы, табл. 2)	100	45	16 031	2,89
Результирующее (формулы (4))	84	37	13 855	3,36

Алгоритмы 2—4 нахождения алгебраических разложений кофакторов в BDDI

Чтобы описать отличия алгоритмов 2—4 от описанного алгоритма 1, заметим, что в **алгоритме 1** в виде алгебраических разложений представляются все кофакторы, т. е. те, которые имеют в своей записи два, три либо четыре литерала.

Формулы разложения Шеннона, включающие только два литерала, назовем *короткими*, а формулы, состоящие из трех либо четырех литералов — *длинными*. Уравнения вида (2), (3), которыми заменяются формулы разложения Шеннона, будем называть также подставляемыми уравнениями либо *подстановками*.

Алгоритм 2 отличается от алгоритма 1 тем, что на этапе 3 корректировки исходного многоуровневого BDDI-представления исходные короткие уравнения не заменяются найденными дизъюнктивными либо конъюнктивными разложениями.

Алгоритм 3 отличается от алгоритма 1 тем, что алгебраические дизъюнктивные и конъюнктивные разложения ищутся только для кофакторов, представленных только длинными уравнениями, при этом для искомых разложений используются кофакторы, представляемые в исходной записи как длинными, так и короткими уравнениями.

Алгоритм 4 отличается от алгоритма 3 дополнительным шагом: если в найденных результирующих формулах алгебраических разложений

$$g_p = \overline{g}_i \vee \overline{g}_j; \quad g_r = \overline{g}_i \,\&\, \overline{g}_j;$$

используются два инверсных кофактора, то, согласно аксиомам булевой алгебры (формулам де Моргана), в решение записываются эквивалентные подстановки, т. е. формулы

$$g_p = \neg\left(g_i \,\&\, g_j\right); \quad g_r = \neg\left(g_i \vee g_j\right),$$

содержащие один оператор инверсии.

Программная реализация

Программная реализация представленных выше алгоритмов была выполнена на языке C++ в рамках кроссплатформенной среды Qt [13], при этом активно применялись ранее разработанные библиотеки работы с булевыми объектами [14] и для обработки описаний, выполненных на языке SF [15, с. 51]. В целях практического использования общая организация программы выполнена по правилам разработки проектных процедур в системе FLC-2 [16].

Программа является параметрически настраиваемой. В качестве параметров выступают:

`-i <src>` — путь доступа к исходному файлу описания схемы, представленного в виде BDD-файла — системы логических уравнений на языке SF;

`-o <tar>` — путь доступа к результирующему, оптимизированному по числу литералов файлу;

`-r <config>` — путь доступа к файлу конфигурации — файлу формата INI системы Windows, содержащему в своем составе секции с набором ключевых параметров, устанавливающих режимы работы программы:

```
[BDD _ OPT]
...
inversion=2
donothandleshort=2
...
[DATA]
RPT _ Name=e:/FLC2/workDir/protokol.txt
```

Здесь параметр `inversion` может принимать значения из множества {0,1,2}: "0" означает, что для подбора оптимизирующих подстановок не будут использованы инверсные функции; "1" — равнозначное использование как прямых, так и инверсных функций; "2" — кроме использования инверсных функций будет осуществлен переход к инверсному представлению для подстановок:

$$g_p = \overline{g}_i \vee \overline{g}_j; \rightarrow g_p = \neg\left(g_i \,\&\, g_j\right);$$
$$g_r = \overline{g}_i \,\&\, \overline{g}_j; \rightarrow g_r = \neg\left(g_i \vee g_j\right).$$

Параметр `donothandleshort` устанавливает правила рассмотрения уравнений на предмет поиска подстановки кофакторов и также может принимать значения из множества {0,1,2}: "0" означает, что проверке на замену будут подвергаться все уравнения рассматриваемой системы; "1" — отсев возможных подстановок, способных сократить число литералов в уравнении, будет выполнен на этапе формирования результирующего файла; "2" — отсев будет проведен на этапе поиска подстановок.

Параметр `RPT _ Name` из секции `DATA` определяет путь доступа к файлу протокола сеанса, содержащего информацию об ошибках в принятых исходных данных.

Рассматриваемая программа имеет ряд ограничений. Среди них фиксированными являются только ограничения по числу n аргументов реализуемой системы булевых функций $n \leqslant 27$; числу возможных представлений отдельного кофактора (не более 96 вариантов) и числу дуг графа G (не более 30 000). Однако достижения таких значений на практике не наблюдалось в силу требуемого объема проводимых вычислений. Известны случаи нормального срабатывания программы при размерах списков смежности описания графа более 15 000 элементов (дуг графа) и порядка 90 шагов срабатывания алгоритма приведения орграфа к бесконтурному виду.

Вычислительный эксперимент

Был проведен эксперимент по проверке эффективности использования программ, реализующих предложенные алгоритмы 1—4, для сокращения площади и увеличения быстродействия блоков комбинационной логики, реализуемых в составе заказных цифровых СБИС (ASIC).

Исходными описаниями *первого набора примеров* комбинационной логики являлись системы ДНФ

булевых функций на языке SF [15, с. 53], взятые из библиотеки [17] примеров схем, представленных в формате PLA, при этом описания из формата PLA переводились в формат SF системы FLC-2 логической оптимизации [16].

Второй набор примеров составляли системы булевых функций, задающих SF-описания таблиц истинности модулярных сумматоров [18].

Для каждого из примеров систем полностью определенных булевых функций выполнялась BDDI-минимизация с помощью программы BDD_Builder [9], также включенной в FLC-2. Полученные схемные реализации были названы *базовыми* для эксперимента. BDDI-описания и дополнительно минимизированные BDDI-описания (дополнительная оптимизация выполнялась с помощью программ, реализующих предложенные в данной работе алгоритмы 1—4 алгебраического разложения кофакторов) конвертировались в VHDL-описания и подавались на вход синтезатора LeonardoSpectrum [19, с. 241]. Для каждого из примеров синтез логической схемы осуществлялся с одними и теми же опциями управления синтезом и для одной и той же целевой библиотеки синтеза. Целевой являлась библиотека проектирования заказных цифровых КМОП СБИС, состав библиотеки приведен в работе [20].

Параметры систем функций и BDDI-описаний для первого набора примеров заданы в табл. 9. Было установлено, что примеры GARY и IN0 задают одну и ту же систему функций в виде различных систем ДНФ, т. е. ДНФ с различными множествами элементарных конъюнкций. Для второго набора примеров имя примера Mod_*i* соответствует сумматору по модулю *i*.

Таблица 9

Параметры исходных систем функций и BDDI

Схема	Система ДНФ			BDDI		
	n	*m*	*k*	Число S_{BDDI} уравнений	Число пар взаимно инверсных кофакторов	Суммарное число литералов *P*
ADD6	12	7	1 092	27	10	91
ADDM4	9	8	512	66	41	590
ADR4	8	5	256	17	6	57
ALU1	12	8	19	16	0	50
B12	15	9	431	66	3	209
B9	16	5	123	73	14	253
BR1	12	8	34	119	4	319
BR2	12	8	35	85	2	216
DC2	8	7	58	58	9	179
DIST	8	5	256	115	32	424
EX7	16	5	123	73	14	253
F51M	8	8	256	37	13	124
GARY	15	11	214	317	17	964
IN0	15	11	138	317	17	964
IN1	16	17	110	756	20	2 552
IN2	19	10	137	261	25	861
INTB	15	7	664	681	105	2 465
LIFE	9	1	512	36	5	133
LOG8MOD	8	5	47	62	10	215
M1	6	12	32	48	7	134
M181	15	9	430	67	3	212
M2	8	16	96	116	28	362
M3	8	16	128	130	30	413

Продолжение табл. 9

Схема	Система ДНФ			BDDI		
	n	*m*	*k*	Число S_{BDDI} уравнений	Число пар взаимно инверсных кофакторов	Суммарное число литералов *P*
M4	8	16	256	175	46	604
MAX1024	10	6	1 024	333	118	1 242
MAX46	9	1	46	72	0	248
MAX512	9	6	512	186	56	686
MLP4	8	8	256	147	37	528
MP2D	14	14	123	78	2	219
NEWAPLA	12	10	17	44	4	112
NEWAPLA1	12	7	10	24	0	55
NEWCPLA1	9	16	38	80	8	212
NEWILL	8	1	8	15	0	47
NEWTAG	8	1	8	8	0	23
NEWTPLA	15	5	23	56	4	147
P82	5	14	24	53	5	162
RADD	8	5	120	17	6	57
RD53	5	3	32	15	5	53
RD73	7	3	147	29	9	107
ROOT	8	5	256	56	15	175
SEX	9	14	23	51	0	142
SQN	7	3	96	46	5	161
SQR6	6	12	64	62	11	199
T3	12	8	152	87	5	248
TIAL	14	8	640	582	126	2 099
VTX1	27	6	110	100	2	298
X9DN	27	7	120	102	2	305
Z4	7	4	128	15	5	51
Z5XP1	7	10	128	40	11	131

Далее в таблицах, задающих результаты экспериментов, используются следующие обозначения для параметров систем функций $\mathbf{f}(\mathbf{x}) = (f_1(x), ..., f_m(\mathbf{x}))$, $\mathbf{x} = (x_1, ..., x_n)$: n — число переменных $x_1, ..., x_n$; m — число функций; k — число общих элементарных конъюнкций, входящих в ДНФ всех компонентных функций $f_j(\mathbf{x})$, $j = 1, ..., m$.

Результирующие логические схемы оценивались двумя параметрами: S_{ASIC} — суммарная площадь всех элементов логической схемы; τ — задержка схемы (нс). Результаты эксперимента приведены в табл. 10—12. Лучшие решения (схемы меньшей площади либо с меньшей задержкой) отмечены жирным шрифтом, символом * помечены решения, улучшающие исходные *базовые* решения, которыми, как уже говорилось, являются логические схемы, построенные по исходным BDDI. В табл. 11 для каждого из алгоритмов 1—4 приведены числа исключаемых (лишних) уравнений после применения соответствующего алгоритма. Такую информацию выдает синтезатор LeonardoSpectrum, если при синтезе выявляются неиспользуемые сигналы в функциональных описаниях, по которым ведется синтез схем. Если для какого-либо примера для каждого из алгоритмов 1—4 число исключаемых уравнений равно нулю, то информация по такому примеру отсутствует, например, для первого примера ADD6 информация в табл. 11 не приводится, так как лишних уравнений не оказывается ни для одного из алгоритмов.

Таблица 10

Результаты эксперимента для первого набора примеров

Схема	Исходные BDDI (базовые решения)		Алгоритм 1		Алгоритм 2		Алгоритм 3		Алгоритм 4	
	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс
ADD6	12 806	8,03	***11 952**	***7,72**	12 806	8,03	12 583	9,05	12 583	9,05
ADDM4	80 782	**7,84**	79 643	8,01	***73 974**	8,17	78 672	11,35	81 602	***7,03**
ADR4	8 074	4,90	***7 589**	***4,82**	8 074	4,90	*7 851	5,60	*7 851	5,06
ALU1	**7 109**	**1,12**	**7 109**	**1,12**	**7 109**	**1,12**	**7 109**	**1,12**	**7 109**	**1,12**
B12	**18 358**	3,59	21 327	3,77	21 293	*3,46	21 293	*3,46	18 492	***3,40**
B9	26 081	4,91	26 204	***3,89**	26 081	4,91	***25 701**	4,65	25 701	4,65
BR1	**23 843**	6,36	27 867	*5,65	24 530	*5,06	26 193	*5,46	28 547	***5,26**
BR2	21 371	6,34	21 790	*5,57	***20 032**	***4,43**	***20 032**	***4,43**	21 371	6,34
DC2	23 302	**4,28**	22 845	5,34	20 780	4,88	*19 976	4,30	***19 067**	4,73
DIST	60 085	6,08	55 627	8,34	***54 159**	6,74	56 062	7,14	54 807	***6,01**
EX7	26 081	4,91	26 204	***3,89**	26 081	4,91	***25 701**	*4,65	***25 701**	*4,65
F51M	18 353	7,67	17 622	*5,39	*16 824	*5,45	17 298	***3,21**	***16 037**	*3,54
GARY	**94 648**	6,74	107 264	7,14	96 032	6,81	101 299	*6,68	***93 978**	**6,43**
IN0	**94 648**	6,74	107 264	7,14	96 032	6,81	98 409	***6,50**	***93 978**	6,43
IN1	**192 655**	**8,26**	235 638	14,12	216 493	10,05	227 262	13,16	221 805	11,62
IN2	75 414	**6,73**	71 619	11,23	***69 471**	9,13	70 102	10,01	*73 851	10,75
INTB	272 555	**8,67**	*241 168	13,90	***237 758**	13,26	*248 561	13,50	*248 316	15,73
LIFE	18 146	**4,60**	*15 256	7,44	*16 450	7,26	***15 150**	7,36	*17 359	7,37
LOG8MOD	**24 022**	4,26	25 160	***4,05**	24 440	4,89	24 161	4,48	24 457	4,76
M1	**15 312**	3,46	17 605	*2,63	16 121	*2,63	16 857	***2,47**	16 037	*2,77
M181	**18 849**	3,48	20 356	3,73	20 914	***3,42**	20 914	***3,42**	***18 576**	3,75
M2	45 086	5,20	*45 058	***4,87**	*43 362	5,39	***42 776**	*5,19	*42 787	*5,07
M3	52 580	**4,49**	54 656	6,62	*52 435	4,92	*51 754	5,81	***49 947**	5,33
M4	78 181	5,87	78 405	*5,27	78 879	6,19	*77 473	6,18	***76 502**	***4,84**
MAX1024	146 888	**7,18**	153 606	11,02	150 777	10,30	148 567	10,50	***144 076**	9,79
MAX46	36 125	**4,89**	*34 892	5,61	***32 035**	4,93	*35 344	**4,89**	*35 344	**4,89**
MAX512	84 643	**5,98**	*84 537	7,29	*79 827	7,72	*81 228	6,29	***79 582**	6,46
MLP4	68 439	**5,60**	70 224	6,28	***66 586**	6,70	68 628	9,17	*68 210	6,98
MP2D	**17 471**	**3,56**	20 239	6,15	19 351	3,48	18 771	4,37	17 767	3,61
NEWAPLA	11 087	**3,84**	***10 184**	3,90	11 087	**3,84**	***10 184**	3,90	11 087	**3,84**
NEWAPLA1	6 869	**3,35**	***6 702**	3,63	6 869	**3,35**	6 869	**3,35**	6 869	**3,35**
NEWCPLA1	20 585	**3,87**	22 158	4,53	20 596	4,58	***19 206**	4,02	22 951	5,14
NEWILL	**5 122**	**3,22**	5 736	4,76	5 736	4,76	5 736	4,76	**5 122**	**3,22**
NEWTAG	**2 126**	**1,90**	**2 126**	**1,90**	**2 126**	**1,90**	**2 126**	**1,90**	**2 126**	**1,90**

Продолжение табл. 10

Схема	Исходные BDDI (базовые решения)		Алгоритм 1		Алгоритм 2		Алгоритм 3		Алгоритм 4	
	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс
NEWTPLA	**11 316**	**3,30**	13 515	3,85	**11 316**	**3,30**	13 264	4,04	**11 316**	**3,30**
P82	19 988	2,93	22 292	3,23	***19 965**	***2,75**	19 982	*2,76	*19 982	*2,76
RADD	8 074	4,90	***7 589**	***4,82**	8 074	4,90	7 851	5,60	*7 851	5,60
RD53	**7 321**	3,38	7 779	***3,22**	**7 321**	3,38	**7 321**	3,38	**7 321**	3,38
RD73	18 090	4,55	***14 212**	***4,49**	15 222	5,30	15 669	4,61	*15 669	4,61
ROOT	26 109	4,78	26 455	*4,68	*24 580	*4,23	***22 532**	***4,08**	*23 581	5,30
SEX	**12 566**	3,69	15 524	*3,56	12 605	***2,08**	12 605	***2,08**	12 605	***2,08**
SQN	22 303	3,33	*20 077	4,87	*19 642	5,04	*19 245	***3,10**	***18 922**	5,57
SQR6	28 737	5,72	*28 157	*4,55	*28 023	*4,33	*27 827	*3,77	***27 035**	***3,72**
T3	**17 276**	**3,59**	19 268	5,85	17 454	6,03	17 454	6,03	17 917	4,16
TIAL	255 531	**8,75**	*221 509	15,25	*212 286	12,65	***209 217**	12,15	*211 633	14,72
VTX1	26 996	5,96	*25 930	6,40	***24 223**	***5,94**	***24 223**	***5,94**	*24 552	6,16
X9DN	26 996	6,00	25 701	***5,81**	***25 155**	*5,84	***25 155**	*5,84	*26 812	6,65
Z4	6 640	**4,25**	***6 339**	4,32	6 640	**4,25**	6 417	4,71	*6 417	4,71
Z5XP1	18 442	4,64	*18 252	4,30	18 442	4,53	***16 400**	***3,61**	***16 400**	***3,61**
Число лучших решений (полужирный шрифт)	16	22	9	12	14	11	15	14	18	16
Число улучшений базовых решений (число *)			17	18	19	12	21	18	28	13

Таблица 11

Результаты эксперимента для первого набора примеров

Схема	Число исключаемых уравнений			Схема	Число исключаемых уравнений		
	Алгоритм 1	Алгоритм 2	Алгоритм 3, 4		Алгоритм 1	Алгоритм 2	Алгоритм 3, 4
ADDM4	1	1	1	MAX1024	3	3	5
B12	1	0	0	MAX512	1	1	2
BR1	4	3	2	MLP4	1	1	1
BR2	6	2	2	MP2D	2	2	2
DC2	4	2	1	NEWAPLA	2	0	0
DIST	5	3	3	NEWCPLA1	2	3	0
GARY	14	6	6	P82	2	0	0
IN0	14	6	6	RD73	1	0	0
IN1	73	60	57	SEX	3	1	3
IN2	22	20	20	SQN	1	1	1
INTB	90	93	96	T3	11	7	7
LIFE	4	4	4	TIAL	59	75	75
M1	5	2	2	VTX1	2	2	2
M181	1	0	0	X9DN	2	2	2
M3	2	0	0	Z5XP1	1	0	0
M4	2	2	2				

Таблица 12

Результаты эксперимента для второго набора примеров

Схема	Исходные BDDI (базовые решения)		Алгоритм 1		Алгоритм 2		Алгоритм 3		Алгоритм 4	
	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс	S_{ASIC}	τ, нс
Mod_5	13 520	**2,68**	***12 923**	2,86	13 520	**2,68**	13 593	3,88	13 520	**2,68**
Mod_7	**16 946**	3,51	20 696	4,50	**16 946**	3,51	19 469	***3,39**	18 777	4,00
Mod_9	32 358	6,21	*30 640	*5,64	32 358	6,21	***30 082**	6,44	*30 679	***5,01**
Mod_15	**41 080**	**5,83**	44 768	7,00	**41 080**	5,83	44 205	6,93	44 428	7,34
Mod_17	54 137	6,02	57 697	7,62	54 137	6,02	*52 703	***5,98**	***48 462**	8,15
Mod_19	49 483	**6,93**	58 311	9,00	49 483	9,93	51 593	7,80	***48 752**	***5,95**
Mod_23	58 746	6,75	***58 171**	***6,17**	58 746	6,75	60 822	7,81	62 624	8,75
Mod_25	**68 043**	**6,60**	75 492	9,00	**68 043**	**6,60**	75 570	8,53	73 913	8,35
Mod_27	**78 600**	**6,77**	92 422	10,28	**78 600**	**6,77**	85 240	8,96	87 059	9,24
Mod_29	77 902	**6,51**	*77 149	10,02	77 902	**6,51**	*72 919	10,39	***71 469**	11,68
Mod_31	73 003	**7,68**	*72 652	8,96	73 003	**7,68**	*72 864	8,08	***69 795**	8,77
Mod_37	113 352	**6,60**	*94 296	9,04	113 352	**6,60**	***87 483**	9,89	*91 395	11,18
Mod_59	**149 628**	**9,35**	159 030	16,62	149 628	**9,35**	150 264	15,73	151 542	13,57
Mod_61	129 863	**8,54**	*102 466	13,36	129 863	**8,54**	*100 261	13,83	***93 549**	18,04
Число лучших решений (полужирный шрифт)	5	10	2	1	4	8	2	2	5	3
Число улучшений базовых решений (число *)			7	2	0	0	6	2	7	2

Проанализируем результаты эксперимента. Первый набор примеров состоял из 59 функциональных описаний схем. Для 10 примеров схем небольшой размерности — это схемы CLPL, CO14, NEWAPLA2, NEWBYTE, NEWCOND, NEWTPLA1, NEWTPLA2, RYY6, SYM10, Z9SYM - применение алгоритмов 1—4 не изменило базовое решение, поэтому исходные данные и результаты вычислительного эксперимента для данных примеров в табл. 9—11 не приводятся.

Для трех примеров B12, LOG8MOD, MP2D применение любого из алгоритмов 1—4 изменения функциональных базовых BDDI описаний приводило к незначительному увеличению площади соответствующей логической схемы. Это связано с тем, что синтезатор LeonardoSpectrum имеет собственные встроенные средства логической оптимизации и технологического отображения. Технологическое отображение заключается в покрытии оптимизированных логических уравнений функциональными описаниями логических элементов, входящих в библиотеку проектирования. Изменение функционального описания приводило к другим результатам технологического отображения. Для 33 примеров схем применение предложенных алгоритмов позволило улучшить их площадь. Сокращение площади для схем большой размерности может быть практически значимым: для схемы INTB сокращение площади составляет 13 %, для схемы Tial — 18 %. В целом, алгоритмы 3 и 4 являются более эффективными по сравнению с алгоритмами 1 и 2. Алгоритм 4 позволяет улучшить базовые решения (по площади) для 27 схем, алгоритм 3 — позволяет увеличить быстродействие для 18 схем. Алгоритмы 3 и 4 можно рекомендовать для практического применения в первую очередь. Не исключается также вариант того, что применение алгоритмов 1 и 2 также может улучшить базовое решение.

Для второго набора примеров (модулярных сумматоров) применение алгоритмов позволило уменьшить площади для девяти схем и уменьшить задержки для пяти схем из 14. Площадь схемы Mod_61 уменьшилась на 28 %, площадь схемы Mod_37 уменьшилась на 23 % (табл. 12).

Программный эксперимент также показал, что для сокращения площади схем всегда более эффективны дизъюнктивные и конъюнктивные разложения кофакторов с использованием их инверсий по сравнению с аналогичными разложениями без инверсий кофакторов.

Заключение

Эксперименты [9] показали, что нахождение пар взаимно инверсных кофакторов при оптимизации BDDI значительно улучшает результаты последующего синтеза по сравнению с оптимизацией BDD, выполняемой без нахождения инверсных кофакторов.

В статье описаны программно реализованные алгоритмы дополнительной минимизации алгебраических многоуровневых BDDI-представлений систем полностью определенных функций на основе дизъюнктивных и конъюнктивных представлений подфункций, находящихся на одном уровне BDDI. Такая дополнительная логическая оптимизация по числу литералов в результирующих логических уравнениях, задающих BDDI, приводит к более простым функциональным описаниям, по которым осуществляется синтез логических схем. Сокращение площади схем, сводящееся к уменьшению числа транзисторов в схемах, позволяет также сокращать энергопотребление схем. Однако сокращение числа литералов при алгебраических разложениях, позволяющее часто сокращать площадь схем из библиотечных КМОП-элементов, может приводить как к уменьшению, так и увеличению временных задержек схем.

Список литературы

1. **Брейтон Р. К., Хэчтел Г. Д., Санджованни-Винчентелли А. Л.** Синтез многоуровневых комбинационных логических схем // ТИИЭР. 1990. Т. 78, № 2. С. 38—83.
2. **Meinel C., Theobald T.** Algorithms and Data Structures in VLSI Design: OBDD — Foundations and Applications. — Berlin, Heidelberg. — Springer-Verlag, 1998. 267 p.
3. **Кнут Д. Э.** Искусство программирования. Том 4, А. Комбинаторные алгоритмы. Часть 1 / Пер. с англ. — М.: Вильямс, 2013. 960 с.
4. **Yang S., Ciesielski M.** BDS: a BDD-based logic optimization system // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2002. Vol. 21, No. 7. P. 866—876.
5. **Ebendt R., Fey G., Drechsler R.** Advanced BDD optimization. — Springer, 2005. 222 p.
6. **Бибило П. Н.** Применение диаграмм двоичного выбора при синтезе логических схем. — Минск: Беларус. навука, 2014. 231 с.
7. **Kubica M., Kania D.** SMTBDD: New form of BDD for logic synthesis // International Journal of Electronics and Telecommunications. 2016. Vol. 62, No. 1. P. 33—41.
8. **Бибило П. Н., Романов В. И.** Минимизация многоуровневых представлений систем полностью определенных булевых функций с использованием разложений Шеннона и алгебраических представлений кофакторов // Информатика. 2021. Т. 18, № 2. С. 7—32.
9. **Бибило П. Н., Ланкевич Ю. Ю.** Использование полиномов Жегалкина при минимизации многоуровневых представлений систем булевых функций на основе разложения Шеннона // Программная инженерия. 2017. Т. 8, № 8. С. 369—384.
10. **Поиск** элементарных циклов в графе. URL: https://vscode.ru/prog-lessons/poisk-elementarnyih-tsiklov-v-grafe.html
11. **Johnson D. B.** Finding all the elementary circuits of a directed graph. // SIAM J. Comput. 1975. Vol. 4, No. 1. P. 77—84.
12. **Торопов Н. Р.** Преобразование многоярусной комбинационной сети в двухъярусную // Логическое проектирование. Вып. 5. — Минск: ИТК НАН Беларуси, 2000. С. 4—14.
13. **Шлее М.** Qt 5.10. Профессиональное программирование на C++. — СПб.: БХВ-Петербург, 2018. 1072 с.
14. **Романов В. И.** Программные средства для решения логико-комбинаторных задач // Информатика. 2005. № 4. С. 114—123.
15. **Бибило П. Н., Романов В. И.** Логическое проектирование дискретных устройств с использованием продукционно-фреймовой модели представления знаний. — Минск: Беларус. навука, 2011. — 279 с.
16. **Бибило П. Н., Романов В. И.** Система логической оптимизации функционально-структурных описаний цифровых устройств на основе продукционно-фреймовой модели представления знаний // Проблемы разработки перспективных микро- и наноэлектронных систем. — 2020. Сб. трудов / под общ. ред. акад. РАН А. Л. Стемпковского. — М.: ИППМ РАН, 2020. № 4. С. 9—16.
17. **The Tests** in the Monograph "Logic Minimization Algorithms for VLSI Synthesis". URL: http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex (дата обращения: 20.11.2020).
18. **Балака Е. С., Тельпухов Д. В., Осинин И. П., Городецкий Д. А.** Сравнительное исследование и анализ методов аппаратной реализации сумматоров по модулю // 7Universum: Технические науки: электрон. научн. журн. 2016. № 1 (23). URL: http://7universum.com/ru/tech/archive/item/2887
19. **Бибило П. Н.** Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum. — М.: СОЛОН-Пресс, 2005. 384 с.
20. **Авдеев Н. А., Бибило П. Н.** Автоматизированное проектирование цифровых операционных устройств с пониженным энергопотреблением // Программная инженерия. 2021. Т. 12, № 2. С. 63—73.

Experimental Study of Algorithms for Minimization of Binary Decision Diagrams using Algebraic Representations of Cofactors

Bibilo P. N., bibilo@newman.bas-net.by, **V. I. Romanov,** rom@newman.bas-net.by,
The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, 220012, Belarus

Corresponding author:

Bibilo Petr N., Head of Laboratory, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, 220012, Minsk, Belarus
E-mail: bibilo@newman.bas-net.by

Received on October 25, 2021
Accepted on December 07, 2021

BDD (Binary Decision Diagram) is used for technology-independent optimization, performed as the first stage in the synthesis of logic circuits in the design of ASIC (application-specific integrated circuit). BDD is an acyclic graph defining a Boolean function or a system of Boolean functions. Each vertex of this graph is associated with the complete or reduced Shannon expansion formula. Binary decision diagrams with mutually inverse subfunctions (cofactors) are considered. We have developed algorithms for finding algebraic representations of cofactors of the same BDD level in the form of a disjunction or conjunction of other inverse or non-inverse cofactors of the same BDD level. The algorithms make it possible to reduce the number of literals by replacing the Shannon expansion formulas with simpler logical formulas and to reduce the number of literals in the description of a system of Boolean functions. We propose to use the developed algorithms for an additional logical optimization of the constructed BDD representations of systems of Boolean functions. Experimental results of the application of the corresponding programs in the synthesis of logic circuits in the design library of custom VLSI CMOS circuits are presented.

Keywords: *system of Boolean functions, Disjunctive Normal Form, Binary Decision Diagram, Shannon expansion, digital logic synthesis, VHDL, VLSI*

For citation:

Bibilo P. N., Romanov V. I. Experimental Study of Algorithms for Minimization of Binary Decision Diagrams using Algebraic Representations of Cofactors, *Programmnaya Ingeneria,* 2022, vol. 13, no. 2, pp. 51—67.

DOI: 10.17587/prin.13.51-67

References

1. **Brayton R. K., Hachtel G. D., Sangiovanni-Vincentelli A. L.** Synthesis of multi-level combinational logic circuits, *Trudy Institute inzhenerov po jelektronike i radiotehnike*, 1990, vol. 78, no. 2, pp. 38—83 (in Russian).

2. **Meinel C., Theobald T.** *Algorithms and Data Structures in VLSI Design: OBDD — Foundations and Applications*, Berlin; Heidelberg, Springer-Verlag, 1998, 267 p.

3. **Knuth D. E.** Combinatorial Algorithms, *The Art of Computer Programming, Pearson Education*, 2011, vol. 4A, 883 p. (Russ. ed.: Moscow, 2013, 960 p.).

4. **Yang S., Ciesielski M.** BDS: a BDD-based logic optimization system, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002, vol. 21, no. 7, pp. 866—876.

5. **Ebendt R., Fey G., Drechsler R.** *Advanced BDD optimization*, Springer, 2005, 222 p.

6. **Bibilo P. N.** *Application of Binary Decision Diagrams in the Synthesis of Logic Circuits*, Minsk, Belaruskaja navuka, 2014, 231 p. (in Russian).

7. **Kubica M., Kania D.** SMTBDD: New form of BDD for logic synthesis, *International Journal of Electronics and Telecommunications*, 2016, vol. 62, no. 1, pp. 33—41.

8. **Bibilo P. N., Romanov V. I.** Minimization of Binary Decision Diagrams for Systems of Completely Defined Boolean Functions using Shannon expansions and algebraic representations of cofactors, *Informatics*, 2021, vol. 18, no. 2, pp. 18—43 (in Russian).

9. **Bibilo P. N., Lankevich Yu. Yu.** The use of Zhegalkin polynomials in minimizing multilevel representations of systems of Boolean functions based on the Shannon expansion, *Programmnaya ingeneria*, 2017, no. 8, pp. 369—384 (in Russian).

10. **Search** for elementary cycles in a graph, available at: https://vscode.ru/prog-lessons/poisk-elementarnyih-tsiklov-v-grafe.html

11. **Johnson D. B.** Finding all the elementary circuits of a directed graph, *SIAM J. Comput.*, 1975, vol. 4, no. 1, pp. 77—84.

12. **Toropov N. R.** Transformation of a multi-level combinational network into a two-level one, *Logicheskoe proektirovanie*, Vyp. 5. Minsk, ITK NAN Belarusi, 2000, pp. 4—14 (in Russian).

13. **Shlee M.** *Qt 5.10. Professional programming in C++,* Saint Petersburg, BHV-Peterburg, 2018, 1072 p. (in Russian).

14. **Romanov V. I.** Software tools for solving logic-combinatorial problems, *Informatics*, 2005, no. 4, pp. 114—123 (in Russian).

15. **Bibilo P. N., Romanov V. I.** *Logical Design of Discrete Devices Using a Production-Frame Knowledge Representation Model*, Minsk, Belaruskaja navuka, 2011, 279 p. (in Russian).

16. **Bibilo P. N., Romanov V. I.** The system of logical optimization of functional structural descriptions of digital circuits based on production-frame knowledge representation model, *Problemy razrabotki perspektivnyh mikro- i nanoelektronnyh sistem.* — 2020. Sb. trudov / pod obshch. red. akad. RAN A. L. Stempkovskogo. — Moscow, IPPM RAN, 2020, no. 4, pp. 9—16 (in Russian).

17. **The Tests** in the Monograph "Logic Minimization Algorithms for VLSI Synthesis", available at: http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex

18. **Balaka E. S., Tel'puhov D. V., Osinin I. P., Gorodeckij D. A.** Comparative study and analysis methods hardware implementation RNS-based adders, *7 Universum: Tekhnicheskie nauki: elektron. nauchn. zhurn.*, 2016, no. 1 (23), available at: http://7universum.com/ru/tech/archive/item/2887 (in Russian).

19. **Bibilo P. N.** *CAD of integrated circuits based on the VHDL. StateCAD, ModelSim, LeonardoSpectrum*, Moscow, SOLON-Press, 2005, 384 p. (in Russian).

20. **Avdeev N. A., Bibilo P. N.,** Design of Digital Operational Units with Low Power Consumption, *Programmnaya Ingeneria,* 2021, no. 2, pp. 63 — 73 (in Russian).

УДК 004.056 DOI: 10.17587/prin.13.68-80

А. М. Вульфин, канд. техн. наук, доц., vulfin.alexey@gmail.com,
Уфимский государственный авиационный технический университет

Обнаружение сетевых атак в гетерогенной промышленной сети на основе технологий машинного обучения

Рассматриваются вопросы совершенствования алгоритмов обнаружения сетевых атак в гетерогенной сети промышленного Интернета вещей на основе технологий машинного обучения для последующей интеграции с подсистемами центра мониторинга и реагирования на инциденты информационной безопасности. Разработана структурная схема системы обнаружения сетевых атак и алгоритм интеллектуального анализа параметров сетевого трафика в задаче обнаружения вредоносной сетевой активности. Проанализированы варианты построения ансамблей классификаторов на основе моделей машинного обучения и гетерогенных нейросетевых моделей. Оценка F1-меры при работе с тестовыми выборками на распространенных общедоступных наборах размеченного сетевого трафика достигает 96 %. Рассмотрена возможность встраивания полученных моделей. Разработан виртуальный полигон для оценки эффективности применения моделей машинного обучения для обнаружения сетевых атак.

Ключевые слова: *сетевые атаки, машинное обучения, интеллектуальный анализ данных, ансамбль классификаторов, гетерогенная промышленная сеть, мониторинг и реагирование на инциденты информационной безопасности*

Введение

На современном этапе цифрой трансформации индустрии актуальными являются вопросы поддержания работоспособности киберфизических систем, т. е. обеспечения устойчивости протекающих в них физических процессов и непрерывности управления технологическими процессами в условиях возможных внутренних и внешних целенаправленных деструктивных воздействий. Наблюдается тенденция [1] к интеграции устройств промышленного (индустриального) Интернета вещей (IIoT) с традиционными системами сбора данных и управления (SCADA) в составе промышленных систем. Глубокое проникновение IIoT в критическую инфраструктуру и производственный сектор также привело к возрастанию вероятности и числа потенциальных кибератак.

По данным отчета Claroty в 2020 г. число уязвимостей, выявленных в компонентах автоматизированных систем управления технологическими процессами (АСУ ТП), выросло почти на 25 % по сравнению с 2019 г. Обнаруженные уязвимости в основном затрагивают секторы промышленного производства, энергетики и водоснабжения. В первом полугодии 2020 г. по сравнению с 2019 г. число уязвимостей в сфере промышленности выросло на 87,3 %, в секторе водоснабжения — на 122 %, в энергетическом секторе — на 58,9 %.

По данным аналитических отчетов Positive Technologies промышленность уже на протяжении двух лет входит в тройку наиболее часто атакуемых отраслей. Число атак на промышленность увеличилось почти в 2 раза по сравнению с 2019 г.: прирост составил 91 %. В IV квартале 2020 г. треть всех инцидентов в промышленной отрасли была связана с кибератаками, в 84 % атак применялось вредоносное программное обеспечение.

Ущерб от кибератак на энергетические и коммунальные отрасли достигает в среднем 13,2 млн долл. США ежегодно, и повышение рисков вынуждает к выработке общих подходов к обеспечению кибербезопасности [2, 3].

Для выявления целевых атак на промышленные системы необходим анализ значительного объема входящего, исходящего и внутреннего сетевого трафика и потока событий информационной безопасности (ИБ) для выявления аномальной активности, анализа вектора атаки и оценки возможного ущерба. Для решения подобных задач применяется комплексный подход — развертывание центра мониторинга и реагирования на инциденты ИБ (*Security Operation Center* — SOC), осуществляющего, в том числе, сбор, хранение и анализ трафика [4] как корпоративного сегмента, так и сегмента промышленной сети. Это позволяет выделять шаблоны проведения атак или использования уязвимостей. Основной целью управления инцидентами ИБ является обеспечение непрерывного мониторинга событий ИБ, своевременное реагирование на инциденты, устранение последствий и формирование шаблонов реагирования для предотвращения возникновения инцидентов в будущем. Следовательно, для промышленного оборудования

и пограничных систем (точек входа в промышленную сеть) необходимо обеспечить анализ трафика для обнаружения сетевых атак с поддержкой анализа промышленных протоколов. Совершенствование средств защиты сетевой инфраструктуры направлено на развитие инструментов интеллектуального мониторинга сетевого трафика и состояния объектов и узлов промышленной сети.

Целью работы, результаты которой представлены в настоящей статье, является совершенствование алгоритмов обнаружения сетевых атак в гетерогенной сети промышленного Интернета вещей на основе технологий машинного обучения для последующей интеграции с подсистемами центра мониторинга и реагирования на инциденты ИБ.

Обнаружение сетевых атак в промышленных сетях на основе методов машинного обучения

Для оперативного анализа и выявления аномалий работы сетевой инфраструктуры, вызванных действиями злоумышленника, применяют технологии интеллектуального анализа и машинного обучения [5], что нашло отражение в ряде публикаций:

- модель обнаружения гибридных аномалий в высоконагруженных сетях связи на основе методов интеллектуального анализа данных (ИАД) [6];
- платформа обнаружения аномалий для выявления кибератак на облачные вычислительные среды [7];
- комплексная система контроля и обеспечения безопасности сбора данных, реализующая мониторинг трафика в реальном времени, обнаружение аномалий, анализ воздействия, стратегии смягчения последствий [8, 9];
- система обнаружения аномалий на основе алгоритмов машинного обучения для устранения угроз кибербезопасности сетей Интернета вещей в умном городе [10];
- подход на основе кластерного анализа сетевого трафика для обнаружения кибератак, вызывающих аномалии в сетях критической информационной инфраструктуры газокомпрессорных станций [11];
- распределенная система обнаружения вторжений для систем диспетчерского управления и сбора данных [12];
- алгоритм обнаружения аномалий и система обнаружения вторжений с фильтрацией ложных срабатываний и возможностью подтверждения атаки [13];
- система обнаружения аномалий для обнаружения утечек конфиденциальной информации в сетевом трафике энергосистем [14];
- методология создания надежных наборов данных для обнаружения аномалий в АСУ ТП [15].

Для создания моделей машинного обучения (ML-моделей) используют общедоступные размеченные по типам атак и режимам работы базы сетевого трафика (NSL-KDD [16], CICIDS2017 [17], UNSW-NB15 [18], BOT-IOT и др.). Для обнаружения новых сетевых атак, реализуемых с помощью постоянно развивающегося инструментария злоумышленников, необходимо периодическое обновление тренировочных наборов с реализацией новых сценариев атак и фиксацией параметров их проведения для дообучения ML-моделей.

Так, например, в работе [19] описан стенд, построенный с применением промышленного оборудования, для исследований алгоритмов машинного обучения в задачах обнаружения сетевых атак. В ходе реализации сложных атак по различным сценариям собран сетевой трафик, соответствующий нормальной работе системы и аномальным состояниям — сетевым атакам. Особенностью этого набора данных является акцент на использование промышленных протоколов, в первую очередь, протокола Modbus в варианте Modbus-over-TCP [20].

Проведя разведку и закрепившись в промышленной сети, злоумышленник может модифицировать управляющие команды или показания датчиков, что может привести к серьезным киберфизическим последствиям. Сетевые атаки на SCADA-системы условно можно разделить на три категории: разведка, внедрение управляющих команд и атаки отказа в обслуживании (DoS/DDoS).

В работе [19] рассмотрены разведывательные сетевые атаки сканирования для выявления возможных уязвимостей, эксплуатация которых позволит злоумышленнику закрепиться в сегменте промышленной сети. Часть атак, при реализации которых существенно возрастает число пересылаемых пакетов, уверенно обнаруживаются стандартными сигнатурными методами. Но большая часть атак с использованием эксплоитов практически не изменяет основные характеристики трафика промышленных протоколов, что делает очень затруднительным подбор сигнатур для их обнаружения. Применение ML-методов позволяет выявить особенности аномального трафика и построить соответствующий детектор.

Система обнаружения сетевых атак в гетерогенной сети промышленного Интернета вещей

Структурная схема системы обнаружения сетевых атак в гетерогенной сети промышленного Интернета вещей на основе интеллектуального анализа данных представлена на рис. 1. Коллектор (4) сетевых сессий собирает параметры трафика с агентов, установленных в ключевых точках сетевой инфраструктуры: агрегирующих коммутаторах, пограничном межсетевом экране, с точек доступа в виде дампа трафика канального уровня и в формате сессий (семейство протоколов netFlow). Модули (5) предобработки, выделения признаков и хранения статистики сетевого трафика позволяют фиксировать в долгосрочном хранилище (ICS БД) компактное описание сетевых сессий, что позволяет проводить ретроспективный анализ накопленных данных и оперативное обновление индикаторов компрометации при взаимодействии (6) с внешними платформами киберразведки. Модуль анализа и генерации признаков (8) используется при подготовке размеченных данных для по-

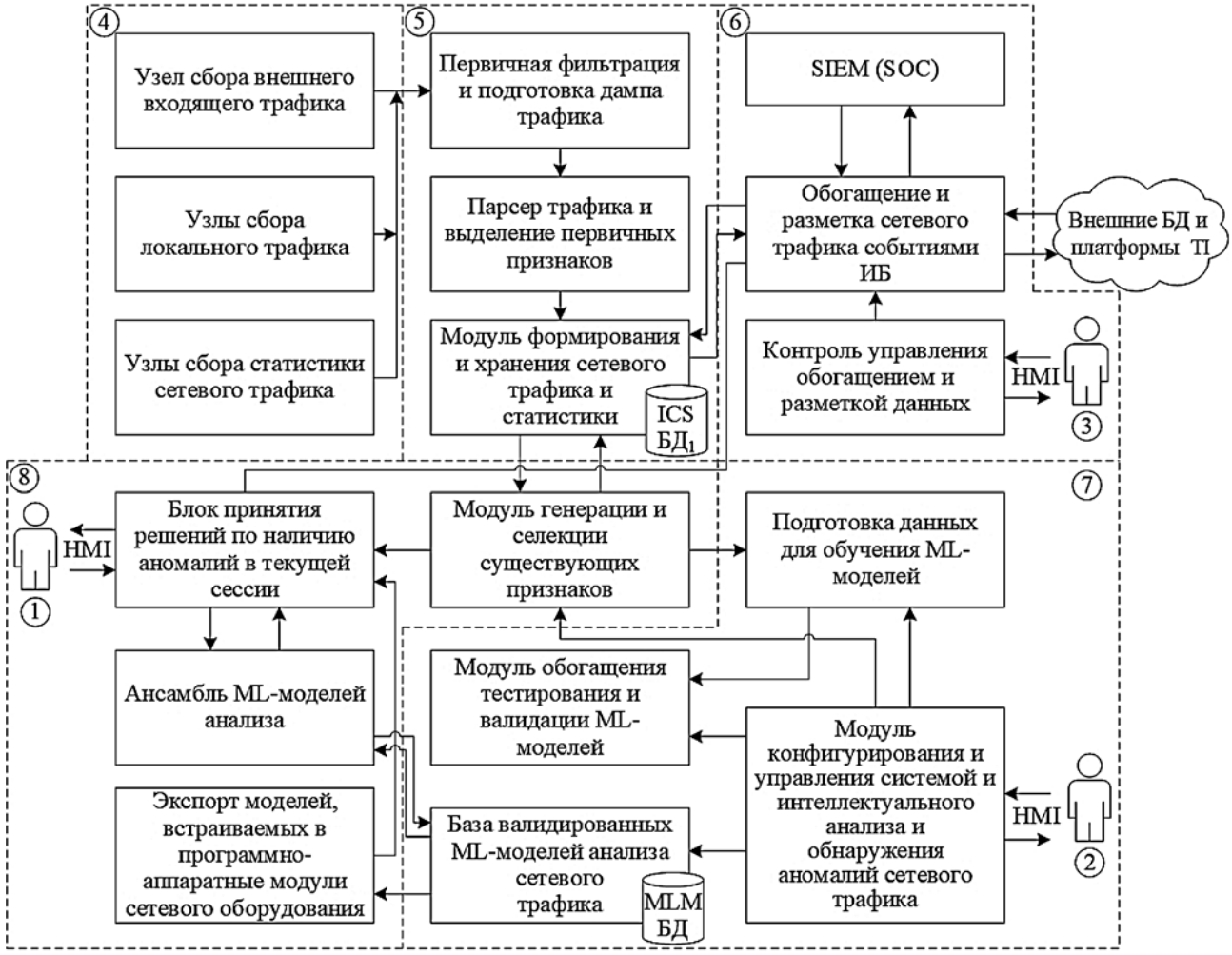


Рис. 1. Структурная схема системы обнаружения сетевых атак на основе интеллектуального анализа данных:
HMI — *Human-machine interface*, человеко-машинный интерфейс; платформа TI — *Threat Intelligence*, платформа управления данными киберразведки

строения и обучения моделей машинного обучения, сохраняемых в БД (MLM БД) для дальнейшего использования при оперативном анализе входящего и внутреннего сетевого трафика.

Модуль обогащения, тестирования и проверки ML-моделей позволяет провести дополнительную разметку сетевого трафика, связав определенные события ИБ с соответствующими сетевыми сессиями.

Оперативное двухстороннее взаимодействие системы в целом с подсистемой управления событиями безопасности и центром мониторинга ИБ и реагирования на инциденты (SIEM (SOC)) позволяет передавать метрики и дополнительную информацию о параметрах текущего состояния сети для последующей агрегации и анализа. Процессом разметки (обогащения) записей сетевых сессий управляет специалист (3) по сетевой безопасности текущего сегмента.

Специалист по интеллектуальному анализу данных (2) управляет работой ансамбля ML-моделей, выполняет задачи по корректировке параметров его работы и своевременного обновления банка моделей.

Итоговый блок принятия решений по обнаружению атак взаимодействует со специалистом (1) по сетевой безопасности и визуализирует результаты анализа ансамбля ML-моделей.

Обобщенный алгоритм интеллектуального анализа параметров сетевого трафика в задаче обнаружения аномалий и вредоносной сетевой активности изображен на рис. 2. Представлены основные этапы сбора и обработки данных для построения и использования ML-моделей.

В целях оценки эффективности предлагаемого решения использовались общедоступные размеченные по типам атак и режимам работы базы данных сетевого трафика (NSL-KDD, CICIDS2017, UNSW-NB15, сети промышленного Интернета вещей — WUSTL-IIOT-2018; беспроводные промышленные сенсорные сети — WSN-DS-2016) и полусинтетические наборы, собранные с использованием полунатурного стенда, моделирующего сегменты корпоративной и промышленной сетей.

Особенностью указанных наборов данных является акцент на использование промышленных протоколов (таких как Modbus). Применение методов машинного обучения и интеллектуального анализа данных позволяет выявить при этом особенности аномального вредоносного трафика атак и внедренных эксплоитов, построить соответствующий детектор.

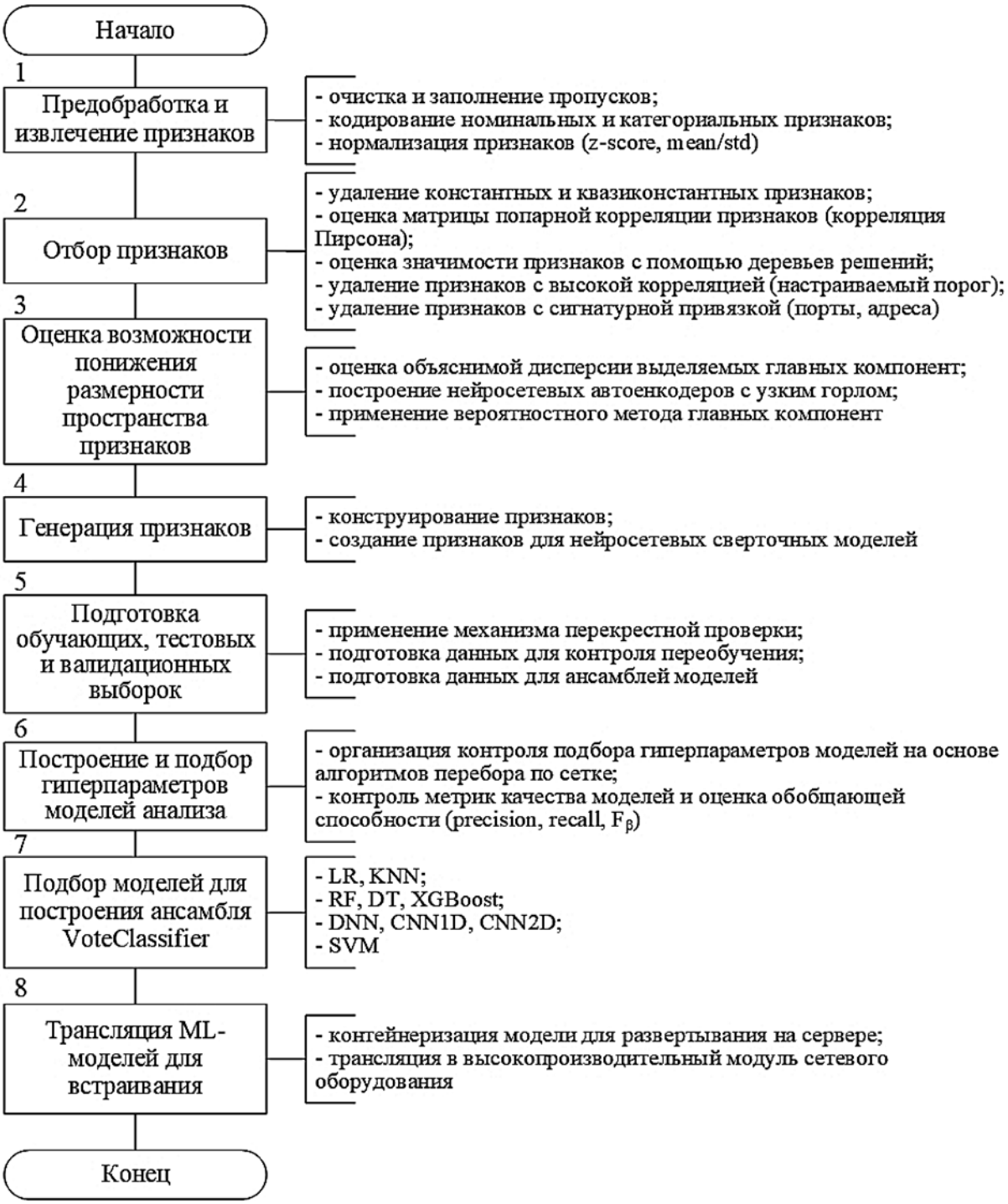


Рис. 2. Обобщенный алгоритм интеллектуального анализа сетевого трафика в задаче обнаружения аномалий и вредоносной сетевой активности

Вычислительные эксперименты

Предлагаемый алгоритм интеллектуального анализа сетевого трафика применялся для анализа наборов данных, представленных в табл. 1.

Далее рассмотрим специфику реализации алгоритма для каждого из наборов данных.

Предобработка и извлечение признаков. На этапе предобработки удаляются идентичные признаки, заполняются или удаляются признаки, содержащие нечисловые значения NaN и Infinity. Значения категориальных признаков (Flow ID, Source IP, Destination IP и Timestamp) преобразуются в численные значения с помощью соответствующей схемы порядкового или унитарного кодирования (Label Encoder или One-Hot-Encoder).

Далее выполняется нормализация признаков с приведением к нулевому среднему и единичному стандартному отклонению.

Таблица 1

Анализируемые наборы данных сетевого трафика

Набор данных	Число сетей (кластеров)	Длительность сбора данных/число записей	Классы атак	Инструменты извлечения признаков	Число признаков	Детальное описание эксперимента
NSL-KDD [17]	2	5 недель/148 517	4	Bro-IDS	41	[21]
CICIDS2017 [18]	1	5 дней/2 830 540	15	CICFlowMeter	84	[22]
UNSW-NB15 [19]	33	16 дней 15 часов/2 059 419	9	Argus, Bro-IDS и др.	47	[23]
WUSTL-IIOT-2018 [19]	1	25 часов/7 037 983	5	ARGUS	4	[24]
WSN-DS-2016 [25]	5	1 день/374 661	4	LEACH protocol	19	[26]

Поскольку наборы данных не сбалансированы, применяют следующие схемы:

- удаление классов с очень малым числом примеров (например, Heartbleed, Web Attack— Sql Injection, Infiltration, Web Attack — XSS и Bot для набора CICIDS2017);
- аугментация имеющейся выборки на основе алгоритмов увеличения числа примеров миноритарного класса (алгоритмы SMOTE) или удаление примеров мажоритарного класса.

Оценка возможности понижения размерности пространства признаков. Оценка суммарной объяснимой дисперсии данных в зависимости от числа главных компонент для набора данных CICIDS2017 представлена на рис. 3.

Ощутимое влияние на долю объясняемого коэффициента дисперсии оказывают первые 20 главных компонент. Дальнейшая процедура отбора признаков позволит существенно сократить их общее число, поэтому применение процедуры понижения размерности пространства признаков не является необходимым. Для набора данных WSN-DS-2016 лучший результат удалось достичь, применив нейросетевой автоенкодер с четырехслойной архитектурой, осуществляющий нелинейное сжатие пространства признаков. Для набора данных NSL-KDD число выделенных главных компонент варьировалось в пределах 4...24.

Отбор признаков. Ярко выраженные сигнатурные признаки, согласно работе [27], удаляются: Flow ID, Source IP, Source Port, Destination IP, Destination Port, Protocol и Timestamp. Это позволит строить модели ML, которые ориентированы на обнаружение статистических особенностей сетевых сессий, соотнесенных с сетевыми атаками, а не с сигнатурными параметрами, которые могут быть изменены или подделаны злоумышленником, и с которыми хорошо справляются традиционные системы обнаружения сетевых атак.

Далее применяют алгоритмы отбора и оценки значимости признаков. Набор данных разделяют на обучающую и тестовую выборки в соотношении 0,7 и 0,3. На обучающей выборке с помощью алгоритма перекрестной проверки с разбиением на 10 групп строится классификатор на основе дерева решений с последующей оценкой значимости признаков. Пример ранжирования признаков по степени значимости для принятия решения о принадлежности к заданному классу для набора данных CICIDS2017 приведен на рис. 4.

Оценка значимости признаков выполняется также с помощью комитета (k = 250) случайных деревьев решений (RF) с использованием процедуры перекрестной проверки. Гистограмма оценки значимости выделенных с помощью RF признаков представлена на рис. 5.

Для набора данных WUSTL-IIOT-2018 оценка значимости признаков по аналогичному сценарию позволяет выдвинуть гипотезу о возможности оставить один или два наиболее значимых признака для построения классификатора (рис. 6, см. вторую сторону обложки).

Используемые методы отбора признаков позволяют сократить их число в 4—5 раз.

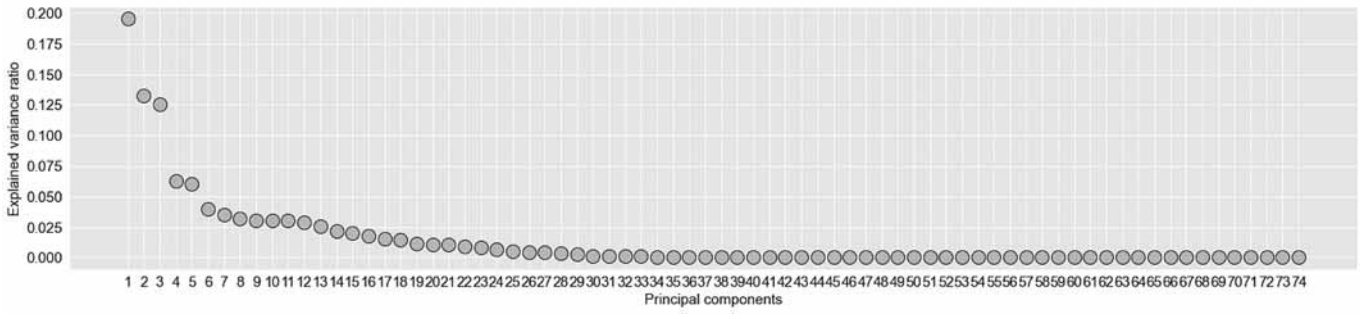


Рис. 3. Зависимость суммарной объяснимой дисперсии (ось ординат) от числа главных компонент (ось абсцисс)

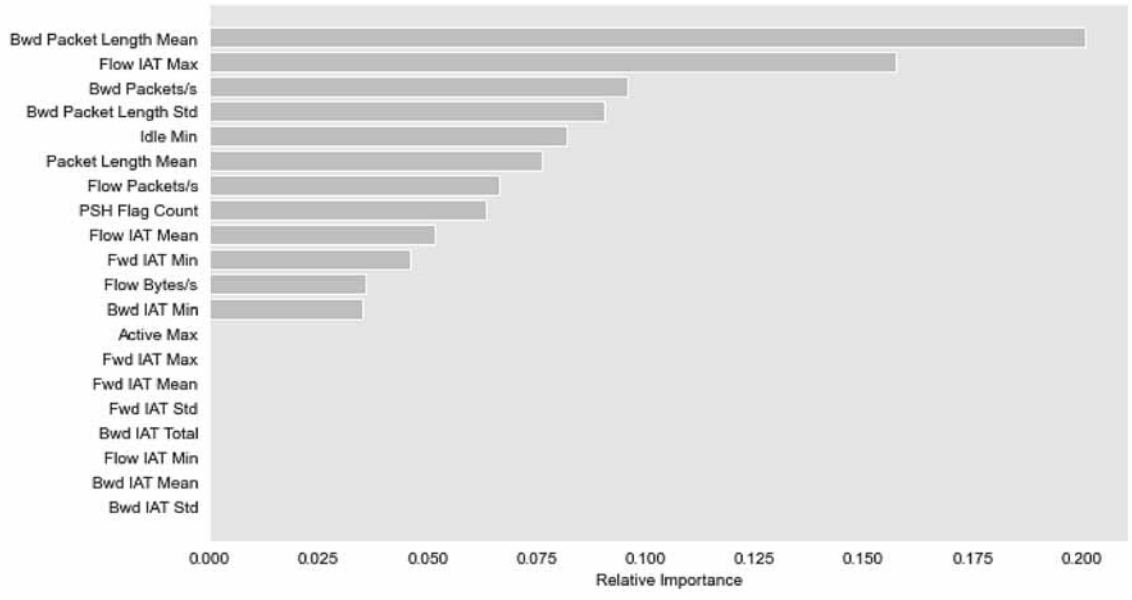


Рис. 4. Гистограмма оценки значимости признаков (ось ординат — признаки), полученная с помощью классификатора на основе дерева решений (ось абсцисс — относительные единицы)

Выполняется поиск и удаление константных и квазиконстантных (с порогом вариабельности (дисперсия) за период анализа (обучающая выборка) 0,005). Далее проводится оценка степени попарной корреляции признаков и удаление признаков с коэффициентом корреляции более установленного порога (например, для набора данных CICIDS2017 порог выбран равным 0,8). Итоговая тепловая карта матрицы попарной корреляции приведена на рис. 7. Полученные результаты согласуются с данными работы [27].

Для понижения размерности пространства признаков CICIDS2017 и визуализации распределения примеров по классам применен метод стохастического вложения соседей с *t*-распределением для понижения размерности пространства признаков и визуализации распределения примеров по классам (рис. 8, см. вторую сторону обложки). Визуализация классов атак и нормальной работы набора данных WUSTL-IIOT-2018, напротив, позволяет сделать однозначный вывод о наличии структуры данных с сокращенным набором признаков и возможности дальнейшего построения классификатора (рис. 9, см. вторую сторону обложки).

Построение классификаторов и ансамблей. Для решения задачи обнаружения сетевых атак на основе формализованного вектора признаков необходимо создание и подбор параметров моделей машинного обучения. Применена процедура оптимизации

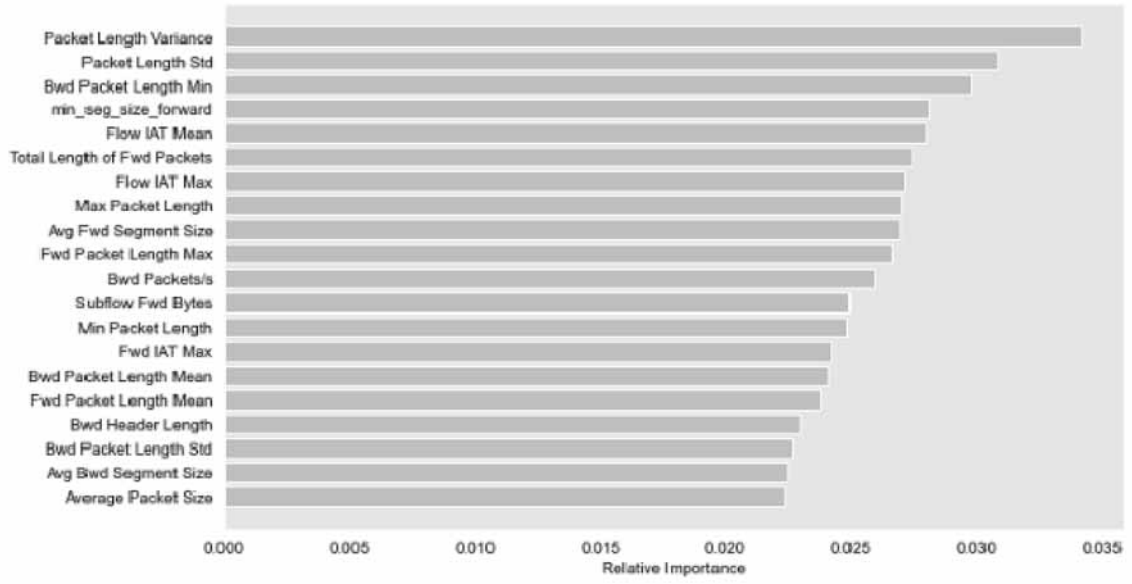


Рис. 5. Гистограмма оценки значимости признаков (ось ординат — признаки), полученная с помощью классификатора на основе комитета деревьев решений (ось абсцисс — относительные единицы)

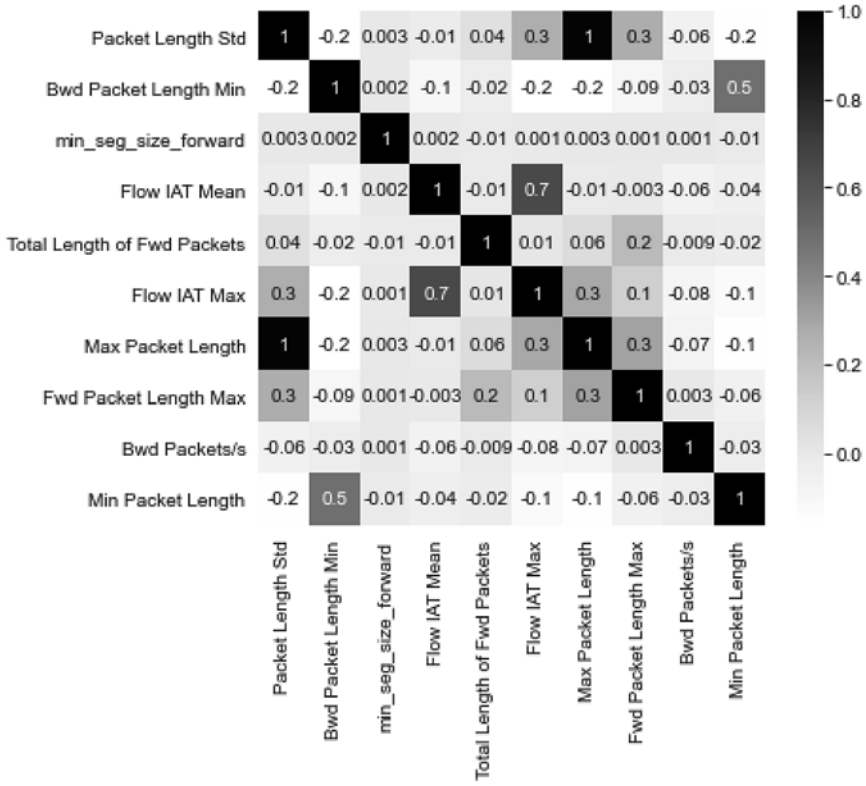


Рис. 7. Матрица попарной корреляции после исключения связанных признаков CICIDS2017 (коэффициент корреляции в диапазоне [−1, 1])

гиперпараметров каждой модели с применением поиска по сетке, перекрестной проверкой с десятью проходами и оценкой качества модели на выделяемой тестовой выборке.

Применяемые классификаторы [28]:

- на основе алгоритма градиентного бустинга для ансамбля деревьев решений (XGBClassifier);
- на основе комитета деревьев решений (*Random Forest*, RF);
- на основе k-ближайших соседей (*K-Nearest Neighbors*, KNN);
- на основе машины опорных векторов (*Support Vector Machines*, SVM);
- на основе логистической регрессии (*Logistic* Regression, LR);
- на основе "мелкой" нейронной сети прямого распространения — многослойный персептрон (shallow MLP);
- на основе сверточных нейронных сетей с одномерным и двумерным входным слоем (CNN1D и CNN2D соответственно);
- на основе глубокой нейронной сети (DNN).

Для классификатора на основе сверточной нейронной сети с двумерным входным слоем признаков CNN2D векторы признаков образцов набора преобразованы в графические примитивы размерностью 5×2 (CICIDS2017) в градациях серого (рис. 10).

На заключительном этапе строится ансамбль классификаторов, включающий в себя комитет деревьев решений (RF), классификатор на основе алгоритма градиентного бустинга на ансамбле деревьев решений (XGBClassifier) и ExtraTreesClassifier. Последний реализует метаоценку, соответствующую набору рандомизированных деревьев решений, или деревьев на различных подвыборках набора данных, использует усреднение для повышения точности прогнозирования и контроля избыточной подгонки отдельных моделей. Параметры комитета: тип голосования — soft (голосование и взвешивание предсказаний моделей для каждого класса); веса моделей распределены как {2, 1, 3}.

После подбора параметров классификаторов и выбора ML-моделей, продемонстрировавших приемлемую обобщающую способность, выполняется итоговая оценка моделей. По сочетанию времени, затрачиваемого на обучение модели, суммарного числа настраиваемых параметров модели, показателей F1-меры и правильности (Acc — Accuracy) итоговой модели на тестовой выборке выбран классификатор на основе комитета деревьев решений (табл. 2).

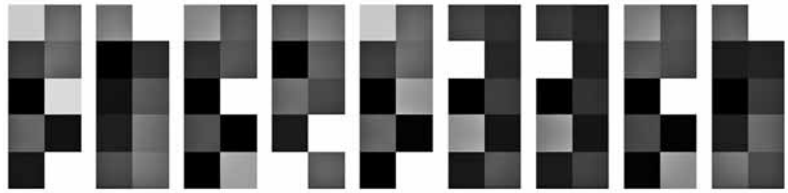


Рис. 10. Двумерное представление признаков примеров набора данных

Таблица 2

Результаты тестирования классификатора

Классификатор	CICIDS2017		NSL-KDD		UNSW-NB15		WUSTL-IIOT-2018		WSN-DS-2016	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
RF	0,967	0,910	0,896	0,885	0,897	0,810	0,999	0,919	0,999	0,947

Одни из лучших результатов показывают классификаторы XGBClassifier, случайный лес и нейросетевые модели. В абсолютном значении лучшую эффективность показал VotingClassifier.

Подготовка ML-моделей для встраивания. Обученные ML-модели RF и MLP предлагается использовать в виде встраиваемых программных модулей соответствующего сетевого оборудования. С помощью транслятора с языка Python созданы заголовочные файлы и файлы реализации на языке C с выгрузкой коэффициентов обученных моделей в качестве статических параметров. Дальнейшая компиляция с помощью кросс-компилятора позволила собрать исполняемые модули для платформы ARM семейства специализированных процессоров NXP LX2160A.

Разработка полигона для тестирования предложенных решений

Для тестирования предлагаемых моделей машинного обучения разработана архитектура стенда промышленного объекта, имитирующая основные элементы инфраструктуры (рис. 11), и включающая основные уровни: полевой, сбора данных, управления и т. п.

Мониторинг состояния информационной и сетевой инфраструктуры реализован на основе развернутого решения на базе ELK-стека (Elasticsearch, Beats, Logstash, Kibana) [29].

Процесс мониторинга [29] разбит на пять шагов.

1. Источником событий выступают AuditBeat и WinlogBeat на серверах.

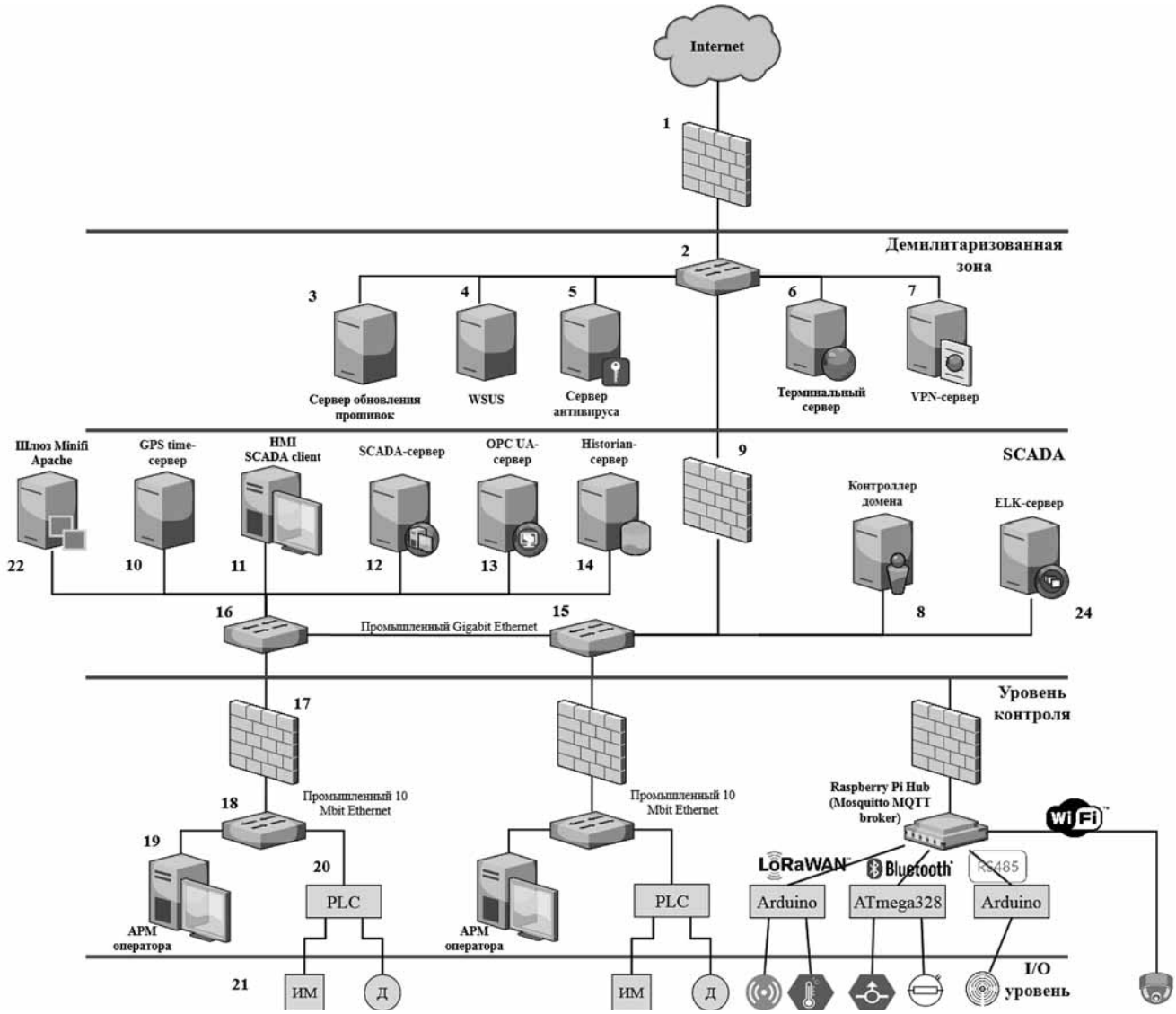


Рис. 11. Архитектура стенда промышленного объекта:
ИМ — исполнительные механизмы; Д — датчики

2. Данные собираются в SIEM-системе Apache NiFi.

3. Хранение происходит в Elasticsearch.

4. Данные обрабатываются и визуализируются с помощью Kibana.

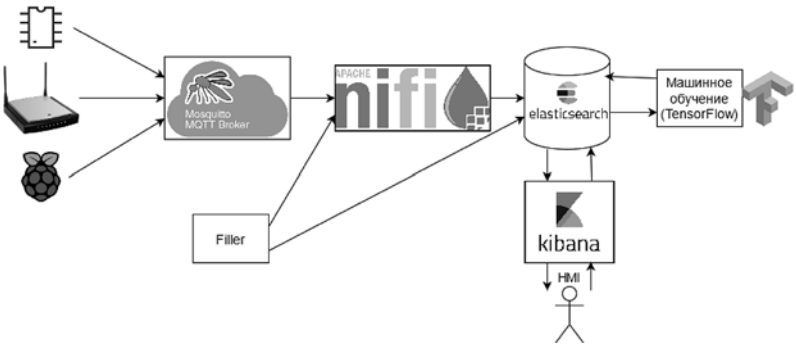


Рис. 12. Упрощенная схема системы сбора и обработки данных о событиях ИБ модельного объекта:
Filler — источник дополнительных данных о событиях ИБ

5. Полученная информация анализируется экспертами.

Упрощенная схема системы сбора и обработки данных о событиях ИБ модельного объекта приведена на рис. 12.

Данные поступают в систему мониторинга, сбора и корреляции событий ИБ в промышленной сети через коллектор данных (реализованный с помощью MQTT-брокера). Связующим звеном является Apache NiFi-сервер, пересылающий полученные данные в Elasticsearch, где к ним может иметь доступ подсистема машинного обучения на базе TensorFlow.

Согласно архитектуре стенда (см. рис. 11 и 12), в системе эмуляции и виртуализации EVE-NG [30] спроектирован и реализован стенд, изображенный на рис. 13.

Краткие описания устройств в составе стенда приведены в табл. 3.

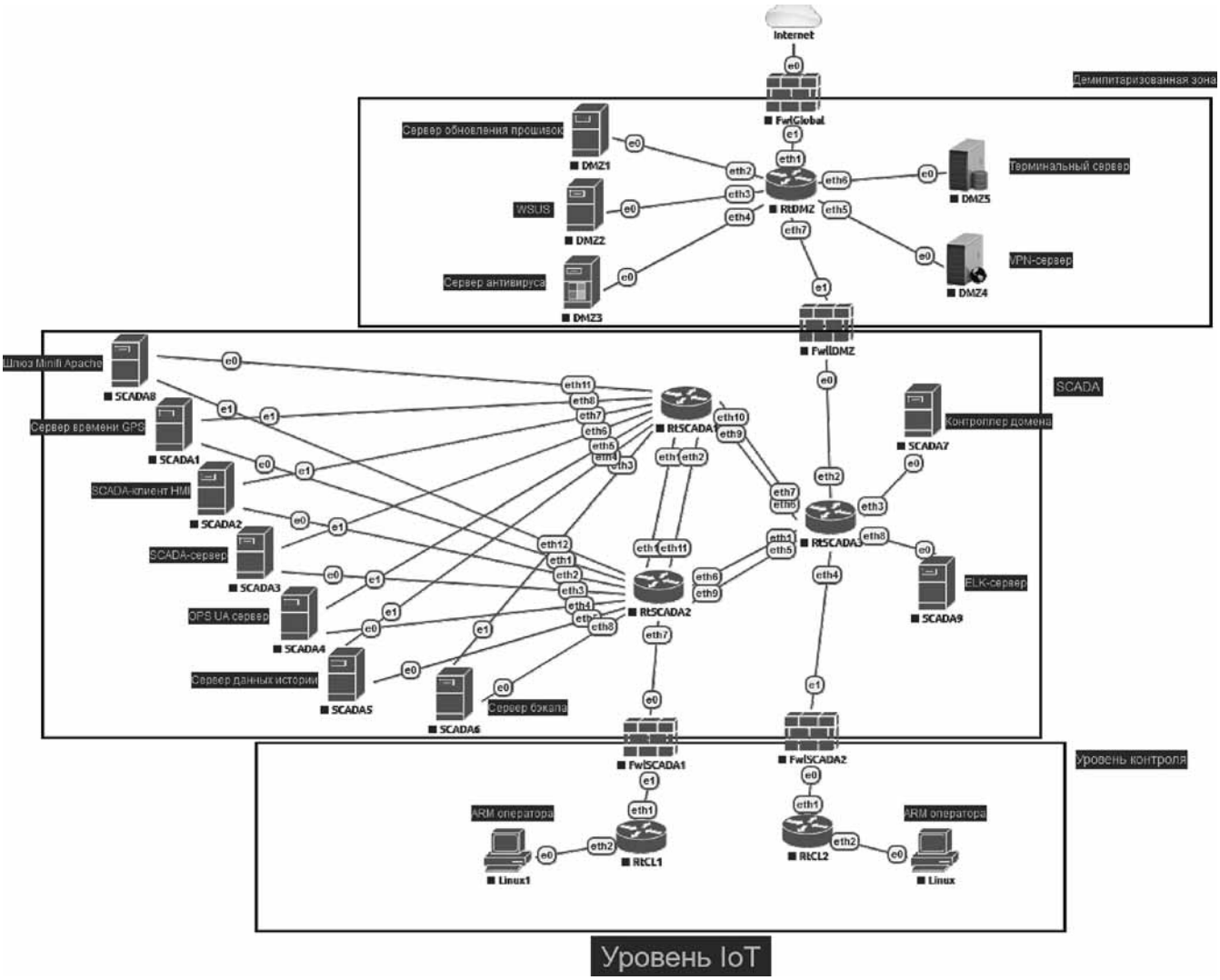


Рис. 13. Архитектура разработанного стенда сегмента промышленной сети:
e0-e1, eth0-eth10 — нумерация сетевых интерфейсов

Таблица 3

Список использованных устройств и их характеристики

№	Зона	Имя устройства	Тип устройства	Название устройства	Используемый образ
1	Демилитаризованная зона	FwlGlobal	Межсетевой экран pfSense	—	Pfsense-CE-2.4.5
2	Демилитаризованная зона	RtDMZ	Маршрутизатор Mikrotik	—	mikrotik-7.0b
3	Демилитаризованная зона	DMZ1	Linux-сервер	Сервер обновления прошивок	linux-Kali-full
4	Демилитаризованная зона	DMZ2	Linux-сервер	WSUS	windows-server-2016
5	Демилитаризованная зона	DMZ3	Linux-сервер	Сервер антивируса	linux-Kali-full
7	Демилитаризованная зона	DMZ4	Linux-сервер	VPN-сервер	linux-Kali-full
6	Демилитаризованная зона	DMZ5	Linux-сервер	Терминальный сервер	linux-Kali-full
9	Демилитаризованная зона	FwlDMZ	Межсетевой экран pfSense	—	pfsense-CE-2.4.5
8	SCADA	SCADA7	Linux-сервер	Контроллер домена	linux-Kali-full
10	SCADA	SCADA1	Linux-сервер	Сервер времени GPS	linux-Kali-full
11	SCADA	SCADA2	Linux-сервер	SCADA-клиент HMI	linux-Kali-full
12	SCADA	SCADA3	Linux-сервер	SCADA-сервер	linux-debian8-openscada
13	SCADA	SCADA4	Linux-сервер	OPS UA-сервер	linux-Kali-full
14	SCADA	SCADA5	Linux-сервер	Сервер данных истории	linux-Kali-full
15	SCADA	RtSCADA1	Маршрутизатор Mikrotik	—	mikrotik-7.0b
16	SCADA	RtSCADA3	Маршрутизатор Mikrotik	—	mikrotik-7.0b
17	SCADA	FwlSCADA1	Межсетевой экран pfSense	—	pfsense-CE-2.4.5
22	SCADA	SCADA8	Linux-сервер	Шлюз Minifi Apache	linux-Debian-10-srv
23	SCADA	RtSCADA2	Маршрутизатор Mikrotik	—	mikrotik-7.0b
24	SCADA	SCADA9	Linux-сервер	ELK-сервер	linux-ELK
25	SCADA	SCADA6	Linux-сервер	Сервер бэкапа	linux-Debian-10-srv
26	SCADA	FwlSCADA2	Межсетевой экран pfSense	—	pfsense-CE-2.4.5
18	Уровень контроля	RtCL1	Маршрутизатор Mikrotik	—	mikrotik-7.0b
19	Уровень контроля	Linux	ПК с ОС Linux	ARM оператора	linux-tinycore-6.4
27	Уровень контроля	RtCL2	Маршрутизатор Mikrotik	—	mikrotik-7.0b
28	Уровень контроля	Linux1	ПК с ОС Linux	ARM оператора	linux-tinycore-6.4

ELK-сервер (24) осуществляет сбор всех типов данных для их последующего использования в системе машинного обучения распознавания аномалий сетевого трафика.

Эксперименты в виртуальном полигоне с разработанными ML-моделями и сценариями реализации сетевых атак подтвердили эффективность предлагаемого решения.

Заключение

Разработана и описана структурная схема системы обнаружения сетевых атак на основе интеллектуального анализа данных.

Разработаны алгоритмы интеллектуального анализа параметров сетевого трафика в задаче обнаружения вредоносной сетевой активности. Приведена общая схема алгоритма. Проанализированы варианты построения ансамблей и комитетов классификаторов на основе традиционных моделей машинного обучения (модели случайного леса, рандомизированные деревья решений и пр.) и гетерогенных нейросетевых моделей (глубокие нейронные сети, сверточные нейронные сети и модели на основе автоенкодеров с долгой краткосрочной памятью). Оценка F1-меры при работе с тестовыми выборками достигает 96 %.

Проанализирована возможность встраивания полученных моделей в качестве модулей сетевого оборудования для повышения оперативности анализа сетевого трафика промышленных систем или использования в составе сетевой системы обнаружения вторжений.

Эффективность полученных решений при оценке качества обнаружения сетевых атак на исходных наборах данных сравнима для протестированных моделей. Наиболее перспективным для применения в специализированных сигнальных процессорах

сетевого оборудования является классификатора на основе комитета случайных деревьев. Данный классификатор обеспечивает хорошее качество обнаружения сетевых атак и не требует значительных вычислительных ресурсов при запуске модели с подобранными в процессе обучения коэффициентами.

Разработан виртуальный полигон для оценки эффективности применения ML-моделей для обнаружения сетевых атак. Дальнейшие исследования направлены на разработку методики испытаний моделей в различных сценариях реализации целенаправленных многошаговых сетевых атак.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-08-00668.

Список литературы

1. **Moore B.** Gartner's top 10 IoT tech trends // IT Brief. URL: https://itbrief.com.au/story/gartner-s-top-10-iot-tech-trends
2. **Актуальные киберугрозы**: IV квартал 2020 года. Отчет Positive Technologies. URL: https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2020-q4/
3. **Ландшафт угроз** для систем промышленной автоматизации. 2019 год // Kaspersky ICS CERT. URL: https://ics-cert.kaspersky.ru/reports/2020/04/24/threat-landscape-for-industrial-automation-systems-2019-report-at-a-glance/
4. **Cecil A.** A Summary of Network Traffic Monitoring and Analysis Techniques. URL: https://www.cse.wustl.edu/~jain/cse567-06/ftp/net_monitoring/index.html
5. **Гайфулина Д. А., Котенко И. В.** Применение методов глубокого обучения в задачах кибербезопасности. Часть 1 // Вопросы кибербезопасности. 2020. № 3. С. 76—86.
6. **Monshizadeh M., Khatri V., Atli B., Kantola R.** Performance evaluation of a combined anomaly detection platform // IEEE Access. 2019. Vol. 7. P. 100964—100978.
7. **Moustafa N., Creech G., Sitnikova E., Keshk M.** Collaborative anomaly detection framework for handling big data of cloud computing // 2017 military communications and information systems conference (MilCIS). IEEE, 2017. P. 1—6.
8. **Ten C. W., Manimaran G., Liu C. C.** Cybersecurity for critical infrastructures: Attack and defense modeling // IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans. 2010. Vol. 40, No. 4. P. 853—865.
9. **Ten C. W., Hong J., Liu C. C.** Anomaly detection for cybersecurity of the substations // IEEE Transactions on Smart Grid. 2011. Vol. 2, No 4. P. 865—873.
10. **Alrashdi I., Alqazzaz A., Al Oufi E.** et al. Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning // 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2019. P. 305—310.
11. **Kiss I., Genge B., Haller P., Sebestyen G.** Data clustering-based anomaly detection in industrial control systems // 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP). IEEE, 2014. P. 275—281.
12. **Cruz T., Rosa L., Proença J.** et al. A cybersecurity detection framework for supervisory control and data acquisition systems // IEEE Transactions on Industrial Informatics. 2016. Vol. 12, No. 6. P. 2236—2246.
13. **Tartakovsky A. G., Polunchenko A. S., Sokolov G.** Efficient computer network anomaly detection by changepoint detection methods // IEEE Journal of Selected Topics in Signal Processing. 2012. Vol. 7, No 1. P. 4—11.
14. **Keshk M., Sitnikova E., Moustafa N.** et al. An integrated framework for privacy-preserving based anomaly detection for cyber-physical systems // IEEE Transactions on Sustainable Computing. 2019. Vol. 6. № 1. P. 66—79.
15. **Gómez Á. L. P., Fernández-Maimó L., Huertas A.** et al. On the generation of anomaly detection datasets in industrial control systems // IEEE Access. 2019. Vol. 7. P. 177460—177473.
16. **Tavallaee M., Bagheri E., Lu W., Ghorbani A.** A detailed analysis of the KDD CUP 99 data set // 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE, 2009. P. 1—6.
17. **Sharafaldin I., Lashkari A. H., Ghorbani A. A.** Toward generating a new intrusion detection dataset and intrusion traffic characterization // ICISSp. 2018. Vol. 1. P. 108—116.
18. **Moustafa N., Slay J.** UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set) // 2015 military communications and information systems conference (MilCIS). IEEE, 2015. P. 1—6.
19. **Teixeira M., Salman T., Zolanvari M., Jain R.** SCADA system testbed for cybersecurity research using machine learning approach // Future Internet. 2018. Vol. 10, No. 8. P. 76.
20. **Miciolino E., Bernieri G., Pascucci F., Setola R.** Communications network analysis in a SCADA system testbed under cyber-attacks // 2015 23rd Telecommunications Forum Telfor (TELFOR). IEEE, 2015. P. 341—344.
21. **Sapozhnikova M. U., Nikonov A. V., Vulfin A. M.** Intrusion detection system based on data mining technics for industrial networks // 2018 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). IEEE, 2018. P. 1—5.
22. **Vulfin A., Vasilyev V., Gvozdev V.** et al Network traffic analysis based on machine learning methods // Journal of Physics: Conference Series. IOP Publishing, 2021. Vol. 2001, No 1. P. 012017.
23. **Gurin M., Vulfin A., Vasilyev V., Nikonov A.** Intrusion detection system on the basis of data mining algorithms in the industrial network // CEUR Workshop Proceedings. 2019. P. 553—565.
24. **Vulfin A., Vasilyev V., Kuharev S.** Algorithms for detecting network attacks in an enterprise industrial network based on data mining algorithms // Journal of Physics: Conference Series. IOP Publishing, 2021. Vol. 2001, No. 1. P. 012004.
25. **Almomani I., Al-Kasasbeh B., Al-Akhras M.** WSN-DS: A dataset for intrusion detection systems in wireless sensor networks // Journal of Sensors. 2016. Vol. 2016. P. 4731953:1-4731953:16.
26. **Васильев В. И., Вульфин А. М., Картак В. М.** и др. Система обнаружения атак в беспроводных сенсорных сетях промышленного Интернета вещей // Труды ИСА РАН. 2019. Т. 69, № 4. С. 70—78.
27. **Горюнов М. Н., Мацкевич А. Г., Рыболовлев Д. А.** Синтез модели машинного обучения для обнаружения компьютерных атак на основе набора данных CICIDS2017 // Труды ИСП РАН. 2020. Т. 32, № 5. С. 81—93.
28. **Kotsiantis S. B.** Supervised machine learning: A review of classification techniques // Informatica. 2007. Vol. 31, No. 3. URL: https://www.researchgate.net/publication/265544297_Supervised_Machine_Learning_A_Review_of_Classification_Techniques
29. **Вульфин А. М.** Система управления данными кибер разведки // Моделирование, оптимизация и информационные технологии. 2021. Т. 9 (1). URL: https://moitvivt.ru/ru/journal/pdf?id=925 DOI: 10.26102/2310-6018/2021.32.1.020.
30. **Tobarra L., Robles-Gómez A., Pastor V.** et al. A Cybersecurity Experience with Cloud Virtual-Remote Laboratories // Multidisciplinary Digital Publishing Institute Proceedings. 2019. Vol. 31, No. 1. P. 3.

Detection of Network Attacks in a Heterogeneous Industrial Network Based on Machine Learning

A. M. Vulfin, vulfin.alexey@gmail.com, Ufa State Aviation Technical University

Corresponding author:

Vulfin Aleksey M., Associate Professor, Ufa State Aviation Technical University
E-mail: vulfin.alexey@gmail.com

Received on January 25, 2021
Accepted on December 15, 2021

The paper discusses the issues of improving algorithms for detecting network attacks in a heterogeneous industrial Internet of Things network based on machine learning technologies for subsequent integration with the subsystems of the center for monitoring and responding to information security incidents. A structural diagram of a network attack detection system and an algorithm for intelligent analysis of network traffic parameters in the task of detecting malicious network activity have been developed. Variants of constructing ensembles of classifiers based on machine learning models and heterogeneous neural network models are analyzed. The F1-measure score when working with test samples reaches 96 %. The possibility of embedding the obtained models as modules of network equipment to increase the efficiency of the analysis of network traffic of industrial systems or use as part of a network intrusion detection system is considered. The efficiency of the obtained solutions in assessing the quality of network attack detection on the original datasets is comparable for the tested models. The most promising for use in specialized signal processors of network equipment is a classifier based on a committee of random trees, since it provides good quality detection of network attacks and does not require significant computing resources when launching a model with coefficients selected during training. Monitoring the state of information and network infrastructure is implemented on the basis of a deployed solution based on the ELK stack. A virtual testing ground has been developed to assess the effectiveness of ML-models for detecting network attacks. Further research is aimed at developing a methodology for testing models in various scenarios for the implementation of targeted multi-step network attacks.

Keywords: *network attacks, machine learning, data mining, ensemble of classifiers, heterogeneous industrial network, monitoring and response to information security incidents*

For citation:

Vulfin A. M. Detection of Network Attacks in a Heterogeneous Industrial Network Based on Machine Learning, *Programmnaya Ingeneria,* 2022, vol. 13, no. 2, pp. 68—80.

DOI: 10.17587/prin.13.68-80

References

1. **Moore B.** Gartner's top 10 IoT tech trends, *IT Brief*, available at: https://itbrief.com.au/story/gartner-s-top-10-iot-tech-trends

2. **Topical Cyber Threats:** Q4 2020. Positive Technologies report, available at: https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2020-q4/

3. **Threat landscape** for industrial automation systems. 2019 year, *Kaspersky ICS CERT,* available at: https://ics-cert.kaspersky.ru/reports/2020/04/24/threat-landscape-for-industrial-automation-systems-2019-report-at-a-glance/

4. **Cecil A.** A Summary of Network Traffic Monitoring and Analysis Techniques, available at: https://www.cse.wustl.edu/~jain/cse567-06/ftp/net_monitoring/index.html.

5. **Gaifulina D. A., Kotenko I. V.** Application of deep learning methods in cybersecurity tasks, *Voprosy kiberbezopasnosti*, 2020, no. 3, pp. 76—86 (in Russian).

6. **Monshizadeh M., Khatri V., Atli B., Kantola R.** Performance evaluation of a combined anomaly detection platform, *IEEE Access*, 2019, vol. 7, pp. 100964—100978.

7. **Moustafa N., Creech G., Sitnikova E., Keshk M.** Collaborative anomaly detection framework for handling big data of cloud computing, *2017 military communications and information systems conference (MilCIS),* IEEE, 2017, pp. 1—6.

8. **Ten C. W., Manimaran G., Liu C. C.** Cybersecurity for critical infrastructures: Attack and defense modeling, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 2010, vol. 40, no. 4, pp. 853—865.

9. **Ten C. W., Hong J., Liu C. C.** Anomaly detection for cybersecurity of the substations, *IEEE Transactions on Smart Grid*, 2011, vol. 2, no. 4, pp. 865—873.

10. **Alrashdi I., Alqazzaz A., Al Oufi E.** et al. Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning, *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC),* IEEE, 2019, pp. 305—310.

11. **Kiss I., Genge B., Haller P., Sebestyen G.** Data clustering-based anomaly detection in industrial control systems, *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP),* IEEE, 2014, pp. 275—281.

12. **Cruz T., Rosa L., Proença, J.** et al. A cybersecurity detection framework for supervisory control and data acquisition systems, *IEEE Transactions on Industrial Informatics*, 2016, vol. 12, no. 6, pp. 2236—2246.

13. **Tartakovsky A. G., Polunchenko A. S., Sokolov G.** Efficient computer network anomaly detection by changepoint detection methods, *IEEE Journal of Selected Topics in Signal Processing*, 2012, vol. 7, no. 1, pp. 4—11.

14. **Keshk M., Sitnikova E., Moustafa N.** et al. An integrated framework for privacy-preserving based anomaly detection for cyber-physical systems, *IEEE Transactions on Sustainable Computing*, 2019, vol. 6, no. 1, pp. 66—79.

15. **Gómez Á. L. P., Fernández-Maimó L., Huertas A.** et al. On the generation of anomaly detection datasets in industrial control systems, *IEEE Access*, 2019, vol. 7, pp. 177460—177473.

16. **Tavallaee M.,Bagheri E., Lu W., Ghorbani A.** A detailed analysis of the KDD CUP 99 data set, *2009 IEEE symposium on computational intelligence for security and defense applications,* IEEE, 2009, pp. 1—6.

17. **Sharafaldin I., Lashkari A. H., Ghorbani A. A.** Toward generating a new intrusion detection dataset and intrusion traffic characterization, *ICISSp*, 2018, vol. 1, pp. 108—116.

18. **Moustafa N., Slay J.** UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), *2015 military communications and information systems conference (MilCIS),* IEEE, 2015, pp. 1—6.

19. **Teixeira M., Salman T., Zolanvari M., Jain R.** SCADA system testbed for cybersecurity research using machine learning approach, *Future Internet*, 2018, vol. 10, no. 8, pp. 76.

20. **Miciolino E., Bernieri G., Pascucci F., Setola R.** Communications network analysis in a SCADA system testbed under cyber-attacks, *2015 23rd Telecommunications Forum Telfor (TELFOR),* IEEE, 2015, pp. 341—344.

21. **Sapozhnikova M. U., Nikonov A. V., Vulfin A. M.** Intrusion detection system based on data mining technics for industrial networks, *2018 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM),* IEEE, 2018, pp. 1—5.

22. **Vulfin A., Vasilyev V., Gvozdev V.** et al. Network traffic analysis based on machine learning methods, *Journal of Physics: Conference Series. IOP Publishing*, 2021, vol. 2001, no. 1, pp. 012017.

23. **Gurin M., Vulfin A., Vasilyev V., Nikonov A.** Intrusion detection system on the basis of data mining algorithms in the industrial network, *CEUR Workshop Proceedings*, 2019, pp. 553—565.

24. **Vulfin A., Vasilyev V., Kuharev S.** Algorithms for detecting network attacks in an enterprise industrial network based on data mining algorithms, *Journal of Physics: Conference Series. IOP Publishing*, 2021, vol. 2001, no. 1, pp. 012004.

25. **Almomani I., Al-Kasasbeh B., Al-Akhras M.** WSN-DS: A dataset for intrusion detection systems in wireless sensor networks, *Journal of Sensors*, 2016, vol. 2016, pp. 4731953:1—4731953:16.

26. **Vasilyev V. I., Vulfin A. M., Kartak V. M.** et al. System of attacks detection in wireless sensor networks of Industrial Internet of Things, *Trudy ISA RAN*, 2019, vol. 69, no. 4, pp. 70—78 (in Russian).

27. **Goryunov M. N., Matskevich A. G., Rybolovlev D. A.** Synthesis of a Machine Learning Model for Detecting Computer Attacks Based on the CICIDS2017 Dataset, *Trudy ISP RAN*, 2020, vol. 32, no. 5, pp. 81—93 (in Russian).

28. **Kotsiantis S. B.** Supervised machine learning: A review of classification techniques, *Informatica,* 2007, vol. 31, no. 3, available at: https://www.researchgate.net/publication/265544297_Supervised_Machine_Learning_A_Review_of_Classification_Techniques

29. **Vulfin A. M.** Cyber threat intelligence data management system, *Modeling, optimization and information technology.* 2021, vol. 9 (1), available at: https://moitvivt.ru/ru/journal/pdf?id=925, DOI: 10.26102/2310-6018/2021.32.1.020 (in Russian).

30. **Tobarra L., Robles-Gómez A., Pastor V.** et al. A Cybersecurity Experience with Cloud Virtual-Remote Laboratories, *Multidisciplinary Digital Publishing Institute Proceedings*, 2019, vol. 31, no. 1, pp. 3.

ИНФОРМАЦИЯ

УДК 004.5 DOI: 10.17587/prin.13.81-87

Д. И. Читалов, мл. науч. сотр., cdi9@yandex.ru, Федеральное государственное бюджетное учреждение науки Южно-Уральский федеральный научный центр минералогии и геоэкологии Уральского отделения Российской академии наук, г. Миасс, Ильменский заповедник

О разработке модуля для работы с решателем buoyantSimpleFoam и утилитой postProcess платформы OpenFOAM

Настоящая статья обобщает результаты исследования по расширению исходного кода графической оболочки платформы OpenFOAM для обеспечения доступа специалиста к новым возможностям — решателю buoyantSimpleFoam и утилите postProcess в рамках постановки экспериментов по моделированию задач механики сплошных сред. Обоснована актуальность исследования, поставлены цели, сформулированы задачи исследования. Статья содержит описание средств разработки, а также диаграммы, описывающие структуру и логику работы программы. Представлены результаты тестирования приложения применительно к одной из фундаментальных задач механики сплошных сред. Сформулированы выводы по результатам работы и практической ценности исследования.

Ключевые слова: *численное моделирование, механика сплошных сред, графический интерфейс пользователя, OpenFOAM, язык программирования Python, открытое программное обеспечение, решатель buoyantSimpleFoam, утилита postProcess, библиотека PyQt*

Введение

В рамках исследования, которое рассматривается в настоящей работе, автором продолжено расширение исходного кода графической оболочки для проведения численного моделирования задач механики сплошных сред (МСС) посредством платформы OpenFOAM [1]. По результатам исследования базовая версия графической оболочки дополнена еще одним компонентом. Это программный модуль, призванный обеспечить доступ специалиста к новым возможностям численного моделирования задач МСС в части их решения и постпроцессинга.

Платформа OpenFOAM считается одним из ведущих программных средств, применяемых отечественными и зарубежными предприятиями сферы тяжелого машиностроения при проектировании продукции. Благодаря данному программному комплексу выявляются особенности взаимодействия изделий с окружающими процессами и оцениваются качественные характеристики изделий. К их числу относятся такие как линейная прочность, жесткость, термопрочность, динамика, аэроупругость, гидроупругость.

У OpenFOAM существуют программы-аналоги, например, комплекс ANSYS [2]. Однако в отличие от OpenFOAM он является коммерческим продуктом. Имея статус свободно распространяемой программной платформы, OpenFOAM практически не уступает по функциональным возможностям комплексу ANSYS. Перечень решаемых на этой платформе задач расширяется с выходом каждой новой версии. Под расширением понимается внедрение новых программ-решателей, а также утилит пре- и постпроцессинга.

Под программой-решателем (солвером) понимается алгоритм численного моделирования задач определенной области МСС, например, движения сжимаемых и несжимаемых жидкостей с поддержкой различных термодинамических моделей и моделей турбулентности, движения газов, изменения параметров деформируемого твердого тела и т. д. В зависимости от особенностей моделируемой задачи специалист вручную формирует специальную директорию (расчетный случай) и создает в ней необходимые для расчета файлы-словари с параметрами задачи МСС.

Слабым звеном в этом процессе является то обстоятельство, что специалист все этапы численного моделирования, в том числе подготовку служебных файлов расчетного случая, запуск утилит и решателей осуществляет вручную, что требует затрат времени и приводит к появлению ошибок. Решение этого вопроса заключается в разработке графической оболочки, которая представила бы возможность пользователю управлять численным экспериментом посредством оконного интерфейса. Такой подход обеспечил бы экономию времени при подготовке расчетного случая и позволил бы минимизировать вероятность появления ошибок.

Коллективы разработчиков, знакомые с отмеченными выше вопросами, предложили подходы к их решению, а именно создали свои версии программных оболочек для платформы OpenFOAM. Самые известные среди них — Salome [3], Helyx-OS [4], VisualCFD [5]. Эти программные средства получили широкое распространение. Однако специалисты, прежде всего отечественные, обнаружили в них недостатки, в частности, документация к данным продуктам не полностью описывает функциональные возможности и предоставляется на английском языке. Кроме того, техническая поддержка обеспечивается на коммерческой основе, а также в ряде случаев для работы с представленными графическими оболочками требуется приобретение лицензии. Таким образом, развитие функциональных возможностей OpenFOAM сохраняет актуальность.

Автором предложен собственный подход к решению задачи, в рамках которого была получена реализация в виде оригинальной графической оболочки. Ее первоначальная версия была представлена в 2016 г. в работе [6]. Впоследствии первоначальная версия была расширена путем создания и интеграции дополнительных программных модулей, отвечающих за этап препроцессинга. Результаты этих исследований изложены в работах [7—9].

В связи с необходимостью применения в численных экспериментах других утилит OpenFOAM, а также необходимостью работы в графической оболочке с новыми программами-решателями, автором предложено дальнейшее расширение возможностей базовой версии графической оболочки в части реализации доступа специалиста к программе-решателю buoyantSimpleFoam и утилите постпроцессинга postProcess.

Назначение решателя buoyantSimpleFoam и утилиты postProcess

Программа buoyantSimpleFoam представляет собой стационарный решатель для плавучего, турбулентного потока сжимаемых жидкостей, в том числе для излучения, вентиляции и теплопередачи. В качестве примера использования данного решателя можно привести расчетный случай, соответствующий задаче расчета передачи теплового потока через структуры [10, 11].

Численный эксперимент на основе решателя buoyantSimpleFoam предусматривает выполнение действий на следующих этапах:

1) подготовка структуры расчетного случая, в том числе файла-словаря sample директории system, который для моделируемой задачи определяет параметры выборки данных поля с возможностью указания различных схем интерполяции (файл sample необходим на этапе препроцессинга решения);

2) построение расчетной сетки (РС), например, с помощью одной из стандартных утилит платформы OpenFOAM (blockMesh, snappyHexMesh и т. д.);

3) запуск процесса численного моделирования задачи МСС посредством решателя buoyantSimpleFoam;

4) выполнение постпроцессинга с помощью утилиты postProcess.

Постпроцессинг обеспечивает специалисту возможность анализа результатов после завершения решения задачи МСС. На этапе постпроцессинга осуществляется обработка или выборка данных, записанных в процессе моделирования задачи, получение дополнительных свойств полей.

Если речь в постпроцессинге идет о выборке данных, то требуется подготовка файла-словаря sample, на основе которого осуществляется выборка полученных в ходе эксперимента данных для эффективного исследования получившейся численной модели. Именно sample содержит настройки, определяющие особенности выборки полей численной модели.

Параметры файла sample и возможные их значения приведены в таблице.

Особая роль отводится вспомогательному словарю sets, где специалист может указать тип метода выборки, название выборки, способ записи координат точек, а также другие параметры в зависимости от используемого метода выборки.

Пример заполненного файла-словаря sample для моделируемой задачи МСС buoyantCavity представлен на рис. 1.

```
FoamFile
{
    version         2.0;
    format          ascii;
    class           dictionary;
    location        "system";
    object          sample;
}

// * * * * * * * * * * * * * * * * * * *
* //

type sets;
libs            ("libsampling.so");

interpolationScheme cellPointFace;

setFormat raw;

sets
(
    y0.1
    {
        type            lineFace;
        axis            x;
        start           (-1 0.218 0);
        end             (1 0.218 0);
    }
    y0.2
    {
        type            lineFace;
        axis            x;
        start           (-1 0.436 0);
        end             (1 0.436 0);
    }
);

fields
(
    T
    U
);
```

Рис. 1. Пример файла-словаря sample

Параметры файла-словаря sample

Параметр	Описание	Значения
setFormat	Установить выходной формат	xmgr — данные в формате xmgr; jplot — данные в формате jplot; gnuplot — данные в формате gnuplot; raw — необработанные данные ASCII в столбцах
surfaceFormat	Формат вывода данных о поверхности	null — подавлять вывод; foamFile — отдельный файл точек, граней и значений; dx — скалярный или векторный формат; vtk — формат VTK ascii; aw — x y z-формат; obj — не содержит значений; stl — формат ascii stl, не содержит значений
interpolationScheme	Схема интерполяции данных	cell — использовать значение центра ячейки; константа по ячейкам (по умолчанию); cellPoint — использовать значения центра ячейки и вершины; cellPointFace — использовать центр ячейки, вершину и значения поверхности
fields	Список полей для выборки	Включает наборы sets и surfaces
sets	Местоположения в домене, в которых выполняется линейная выборка полей	uniform — равномерно распределенные точки на линии; face — столкновение с одной точкой на пересечении граней; midPoint — указание по одной точке на ячейку между двумя пересечениями граней; midPointAndFace — комбинации поверхности и midpoint; curve — точки, указанные на кривой, не обязательно на линии; cloud — облако определенных точек; Вариант оси: как записать координату точки. Выбор: • x / y / z: только координаты x / y / z; • xyz: три столбца (для raw); • distance: расстояние от начала линии
surfaces	Местоположения в пределах области, где проводится поверхностная выборка полей	plane — значения на плоскости, определяемой точкой, нормалью; patch — значения на патче

Постановка цели и задач разработки

Объектом настоящего исследования являются программа-решатель buoyantSimpleFoam и утилита postProcess, а также особенности их применения в рамках численного моделирования задач МСС. Автором проанализирована документация OpenFOAM [10, 11], по которой изучен алгоритм подготовки расчетного случая, изучены необходимые служебные файлы-словари и входящие в них параметры. Также выполнена постановка численного эксперимента на основе решателя buoyantSimpleFoam посредством традиционного подхода, когда структура расчетного случая формируется вручную, а также вручную задаются и параметры численной модели.

Цель работы заключается в замене традиционного подхода на использование программного модуля с графическим интерфейсом, который позволит все этапы работы с расчетным случаем выполнять посредством привычного оконного интерфейса, что обеспечит экономию времени специалиста. Также предполагается реализовать механизм валидации для предотвращения указания в файлах-словарях неверных параметров и механизм проверки комплектности директории расчетного случая.

Предполагается реализовать предложенный модуль в виде компонента базовой версии графической оболочки, т. е. расширить ее исходный код. Так как оболочка функционирует только в привязке к платформе OpenFOAM, настоящее исследование предполагает в том числе расширение исходного кода свободно-распространяемой платформы OpenFOAM. В рамках достижения поставленной цели автором намечен представленный далее перечень задач.

- Разработка программного кода, отвечающего за формирование графической составляющей модуля (интерфейса), в частности, форм редактирования параметров файлов-словарей расчетного случая, например, файла sample, а также элементов управления для запуска решателя buoyantSimpleFoam и утилиты postProcess.
- Подготовка алгоритмов для извлечения параметров, введенных через формы и для записи этих параметров в соответствующие файлы-словари расчетного случая.
- Перенос подготовленных алгоритмов на выбранный язык программирования, включая подключение к системе хранения данных.
- Разработка для форм редактирования файлов-словарей расчетного случая механизма валидации в целях обеспечения корректности вводимых параметров. А также разработка механизма проверки комплектности директории расчетного случая перед запуском решателя buoyantSimpleFoam и утилиты postProcess.

- Тестирование и отладка разработанного программного кода и его интеграция в исходный код базовой версии созданной автором графической оболочки.

Средства разработки

Для разработки программного модуля, как и оригинальной графической оболочки [6], автором определен необходимый стек технологий. Каждая позиция этого стека представляет собой свободно распространяемую технологию, не требующую приобретения лицензии.

- Программно-аппаратная часть модуля. Она отвечает за функционирование внутренней составляющей приложения. Это логика работы программы, т. е. программный код, отвечающий за выполнение приложением свих задач. Принято решение продолжить использование высокоуровневого языка Python 3 [12, 13], который по рейтингу TIOBE занимает первую позицию в списке наиболее популярных языков программирования [14].
- Клиентская сторона. Это интерфейс для связи пользователя с программно-аппаратной частью приложения. По сути через клиентскую часть специалист "отдает команды" программе. Возможности клиентской стороны реализованы посредством библиотеки PyQt — популярного инструмента создания пользовательских интерфейсов Python-приложений [15].
- СУБД. Встроенная в виде модуля sqlite3 в стандартную библиотеку Python реляционная СУБД SQLite позволяет сохранять передаваемые через экранные формы параметры в привычном табличном формате [16]. Это простая, при этом полнофункциональная система с минимальными настройками идеально подходит для настольных программных продуктов.
- ORM-библиотека. Благодаря ORM-подходу разработчик избавлен от необходимости взаимодействия с базой данных с помощью языка запросов SQL. Все операции по созданию таблиц, сохранению и получению данных из базы осуществляются посредством привычного Python-синтаксиса. Принято решение продолжить работу с ORM-библиотекой SQLAlchemy [17].

Структура и логика работы модуля

Связь компонентов графической оболочки приведена на рис. 2. На диаграмме представлены файлы с исходным кодом графической составляющей

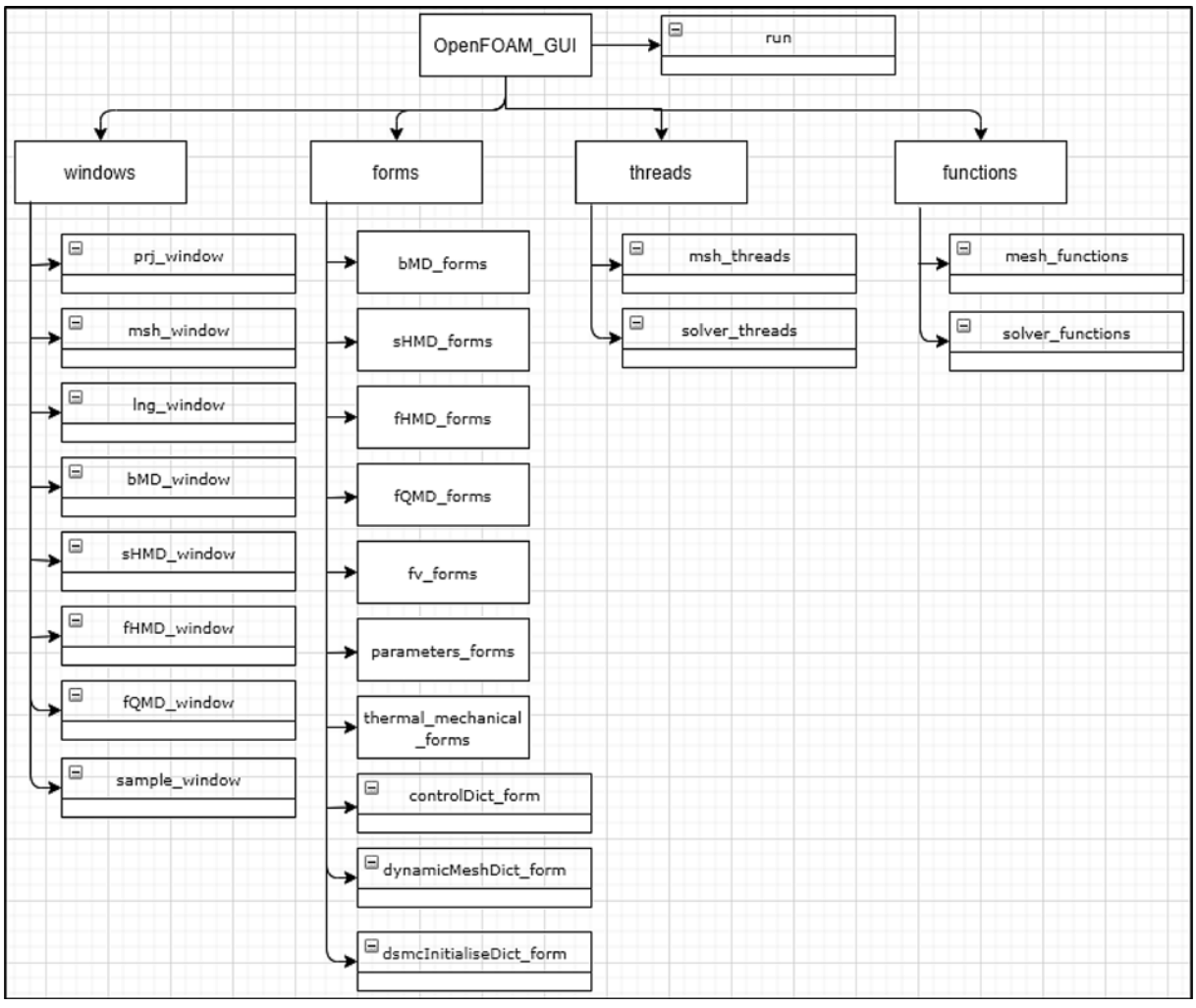


Рис. 2. Структура графической оболочки OpenFOAM_GUI

(экранные формы) приложения, файлы с программным кодом, обеспечивающим отображение интерфейса, и программным кодом, отвечающим за заполнение файлов-словарей расчетного случая и за валидацию вводимых через форму параметров. Структура приложения приведена с учетом интеграции в исходный код графической оболочки компонентов для работы с решателем buoyantSimpleFoam и утилитой postProcess.

Приложение включает следующие структурные блоки:

- директория windows, содержащая файлы-компоненты с исходным кодом, отвечающим за визуализацию экранных форм графической оболочки для указания параметров численной модели;
- директория forms, содержащая файлы с программным кодом основных блоков экранных форм;
- директория threads, содержащая программный код скриптов, обеспечивающих возможность параллельного выполнения задач, решаемых с помощью графической оболочки для OpenFOAM;
- директория functions, содержащая исходный код вспомогательных компонентов графической оболочки;
- файл run.py, обеспечивающий взаимосвязь всех компонентов приложения и визуализацию главного окна.

На рис. 3 представлена диаграмма, описывающая логику работы специалиста с решателем buoyantSimpleFoam и утилитой postProcess.

Представленный в работе программный модуль применяется на этапе решения задачи МСС и на этапе постпроцессинга. Разработчик дополняет директорию расчетного случая необходимыми файлами-словарями с параметрами численной модели. Далее выполняется запуск программы-решателя buoyantSimpleFoam и последующий постпроцессинг с помощью утилиты postProcess. После визуализации результатов численного эксперимента с помощью пакета ParaView [18] специалист может завершить эксперимент или выполнить корректировки и повторить необходимые этапы численного моделирования.

Результаты исследования

В рамках выполненного автором исследования исходный код базовой версии графической оболочки [6] расширен за счет интеграции программного модуля для работы с решателем buoyantSimpleFoam и утилитой postProcess. Программный модуль может применяться в различных областях промышленности, где требуется численное исследование для плавучего турбулентного потока сжимаемых жидкостей, в том числе излучения, вентиляции и теплопередачи.

Автором выполнена доработка графической составляющей базовой версии приложения путем

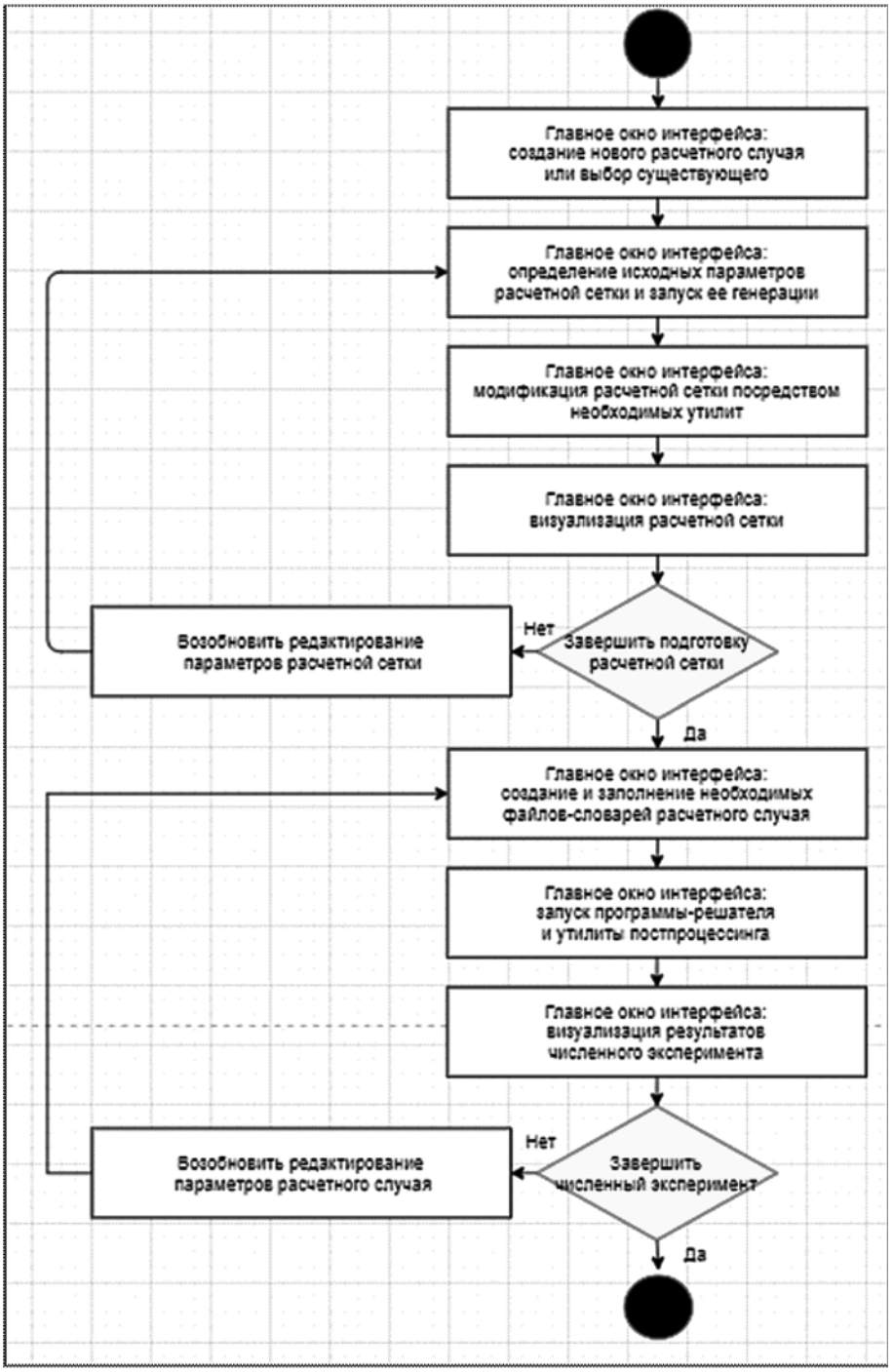


Рис. 3. Логика работы модуля

внедрения в интерфейс дополнительных элементов управления, а также доработка программных скриптов приложения и системы валидации параметров файлов-словарей расчетного случая. За счет программного модуля специалист имеет возможность посредством экранных форм работать с файлами-словарями расчетного случая, в частности, с файлом sample. Также в распоряжении специалиста есть графические элементы управления для запуска необходимой программы-решателя и утилиты постпроцессинга.

Возможности разработанного программного модуля протестированы на примере одной из фундаментальных задач МСС — buoyantCavity, которая соответствует задаче численного моделирования плавучего потока в каверне. Численное решение этой задачи осуществляется посредством решателя buoyantSimpleFoam. Постпроцессинг предусматривает выборку на основе файла-словаря sample полученных в ходе эксперимента данных для эффективного исследования численной модели.

На рис. 4—7 (см. третью и четвертую стороны обложки) представлены результаты численного моделирования указанной задачи МСС с отображением параметров изменения импульса, давления и температуры.

Заключение

Настоящая статья обобщает результаты исследования по расширению исходного кода графической оболочки [6] за счет разработки и интеграции программного модуля, обеспечивающего управление этапами численного эксперимента посредством оконного интерфейса. Речь идет об этапе решения задач МСС с помощью решателя buoyantSimpleFoam и этапе запуска постпроцессинга численного эксперимента с помощью утилиты postProcess.

Автором изучен механизм подготовки необходимых для работы программы-решателя buoyantSimpleFoam и утилиты postProcess служебных файлов-словарей, проанализированы определяемые в них параметры РС и моделируемой задачи МСС. По результатам этой работы модифицирована базовая версия графической оболочки платформы OpenFOAM [19], в которую интегрированы:

- экранные формы для редактирования параметров, определяемых в файлах-словарях расчетного случая при работе с решателем buoyantSimpleFoam;
- экранные формы для настройки параметров постпроцессинга численного решения, определяемых, например, через файл-словарь sample;
- алгоритм визуализации экранных форм для редактирования параметров моделируемой задачи МСС, а также алгоритм записи параметров, введенных через формы в соответствующие таблицы базы данных;
- механизм валидации параметров, определяемых через формы (ввод по маске, проверка типа данных параметров);
- механизм генерации нескольких версий файлов-словарей расчетного случая для обеспечения постановки численного эксперимента применительно к различным условиям среды.

Благодаря возможности работы с программой-решателем buoyantSimpleFoam и утилитой postProcess посредством графической оболочки пользователь может добиться экономии рабочего времени в процессе проведения численного эксперимента, а также избежать ошибок при подготовке расчетного случая и заполнении служебных файлов-словарей. Последнее гарантирует соответствие результатов полученной численной модели реальному объекту или процессу.

Список литературы

1. **OpenFOAM.** The open source CFD toolbox. URL: https://www.openfoam.com/
2. **ANSYS.** URL: https://www.ansys.com/
3. **Salome.** The Open Source integration Platform for Numerical Simulation. URL: https://www.salome-platform.org/
4. **Helyx-OS.** Open-Source GUI for OpenFOAM. URL: https://engys.com/products/helyx-os
5. **Visual-CFD.** URL: https://www.esi-group.com/products/computational-fluid-dynamics
6. **Читалов Д. И., Меркулов Е. С., Калашников С. Т.** Разработка графического интерфейса пользователя для программного комплекса OpenFOAM. Программная инженерия. 2016. Т. 7, № 12. С. 568—574. DOI: 10.17587/prin.7.568-574.
7. **Читалов Д. И., Калашников С. Т.** Разработка модуля для реализации зеркального отображения расчетных сеток вокруг заданной плоскости в графическом интерфейсе пользователя платформы openfoam // Программная инженерия. 2019. Т. 10, № 7-8. С. 297-304. DOI: 10.17587/prin.10.297-304.
8. **Читалов Д. И.** О разработке модуля для реализации движения и топологического изменения расчетных сеток и его интеграции в графическую оболочку для платформы openfoam // Программная инженерия. 2020. Т. 11, № 2. С. 108—114. DOI: 10.17587/prin.11.108-114.
9. **Читалов Д. И.** Разработка модуля для измельчения ячеек расчетных сеток в нескольких направлениях и его интеграция в gui для программной среды openfoam // Системы и средства информатики. 2020. Т. 30, № 3. С. 133—144. DOI: 10.14357/08696527200312.
10. **OpenFOAM.** User Guide. URL: http://foam.sourceforge.net/docs/Guides-a4/OpenFOAMUserGuide-A4.pdf
11. **OpenFOAM.** Tutorial Guide. URL: http://openfoam.com/documentation/tutorial-guide/index.php
12. **Python 3.7** documentation. URL: https://docs.python.org/3.7/
13. **Прохоренок Н. А.** Python 3 и PyQt. Разработка приложений. СПб.: БХВ-Петербург, 2012. 704 с.
14. **TIOBE** Index. URL: http://www.tiobe.com/tiobe-index/
15. **PyQt5** Reference Guide. URL: http://pyqt.sourceforge.net/Docs/PyQt5/
16. **SQLite.** URL: https://www.sqlite.org/index.html
17. **SQLAlchemy.** URL: https://www.sqlalchemy.org/
18. **ParaView.** URL: https://www.paraview.org/
19. **OpenFOAM_GUI.** URL: http://github.com/DmitryChitalov/OpenFOAM_GUI

On the Development of a Module for Working with the buoyantSimpleFoam Solver and the postProcess Utility of the OpenFOAM Platform

D. I. Chitalov, cdi9@yandex.ru, South Urals Federal Research Centre of Mineralogy and Geoecology of the UB RAS, Chelyabinsk Region, Miass, Ilmen reserve, 456317, Russian Federation

Corresponding author:

Chitalov Dmitry I., Junior Researcher, South Urals Federal Research Centre of Mineralogy and Geoecology of the UB RAS, Chelyabinsk Region, Miass, Ilmen reserve, 456317, Russian Federation
E-mail: cdi9@yandex.ru

Received on December 19, 2021
Accepted on December 27, 2021

The paper summarizes the results of research on the development of a software module that expands the source code of the OpenFOAM platform in terms of providing a specialist with access to new possibilities of a numerical experiment in relation to problems of continuum mechanics. The module provides the user with graphical and software tools for working with the buoyantSimpleFoam solver and postProcess utility. This work contains a description of the shortcomings of existing software solutions – analogs, the urgency of the problem under study is formulated. The author has set goals and defined the tasks necessary to achieve them. A description of the operation of the postProcess utility and the buoyantSimpleFoam solver is given, as well as the structure and parameters of the corresponding dictionary files of the design case. The author presents a set of technologies necessary for the implementation of the capabilities of a software module, typing, debugging and testing its program code. The performance of the developed software solution has been tested on the example of one of the fundamental problems of continuum mechanics, and the results of testing are presented. Based on the results of the study, the final conclusions are presented, as well as information on the scientific novelty and potential practical significance of the study.

Keywords: *numerical simulation, continuum mechanics, graphical user interface, OpenFOAM, Python, open source software, postProcess utility, buoyantSimpleFoam solver, PyQt*

For citation:

Chitalov D. I. On the Development of a Module for Working with the buoyantSimpleFoam Solver and the postProcess Utility of the OpenFOAM Platform, *Programmnaya Ingeneria,* 2022, vol. 13, no. 2, pp. 81—87.

DOI: 10.17587/prin.13.81-87

References

1. **OpenFOAM.** The open source CFD toolbox, available at: https://www.openfoam.com/
2. **ANSYS,** available at: https://www.ansys.com/
3. **Salome.** The Open Source integration Platform for Numerical Simulation, available at: https://www.salome-platform.org/
4. **Helyx-OS.** Open-Source GUI for OpenFOAM, available at: https://engys.com/products/helyx-os
5. **Visual-CFD,** available at: https://www.esi-group.com/products/computational-fluid-dynamics
6. **Chitalov D. I., Merkulov Ye. S., Kalashnikov S. T.** Development of a Graphical User Interface for the OpenFOAM Toolbox, *Programmnaya Ingeneria,* 2016, vol. 7, no. 12, pp. 568—574. DOI: 10.17587/prin.7.568-574 (in Russian).
7. **Chitalov D. I., Kalashnikov S. T.** Development of a Module for Implementing the Mirroring of Computational Meshes around a Given Plane in the Graphical User Interface of the OpenFOAM Platform, *Programmnaya Ingeneria,* 2019, vol. 10, no. 7—8, pp. 297—304. DOI: 10.17587/prin.10.297-304 (in Russian).
8. **Chitalov D. I.** On the Development of a Module for Implementing Motion and Topological Changes in Computational Meshes and its Integration into the Graphical Shell for the OpenFOAM Platform, *Programmnaya Ingeneria,* 2020, vol. 11, no. 2, pp. 108—114 DOI: 10.17587/prin.11.108-114 (in Russian).
9. **Chitalov D. I.** Development of a module for grinding meshes cells in several directions and its integration into the GUI for the openFoam software environment, *Systems and Means of Informatics*, 2020, vol. 30, no. 3, pp. 133—144. DOI: 10.14357/08696527200312 (in Russian).
10. **OpenFOAM.** User Guide, available at:http://foam.sourceforge.net/docs/Guides-a4/OpenFOAMUserGuide-A4.pdf
11. **OpenFOAM.** Tutorial Guide, available at: http://openfoam.com/documentation/tutorial-guide/index.php
12. **Python 3.7** documentation, available at: https://docs.python.org/3.7/
13. **Prohorenok N. A.** *Python 3 i PyQt. Razrabotka prilozhenij.* Saint Petersburg, BHV-Peterburg, 2012. 704 p. (in Russian).
14. **TIOBE** Index, available at: http://www.tiobe.com/tiobe-index/
15. **PyQt5** Reference Guide, available at:http://pyqt.sourceforge.net/Docs/PyQt5/
16. **SQLite,** available at: https://www.sqlite.org/index.html
17. **SQLAlchemy**, available at: https://www.sqlalchemy.org/
18. **ParaView,** available at: https://www.paraview.org/
19. **OpenFOAM_GUI**, available at: http://github.com/Dmitry-Chitalov/OpenFOAM_GUI

УДК 004.514.2 DOI: 10.17587/prin.13.88-93

А. А. Коротышева, студент магистратуры, ania.korotishewa@yandex.ru,
С. Н. Жуков, доц. канд. физ.-мат. наук, jsn@rf.unn.ru,
Нижегородский государственный университет им. Н. И. Лобачевского

Реализация алгоритмов дополненной реальности в навигации автомобильного транспорта с использованием открытых сервисов

Предложен алгоритм визуализации навигационной информации для управления автомобилем в виде объектов дополненной реальности (augmented reality, AR). Работа алгоритма реализуется с использованием открытых сервисов OpenStreetMap и OSRM. На основе полученных от сервисов данных формируется карта, выполняется построение и оптимизация маршрута. Предоставление водителю навигационной информации в виде объектов AR на лобовом стекле автомобиля, интегрированных в дорожную обстановку, обеспечивает ее контекстное восприятие и повышает удобство использования навигационной системы.

Ключевые слова: *информационные технологии, дополненная реальность, AR, алгоритмы визуализации, геоинформационные данные, геоинформационные системы, GIS, построение маршрута, HUD*

Введение

В современном мире для построения маршрутов движения водители автомобильного транспорта вместо обычных бумажных карт используют навигаторы, предоставляющие информацию о местоположении транспортного средства, основанную на данных геоинформационных систем (ГИС) [1]. При этом частое переключение внимания водителя на навигатор отвлекает его от дорожной обстановки и порождает проблему безопасности движения.

Одним из перспективных направлений применения информационных систем в этой сфере является применение технологии дополненной реальности (*augmented reality*, AR) [2] в виде проекции информации на лобовое стекло автомобиля либо на полупрозрачный экран перед лобовым стеклом. Такой способ вывода информации называется технологией HUD (*head-up display*, проекционный дисплей) [3]. Навигационная информация, предоставляемая непосредственно на уровне направления взгляда на дорогу, легко воспринимается водителем, не отвлекая его внимания от дорожной ситуации.

Таким образом, разработка и применение алгоритмов дополненной реальности в системах навигации автомобильного транспорта является актуальной задачей, позволяющей решать проблему повышения безопасности дорожного движения.

В настоящее время в мире широко проводятся исследования по совершенствованию технологии HUD и ее внедрению в системы автомобильной навигации, однако существующие навигационные системы с HUD обладают высокой стоимостью и доступны только для премиального сегмента автомобильной промышленности.

В целях создания недорогих вариантов эффективно работающих навигационных систем в данной работе предложен программно реализованный алгоритм визуализации объектов с динамическими параметрами, функционально зависящими от геоинформационных данных, использующий открытые сервисы формирования карты и построения маршрута.

Применение дополненной реальности в автомобиле

Дополненная реальность представляет собой технологию интеграции виртуальных объектов в реальную обстановку. В применении к автомобильной навигации этот процесс заключается во встраивании (наложении) виртуальных объектов, например, стрелок — указателей направления с прозрачным окружением в дорожную обстановку за лобовым стеклом путем их проецирования непосредственно на стекло (рис. 1).

Отличительными особенностями предложенного алгоритма для навигационной системы автомобиля являются:

— формирование карты и построения маршрута с использованием открытых сервисов OSRM (Open Street Routing Machine) и OpenStreetMap [4, 5];

— система геопространственных слоев, позволяющая объединять и рекомбинировать данные из различных веб-сервисов в единую карту;

— обработка в режиме реального времени геоинформационных данных, построение и оптимизация

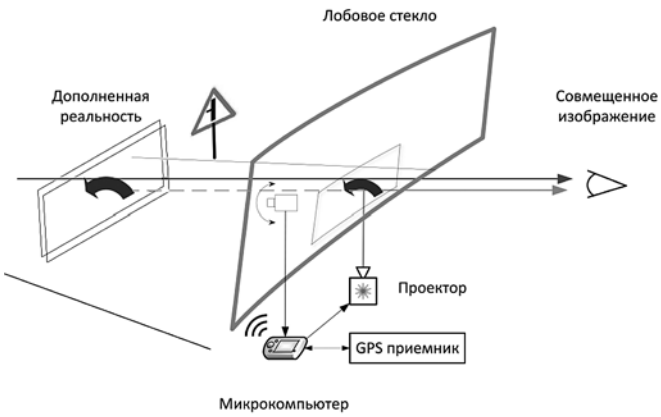


Рис. 1. Навигационная система автомобиля с HUD

маршрута с последующей визуализацией навигационной информации;

— интерактивный интерфейс представления водителю навигационной информации на основе технологии дополненной реальности в виде проекции на лобовое стекло автомобиля, что значительно повышает уровень информативности и комфорта использования навигационной системы [6].

Алгоритм построения маршрута

Основная задача ГИС — построение оптимального маршрута движения транспортного средства (по нескольким точкам). Ключевыми критериями оценки задачи могут выступать минимизация расстояния между точками маршрута, минимизация общего времени прохождения маршрута или их комбинация.

Дорожную сеть, состоящую из местных дорог, городских улиц и магистралей, при поиске маршрута можно представить в виде графа, где ребра — это дороги, а вершины — перекрестки, промежуточные и конечные точки.

Обычно в качестве алгоритмов маршрутизации применяют известные алгоритмы Дейкстры или A* [7, 8]. Логическое условие выполнения алгоритма Дейкстры определяется выражением

$$d[v] = \min_{p:u[p]=\text{false}} d[p],$$

где $d[v]$ — текущая длина кратчайшего пути из s в v для вершины v; $u[p]$ — булевский массив, в котором сначала все узлы не помечены (значение элементов — false), а на очередной итерации выбирается вершина v с самым маленьким значением $d[v]$ среди тех, которые еще не помечены. Выбранная вершина отмечается помеченной, после чего на текущей итерации из вершины v выполняются релаксации — просматриваются все ребра (v, t_0), которые исходят из вершины, причем для каждого узла алгоритм пробует улучшить значение $d[t_0]$:

$$d[t_0] = \min\left(d[t_0], d[v] + len\right),$$

где len — длина текущего ребра. В конечном итоге после n итераций все узлы станут помеченными, а алгоритм завершит работу.

Порядок обхода вершин в алгоритме A* определяется функцией

$$f(n) = g(n) + h(n),$$

где $f(n)$ — минимальная стоимость перехода в соседний узел; $g(n)$ — стоимость пути от начальной вершины до любой другой; $h(n)$ — эвристическое приближение стоимости пути от узла n до конечного узла.

В данной работе для построения маршрутов выбран маршрутный сервис OSRM, который просчитывает и возвращает оптимальные маршруты в условиях переданных ему координат. Сервис OSRM поставляется с открытым исходным кодом, использующим более производительный алгоритм маршрутизации Contraction Hierarchies [9].

Алгоритм Contraction Hierarchies позволяет ускорить вычисление кратчайших путей графа в среднем в 47 раз в сравнении с алгоритмом Дейкстры [10], требует меньше памяти при обработке и состоит из двух стадий.

Первая стадия — это стадия препроцессинга: вершины сортируются в каком-либо жестком порядке, далее проводится предварительная обработка графа, представляющая процесс сжатия узлов графа по одному за проход. Чтобы выполнить сжатие, вычисляется кратчайший путь между ближайшими соседями узла, на них размещаются метки и узел помечается как обработанный. В дальнейшем эта информация используется при построении маршрута или его перестроении, исходя из новых данных, что позволяет ускорить выполнение отдельных обращений (запросов) к графу и более эффективно использовать память компьютера.

Появление дополнительной информации, например, об аварии или пробке на участке дороги, которые увеличивают время проезда по данному участку, меняет вес ребер и приоритеты меток, что запускает процесс повторной обработки этого узла.

Вторая стадия — стадия запроса: с начальной и конечной точек маршрута запускается двусторонний алгоритм Дейкстры с условием, что волны идут только вверх по иерархии (когда они встретятся — путь найден), далее последовательно восстанавливаются сокращенные ребра (рис. 2, см. четвертую сторону обложки).

Штриховыми линиями на рис. 2 показаны ребра, которые сокращаются при первой волне прохождения по графу. Сплошными линиями — ребра, которые сокращаются при второй волне прохождения по графу. Зеленые стрелки показывают процесс сжатия — замены двух последовательных ребер одним. Красным цветом выделен итоговый маршрут.

В качестве источника геоинформационных данных при реализации алгоритма визуализации объектов дополненной реальности был выбран OpenStreetMap — проект с открытым исходным кодом, которой является аналогом плиточных картографических сервисов, используемых такими систе-

мами, как OpenLayers. OpenStreetMap содержит глобальные векторные данные на уровне улиц и других пространственных объектов, а также предоставляет возможность обращения к точкам маршрута по специфической нумерации российских адресов. Данная картографическая база бесплатна и может быть использована для коммерческих целей, в отличие от Яндекс- и Гугл-карт.

Структуры и форматы геопространственных данных

Геопространственные данные в настоящее время представляются в десятке форматов файлов и структур баз данных и продолжают развиваться и расти, включая новые типы данных и стандарты. При этом данные могут быть как в растровом формате, так и в векторном. В предложенном алгоритме используются векторные данные, что связано с их компактностью, большой точностью и легкостью управления. Обеспечивается поддержка наиболее распространенных стандартов, например:

— shape-файл (файл фигур) — открытая спецификация, разработанная ESRI "Институт исследования систем окружающей среды";

— Coverage (пространственное покрытие) компании ESRI для хранения географических объектов в виде точек, дуг и многоугольников со связанными с ними таблицами атрибутов;

— Simple Features (простые геообъекты) — стандарт OpenGIS для хранения географических данных (точек, линий и многоугольников) вместе со связанными с ними атрибутами.

При описании отдельных сегментов геоданных обычно используются несколько распространенных микроформатов, таких как:

- WKT (Well-known Text) — текстовый формат для представления одного географического объекта, такого как многоугольник или ломаная;
- GeoJSON — открытый формат для кодирования географических структур данных, который основан на формате для обмена данными JSON;
- GML (Geography Markup Language), или язык географической разметки, — открытый стандарт на основе XML для обмена данными ГИС.

В рабочей программе для обмена данными с ГИС используется библиотека json и формат GeoJSON, отличающийся простотой интерпретации, описание объектов в нем определяется наборами типовых свойств вида "ключ/значение".

Существует несколько перечисленных далее подходов рендеринга в зависимости от формата геоданных, которые имеют различную поддержку среди готовых библиотек (сервисов).

◇ Локальная трансформация OSM XML (.osm) файлов формата shp в SVG, BMP, PNG. Такой подход используется, например, настольной программой рендеринга Maperitive, работающей под операционными системами Windows, Linux, Mac.

◇ Загрузка слоев, однократный рендеринг. Поддерживается проектами Osmarender, Kosmos, Maposmatic, CartoType и др.

◇ Генерация набора статичных тайлов. Поддерживается Mapnik (требуются PostgreSQL и различные библиотеки C++), Pyrender и др.

В описываемом алгоритме используется концепция мэшапа [11], представляющая систему геопространственных слоев, которая позволяет пользователю объединять и рекомбинировать данные из различных веб-сервисов в единую карту.

Алгоритм визуализации дополненной реальности

При построении маршрута определяются начальная и конечная координаты маршрута (рис. 3), их значения обрабатываются и передаются в маршрутный сервис OSRM. Для определения координат используется приемник GPS спутниковой системы навигации [12].

Расчет расстояния между точками маршрута проводится по формуле гаверсинусов, чтобы избежать проблем с небольшими расстояниями:

$$\Delta\sigma = 2\arcsin\left\{\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right\},$$

где ϕ_1, λ_1; ϕ_2, λ_2 — широта и долгота двух точек в радианах; $\Delta\lambda$ — разница координат по долготе; $\Delta\sigma$ — угловая разница.

Начальный азимут от начальной точки к конечной точке вычисляется по формуле

$$\theta = \operatorname{atan}2\left(\sin\Delta\lambda\cos\phi_2,\ \cos\phi_1\sin\phi_2 - \sin\phi_1\cos\phi_2\cos\Delta\lambda\right).$$

Скаченные с серверов OpenStreetMap тайлы (фрагменты карты, характеризующиеся тремя переменными: x, y — координаты верхнего левого угла фрагмента; z — размер масштабирования) объединяются в карту, затем полученные координаты переводятся в систему координат карты и визуализируются посредством 2D-графики.

Поступающий от видеокамеры видеопоток подвергается обработке. Проводится последовательный захват видеокадров и их декодирование. Затем полученные данные поступают в функцию отрисовки объекта, в которой проводится обработка и анализ каждого кадра. Вычисляется маска объекта, его растяжение и поворот, что позволяет однозначно задать положение объекта в пространстве. Затем объект выводится на экран с помощью графической библиотеки.

Для наложения объекта на кадр установим маску на растровое изображение, т. е. объединим маску с альфа-каналом растрового изображения. Тогда значение пикселя 1 на маске означает, что пиксель пиксельного изображения остается неизменным; значение 0 означает, что пиксель прозрачен. При этом маска должна иметь тот же размер, что и это пиксельное изображение. Таким образом, реализуется операция альфа-смешивания (*alpha blending*) — наложение изображений друг на друга в целях создания эффекта частичной прозрачности или вычисление взвешенной суммы двух массивов:

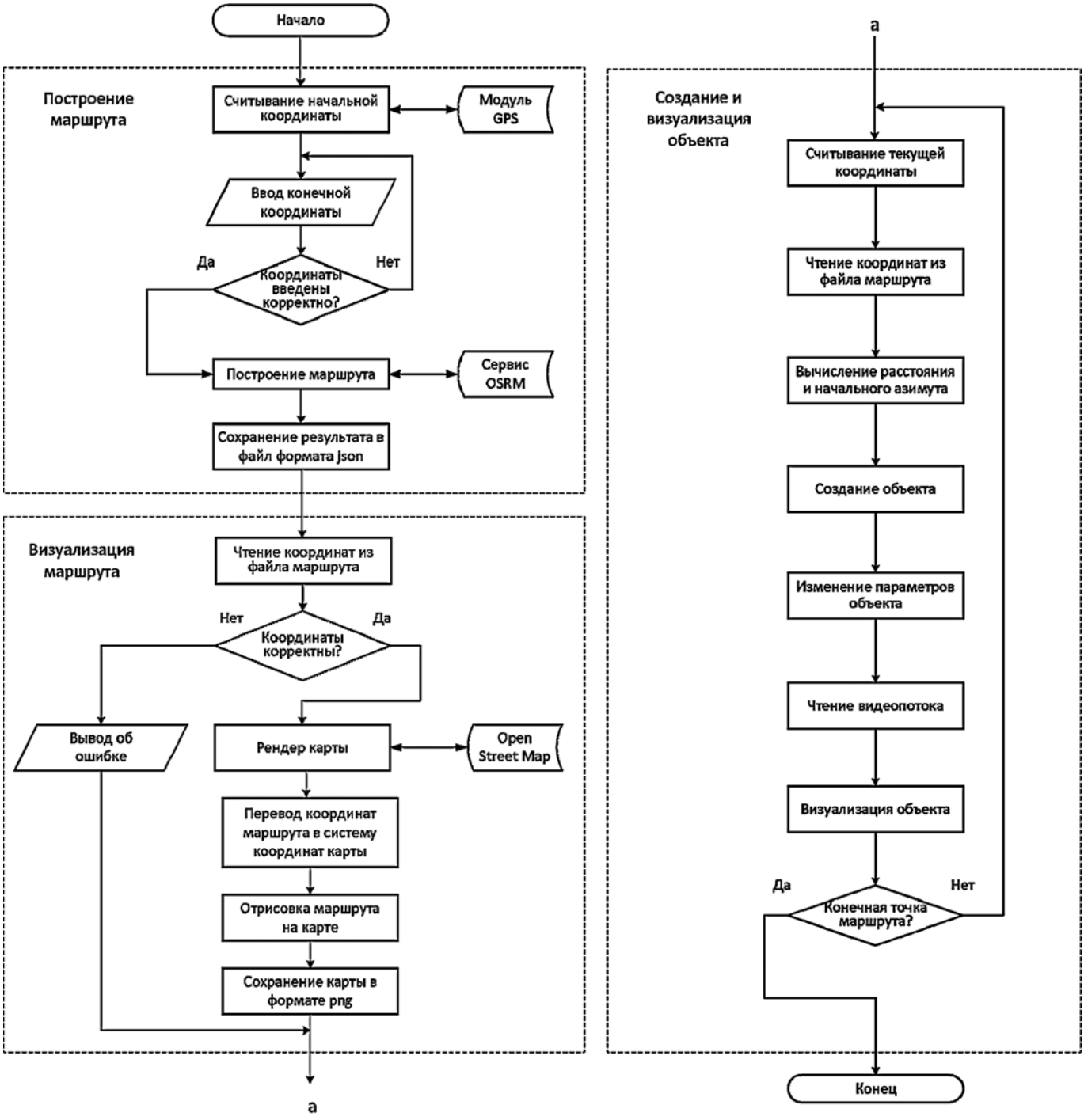


Рис. 3. Алгоритм визуализации объектов дополненной реальности

$$dst = src_1 * alpha + src_2 * beta + gamma,$$

где src_1 — первый исходный массив; *alpha* — вес элементов первого массива; src_2 — второй исходный массив; *beta* — вес элементов второго массива; *gamma* — добавочное значение к каждой сумме; *dst* — массив для сохранения результата.

Рассмотренный алгоритм визуализации объектов дополненной реальности запрограммирован в коде, с использованием которого реализуется навигация автомобиля. При задании точек маршрута сначала в соответствии с алгоритмом строится оптимальный граф. При изменении координат маршрута граф перестраивается. Пример части смоделированного графа представлен на рис. 4, где обозначение вершин графа соответствует координатам точек маршрута.

Одновременно с формированием карты и построением графа на экран в режиме реального времени

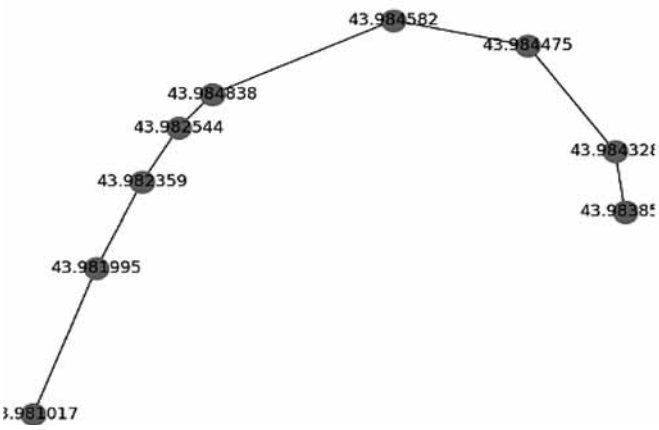


Рис. 4. Результаты моделирования — граф маршрута

выводится видеопоток с изменяющимся изображением дороги, полученный от видеокамеры, а также дополнительный слой с визуализацией направления движения по заданному маршруту в виде подсказок — стрелок-указателей (рис. 5, см. четвертую сторону обложки).

Заключение

Создан и опробован алгоритм визуализации объектов с динамическими параметрами, функционально зависящими от геоинформационных данных, с использованием открытых сервисов OSRM и OpenStreetMap. Интерактивный интерфейс программы визуализации дополненной реальности с проекцией на лобовое стекло автомобиля обладает интегрированным эффектом от совмещения преимуществ навигационных систем и сервисов геоинформационных данных. Предложенный алгоритм может быть использован в программном обеспечении системы навигационного оснащения автомобиля для повышения удобства ее использования и обеспечения безопасности движения.

Список литературы

1. **Капралов Е., Кошкарев А., Тикунов В.** и др. Геоинформатика. М.: Academia, 2010. 400 с.

2. **Смолин А. А., Жданов Д. Д., Потемин И. С., Меженин А. В., Богатырев В. А.** Системы виртуальной, дополненной и смешанной реальности. Уч. пос. Санкт-Петербург: Университет ИТМО, 2018. 59 с.

3. **Charissis V., Papanastasiou S.** Human—machine collaboration through vehicle head up display interface // Cogn. Tech. Work. 2010. Vol. 12. P. 41—50.

4. **OSRM** API Documentation URL: http://project-osrm.org/docs/v5.22.0/api/#general-options

5. **OpenStreetMap** — wiki-карта мира. URL: https://www.openstreetmap.org/

6. **Горячкин Б. С., Гаранов К. В., Бгатцев А. В.** Повышение эргономичности транспортных средств путем внедрения проекционных дисплеев // E-Scio. 2020. № 6 (45). С. 104—118.

7. **Dijkstra E. W.** A Note on Two Problems in Connexion with Graphs // Numerische Mathematik. 1959. Vol. 1. P. 269—271.

8. **Hart P. E., Nilsson N. J., Raphael B.** A Formal Basis for the Heuristic Determination of Minimum Cost Paths // IEEE Transactions on Systems Science and Cybernetics SSC4. 1968. Vol. 2. P. 100—107.

9. **Geisberger R., Sanders P., Schultes D., Delling D.** Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks // Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08). 5038 of Lecture Notes in Computer Science. 2008. P. 319—333.

10. **Ураков А. Р., Тимеряев Т. В.** Алгоритм поиска кратчайших путей для разреженных графов большой размерности // Прикладная дискретная математика. 2013. № 1 (19). С. 84—92.

11. **Yee R.** Pro Web 2.0 Mashups: Remixing data and web services. New York: Apress, 2008. 603 p.

12. **u-blox** 5 NMEA, UBX Protocol Specification. URL: https://www.sparkfun.com/datasheets/GPS/Modules/u-blox5_Protocol_Specifications(GPS.G5-X-07036).pdf

Implementation of Augmented Reality Algorithms in Road Transport Navigation using Open Services

A. A. Korotysheva, ania.korotishewa@yandex.ru, **S. N. Zhukov**, jsn@rf.unn.ru, National Research Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, 603950, Russian Federation

Corresponding author:

Korotysheva Anna A., Master's Degree Student, National Research Lobachevsky State University of Nizhny Novgorod, 603950, Russian Federation
E-mail: ania.korotishewa@yandex.ru

Received on September 24, 2021
Accepted on November 17, 2021

In the modern world, vehicle drivers use geographic information systems (GIS) — based navigators instead of the conventional paper maps to build traffic routes. One of the promising branches in this area is the use of augmented reality (AR) technology and head-up display (HUD) technology in the form of information projection onto the windshield of a car. The development and implementation of augmented reality algorithms together with the implementation of HUD technology is an urgent task to solve the problem of improving road safety. The objective of the study is to

optimize the algorithm for constructing traffic routes for a car using geographic information services and augmented reality technology with projection of information onto the windshield of a car. In the present paper, an algorithm for visualizing objects with dynamic parameters functionally dependent on geographic information data using OpenStreetMap and OSRM (Open Street Routing Machine) services is proposed and implemented as a software. Features of route optimization in OSRM based on efficient Contraction Hierarchies algorithm are shown, which has a number of advantages compared to commonly used Dijkstra and A route construction algorithms — higher speed, less memory requirements, a relatively fast preprocessing. The simulation results are provided. Possibilities and prospects of using the proposed algorithm in the software of the vehicle navigation system are presented. The developed algorithm using geographic information services and augmented reality technology with the projection of information onto the windshield of a car makes it possible to optimize the route construction process. The interactive interface of the augmented reality visualization program with the projection of route indicators and other important information onto the windshield of a car has the integrated effect of combining the advantages of AR, HUD technologies and geographic information services. The use of the interface in the vehicle navigation system will help to improve traffic safety.*

Keywords: *information technology, object visualization, augmented reality, visualization algorithms, geographic information data, geographic information system, graphic primitives*

For citation:

Korotysheva A. A., Zhukov S. N. Implementation of Augmented Reality Algorithms in Road Transport Navigation Using Open Services, *Programmnaya Ingeneria*, 2022, vol. 13, no. 2, pp. 88—93.

DOI: 10.17587/prin.13.88-93

References

1. **Kapralov E. G., Koshkarev A. V., Tikunov V. S.** et al. *Geoinformatics*, Moscow, Academiya, 2010, 400 p. (in Russian).

2. **Smolin A. A., Zhdanov D. D., Potemin I. S., Mezhenin A. V., Bogatyrev V. A.** *Virtual, augmented, and mixed reality systems*, Saint-Petersburg, University ITMO, 2018, 59 p. (in Russian).

3. **Charissis V., Papanastasiou S.** Human—machine collaboration through vehicle head up display interface, *Cognition Technology and Work*, 2010, vol. 12, pp. 41—50.

4. **OSRM** API Documentation. September 14, 2021, available at: http://project-osrm.org/docs/v5.22.0/api/#general-options

5. **OpenStreetMap** — site of world map wiki. September 14, 2021, available at: https://www.openstreetmap.org (in Russian).

6. **Goriachkin B. S., Garanov K. V., Bgattsev A. V.** Improving the ergonomics of vehicles by introducing projection displays, *E-Scio*, 2020, no. 6 (45), pp. 104—118 (in Russian).

7. **Dijkstra E. W**. A Note on Two Problems in Connexion with Graphs, *Numerische Mathematik*, 1959, vol. 1, pp. 269—271.

8. **Hart P. E., Nilsson N. J., Raphael B.** A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions on Systems Science and Cybernetics SSC4*, 1968, vol. 2, pp. 100 — 107.

9. **Geisberger R., Sanders P., Schultes D., Delling D.** Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks, *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)*, 5038 of Lecture Notes in Computer Science, 2008, pp. 319—333.

10. **Urakov A. R., Timeryaev T. V.** All-Pairs Shortest Paths Algorithm for Highdimensional Sparse Graphs, *Prikladnaya diskretnaya matematika*, 2013, no. 1 (19), pp. 84—92 (in Russian).

11. **Yee R.** *Pro Web 2.0 Mashups: Remixing data and web services,* New York, Apress, 2008, 603 p.

12. **u-blox** 5 NMEA, UBX Protocol Specification. September 14, 2021, available at: https://www.sparkfun.com/datasheets/GPS/Modules/u-blox5_Protocol_Specifications (GPS.G5-X-07036).pdf

УДК 519.684.6 DOI: 10.17587/prin.13.94-101

Н. С. Капралов, магистр[1], мл. инженер-программист[2], nskaprl@gmail.com,
А. Ю. Морозов, канд. физ.-мат. наук, ст. препод.[1], науч. сотр.[3], morozov@infway.ru,
С. П. Никулин, канд. физ.-мат. наук, доц.[1], sergeynp@yandex.ru
[1] Федеральное государственное бюджетное образовательное учреждение высшего образования "Московский авиационный институт (национальный исследовательский университет)" (МАИ),
[2] ООО "Технологический центр Дойче Банка", Москва
[3] Федеральный исследовательский центр "Информатика и управление" Российской академии наук (ФИЦ ИУ РАН), Москва

Параллельная аппроксимация многомерных тензоров с использованием графических процессоров

При решении многих исследовательских задач прикладного характера возникает необходимость работать с многомерными массивами (тензорами). На практике используется эффективное и компактное представление данных объектов в виде так называемых тензорных поездов. Рассматривается параллельная реализация алгоритма TT-cross, который позволяет получить разложение многомерного массива в тензорный поезд, с использованием графического процессора архитектуры CUDA. Представлены основные аспекты и особенности выполнения параллельной реализации алгоритма. На ряде примеров проведены апробации полученной параллельной версии алгоритма. Продемонстрировано существенное сокращение вычислительного времени по сравнению с аналогичной последовательной реализацией алгоритма, что свидетельствует об эффективности предлагаемых подходов к распараллеливанию.

Ключевые слова: *распараллеливание, тензорный поезд, тензорные разложения, большие размерности, проклятие размерности, многомерные массивы, крестовая аппроксимация, TT-cross, maxvol, малоранговая аппроксимация, CUDA, GPU, Nvidia*

Введение

В настоящее время существует много задач, для решения которых требуются высокопроизводительные ресурсы. Рассмотрим в качестве примера моделирование поведения объекта, зависящего от большого числа параметров, значения которых нельзя точно указать, но можно привести интервалы, в которых они находятся. Для решения таких задач был разработан [1], теоретически обоснован [1, 2], апробирован на прикладных задачах [3] и программно реализован [4, 5] алгоритм адаптивной интерполяции. В процессе работы алгоритма над множеством, образованным интервальными параметрами задачи, строится интерполяционная сетка, в которой число узлов экспоненциально зависит от числа параметров. В результате алгоритм имеет экспоненциальную сложность относительно числа интервальных неопределенностей как по вычислительным затратам, так и по необходимому объему памяти [3—5]. Также отметим, что зачастую моделируемый объект может изначально подразумевать использование некоторой пространственной сетки, что приводит к дополнительному увеличению общей вычислительной сложности.

Решение подобных задач, подразумевающих работу с многомерными структурами, требует очень больших вычислительных ресурсов. Одной из ключевых подзадач, которую можно выделить, является подзадача эффективного хранения и взаимодействия с многомерными массивами — тензорами (данная терминология используется в соответствии с работами [6, 7]). В результате такого подхода к решению рассматриваемой подзадачи хотелось бы, имея некоторое подмножество элементов с меньшей размерностью, уметь вычислять остальные значения многомерного массива. Если рассматривать тензор как функцию многих переменных, то необходимо провести разделение переменных. С помощью произведения функций одной переменной это получится крайне неточно. Однако если представить исходную функцию с помощью суммы таких произведений, то это уже позволит получить в некоторой мере желаемый результат и представление будет иметь $O(dnr)$ элементов, где d — размерность; r — ранг тензора (в данном случае многомерного массива); n — число значений параметров. Представление тензора в таком виде с минимальным r носит название канонического разложения [8].

К сожалению, нет методов, гарантирующих нахождение такого разложения.

Рассмотрим формат, называемый тензорным поездом (TT-формат) [6]. TT-формат или, по-другому, TT-разложение представляется в виде набора $(d-2)$ трехмерных тензоров и двух матриц. Вычисление любого элемента исходного тензора сводится к перемножению вектора-строки на цепочку матриц и на вектор-столбец. К преимуществам данного формата относятся: возможность получения искомого разложения без вычисления всех элементов исходного тензора; доступность всех арифметических (и не только) операций над тензорами в рамках этого представления.

Построение TT-разложения сводится к обычным матричным разложениям. Выполняется переход от d измерений к двум с помощью группировки индексов. Для полученной матрицы строится сингулярное разложение (SVD-разложение) [9]. Строки и столбцы, соответствующие несущественным сингулярным числам, отбрасывают. Урезанные таким способом матрицы разложения превращаются обратно в тензоры меньшей размерности, для которых выполняются все те же самые действия (тензоры превращаются в матрицы, строится SVD-разложение и т. д). В результате получается набор из трехмерных тензоров. Описанный выше подход лежит в основе алгоритма TT-SVD. Идея алгоритма TT-cross [7], позволяющего построить TT-разложение без вычисления всех элементов тензора, заключается в замене SVD-разложения на разложение с меньшей алгоритмической сложностью и не требующего полного знания всех элементов тензора. Здесь применяются подходы [10—13], позволяющие уменьшить вычислительную сложность задач за счет выявления скрытых структур и устранения избыточности, в частности, крестовая аппроксимация.

При решении прикладных задач получаемые тензоры могут иметь сложную структуру, и применение алгоритма TT-cross в общем случае будет связано с большими вычислительными затратами. В связи с этим обстоятельством существует необходимость в параллельной программной реализации данного алгоритма на высокопроизводительной вычислительной установке (системе).

Наиболее доступными и высокопроизводительными системами на настоящее время являются такие, в основе которых находятся графические процессоры (GPU). Графический процессор состоит из большого числа специализированных ядер и в общей сложности позволяет выполнять намного больше операций в единицу времени по сравнению с центральным процессором. Использование этого преимущества накладывает некоторые ограничения, а именно — алгоритм должен иметь возможность выполняться в параллельном варианте, т. е. необходима возможность часть операций проводить независимо друг от друга, иначе ядра будут ждать получения промежуточных результатов, и это может не только не дать желаемого ускорения, но и значительно замедлить выполнение программы. Поэтому необходимо выполнить детальный анализ алгоритма в целях поиска мест, которые могут быть распараллелены.

Реализация алгоритма выполняется с использованием технологий CUDA [14] компании Nvidia. Технология CUDA реализуется в виде надстройки над языком C [15], и для компиляции ее программ используется специальный компилятор, поставляемый производителем. Удобство состоит в том, что при знании языка C гораздо проще использовать CUDA. Кроме того, компилятор поддерживает язык C++ на центральном процессоре и позволяет использовать его стандартную библиотеку [16]. В идеале при работе графического процессора центральный процессор не простаивает и готовит данные для последующей их обработки. За счет такого подхода при грамотной реализации, для которой необходимо знать детали работы графического процессора, можно сократить временны́е затраты во много раз. В настоящее время графический процессор имеется в каждом среднем компьютере, соответственно, реализация алгоритма построения TT-разложения с использованием технологии CUDA может быть широко использована общественностью.

Алгоритм TT-cross

Приведем формальное описание TT-формата. Рассматривается d-мерный тензор $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$. Получение произвольного элемента с индексами $i_1, i_2, \ldots, i_d$ исходного тензора, представленного в TT-формате, выглядит следующим образом:

$$\mathbf{A}(i_1, i_2, \ldots, i_d) = \sum_{\alpha_1=1}^{r_1} \ldots \sum_{\alpha_{d-1}=1}^{r_{d-1}} \mathbf{G}_1(i_1, \alpha_1)\mathbf{G}_2(\alpha_1, i_2, \alpha_2)\ldots$$

$$\ldots \mathbf{G}_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1})\mathbf{G}_d(\alpha_{d-1}, i_d),$$

где $\mathbf{G}_1$, $\mathbf{G}_d$ — матрицы размером $n_1 \times r_1$ и $r_{d-1} \times n_d$ соответственно; а $\mathbf{G}_k$, $2 \leqslant k \leqslant (d-1)$ — трехмерные тензоры размером $r_{k-1} \times n_k \times r_k$.

Вычисление $\mathbf{A}(i_1, i_2, \ldots, i_d)$ по сути является перемножением вектора-строки на цепочку матриц и на вектор-столбец. Отметим, что у векторно-матричных операций зачастую высокая степень распараллеливания, и существует ряд готовых библиотек, их реализующих [17—19].

В данном контексте r_k называют рангами аппроксимации, а $\mathbf{G}_k$ — ядрами разложения или вагонами. Именно схожесть формата с поездом из-за сцепки ядер друг с другом индексами суммирования α_k повлекла за собой появление названия "тензорный поезд".

Для краткости далее при работе с тензорами или матрицами будет использована нотация MATLAB [20]. Например, запись $\mathbf{A}(i,:)$ означает взятие строк i, т. е. по первой размерности берется только индекс i, а по второй — все индексы. Заглавные буквы в индексах означают некоторое множество индексов. Диапазон значений от 1 до n будет обозначаться как $1{:}n$. Функция *reshape* — изменение размера матрицы (например, преобразование матрицы размером 3×4 к 6×2).

Разверткой тензора $\mathbf{A}_k$ будем называть матрицу, полученную в результате группировки первых k индексов как строчных, а остальных $d-k$ индексов как столбцовых:

$$\mathbf{A}_k = \mathbf{A}\left([i_1, i_2, ..., i_k], [i_{k+1}, i_{k+2}, ..., i_d]\right).$$

Таким образом, из тензора размером $n_1 \times n_2 \times ... \times n_d$ получается матрица размером $(n_1 n_2 ... n_k) \times (n_{k+1} n_{k+2} ... n_d)$. Приведя тензор к такому виду, можно удобно хранить значения в памяти компьютера и использовать все традиционные матричные методы. Отметим, что доказано существование TT-разложения с рангами r_k, равными рангу соответствующих матриц развертки $\mathbf{A}_k$ [21].

Описание всех алгоритмов приводится в соответствии с работами [21, 22]. Основная идея TT-cross заключается в последовательном применении матричного разложения к разверткам тензора и выборке наиболее важных элементов, по которым в дальнейшем можно будет восстановить все остальные.

Входные данные алгоритма: тензор **A** размером $n_1 \times n_2 \times ... \times n_d$, ранги аппроксимации $r_1, r_2, ..., r_{d-1}$. Результат выполнения алгоритма — ядра разложения $\mathbf{G}_1, \mathbf{G}_2, ..., \mathbf{G}_d$:

1) $N = \prod_{k=2}^{d} n_k$;

2) $\mathbf{A}_1 = reshape\left(\mathbf{A}, [n_1, N]\right)$ // первая развертка тензора **A**;

3) $\mathbf{Q}\widehat{\mathbf{Q}}^{-1}\mathbf{R} = crossApproximation\left(\mathbf{A}_1, r_1\right)$ // крестовая аппроксимация матрицы-развертки $\mathbf{A}_1$, где **Q** — матрица размера $n_1 \times r_1$; $\widehat{\mathbf{Q}}$ — матрица размера $r_1 \times r_1$; **R** — матрица размера $r_1 \times N$, составленная из r_1 строк матрицы $\mathbf{A}_1$;

4) $\mathbf{G}_1 = \mathbf{Q}\widehat{\mathbf{Q}}^{-1}$ // первое ядро разложения;

5) for $k = 2$ to $(d - 1)$ do // цикл по всем размерностям;

6) $N = \frac{N}{n_k}$;

7) $\mathbf{A}_k = reshape\left(\mathbf{R}, [r_{k-1} n_k, N]\right)$ // i-я развертка оставшейся части тензора;

8) $\mathbf{Q}\widehat{\mathbf{Q}}^{-1}\mathbf{R} = crossApproximation\left(\mathbf{A}_k, r_k\right)$ // крестовая аппроксимация матрицы-развертки $\mathbf{A}_k$, где **Q** — матрица размера $r_{k-1} n_k \times r_k$; $\widehat{\mathbf{Q}}$ — матрица размера $r_k \times r_k$ и **R** — матрица размера $r_k \times N$;

9) $\mathbf{G}_k = reshape\left(\mathbf{Q}\widehat{\mathbf{Q}}^{-1}, [r_{k-1}, n_k, r_k]\right)$ // преобразовать матрицу $\mathbf{Q}\widehat{\mathbf{Q}}^{-1}$ в тензор и приравнять k-е ядро разложения к нему;

10) end for;

11) $\mathbf{G}_d = \mathbf{R}$.

Важным свойством данного алгоритма является тот факт, что он имеет фиксированные ранги аппроксимации, которые в зависимости от ситуации можно увеличивать или уменьшать. Для хранения тензора в таком формате необходимо $O(dnr^2)$ элементов. Для приведения тензора в формат тензорного поезда необходимо выполнить $O(dnr^3 + n^d)$ операций, что составляет линейную сложность относительно числа элементов тензора.

Ключевым моментом в TT-cross является алгоритм крестовой аппроксимации *crossApproximation* [7]. Он позволяет построить компактное представление матрицы за счет выявления скрытых зависимостей с использованием при этом только части элементов исходной матрицы. По сути, *crossApproximation* выполняет сжатие данных (с потерями, если ранг матрицы превышает ранг аппроксимации). Входные данные алгоритма: матрица **A** размером $m \times n$, ранг аппроксимации r, параметр остановки δ. Результат выполнения алгоритма: разложение матрицы $\mathbf{Q}\widehat{\mathbf{Q}}^{-1}\mathbf{R} \approx \mathbf{A}$, где **Q** — матрица размера $m \times r$, полученная с помощью *QR*-разложения матрицы, составленной из r столбцов матрицы **A**; $\widehat{\mathbf{Q}}$ — матрица размера $r \times r$, составленная из r строк матрицы **Q**; **R** — матрица размера $r \times n$, составленная из r строк матрицы **A**:

1) $\mathbf{I} = random\left([1, m], r\right)$ // r случайных строчных индексов;

2) $\mathbf{A}_0 = zeros\left(m, n\right)$ //$zeros\left(m, n\right)$ — матрица с нулевыми элементами размером $m \times n$;

3) $k = 1$;

4) do;

5) $\mathbf{R} = \mathbf{A}(\mathbf{I},:)$// строки с индексами **I**;

6) $\mathbf{R} = \mathbf{R}^{\mathrm{T}}$;

7) $\mathbf{QT} = QR\left(\mathbf{R}\right)$ // *QR*-разложение матрицы **R**;

8) $\mathbf{J} = maxvol(\mathbf{Q})$ // поиск индексов подматрицы максимального объема;

9) $\mathbf{C} = \mathbf{A}(:,\mathbf{J})$ // столбцы с индексами **J**;

10) $\mathbf{QT} = QR(\mathbf{C})$;

11) $\mathbf{I} = maxvol(\mathbf{Q})$;

12) $\widehat{\mathbf{Q}} = \mathbf{Q}(\mathbf{I}:)$;

13) $\mathbf{A}_k = \mathbf{Q}\widehat{\mathbf{Q}}^{-1}\mathbf{A}\left(\mathbf{I},:\right)$ // k-е приближение;

14) $k = k + 1$;

15) while$\left(\left\|\mathbf{A}_k - \mathbf{A}_{k-1}\right\| > \delta\left\|\mathbf{A}_k\right\|\right)$;

16) $\mathbf{R} = \mathbf{A}(\mathbf{I},:)$.

Алгоритм *maxvol* используется для нахождения индексов подматрицы максимального объема и является основным вспомогательным алгоритмом, используемым в ходе работы крестовой аппроксимации. На каждой итерации *crossApproximation* рассматривается подматрица, состоящая из r строк исходной матрицы, для которой определяются номера r столбцов, содержащие подматрицу максимального объема. Далее берется подматрица матрицы **A**, состоящая уже из r соответствующих столбцов, и в ней определяются номера r строк, содержащих подматрицу максимального объема. Алгоритм завершается, когда полученные на двух подряд идущих итерациях аппроксимации исходной матрицы отличаются меньше, чем на некоторое значение, характеризующееся параметром остановки δ.

Подматрица максимального объема — это подматрица, имеющая максимальный по модулю определитель. На практике обычно ищется такая подматрица $\widehat{\mathbf{C}}$ матрицы **C** $(m \times r, m > r)$, что элементы матрицы $\mathbf{C}\widehat{\mathbf{C}}^{-1}$ по модулю не превосходят 1 [22].

Входные данные алгоритма *maxvol*: матрица **A** размером $m \times r$, $m > r$. Результат выполнения алгоритма: массив размером r, содержащий номера строк матрицы **A**, составляющих подматрицу максимального объема:

1) $\mathbf{I} = 1 : m$ // массив, который отображает индекс строки в исходной матрице;

2) for $i = 1$ to r do // выбрать случайные строки в исходной матрице и передвинуть их в верхний блок;

3) $ind = random\ (2, m)$ // случайный индекс;

4) $swapRows\left(\mathbf{A}, i, ind\right)$// поменять местами строки матрицы;

5) $swap\left(\mathbf{I}, i, ind\right)$// поменять местами значения в массиве индексов;

6) end for;
7) $\hat{\mathbf{A}} = getRows(\mathbf{A}, [1:r])$ // взять первые r строк;
8) $\mathbf{B} = \mathbf{A}\hat{\mathbf{A}}^{-1}$;
9) do;
10) $i, j = argmax(\mathbf{B})$ // индексы максимального по модулю элемента в $\mathbf{B}$;
11) $B_{\max} = |\mathbf{B}(i, j)|$;
12) if $B_{\max} > 1$ then;
13) $\mathbf{B} = \mathbf{B} - \frac{1}{\mathbf{B}(i, j)}\left(\mathbf{B}(:, j) - \mathbf{e}_j + \mathbf{e}_i\right)\left(\mathbf{B}(i,:) - \mathbf{e}_j^{\mathrm{T}}\right)$;
14) $swap(\mathbf{I}, i, j)$;
15) end if;
16) while$(B_{\max} > 1 + \varepsilon)$;
17) $resultIndices = \mathbf{I}(1:r)$ // индексы строк изначальной матрицы $\mathbf{A}$, которые составляют подматрицу максимального объема.

Общая вычислительная сложность *maxvol* $O\left(r^3 + 3rk(n-r)\right)$, где k — число итераций алгоритма. Параметр ε — небольшое число (обычно $\varepsilon = 0{,}02$), позволяющее ускорить сходимость алгоритма (получить дополнительный прирост производительности).

В данном разделе рассмотрен алгоритм TT-cross, в основе которого лежат алгоритм крестовой аппроксимации и алгоритм *maxvol*. Описанные алгоритмы построены на операциях с векторами и матрицами и, следовательно, имеют потенциал для параллельной реализации, а также использования возможностей технологии CUDA.

Распараллеливание и реализация

Выполнена реализация алгоритма TT-cross в двух вариантах — последовательном и параллельном. В TT-cross используются операции с векторами и матрицами, их реализация была взята из готовых библиотек.

В последовательной реализации была использована библиотека LAPACKE, которая представляет собой интерфейс к открытой библиотеке LAPACK [23] на языке Fortran. Данная библиотека является широко распространенной библиотекой алгоритмов линейной алгебры. За долгое время ее существования она была качественно отлажена и оптимизирована.

Для реализации, выполненной на графическом процессоре, была использована библиотека MAGMA [17]. Данная библиотека является аналогом LAPACK для графических процессоров и предоставляет реализацию алгоритмов линейной алгебры уже в параллельном варианте. Стоит принимать во внимание, что обычные графические процессоры изначально предназначены для работы с числами одинарной точности, так как для решения задач компьютерной графики большая точность не требуется. Однако численные методы чаще всего подразумевают использование чисел двойной точности. Отметим, что для работы с числами повышенной точности компании выпускают специальные научные графические процессоры, которые оптимизированы специально для этого. Были проведены сравнительные тесты производительности [24], которые показали, что MAGMA при работе с 64-битными вещественными числами не уступает по производительности библиотеке cuBLAS [18] от разработчиков Nvidia, при этом имеет более широкий функционал, поэтому именно она была выбрана к использованию.

Все многомерные объекты в памяти хранятся в линеаризованном виде. Например, матрицы хранятся в массиве по столбцам. Это означает, что элемент $\mathbf{A}(i, j)$ в одномерном массиве будет иметь индекс $z = i + jm$, где $\mathbf{A}$ — матрица размером $l \times m$. Аналогично для трехмерного тензора $l \times m \times n$ индекс для элемента $\mathbf{A}(i, j, k)$ будет преобразован в линейный индекс $z = i + jm + kmn$. Нетрудно заметить, что организация многопоточной программы для графического процессора с выбором номера потока имеет схожие принципы, поэтому данный метод линеаризации хорошо ложится на GPU. Следует отметить, что при данном способе хранения операция *reshape*, которая используется в алгоритме TT-cross для получения следующей развертки тензора, выполняется за константное время и не требует перестановки элементов тензора.

С использованием описанных выше подходов была выполнена реализация алгоритмов TT-cross, крестовой аппроксимации и *maxvol* на языке программирования C++ с использованием технологии CUDA.

Результаты

Каждая часть алгоритма TT-cross тестировалась отдельно: сначала алгоритм *maxvol*, далее крестовая аппроксимация, а в заключение весь алгоритм TT-cross. Характеристики CPU: Intel Core i5-9400F, ОП: 48 Гбайт, 3200 МГц; GPU: GeForce GTX 1060 6 Гбайт.

Апробация реализаций алгоритма *maxvol* выполнялась на нескольких матрицах, сгенерированных случайным образом. В табл. 1 представлены замеры времени работы.

Таблица 1

Результаты работы алгоритма *maxvol*

Результаты	Размер матрицы, $m \times n$			
	500×3000	5000×500	10000×300	10000×500
Время CPU, мс	12 589	31 622	25 218	60 195
Время GPU, мс	316	501	398	630
Ускорение, раз	39,8	63,1	63,4	95,5

По данным табл. 1 видно, что за счет использования GPU удалось сократить расчетное время более чем в 95 раз. Самой трудозатратой частью алгоритма является обращение верхнего блока матрицы, а данная операция имеет хороший потенциал для распараллеливания, что позволяет получать отличные результаты.

Реализации крестовой аппроксимации тестировались на матрицах определенного ранга. Для того чтобы получить матрицу $m \times n$ рангом $\leqslant r$, следует перемножить матрицы $m \times r$ и $r \times n$ ранга r. Алгоритм корректно работает, если при ранге аппроксимации, большем или равном r, матрица будет в точности восстановлена из полученного разложения, причем аппроксимация рангом более чем r должна находиться корректно. Следует отметить, что при ранге аппроксимации меньшем r должны возникать погрешности.

Сначала выполняется апробация на матрице размером 10000×10000 ранга 1000. В табл. 2 представлены замеры времени при различном значении задаваемого ранга аппроксимации.

Максимальное ускорение в 40 раз достигается, когда заданный ранг совпадает с фактическим рангом матрицы. Время нахождения аппроксимации сокращается примерно с 4 мин до 6 с, что является очень значительным приростом. Качество восстановления матрицы при этом остается неизменным. Поэлементные разности между исходными и восстановленными матрицами, полученными в результате работы обеих реализаций, составляют около 10^{-16}, что граничит с машинной точностью для 64-битных вещественных чисел. А Евклидова норма разности исходной и восстановленной матриц составила около 10^{-9}.

Далее выполняется тестирование на матрицах разного размера с одинаковым рангом — 95 (табл. 3).

Для матрицы размером 100 × 100 реализация на CPU оказалась быстрее, так как подготовка запуска вычислений на GPU требует дополнительного времени, сопоставимого со временем самих вычислений. Однако при увеличении размера матрицы ускорение возрастает до 25 раз, хотя ранг является довольно малым относительно размера матрицы.

Таблица 2

Результаты работы крестовой аппроксимации на матрице 10000×10000 ранга 1000

Результаты	Ранг аппроксимации		
	10	100	1000
Время CPU, мс	9347	21 032	240 763
Время GPU, мс	310	781	6046
Ускорение, раз	30,2	26,9	39,8

Таблица 3

Результаты крестовой аппроксимации на матрицах разного размера

Результаты	Размер матрицы $m \times n$		
	100×100	1000×1000	10000×10000
Время CPU, мс	9	488	19 640
Время GPU, мс	32	126	740
Ускорение, раз	0,3	3,9	26,5

Алгоритм TT-cross тестировался на нескольких тензорах, заданных в виде функций. Ранги аппроксимации, участвующие в алгоритме, подбирались так, чтобы по построенному тензорному поезду можно было в точности восстановить все элементы исходного тензора.

Элементы тензора Гильберта [7] задаются следующим образом:

$$\mathbf{A}(i_1, i_2, \ldots, i_d) = \frac{1}{i_1 + i_2 + \ldots + i_d}, \quad 1 \leqslant i_j \leqslant n, \quad 1 \leqslant j \leqslant d.$$

В табл. 4 приведены временные затраты на построение тензорного поезда и восстановление тензора при различных значениях n с фиксированной размерностью $d = 5$. Ранги для данного тензора

Таблица 4

Результаты разложения и восстановления тензора Гильберта $d = 5$

Операция	Результаты	n		
		10	20	40
Построение тензорного поезда	Время CPU, мс	47	866	22 674
	Время GPU, мс	150	222	2415
	Ускорение, раз	0,3	3,9	9,4
Восстановление элементов тензора	Время CPU, мс	4	107	3648
	Время GPU, мс	2	4	35
	Ускорение, раз	2	26,8	104,2

равны 5, и при построении тензорного поезда они задавались соответствующими. Здесь наблюдаются десятикратное ускорение при построении тензорного поезда и стократное ускорение при его восстановлении. Получение такого большого ускорения связано с тем обстоятельством, что восстановление состоит исключительно из операции перемножения матрицы на вектор, которая обладает высокой степенью параллелизма.

Исследуем, как изменяется время работы при разных рангах аппроксимации на тензоре размером $80 \times 80 \times 80 \times 80$ (табл. 5).

Здесь ускорение при построении тензорного поезда составляет около 17 раз, а при восстановлении тензора — 70—80 раз. Данный расчет показал, что увеличение ранга аппроксимации приводит к увеличению вычислительного времени. Это связано с тем, что возрастают размеры соответствующих матриц, получающихся в процессе работы алгоритма.

Далее рассмотрим тензор, элементы которого задаются с помощью функции Гривонка [25]:

$$\mathbf{A}(i_1, i_2, ..., i_d) = 1 + \frac{1}{4000}\sum_{k=1}^{d} i_k^2 - \prod_{k=1}^{d} \cos\left(\frac{i_k}{\sqrt{k}}\right),$$
$$1 \leqslant i_j \leqslant n,\ 1 \leqslant j \leqslant d.$$

В отличие от тензора Гильберта, здесь есть тяжелые в вычислительном плане функции, такие как cos и $\sqrt{}$.

Для восстановления тензора с абсолютной точностью 10^{-9} достаточно использовать ранги аппроксимации, равные 3. В табл. 6 показаны замеры времени на тензорах разной размерности при $n = 75$.

При размерности тензора больше 3 параллельная реализация TT-cross работает в 10 и более раз быстрее прямого вычисления всех элементов и последовательной реализации. Необходимо отметить, что суммарное время на построение тензорного поезда и восстановление всех его элементов меньше, чем время на прямое вычисление всех элементов тензора, что еще раз дополнительно подтверждает эффективность тензорных поездов. Кроме этого, для хранения элементов пятимерного тензора с двойной точностью (8 байт на элемент) потребовалось бы 18 Гбайт памяти, что достаточно существенно. При этом отметим, что построение тензорного поезда успешно выполнилось на видеокарте с 6 Гбайт графической памяти.

Приведенные в данном разделе результаты демонстрируют эффективность параллельной реализации алгоритмов TT-cross, крестовой аппроксимации и *maxvol*.

Таблица 5

Временны́е затраты на разложение и восстановление тензора Гильберта размером $80 \times 80 \times 80 \times 80$ при разных значениях ранга

Операция	Результаты	Ранг аппроксимации			
		5	10	20	40
Построение тензорного поезда	Время CPU, мс	7963	10 454	19 822	51 097
	Время GPU, мс	450	601	1142	2979
	Ускорение, раз	17,7	17,4	17,4	17,2
Восстановление элементов тензора	Время CPU, мс	1269	1613	2566	4822
	Время GPU, мс	16	23	32	62
	Ускорение, раз	79,3	70,1	80,2	77,8

Таблица 6

Временны́е затраты на разложение и восстановление тензора, полученного с помощью функции Гривонка

Операция	Результаты	Размерность, d		
		3	4	5
Построение тензорного поезда	Время CPU, мс	65	5755	220 923
	Время GPU, мс	71	249	11 628
	Ускорение, раз	0,9	23,1	19
Восстановление элементов тензора	Время CPU, мс	7	873	31 622
	Время GPU, мс	1	13	502
	Ускорение, раз	7	67,2	63
Прямое вычисление всех элементов тензора, CPU, мс		44	4086	350 603

Заключение

Необходимость работать с многомерными данными возникает во многих современных областях. Формат тензорного поезда является эффективным форматом представления многомерных данных. В работе рассматриваются вопросы распараллеливания алгоритма TT-cross, который позволяет строить разложение в тензорный поезд без вычисления всех элементов исходного многомерного тензора, на графических процессорах. Алгоритм TT-cross включает в себя алгоритм крестовой аппроксимации и алгоритм *maxvol*. Каждый из алгоритмов был реализован с использованием технологии CUDA и библиотеки параллельных матрично-векторных операций MAGMA. Выполнено сравнение параллельных реализаций алгоритмов с последовательными реализациями. По результатам, полученным на представительном наборе тестовых примеров, можно сделать следующие выводы: алгоритм *maxvol* ускорился в 90 раз; алгоритм крестовой аппроксимации — в 40 раз; алгоритм TT-cross — в 20 раз. Отметим, что вычисления выполнялись с использованием обычной видеокарты прошлого поколения. Полученные результаты демонстрируют эффективность разработанных параллельных реализаций соответствующих алгоритмов.

Список литературы

1. **Морозов А. Ю., Ревизников Д. Л.** Алгоритм адаптивной интерполяции на основе kd-дерева для численного интегрирования систем ОДУ с интервальными начальными условиями // Дифференциальные уравнения. 2018. Т. 54, № 7. С. 963—974. DOI: 10.1134/S0374064118070130.

2. **Morozov A. Yu., Reviznikov D. L.** Modelling of Dynamic Systems with Interval Parameters on Graphic Processors // Программная инженерия. 2019. Vol. 10, No. 2. P. 69—76. DOI: 10.17587/prin.10.69-76.

3. **Морозов А. Ю.** Параллельный алгоритм адаптивной интерполяции на основе разреженных сеток для моделирования динамических систем с интервальными параметрами // Программная инженерия. 2021. Т. 12, № 8. С. 395—403. DOI: 10.17587/prin.12.395-403.

4. **Морозов А. Ю., Журавлев А. А., Ревизников Д. Л.** Анализ и оптимизация алгоритма адаптивной интерполяции численного решения систем обыкновенных дифференциальных уравнений с интервальными параметрами // Дифференциальные уравнения. 2020. Т. 56, № 7. С. 960—974. DOI: 10.1134/S0374064120070122.

5. **Гидаспов В. Ю., Морозов А. Ю., Ревизников Д. Л.** Алгоритм адаптивной интерполяции с использованием TT-разложения для моделирования динамических систем с интервальными параметрами // Журнал вычислительной математики и математической физики. 2021. Т. 61, № 9. С. 1416—1430. DOI: 10.31857/S0044466921090106.

6. **Oseledets I. V.** Tensor-train decomposition // SIAM Journal on Scientific Computing. 2011. Vol. 33, No. 5. P. 2295—2317. DOI: 10.1137/090752286.

7. **Oseledets I., Tyrtyshnikov E.** TT-cross approximation for multidimensional arrays // Linear Algebra and its Applications. 2010. Vol. 432, Is. 1. P. 70—88. DOI: 10.1016/j.laa.2009.07.024.

8. **Hitchcock F. L.** The expression of a tensor or a polyadic as a sum of products // J. Math. Phys. 1927. Vol. 6, No. 1. P. 164—189. DOI: 10.1002/sapm192761164.

9. **Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.** 2.6 Singular Value Decomposition. Numerical Recipes in C. 1992. 2nd ed. Cambridge: Cambridge University Press.

10. **Тыртышников Е. Е.** Тензорные аппроксимации матриц, порожденных асимптотически гладкими функциями // Математический сборник. 2003. Т. 194, № 6. С. 147—160. DOI: 10.4213/sm747.

11. **Тыртышников Е. Е., Щербакова Е. М.** Методы неотрицательной матричной факторизации на основе крестовых малоранговых приближений // Журнал вычислительной математики и математической физики. 2019. Т. 59, № 8. С. 1314—1330. DOI: 10.1134/S0044466919080179.

12. **Желтков Д. А., Тыртышников Е. Е.** Параллельная реализация матричного крестового метода // Вычислительные методы и программирование. 2015. Т. 16. С. 369—375. DOI: 10.26089/NumMet.v16r336.

13. **Горейнов С. А., Замарашкин Н. Л., Тыртышников Е. Е.** Псевдоскелетные аппроксимации при помощи подматриц наибольшего объема // Матем. заметки. 1997. Т. 62, Вып. 4. С. 619—623. DOI: 10.4213/MZM1644.

14. **CUDA** Zone. URL: https://developer.nvidia.com/cuda-zone

15. **Керниган Б. У., Ритчи Д. М.** Язык программирования С. М.: Вильямс, 2019. 288 с.

16. **Джосаттис Н. М.** Стандартная библиотека C++. Справочное руководство. М.: Вильямс, 2017. 1129 с.

17. **Matrix** Algebra on GPU and Multicore Architectures (MAGMA). URL: https://icl.cs.utk.edu/magma

18. **cuBLAS**. URL: https://docs.nvidia.com/cuda/cublas

19. **ScaLAPACK**. URL: http://www.netlib.org/scalapack

20. **MATLAB.** URL: https://www.mathworks.com/help/matlab

21. **Оселедец И. В.** Тензорные методы и их применение: дис. ... д-ра физ.-мат. наук: 01.01.07. М.: Учреждение Российской академии наук Институт вычислительной математики РАН, 2009. 282 с.

22. **Goreinov S. A., Oseledets I. V., Savostyanov D. V., Tyrtyshnikov E. E.** How to find a good submatrix. Institute for Computational Mathematics Hong Kong Baptist University, China, 2008. 10 p. DOI: 10.1142/9789812836021_0015.

23. **LAPACK.** URL: http://www.netlib.org/lapack

24. **Chrzeszczyk A., Anders J.** Matrix computations on the GPU. Jan Kochanowski University, Poland, 2017. 455 p.

25. **Griewank A. O.** Generalized Decent for Global Optimization // Journal of Optimization Theory and Applications. 1981. Vol. 34. P. 11—39. DOI:10.1007/BF00933356.

Parallel Approximation of Multivariate Tensors using GPUs

N. S. Kapralov[1,2], nskaprl@gmail.com, **A. Yu. Morozov**[1,3], morozov@infway.ru,
S. P. Nikulin[1], sergeynp@yandex.ru
[1]Moscow Aviation Institute (MAI), Moscow, Russian Federation
[2]Deutsche Bank Technology Center, Moscow, Russian Federation
[3]Federal Research Center "Computer Science and Control" of Russian Academy of Sciences (FRC CSC RAS), Moscow, Russian Federation

Corresponding author:

Morozov Alexander Yu., Senior Lecturer, Moscow Aviation Institute (MAI), Moscow, Russian Federation, Researcher, Federal Research Center "Computer Science and Control" of Russian Academy of Sciences (FRC CSC RAS), Moscow, Russian Federation
E-mail: morozov@infway.ru

Received on December 16, 2021
Accepted on December 28, 2021

When solving many applied and research problems, it becomes necessary to work with multidimensional arrays (tensors). In practice, an efficient and compact representation of these objects is used in the form of so-called tensor trains. The paper considers a parallel implementation of the TT-cross algorithm, which allows one to obtain a decomposition of a multidimensional array into a tensor train using a graphics processor of the CUDA architecture. The main aspects and features of parallelization and implementation of the algorithm are presented. The obtained parallel implementation was tested on a representative number of examples. A significant reduction in computational time is demonstrated in comparison with a similar sequential implementation of the algorithm, which indicates the effectiveness of the proposed approaches to parallelization.

Keywords: *parallelization, tensor train, tensor decomposition, high dimensions, curse of dimensionality, multidimensional arrays, cross approximation, TT-cross, maxvol, low-rank approximation, CUDA, GPU, Nvidia*

For citation:

Kapralov N. S., **Morozov A. Yu.**, **Nikulin S. P.** Parallel Approximation of Multivariate Tensors using GPUs, *Programmnaya Ingeneria*, 2022, vol. 13, no. 2, pp. 94—101.

DOI: 10.17587/prin.13.94-101

References

1. **Morozov A. Y., Reviznikov D. L.** Adaptive Interpolation Algorithm Based on a kd-Tree for Numerical Integration of Systems of Ordinary Differential Equations with Interval Initial Conditions, *Differential Equations,* 2018, vol. 54, no. 7, pp. 945—956, DOI: 10.1134/S0012266118070121.

2. **Morozov A. Yu., Reviznikov D. L.** Modelling of Dynamic Systems with Interval Parameters on Graphic Processors, *Programmnaya Ingeneria*, 2019, vol. 10, no. 2, pp. 69—76. DOI: 10.17587/prin.10.69-76.

3. **Morozov A. Yu.** Parallel Adaptive Interpolation Algorithm based on Sparse Grids for Modeling Dynamic Systems with Interval Parameters, *Programmnaya Ingeneria,* 2021, vol. 12, no. 8, pp. 395—403. DOI: 10.17587/prin.12.395-403 (in Russian).

4. **Morozov A. Y., Zhuravlev A. A., Reviznikov D. L.** Analysis and Optimization of an Adaptive Interpolation Algorithm for the Numerical Solution of a System of Ordinary Differential Equations with Interval Parameters, *Differential Equations,* 2020, vol. 56, no. 7, pp. 935—949, DOI:10.1134/s0012266120070125.

5. **Gidaspov V. Yu., Morozov A. Yu., Reviznikov D. L.** Adaptive Interpolation Algorithm Using TT-Decomposition for Modeling Dynamical Systems with Interval Parameters, *Computational Mathematics and Mathematical Physics,* 2021, vol. 61, no. 9, pp. 1387—1400, DOI: 10.1134/S0965542521090098.

6. **Oseledets I. V.** Tensor-train decomposition, *SIAM Journal on Scientific Computing,* 2011, vol. 33, no. 5, pp. 2295—2317. DOI: 10.1137/090752286.

7. **Oseledets I., Tyrtyshnikov E.** TT-cross approximation for multidimensional arrays, *Linear Algebra and its Applications*, 2010, vol. 432, is. 1, pp. 70—88. DOI: 10.1016/j.laa.2009.07.024.

8. **Hitchcock F. L.** The expression of a tensor or a polyadic as a sum of products, *J. Math. Phys.* 1927, vol. 6, no. 1, pp. 164—189. DOI: 10.1002/sapm192761164.

9. **Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.** *2.6 Singular Value Decomposition, Numerical Recipes in C.* 1992, 2nd edition, Cambridge, Cambridge University Press.

10. **Tyrtyshnikov E. E.** Tensor approximations of matrices generated by asymptotically smooth functions, *Sbornik: Mathematics*, 2003, vol. 194, no. 6. pp. 941—954. DOI: 10.1070/SM2003v194n06ABEH000747.

11. **Tyrtyshnikov E. E., Shcherbakova E. M.** Methods for nonnegative matrix factorization based on low-rank cross approximations, *Computational Mathematics and Mathematical Physics,* 2019, vol. 59, no. 8, pp. 1251—1266. DOI: 10.1134/S0965542519080165.

12. **Zheltkov D. A. Tyrtyshnikov E. E.** A parallel implementation of the matrix cross approximation method, *Numerical Methods and Programming,* 2015, vol. 16, no. 3, pp. 369—375. DOI: 10.26089/NumMet.v16r336.

13. **Goreinov S. A., Zamarashkin N. L., Tyrtyshnikov E. E.** Pseudo-skeleton approximations by matrices of maximal volume, *Mathematical Notes*, 1997, vol. 62, pp. 515—519. DOI:10.1007/2FBF02358985.

14. **CUDA** Zone, available at: https://developer.nvidia.com/cuda-zone

15. **Kernighan B. W., Ritchie D. M.** *C Programming Language*, Prentice-Hall, 1988.

16. **Josuttis N. M.** *The C++ Standard Library: A Tutorial and Reference*, Addison Wesley, 2012, 1136 p.

17. **Matrix** Algebra on GPU and Multicore Architectures (MAGMA), available at: https://icl.cs.utk.edu/magma

18. **cuBLAS,** available at: https://docs.nvidia.com/cuda/cublas

19. **ScaLAPACK,** available at: http://www.netlib.org/scalapack

20. **MATLAB,** available at: https://www.mathworks.com/help/matlab

21. **Oseledets I. V.** Tensor methods and their applications: Dissertation of Doctor of Physical and Mathematical Sciences: 01.01.07. Moscow Center of Fundamental and Applied Mathematics at INM RAS, Moscow, 2009, 282 p. (in Russian).

22. **Goreinov S. A., Oseledets I. V., Savostyanov D. V., Tyrtyshnikov E. E.** *How to find a good submatrix*, Institute for Computational Mathematics Hong Kong Baptist University, China, 2008, 10 p. DOI: 10.1142/9789812836021_0015.

23. **LAPACK**, available at: http://www.netlib.org/lapack

24. **Chrzeszczyk A., Anders J.** *Matrix computations on the GPU,* Jan Kochanowski University, Poland, 2017, 455 p.

25. **Griewank A. O.** Generalized Decent for Global Optimization, *Journal of Optimization Theory and Applications,* 1981, vol. 34, pp. 11—39. DOI: 10.1007/BF00933356.

4—6 октября 2022 г. в Санкт-Петербурге
на базе АО "Концерн" ЦНИИ "Электроприбор" состоится

15-я мультиконференция

по проблемам управления (МКПУ-2022)

Председатель президиума мультиконференции — академик РАН **В. Г. Пешехонов**

Мультиконференция включает следующие пять локальных конференций, объединенных общей идеей:

- **XXXIII конференция памяти выдающегося конструктора гироскопических приборов Н. Н. Острякова**
  Председатель программного комитета — акад. РАН **В. Г. Пешехонов**
- **Конференция "Информационные технологии в управлении" (ИТУ-2022)**
  Сопредседатели программного комитета: член-корр. РАН **В. Н. Васильев**, д.т.н., проф. **В. Н. Шелудько**, член-корр. РАН **Р. М. Юсупов**
- **Конференция "Математическая теория управления и ее приложения" (МТУиП-2022)**
  Сопредседатели программного комитета: член-корр. РАН **Д. А. Новиков** и д.ф.-м.н., проф. **Н. В. Кузнецов**
- **Конференция "Управление в аэрокосмических системах" имени академика Е. А. Микрина (УАКС-2022)**
  Сопредседатели программного комитета: акад. РАН **С. Ю. Желтов**, член-корр. РАН **В. А. Соловьев**, член-корр. РАН **М. В. Сильников**
- **Конференция "Управление в морских системах" (УМС-2022)**
  Сопредседатели программного комитета: акад. РАН **Е. И. Якушенко**, акад. РАН **С. Н. Васильев**, член-корр. РАН **А. Ф. Щербатюк**

В рамках мультиконференции пройдет Семинар по закрытой тематике

Информация для связи:

ГНЦ РФ АО "Концерн "ЦНИИ "Электроприбор",
Тел.: +7 (812) 499-82-10 — Истомина Елена Анатольевна
+7 (812) 499 82 67 — Тарановский Дмитрий Олегович
Факс: +7 (812) 232 33 76, E-mail: icins@eprib.ru

XXXV Международная научная конференция МАТЕМАТИЧЕСКИЕ МЕТОДЫ В ТЕХНИКЕ И ТЕХНОЛОГИЯХ — ММТТ-35

Конференция ММТТ-35 (http://mmtt.sstu.ru/) проводится 30 мая — 3 июня 2022 г. в Ярославле на базе Ярославского государственного технического университета (ЯГТУ), Ярославского государственного университета им. П. Г. Демидова (ЯрГУ), Государственной академии промышленного менеджмента имени Н. П. Пастухова (Академия Пастухова); 05—07 октября 2022 г. в Саратове на базе Саратовского государственного технического университета (СГТУ); 24—28 октября 2022 г. в Минске на базе Белорусского национального технического университета (БНТУ).

Научная конференция ММТТ-35 проводится для обсуждения опыта использования математических методов в технике и технологиях и современных направлений математического и компьютерного обеспечения технологических и технических систем.

ЯРОСЛАВСКАЯ НАУЧНАЯ КОНФЕРЕНЦИЯ ММТТ-35

состоит из следующих секций

- Качественные и численные методы исследования дифференциальных и интегральных уравнений
- Оптимизация, автоматизация и оптимальное управление технологическими процессами
- Математическое моделирование технологических и социальных процессов
- Математическое моделирование и оптимизация в задачах САПР, аддитивных технологий
- Математические методы в задачах радиотехники, радиоэлектроники и телекоммуникаций, геоинформатики, авионики и космонавтики
- Математические методы и интеллектуальные системы в робототехнике и мехатронике
- Математические методы в медицине, биотехнологии и экологии
- Математические методы в экономике и гуманитарных науках
- Информационные и интеллектуальные технологии в технике и образовании
- Математические и инструментальные методы технологий Индустрии 4.0
- Обсуждение квалификационных работ
- Круглые столы

ЯРОСЛАВСКАЯ ШКОЛА МОЛОДЫХ УЧЕНЫХ — ШМУ-38

включает цикл лекций по современным проблемам математического моделирования, информатизации и интеллектуализации технических систем различного назначения, а также следующие секции:

- интеллектуализация управляемых технологических процессов
- информатизация технических систем и процессов

Научные доклады принимаются до 20 мая 2022 г.

САРАТОВСКАЯ ШКОЛА МОЛОДЫХ УЧЕНЫХ — ШМУ-39

включает цикл лекций по современным проблемам вычислительной математики, интеллектуальным алгоритмам и системам, информационным технологиям, а также две секции:

- интеллектуализация управляемых технологических процессов
- информатизация технических систем и процессов

В рамках ШМУ-39 будет проведен полуфинал Конкурса по программе УМНИК (более подробная информация на сайте http://www.fasie.ru).
Доклады на ШМУ-39 и на Конкурс принимаются до 30 сентября 2022 г.

МИНСКАЯ НАУЧНАЯ КОНФЕРЕНЦИЯ ММТТ-35

проводится по направлениям Санкт-Петербургской конференции в рамках XI Форума вузов инженерно-технологического профиля в Минске на базе БНТУ.
Научные доклады принимаются до 30 сентября 2022 г.

Подробности: http://mmtt.sstu.ru/mmtt-35.nsf

МЕЖДУНАРОДНАЯ НАУЧНАЯ КОНФЕРЕНЦИЯ

"Параллельные вычислительные технологии (ПаВТ) 2022"

29—31 марта 2022 г., Дубна, Объединенный институт ядерных исследований

"Параллельные вычислительные технологии (ПаВТ) 2022" — международная научная конференция, шестнадцатая в серии ежегодных конференций, посвященных развитию и применению параллельных вычислительных технологий и машинного обучения в различных областях науки и техники.

Главная цель конференции — предоставить возможность для представления и обсуждения результатов, полученных ведущими научными группами в использовании суперкомпьютерных и нейросетевых технологий для решения практических задач.

Организаторы конференции

- Министерство науки и высшего образования РФ
- Суперкомпьютерный консорциум университетов России

Тематика конференции охватывает следующие основные направления (но не ограничивается ими)

- Технологии параллельных и распределенных вычислений
- Облачные вычисления
- Перспективные многопроцессорные архитектуры
- Параллельные и распределенные системы баз данных
- Искусственные нейронные сети и глубокое обучение
- Управление, администрирование, мониторинг и тестирование многопроцессорных систем
- Вычислительная математика
- Вычислительная физика
- Вычислительная химия
- Гидро-газодинамика и теплообмен
- Высоконелинейные и быстротекущие процессы в задачах механики
- Биоинформатика и медицина
- Нанотехнологии
- Геоинформатика
- Криптография
- Обработка изображений и визуализация
- Компьютерная алгебра
- Суперкомпьютерные НОЦ.

В первый день работы конференции будет объявлена **36-я редакция списка Top50** самых мощных компьютеров СНГ.

Во все дни работы конференции будет действовать **суперкомпьютерная выставка**, на которой ведущие производители аппаратного и программного обеспечения представят свои новейшие разработки в области высокопроизводительных вычислений.

Языки конференции: русский, английский.

Официальный сайт конференции: http://agora.guru.ru/pavt2022/

ООО "Издательство "Новые технологии". 107076, Москва, ул. Матросская Тишина, д. 23, стр. 2
Технический редактор *Е. М. Патрушева.* Корректор *А. В. Чугунова.*

Сдано в набор 28.12.2021 г. Подписано в печать 03.02.2022 г. Формат 60×88 1/8. Заказ PI221
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**

Рисунки к статье Д. И. Читалова

«О РАЗРАБОТКЕ МОДУЛЯ ДЛЯ РАБОТЫ С РЕШАТЕЛЕМ buoyantSimpleFoam И УТИЛИТОЙ postProcess ПЛАТФОРМЫ OpenFOAM»

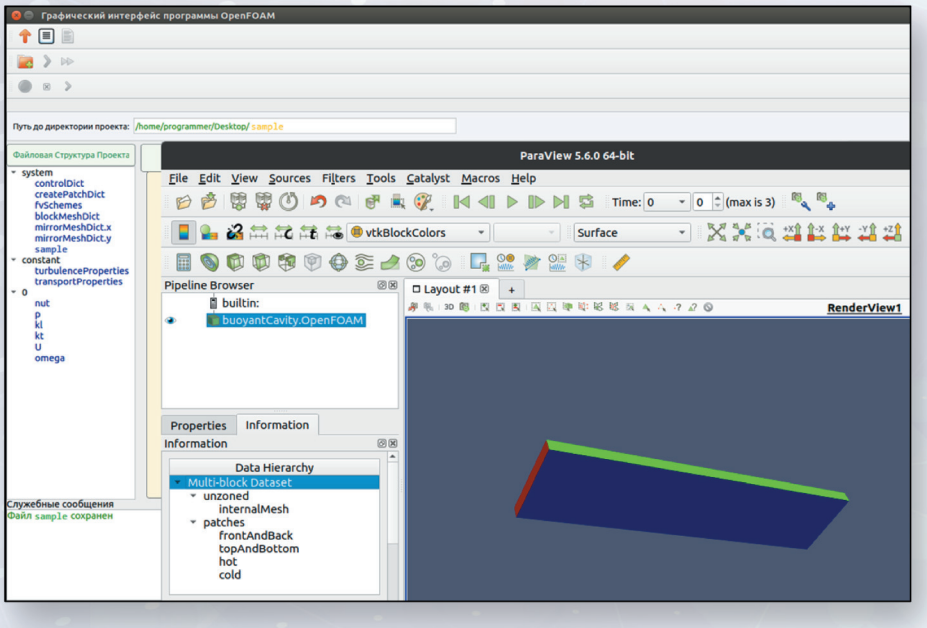


Рис. 4. Результаты генерации расчетной сетки

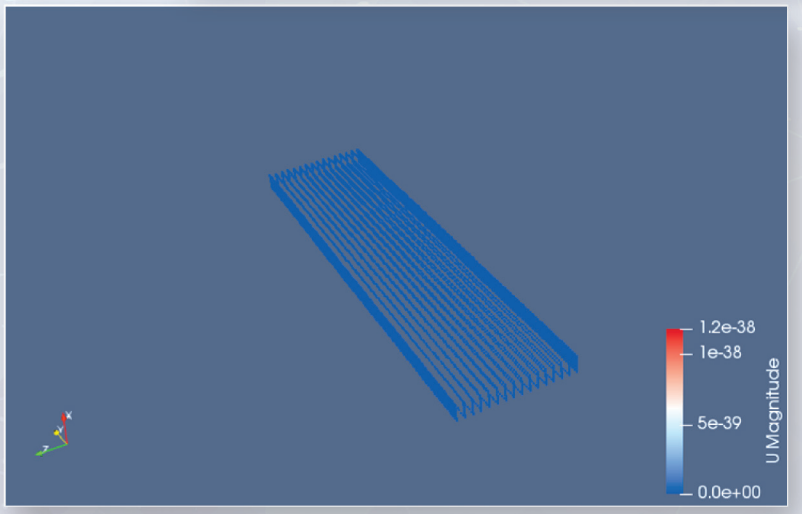


Рис. 5. Параметры изменения импульса

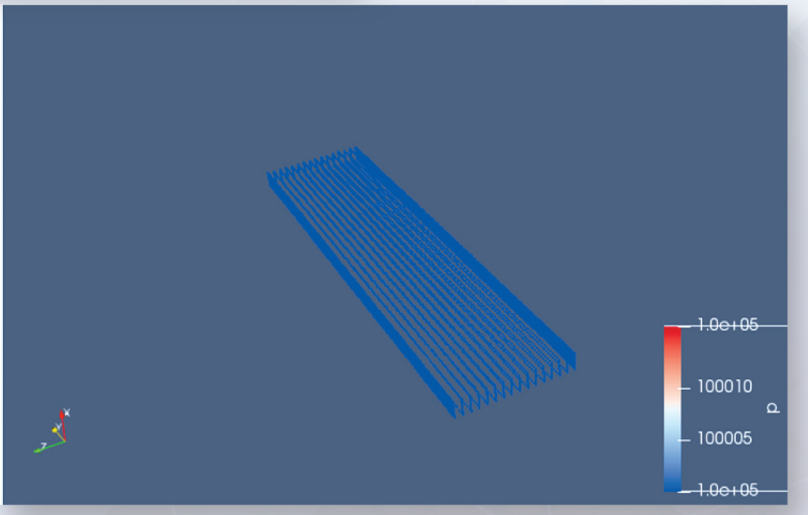


Рис. 6. Параметры изменения давления

Рисунок к статье **Д. И. Читалова**

«О РАЗРАБОТКЕ МОДУЛЯ ДЛЯ РАБОТЫ С РЕШАТЕЛЕМ buoyantSimpleFoam И УТИЛИТОЙ postProcess ПЛАТФОРМЫ OpenFOAM»

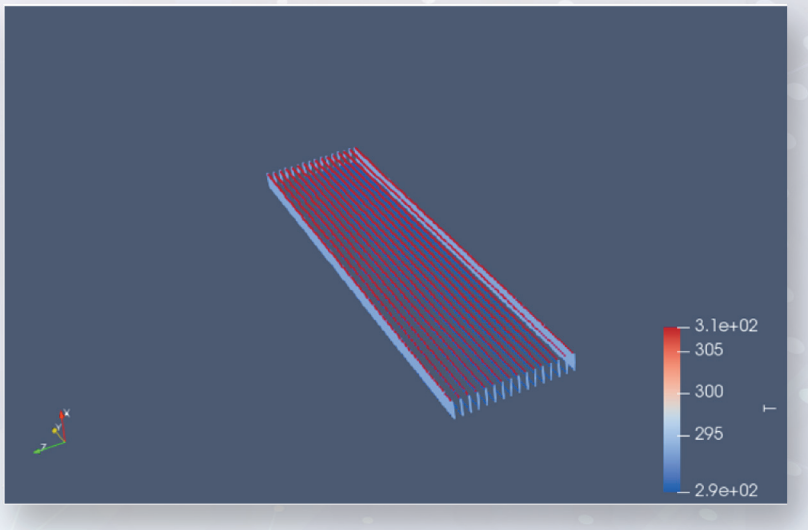


Рис. 7. Параметры изменения температуры

Рисунки к статье **А. А. Коротышевой, С. Н. Жукова**

«РЕАЛИЗАЦИЯ АЛГОРИТМОВ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ В НАВИГАЦИИ АВТОМОБИЛЬНОГО ТРАНСПОРТА С ИСПОЛЬЗОВАНИЕМ ОТКРЫТЫХ СЕРВИСОВ»

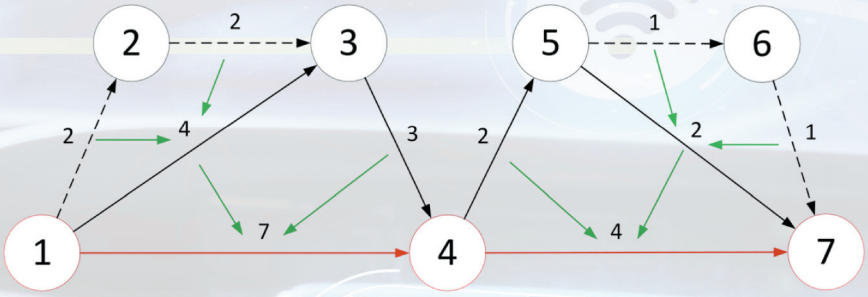


Рис. 2. Пример построения алгоритма Contraction Hierarchies

Рис. 5. Визуализация объекта дополненной реальности

ISSN 2220-3397. Программная инженерия. 2022. Том 13. № 2. 49 – 104. Подписной индекс по Объединенному каталогу "Пресса России" – 22765